Universidad de Burgos

Tesis Doctoral

Combinación de Clasificadores: Construcción de Características e Incremento de la Diversidad

Jesús Maudes

4-Octubre-2010

Directores:
Dr. Juan José Rodríguez Diez
Dr. César García Osorio

Agradecimientos

A mis directores de tesis, por su paciencia y dedicación, y sobretodo por su amistad. Sin ellos, sin sus conocimientos, sin sus ánimos y su comprensión, hubiera sido poco probable que hubiera conseguido acabar esta tesis.

A los compañeros del Área de Lenguajes y Sistemas Informáticos, por todos los cafés compartidos y por no cargarme de trabajo de gestión en estos últimos años, lo cual me ha facilitado en gran medida las cosas. También quiero agradecerles su disposición para compartir sus PCs en los inicios de esta tesis, cuando aún no disponíamos de máquinas específicas de cálculo. Dentro de los compañeros del área también quiero dar un agradecimiento especial a Carlos Pardo, por sus conocimientos enciclopédicos de Linux que sirvieron para facilitarme las validaciones experimentales. También quisiera animar, en general, a mis compañeros no doctores: si lo he hecho yo, qué no podríais hacer cualquiera de vosotros.

A Nicolás García Pedrajas por su amabilidad al permitirme usar el cluster de su equipo de investigación en la Universidad de Córdoba para completar los resultados experimentales. Sin esta ayuda algunos experimentos hubieran tardado mucho más. A Colin Fyfe, por sus comentarios (¡en castellano!).

Vaya también mi agradecimiento para todas aquellas personas e instituciones que de manera gratuita y, muchas veces, desinteresada han puesto a disposición del público recursos que han sido utilizados en la elaboración de la tesis. En concreto sobretodo a los desarrolladores de WEKA, pero también a los de TeXnicCenter, MiKTeX, Yap, OpenOffice, Linux, GNUPlot, FoxIt y GhostView; así como a los donantes de datos y mantenedores de los repositorios UCI y Statlib.

A mis vecinos, por no ser en absoluto *disturbing*. A los que les he robado horas: padres, hermanos, amigos, hijo, esposa y perro, que lleva más de un año en arresto domiciliario. Finalmente, a Maria José por convivir con mi yo más inaguantable.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Esta tesis presenta varios algoritmos de *Pattern Recognition* o *Reconocimiento de Patrones* [46]. Esta disciplina hace tiempo que salió de los laboratorios y las publicaciones científicas para impregnar nuestro día a día. Sistemas que reconocen la escritura [70], la voz [56], las imágenes [120], que descifran los genes [73], diagnostican enfermedades [6], interpretan las señales de tráfico [86], o rechazan el correo basura [105]. Todos ellos, son unos pocos ejemplos de estos sistemas con los que de manera casi imperceptible nos hemos acostumbrado poco a poco a convivir.

En reconocimiento de patrones un algoritmo *aprende* o analiza de forma automatizada un conjunto de datos existentes sobre una población de individuos o *instancias*. Este proceso de aprendizaje se conoce como *entrenamiento*. Cada uno de los individuos del conjunto de datos se caracteriza mediante un conjunto de valores o *atributos*.

El *Aprendizaje Supervisado* es un tipo especial de reconocimiento de patrones. En el aprendizaje supervisado cada instancia del conjunto de datos se puede representar como un par $(\mathbf{x}, y)$, donde y es un atributo especial en tanto el algoritmo ha de aprender a predecirlo a partir de los valores del vector $\mathbf{x}$, que son el resto de atributos de la instancia. El adjetivo *supervisado* se aplica debido a que en el proceso de entrenamiento se conocen y utilizan los valores de y de cada una de las instancias para realizar ese aprendizaje. También existe el aprendizaje *no supervisado*, en el que los posibles valores de y no son conocidos a priori, y el algoritmo trata de descubrir agrupaciones de instancias entre las que poder establecer una relación, por ejemplo, asociarles una etiqueta.

Dentro del aprendizaje supervisado puede ocurrir que los valores de y sean un conjunto finito de etiquetas, o bien un conjunto de valores continuos. En el primer caso el aprendizaje resolvería un problema denominado de *clasificación*, mientras que en el segundo el problema sería de *regresión*. Esta tesis está centrada en los problemas de clasificación dentro del aprendizaje supervisado. Un *clasificador* es un modelo que sirve para clasificar las entradas $\mathbf{x}$ de un conjunto de datos.

Para obtener un clasificador, previamente un algoritmo procesa un conjunto

de pares $(\mathbf{x}, y)$, el cual se denomina *conjunto de entrenamiento*. El algoritmo, analiza este conjunto de datos mediante un proceso que se conoce como *entrenamiento*, fruto del cual se obtiene un modelo predictivo (i.e., el clasificador) que es capaz de asignar a futuros valores del vector $\mathbf{x}$, los valores y que probablemente le corresponderían. El mérito del algoritmo estará en obtener modelos que yerren lo menos posible en estas asignaciones.

Una forma relativamente reciente de abordar el problema de la clasificación, es la utilización de *Multiclasificadores* o *Ensembles* [66, 102, 88]. Un multiclasificador es una agrupación de clasificadores, que se conocen como *clasificadores base* que combinan sus predicciones siguiendo un determinado esquema, con el fin de obtener una predicción más fiable que la que normalmente serían capaces de obtener en solitario.

El tipo de algoritmos que se presentan en esta tesis o son multiclasificadores, o bien son modificaciones aplicables a multiclasificadores existentes que son capaces de mejorarlos en determinadas situaciones. El común denominador a dos de ellos es la *construcción de características*. La construcción de características es el proceso mediante el cual se descubre información no presente sobre las relaciones de los atributos, aumentando el espacio de características al deducir o crear otras nuevas [51]. La construcción de características no presentes en el conjunto de datos es distinta a:

1. La extracción de características, que persigue encontrar un conjunto mínimo de nuevas características a través de alguna transformación y según un criterio de optimización.

2. La selección de características, que tiene por objeto eliminar aquellas que resulten redundantes.

3. La combinación de características que sirve para obtener grupos de las mismas que faciliten la tarea de clasificación.

Las nuevas características que construyen dos de los algoritmos que se presentan en esta tesis, en unos casos servirán para adaptar determinados tipos de problemas a determinados tipos de clasificadores, y en otros casos han servido para mejorar el comportamiento del multiclasificador, alterando el funcionamiento de sus clasificadores base.

La *diversidad* es una cualidad que idealmente debieran presentar los clasificadores base de un multiclasificador, en virtud de la cual las predicciones de los mismos tienden a ser distintas. Construir un multiclasificador exitoso requiere que sus clasificadores base acierten casi siempre en sus predicciones, pero que cuando yerren cada uno lo haga en distintas instancias; de lo contrario la ventaja de tener varios clasificadores colaborando, frente a uno solo, no existiría.

Dos de los métodos presentados en esta tesis servirán para aumentar la diversidad en sus clasificadores base, consiguiendo generalmente mejorar el rendimiento de los multiclasificadores a los que pertenecen.

1.1. Objetivos

Los métodos que se incluyen en esta memoria pueden agruparse en dos:

1. Por un lado, se presenta un método orientado a mejorar el acierto de métodos de clasificación numéricos frente a *datos nominales*.

2. Por otro lado, se presentan dos métodos orientados a incrementar la diversidad en multiclasificadores cuyos *clasificadores miembros* o *clasificadores base* sean del mismo tipo.

El primero de los métodos está enfocado a un tipo de problemas específico: aquellos en los que predominan los datos *nominales* o *categóricos*. Este tipo de datos son aquellos que toman sus valores de entre un conjunto de etiquetas finitas (e.g., la clase es un ejemplo de atributo categórico).

Hay clasificadores capaces de trabajar con este tipo de datos directamente, mientras que otros sólo admiten entradas numéricas, por ejemplo, los clasificadores lineales, en los que un hiperplano separa las instancias de una clase de las de otras.

Una de las aportaciones de esta tesis es un algoritmo para la construcción de nuevas características numéricas a partir de las características nominales de partida. El resultado será un multiclasificador en tanto que las nuevas características a construir, provienen de la salida de otro clasificador capaz de trabajar con datos nominales directamente (i.e., *Cascading* utilizando árboles de decisión como clasificadores base). Los resultados experimentales obtenidos indican una mejora frente a otras técnicas conocidas para resolver el mismo problema.

En cuanto al segundo grupo de métodos que se presentan, están orientados a mejorar el acierto en multiclasificadores que utilizan un único tipo de clasificador base replicado durante un número dado de iteraciones. La idea fundamental de estas mejoras es la introducción de algún tipo de perturbación aleatoria en el entrenamiento de los clasificadores base, de manera que dicha perturbación acabe por generar clasificadores base más diversos.

Los métodos correspondientes a este segundo grupo que se presentan en la tesis son:

1. Uno llamado *Disturbing Neighbors*, que es genérico en cuanto puede valer tanto para múltiples tipos de clasificadores base como para múltiples tipos de multiclasificador, y que está basado en construcción de características. El método obtendrá esas características adicionales a partir de la pertenencia a ciertas regiones de Voronoi definidas a partir de una selección aleatoria de un conjunto reducido de instancias de entrenamiento.

2. Otro llamado *Random Feature Weights*, que es más específico en cuanto está orientado a *bosques*, esto es: a multiclasificadores cuyos clasificadores base son todos de tipo árbol. Sin embargo, aunque es un esquema que restringe el tipo de clasificador base, su aplicación no está restringida a ningún tipo de bosque en concreto. En este caso, lo que se trastoca es el

normal funcionamiento del proceso de entrenamiento de los árboles mediante la introducción de un elemento aleatorio en el criterio de bifurcación de las ramas de cada árbol.

La mejora de la diversidad de ambas técnicas ha sido probada experimentalmente, así como la mejora de las tasas de acierto de los multiclasificadores así generados.

1.2. Organización de la tesis

El capítulo 2 introduce los conceptos que son comunes al resto de capítulos de la tesis. En concreto, se centra en explicar brevemente los clasificadores base y multiclasificadores de referencia utilizados. También analiza la técnicas de validación experimental utilizadas, junto con los diagramas que han servido para explicar los resultados obtenidos en términos de aumento de la diversidad. Estos diagramas son unos de los productos más relevantes que ha dado lugar el desarrollo de esta tesis.

El capítulo 3 presenta una configuración concreta de *Cascada* que permite mejorar la utilización de datos nominales por clasificadores que sólo admiten entradas de tipo numérico.

El capítulo 4 es el más extenso de la tesis. Presenta el método de *Disturbing Neighbors* que es capaz de aumentar la diversidad en clasificadores base, lo que generalmente mejora los resultados del multiclasificador al que pertenecen. Este método se ha probado utilizando árboles de decisión y máquinas de vectores soporte como clasificadores base. La elección de estos dos tipos de clasificadores responde a que entre si presentan grandes diferencias en un elemento que es clave para los miembros de un multiclasificador, como es el caso de su estabilidad frente a pequeños cambios en el conjunto de entrenamiento. El éxito del método es analizado desde la perspectiva del aumento de la diversidad en los clasificadores base, para lo que se aportan diagramas obtenidos experimentalmente, basados en la estadística *Kappa-Error*. Además, el capítulo presenta un análisis de lesiones que muestra qué elementos del algoritmo de Disturbing Neighbors son fundamentales y cuáles no son influyentes.

El capítulo 5 presenta el método *Random Feature Weights* que sirve para aumentar la diversidad en multiclasificadores con clasificadores base que sean árboles de decisión. En este capítulo se incluyen resultados experimentales para este método con y sin ruido, dado el buen comportamiento que tiene en ambos escenarios. Para comprobar el aumento de la diversidad en los Random Feature Weights también se aportan los correspondientes diagramas basados en la estadística *Kappa-Error*.

Finalmente, el capítulo 6 presenta cuáles son las conclusiones que se extraen de la tesis, y cuáles son las posibles líneas de investigación futuras.

Además, al final del volumen, se aporta un apéndice conteniendo las tasas de error correspondientes a las validaciones experimentales. Debido a que las validaciones experimentales incluyen un gran número de conjuntos de datos y

métodos, se ha preferido ubicarlas de esta manera, ya que si se hubieran incluido en el interior de sus correspondientes capítulos, debido a su tamaño, posiblemente no hubieran facilitado la lectura de los mismos. En los capítulos ya se incluyen las tablas y gráficos necesarios que permiten sintetizar toda esta información de una manera más concisa y eficiente, mientras que el apéndice permite comprobar de qué conjuntos de datos y métodos provienen esos resúmenes.

1.3. Aportaciones de esta tesis

Como resumen de todo lo comentado anteriormente en este capítulo, a continuación se numeran las principales aportaciones de esta tesis:

- Dos diagramas para facilitar la comparación visual de métodos de clasificación (subsección 2.3.3, página 55):
  - *Diagramas de Movimiento Kappa-Error.*
  - *Diagramas de Movimiento Relativo Kappa-Error.*
- Un nuevo clasificador para conjuntos de datos nominales, *Cascadas para Datos Nominales* (Capítulo 3, página 61).
- Dos nuevas familias de algoritmos de construcción de multiclasificadores:
  - *Disturbing Neighbors* (Capítulo 4, página 81).
  - *Random Feature Weights* (Capítulo 5, página 129).

Los resultados de esta tesis han tenido sus reflejos en las siguientes publicaciones:

- Capítulos de Libros: [78], [80].
- Actas de Congresos: [76], [77], [79], [81].

Capítulo 2

Conceptos Previos y Estado del Arte

2.1. Clasificadores base utilizados en esta tesis

En Reconocimiento de Patrones un clasificador no es más que una función que dado un vector de atributos $\mathbf{x}$ le asigna a éste una etiqueta y perteneciente al conjunto de clases del problema. Los *multiclasificadores* o *ensembles* son clasificadores que se construyen a partir de otros más simples, llamados *Clasificadores Base*. El entrenamiento de los clasificadores base puede seguir pautas distintas de un multiclasificador a otro, y la predicción final de un multiclasificador seguirá un esquema de combinación de las predicciones de los clasificadores base que también será propio de cada multiclasificador.

Los clasificadores base que se han utilizado en esta tesis son dos:

1. Máquinas de vectores soporte [1] (SVM) [115, 16, 108, 54, 24]. Una SVM es un clasificador lineal en un espacio que podría ser distinto al espacio original donde están definidos los vectores $\mathbf{x}$, y por tanto un hiperplano que clasifica las instancias por la pertenencia a cada una de las regiones de ese espacio que son limitadas por dicho hiperplano. La SVM se obtiene siguiendo unos determinados criterios de optimización, y aunque en principio parece requerir que el problema de clasificación presente regiones linealmente separables, es capaz de tratar problemas no separables linealmente a partir de la modificación de ciertos parámetros y/o de la elección del espacio donde se defina el hiperplano.

2. Árboles de decisión [103, 13, 63, 54]. Los árboles de decisión son una colección de nodos conectados entre sí, cada uno con un ascendiente -excepto el nodo raíz que no tiene ascendientes - y cero o más descendientes. Los

[1] En lo sucesivo se utilizará indistintamente la/s SVM (la/s máquinas de vectores soporte) y el/los SVM (el/los clasificadores tipo SVM).

nodos sin descendientes se conocen como hojas y tienen asociada una clase. Los nodos que no son hojas contienen una condición con la que evaluar la instancia que se esté clasificando. El proceso de clasificación consiste en ir recorriendo los nodos desde la raíz hasta una hoja. El recorrido viene dado por cómo «responda» la instancia a cada una de las decisiones que el árbol planteará en cada nodo, de forma que cada posible respuesta de la instancia se traducirá en por que nodo o rama continuará la instancia el proceso de clasificación.

Los dos clasificadores base elegidos son muy distintos en muchos aspectos, entre otros cabe destacar:

1. Por un lado una SVM al ser un hiperplano es un clasificador orientado a trabajar con datos numéricos, mientras que los árboles de decisión pueden trabajar directamente con datos nominales, pues en sus nodos intermedios pueden albergar comparaciones del tipo «este atributo es igual a tal etiqueta».

2. Las SVM dividen el espacio del problema en dos, por tanto son clasificadores binarios; si bien existen técnicas basadas en generar múltiples SVM para tratar el caso multiclase. Los árboles de decisión, sin embargo, pueden trabajar directamente con problemas multiclase.

3. Las SVM, en su planteamiento más simple, se benefician de que el problema sea linealmente separable, mientras que a los árboles de decisión no les influye esta propiedad del problema.

Finalmente, los árboles de decisión son capaces de aprender todos los ejemplos del conjunto de entrenamiento, incluso aquellos que sean banales o espurios y tengan poco que ver con la hipótesis predictiva que el clasificador pretende modelar. Se dice que los árboles de decisión podrían sufrir en ese caso un problema de *sobreentrenamiento*. Existen técnicas en la construcción de árboles que palían con éxito este efecto. A diferencia de los árboles, las SVM no suelen tener este problema. Se dice que un clasificador *generaliza*, cuando sus predicciones no acusan el efecto del sobreentrenamiento.

2.1.1. Máquinas de Vectores Soporte, SVM

Una forma de abordar el problema de la clasificación es utilizar un *modelo lineal*. Sea $\mathbf{x}$ una instancia de un conjunto de datos $X \subseteq \mathbb{R}^n$ cuyas clases toman los valores $y \in \{-1, +1\}$. Entonces, un modelo lineal viene definido por la siguiente ecuación del hiperplano en $\mathbb{R}^n$

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + b = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \tag{2.1}$$

Donde w_i son los coeficientes de ese hiperplano y b el término independiente. El sumatorio indica el producto escalar entre el vector normal al hiperplano y

cada instancia $\mathbf{x}$. El signo de $f(\mathbf{x})$ para un $\mathbf{x}$ determinado, indicará si la instancia queda a un lado u otro de dicho hiperplano, clasificándola como -1 o $+1$. En principio, los modelos lineales asumen que el conjunto de datos es *linealmente separable*. Es decir, es posible encontrar al menos un hiperplano que separe a todas las instancias de una clase de las instancias de la otra.

La utilización de modelos lineales no es reciente (e.g., el discriminante lineal de Fisher, 1936 [37] o el perceptrón de Rosenblatt, 1956 [104]). Las máquinas de vectores soporte (SVM) [115] pueden considerarse que pertenecen a esta familia. La peculiaridad de las SVM sobre sus predecesoras es que *maximizan el margen* y ello las permite obtener muy buenos resultados, porque son capaces de generalizar muy bien. Se adjunta a continuación, a modo de resumen, una serie de explicaciones acerca de esta propiedad de maximización del margen. Dichas explicaciones están extraídas de [16, 108].

Maximización del Margen Sea un conjunto de datos $\mathbf{X}$ linealmente separable por un hiperplano, y sean d_+ y d_- la distancias que separan a dicho hiperplano de las instancias más cercanas correspondiente a cada clase. La maximización del margen significa que el hiperplano es tal que maximiza la suma de esas dos distancias. Esto ocurre cuando $d_+ = d_-$, es decir cuando el hiperplano equidista de esos puntos. Escalando los coeficientes $\mathbf{w}$ del hiperplano, se puede hacer que para $\mathbf{x}_i \in \mathbf{X}$ e $y_i \in \{-1, +1\}$ se verifique:

$$\begin{aligned} \langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b \geq +1 \text{ para } y_i = +1 \\ \langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b \leq -1 \text{ para } y_i = -1 \end{aligned} \tag{2.2}$$

Cada una de estas dos restricciones indican que todas las instancias están separadas del hiperplano por una cierta magnitud, que en este caso se ha considerado unitaria (podría haberse elegido cualquier otra constante). Al combinar ambas restricciones en una sola expresión queda:

$$y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1 \geq 0, \forall i \tag{2.3}$$

De donde $d_+ = d_- = 1/\|\mathbf{w}\|$, por lo que el margen es $2/\|\mathbf{w}\|$, que se maximiza minimizando $\|\mathbf{w}\|^2$ sujeto a la restricción (2.3). Supuesto que $\mathbf{X}$ tiene l instancias, el problema se puede plantear tomando multiplicadores de Lagrange $\alpha_i \geq 0, i = 1, ..., l$.

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i (y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1) \tag{2.4}$$

Que es la forma *Primaria*[2] del problema. Ahora el problema se traduce en minimizar L_P respecto a las variables $\mathbf{w}$ y b, maximizándolo respecto las variables α_i. Igualando las derivadas parciales de $\mathbf{w}$ y b en (2.4) a cero se llega a:

[2] En lo que sigue la terminología en castellano de este apartado se tomará de [54].

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{2.5}$$

$$0 = \sum_{i=1}^{l} \alpha_i y_i \tag{2.6}$$

Sustituyendo (2.5) y (2.6) en (2.4), se obtiene la forma *Dual* del problema:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{2.7}$$

Como se verá más adelante, la forma dual tiene otras ventajas, pues permite hallar el hiperplano en espacios diferentes al original.

Si se denotasen como $\mathcal{H}_+$ y $\mathcal{H}_-$ a los dos hiperplanos que: (i) son paralelos al hiperplano de la SVM y, (ii) contienen los puntos que distan d_+ (d_-) del hiperplano de la SVM; se tendría que los puntos que verifican la igualdad en (2.3) son los que pertenecen a $\mathcal{H}_+$ y $\mathcal{H}_-$, que son los que están más cercanos a la zona limítrofe entre las dos regiones del problema, y que se conocen como *Vectores Soporte*, mientras los que verifican la desigualdad estricta en (2.3) son los que quedan del lado del plano más alejado de esa zona limítrofe. Los Vectores Soporte son los que dan nombre al método. Como se verá más adelante las instancias que no son vectores soporte no influyen en el cálculo de la SVM.

En el caso de que las regiones correspondientes a las clases no fuesen linealmente separables, se introducen en la formulación del problema unas variables positivas $\xi_i, i = 1...l$, llamadas *variables de holgura*, de manera que las restricciones en (2.2) quedan en la forma:

$$\begin{aligned} \langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b &\geq +1 - \xi_i \text{ para } y_i = +1 \\ \langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b &\leq -1 + \xi_i \text{ para } y_i = -1 \\ \xi_i &\geq 0 \ \forall i \end{aligned} \tag{2.8}$$

De donde se deduce que cuando una instancia $\mathbf{x}_i$ esta mal clasificada por el hiperplano, le ha de corresponder un ξ_i mayor que uno, de donde, a su vez se deduce que $\sum_i \xi_i$ es una cota superior del número de errores de entrenamiento. Al incorporar el coste de estos errores de entrenamiento, la función objetivo a minimizar pasa de ser $\|w\|^2/2$ a ser $\|w\|^2/2 + C \sum_i \xi_i$, donde C es un parámetro de coste a elegir para cada conjunto de datos, de forma que cuanto mayor sea C más se penalizarán los errores. La forma primal del Lagrangiano queda ahora como:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i - \sum_{i=1}^{l} \alpha_i \{y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1 + \xi_i\} - \sum_{i=1}^{l} \mu_i \xi_i \tag{2.9}$$

Donde los μ_i son los multiplicadores de Lagrange asociados a la restricción de que todos los ξ_i sean mayores o iguales que cero. Al igualar las derivadas parciales de $\mathbf{w}$, b y ξ_i a cero se llega a:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{2.10}$$

$$0 = \sum_{i=1}^{l} \alpha_i y_i \tag{2.11}$$

$$0 = C - \alpha_i - \mu_i \tag{2.12}$$

Sustituyendo (2.10), (2.11) y (2.12) en (2.9) queda la forma dual para el caso no linealmente separable:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{2.13}$$

que parece la misma que para el caso separable (2.7). La diferencia es que la restricción (2.12) unida a que las μ_i son mayores o iguales que cero, da lugar a que en el caso no separable habrá de verificarse $0 \leq \alpha_i \leq C$. Es decir, ahora las α_i están acotadas también superiormente. En [108] puede encontrarse una explicación intuitiva de esta cota superior. Esa explicación se basa en que la forma primal del Lagrangiano para el caso no separable (2.4) ha de minimizarse respecto a $\mathbf{w}$ y b, pero a la vez ha de maximizarse respecto a los α_i, es decir se trata de hallar un punto de silla.

Cuando una instancia esté mal clasificada la restricción $y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1 \geq 0, \forall i$, no se verifica haciendo que el término $(y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1)$ sea negativo. Como $\alpha_i \geq 0$ y el sumatorio tiene signo menos, haciendo los multiplicadores α_i todo lo grandes que se desee, el Lagrangiano va aumentando respecto a esa variable. Por tanto, en esas instancias la maximización del Lagrangiano respecto a los α_i lleva a que los α_i tiendan a infinito. Al añadir el coste C lo que se hace es limitar el crecimiento de los α_i para esos casos. Esta misma línea de razonamiento sirve para ver como los α_i toman el valor cero en el caso de que la instancia a clasificar verifique la desigualdad estricta $y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1 > 0$ (i.e., esté bien clasificada pero no sea un vector soporte), pues $y_i(\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1$ contribuye con signo positivo al sumatorio que a su vez tiene signo menos, luego $\alpha_i = 0$ anula ese término, maximizando el Lagrangiano.

La forma dual (2.13), tiene la ventaja de que el Lagrangiano queda expresado únicamente en términos del producto escalar de las instancias del conjunto de datos. Esto permite generalizar el método utilizando funciones de decisión que no sean hiperplanos en el espacio del conjunto de datos, pero si que lo sean en otros espacios quizás de dimensión superior. Una *función núcleo* es una función K, tal que para dos instancias cualquiera del conjunto de datos $\mathbf{x}_i$ e $\mathbf{x}_j$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i).\phi(\mathbf{x}_j) \rangle$$
$$\phi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathcal{F} \tag{2.14}$$

Es decir, $\phi(\mathbf{x})$ es una transformación que permite llevar a las instancias del espacio original a otro espacio $\mathcal{F}$, con otra dimensión (incluso de dimensión infinita). La función K permite calcular el producto escalar $\mathbf{x}_i \cdot \mathbf{x}_j$ sin necesidad de utilizar la transformación ϕ. Problemas que no son linealmente separables en el espacio original del conjunto de datos pueden serlo, o estar más cerca de serlo en $\mathcal{F}$.

En esta tesis no se han utilizado nunca una SVM con ninguna función núcleo que trasladara el problema a otro espacio (i.e., el hiperplano siempre ha estado definido en el espacio original del problema), o dicho de otra forma, la función núcleo utilizada ha sido la *lineal*. Esto ha sido debido a que:

1. En los métodos propuestos, las SVM se han utilizado como clasificadores base de algún multiclasificador, y la opción lineal es la más rápida computacionalmente, lo cual es muy interesante cuando cada multiclasificador tiene que calcular por ejemplo 50 SVM, como ha ocurrido en bastantes de los experimentos.

2. El interés de los experimentos no ha sido tanto el de encontrar la mejor SVM para cada problema, sino tener una idea comparativa de los métodos multiclasificadores que usan SVM. Para ello, una opción razonable y a la vez manejable en cuanto a número de casos posibles a experimentar, es que todos usaran el mismo tipo de SVM.

3. El resto de funciones núcleo necesitan de más parámetros además del ya mencionado parámetro de coste. La sensibilidad a estos parámetros por parte de la SVM puede ser grande, y no existe una razón objetiva para dar unos valores concretos; habría que hallar los valores óptimos de los parámetros para cada caso. Optimizar los parámetros en multiclaficadores de 50 SVM, con validaciones experimentales sobre un gran número de conjuntos de datos, tiene un elevado coste computacional y por ello se ha renunciado a hacerlo. Por ello, parece razonable usar un tipo de SVM lo menos sensible posible a la optimización de parámetros.

Existen funciones núcleo para SVM que permiten trabajar con *strings* o secuencias de símbolos directamente (i.e., *String Kernels* [53]). Dos aplicaciones fundamentales de este tipo de funciones núcleo son la categorización de textos [24] y la de secuencias en bioinformática (e.g., proteínas formadas por aminoácidos) [24, 71]. En el primer caso la función núcleo puede depender de qué palabras tienen en común los dos textos, en el segundo de alguna función de similitud entre las secuencias.

En el caso de la categorización de textos es posible conocer de antemano el *léxico* o conjunto de secuencias posibles, y la clasificación suele ser multietiqueta (i.e., a una instancia se le asignan varias clases simultáneamente).

En el caso de la clasificación de proteínas, el léxico se asume que consiste en todas las combinaciones posibles de símbolos de una longitud máxima dada. Además, en este caso, la coincidencia de las secuencias no tiene porqué ser exacta, pues puede haber símbolos intercalados (e.g. en una proteína podría haber bases intercaladas en la cadena de aminoácidos).

Los SVM con String Kernel no necesitan convertir los datos a numéricos, sino que procesan directamente las cadenas. Para ello, las funciones núcleo asociadas se definen internamente en base a expresiones que ponderan la importancia de las secuencias según su peso, y que de alguna forma hacen un recuento del número de secuencias en común entre las dos instancias x_i y x_j que sirven de parámetros en la ecuación (2.14), de forma que si las instancias son parecidas, su la función núcleo devolverá un valor elevado.

Estos problemas específicos para los que se utilizan String Kernels no son los que se abordan en esta tesis, que expone métodos que utilizan SVM que trabajan con conjuntos de datos con atributos numéricos y/o nominales (ver capítulo 4), y métodos que utilizan SVM con datos únicamente nominales (ver capítulo 3). En el caso de los atributos nominales, cada atributo toma un valor de entre varias etiquetas posibles, las cuales no pueden ser entendidas como secuencias de símbolos.

Por ello, se asume que las SVM que se van a utilizar, al ser clasificadores que trabajan con entradas numéricas, necesitan que las entradas categóricas o nominales sean transformadas en números. Para ello, en esta tesis, mientras no se indique lo contrario, las SVM utilizadas realizan la transformación *nominal a binario* (NBF). Esto es, cada atributo nominal que pudiese tomar m valores se sustituye por m atributos binarios, de forma que cuando un atributo categórico original tome el valor nominal i-ésimo, el atributo binario i-ésimo tomará el valor uno, y el resto tomará valores cero.

Las SVM, por otro lado, son clasificadores binarios. Es decir, en principio no son capaces de trabajar directamente con conjuntos de datos con más de dos clases. En esta tesis, se asume que las SVM utilizadas usan la aproximación *uno contra uno* [60], para resolver dicho problema. Es decir, se crea una SVM por cada combinación de dos clases, entrenado únicamente con instancias de dichas dos clases. En un problema de c clases esto da lugar a $c(c-1)/2$ SVM. Al predecir se toma la clase que más veces es predicha por todas estas SVM.

Asimismo, también se asumirá en adelante, que los atributos que sirven de entrada a una SVM están normalizados a partir de los valores del conjunto de entrenamiento, para evitar que unos predominen artificialmente sobre otros.

Finalmente, conviene comentar que las SVM son clasificadores sensibles a cambios de pesos en las instancias y ello permite que sean utilizadas dentro de multiclasificadores que puedan aprovechar esta característica (p.e Boosting, como se verá más adelante). Para ello, hay que ponderar el coste del error de clasificación según la importancia de cada instancia. Esto se logra redefiniendo el problema de minimizar $\|w\|^2/2+C\sum_i \xi_i$ como minimizar $\|w\|^2/2+C\sum_i \xi_i p_i$, donde p_i es el peso de la instancia $\mathbf{x}_i$ dentro del conjunto de datos. La forma dual que se obtiene es la misma, pero la restricción $0 \leq \alpha_i \leq C$, queda ahora de la forma $0 \leq \alpha_i \leq Cp_i$.

2.1.2. Árboles de Decisión

Un árbol de decisión es un clasificador que se compone de un conjunto de nodos unidos por arcos dirigidos. Cada nodo representa una decisión o comparación. Normalmente recibe un arco entrante y es el origen de varios arcos salientes. Hay un nodo especial en el árbol que se conoce como *raíz*, el cual es el único que no recibe ningún arco entrante. Cuando se va a clasificar una instancia, se somete a la comparación correspondiente al nodo raíz. Dependiendo del resultado de esta comparación, el nodo raíz encamina la instancia hacia otro nuevo nodo (un *descendiente*) siguiendo uno de los arcos dirigidos. De este nuevo nodo cuelga un subárbol o *rama*. El nuevo nodo actúa como raíz de la rama, por lo que puede repetirse el proceso de comparación y encaminamiento hacia otra rama de nivel inferior de manera recursiva, así hasta llegar a un nodo terminal u *hoja* del que ya no sale arco alguno. Las hojas no tienen asociadas ninguna decisión o comparación, y por tanto no se ramifican más. En cambio, las hojas tienen asociada la clase que el clasificador árbol va a asignar a las instancias que lleguen a esa hoja, o dependiendo de la implementación una distribución de la probabilidad con la que la instancia a evaluar pertenece a cada una de las clases. Es normal que varias hojas predigan la misma clase y habrá normalmente al menos una hoja por cada clase.

Una ventaja de los árboles de decisión es que representan la estructura del conjunto de datos de una forma que es fácil de comprender para un humano, pues la ruta seguida por una determinada instancia desde el nodo raíz hasta un nodo hoja, se interpreta como una secuencia de predicados (las comparaciones de cada nodo) encadenadas por conectores lógicos *AND*. Es más, la comparación en cada nodo normalmente sólo afecta a un atributo, aunque hay algoritmos que permiten obtener árboles en los que las comparaciones en los nodos pueden involucrar a varios atributos, dificultando la comprensión del árbol. En esta tesis se han considerado siempre árboles con nodos que efectúan la comparación sobre un solo atributo.

El algoritmo de construcción de árboles utilizado casi siempre en esta tesis es *C4.5* [92]. En [119] se puede encontrar un resumen de C4.5. Indirectamente, también se ha utilizado una modificación de *REPTree* (i.e., *Reduced-Error Pruning Tree* [117]), como clasificador base en la implementación del multiclasificador *Random Forest* [11] (descrito más adelante en la sección 2.2.2), que es uno de los multiclasificadores de referencia que se ha considerado en algunos de los capítulos. La herramienta con la que se han hecho las validaciones experimentales (i.e., WEKA [117]), provee de una implementación eficiente de Random Forests basada en REPTrees.

La construcción de un árbol de decisión es desde arriba hacia abajo. Se comienza buscando la comparación que se va a asociar al nodo raíz. Entonces se divide el conjunto de entrenamiento en tantas partes como ramas, de manera que a cada parte se asocian todas las instancias de entrenamiento que al ser testadas visitarían la raíz de esa rama. Este proceder es recursivo, de manera que cada rama vuelve a dividirse así en otras, hasta llegar a algún criterio de parada del algoritmo.

Los árboles de decisión pueden trabajar tanto con datos nominales como numéricos.

En el caso de atributos numéricos hay que buscar un valor umbral. Una vez encontrado dicho umbral, el árbol se bifurca en dos ramas: los que son menores que el umbral y los que no lo son. Para determinar ese valor umbral, es necesario ordenar las instancias por el valor del atributo. Tanto REPTRee como C4.5 están optimizados de manera que sólo hacen esa ordenación en el nodo raíz, manteniendo una estructura de datos que permite aprovechar esa reordenación a medida que van construyendo las ramas.

En el caso de un atributo nominal, por el contrario, el número de bifurcaciones en un nodo puede ser mayor que dos. En el caso de C4.5 y REPTRee el número de bifurcaciones que genera un nodo que realiza una comparación contra un atributo nominal es igual al número de posibles valores que pueda tomar ese atributo; si bien es posible modificar de manera muy simple el algoritmo de construcción del árbol para forzar también en este caso bifurcaciones binarias con el criterio «instancias para las que el atributo en cuestión toma un determinado valor vs. instancias para las que no lo toma».

La decisión de cuál es el atributo sobre el que comparar en cada nodo se hace de la siguiente manera. Primero, para cada posible atributo sobre el que bifurcar, se estiman las probabilidades P_i, de pertenecer a cada una de las clases c_i, que tenga una instancia que visite ese nodo, lo cual se hace computando la proporción de instancias de entrenamiento de cada clase que llegarían al nodo. Con esas probabilidades se mide la *impureza* del nodo candidato, para lo cual hay diferentes técnicas.

En el caso de los árboles C4.5 y REPTree, el criterio seguido para medir esa impureza es la *entropía*:

$$i(t) = -\sum_{j=1}^{c} P_j \log_2 P_j \tag{2.15}$$

Donde $i(t)$ es la impureza (o entropía) del nodo t, y c es el número de clases. Para una rama pura, todas las instancias de entrenamiento son de la misma clase, y la entropía alcanza su valor mínimo cero (se considera $0 \log 0 = 0$). En el otro extremo, si las instancias presentasen una distribución uniforme en cuanto a sus clases, se alcanza la impureza máxima que sería $\log c$.

La entropía es una medida de la información necesaria para clasificar a una instancia que haya recorrido el camino hasta ese nodo. Si el nodo es puro, no se necesita más información, y su valor es cero. Si la distribución de clases en las instancias correspondientes al nodo fuese una distribución uniforme, se necesitaría la máxima información posible.

Una vez calculado $i(t)$, se calcula de la misma forma la entropía por cada valor posible de ese atributo, y se pondera multiplicándola por la probabilidad de que una instancia de entrenamiento tome dicho valor (i.e., sea enrutada hacia la rama correspondiente a ese valor). Al sumar en un atributo todas estas entropías ponderadas, lo que se tiene es la media de la entropía en cada una de las bifurcaciones.

Se define la *ganancia de información* asociada a elegir bifurcar por un determinado atributo, como la diferencia entre la entropía del nodo actual y la suma ponderada de las entropías correspondientes a bifurcar por ese atributo.

$$g(t) = i(t) - \sum_{v} \frac{|t_v|}{|t|} i(t_v) \tag{2.16}$$

Donde, $g(t)$ es la ganancia de información del nodo t respecto a sus v nodos hijos. En la ecuación 2.16, $|t|$ representa el número de instancias de entrenamiento que llegan a un nodo t, y $|t_v|$ el número de instancias que llegan a un nodo hijo de t.

En el caso de REPTree la ganancia de información basta para decidir cuál es el atributo más adecuado para bifurcar, pues se toma aquel que presente mayor ganancia.

Un árbol que resulte de bifurcar por un atributo nominal que tenga muchos valores, tendrá también muchas ramas, el número de instancias de entrenamiento que correspondan a cada una de las ramas será por tanto menor que si, por ejemplo, la bifurcación fuese binaria, y por tanto es más fácil que esas ramas tiendan a ser puras. Por ello, si se toma como criterio la ganancia de información, hay una cierta preferencia implícita a bifurcar por los atributos nominales con muchos valores. Este efecto no es deseable, pues se llega a árboles que desprecian de forma prematura el resto de atributos, al llegar en seguida a ramas con pocas instancias.

C4.5 introduce una mejora en este criterio para evitar este efecto, pues utiliza como criterio el *gain ratio* en lugar de la ganancia de información. El gain ratio $r(t)$ de un nodo t se define como el cociente entre la ganancia de información y la entropía en ese nodo sin tener en cuenta la clase. Esto es:

$$r(t) = \frac{g(t)}{e(t)} \tag{2.17}$$

Donde

$$e(t) = -\sum_{j=1}^{v} P_j \log_2 P_j \tag{2.18}$$

Siendo P_j la probabilidad de que el atributo tome uno de los v valores posibles de ese atributo en ese subárbol. Intuitivamente la mejora se explica porque al bifurcar por más ramas, el número de instancias en cada rama es menor. Debido a que el valor absoluto de los logaritmos de P_j (que sólo toma valores entre cero y uno) crecen muy rápidamente a medida que P_j se acerca a cero, una pequeña disminución en el número de instancias que corresponden a las ramas disminuye el valor de P_j, y ésto, a su vez, aumenta el valor de $\log_2 P_v$ a mayor ritmo que el de la disminución de P_j. Esto hace que $e(t)$ crezca, y al crecer $e(t)$, $r(t)$ disminuye.

En cuanto al criterio de parada en la construcción del árbol, el más habitual es que en la rama en curso sólo haya instancias de una única clase (*ramas puras*).

Cuando el algoritmo se encuentra con una rama de este tipo ya no puede seguir bifurcando, y lo marca como hoja.

Sea un conjunto de datos en el que no existen instancias que, teniendo idénticos valores en todos su atributos, difieran en el valor de la clase (i.e., no contiene instancias que no sean discernibles). En este tipo de conjuntos de datos es posible construir un árbol con error cero sobre el conjunto de entrenamiento. Por tanto, los árboles tienen la capacidad de «memorizar» el conjunto de datos de entrenamiento. Es más, si hubiese ruido en el conjunto de datos, el algoritmo sería capaz de aprenderlo, provocando que algunas predicciones sean erróneas.

Para evitar este problema, opcionalmente la construcción de árboles se puede complementar con un un proceso de *poda* que acorte la profundidad de ciertas ramas, de manera que el árbol «olvide» aquellas ramas que representan situaciones poco frecuentes, que son la principal fuente de la mencionada inestabilidad. Esta es la razón que hace conveniente la poda. La poda ayudará a que el árbol fuese más estable y generalizase mejor.

No obstante, en esta tesis habrá ocasiones en las que se utilicen árboles sin poda, pues la inestabilidad, aunque es una característica no deseable en clasificadores que actúan en solitario, si puede ser deseable cuando actúa como clasificador base de un multiclasificador.

Hay dos tipos de podas. En la *postpoda* primero se construye el árbol por completo, típicamente hasta llegar a que todas sus hojas sean puras. En la *prepoda*, la propia poda es parte del criterio de parada en la construcción del árbol, pues una rama puede pasar a ser hoja directamente sin llegar a ser un nodo puro. En [66] se comentan algunos de estos criterios de prepoda.

La prepoda puede evitar mucho cálculo extra, pero presenta el llamado «efecto horizonte», en tanto no es posible vislumbrar cuales son las consecuencias de una poda, pues una decisión de poda prematura puede detener el crecimiento del árbol hacia nodos con los que a la postre si que hubiera sido importante contar. Por ejemplo, en la construcción de un nodo, un determinado atributo puede parecer que no va a servir para discriminar la clase y por lo tanto puede ser candidato a sufrir una prepoda. Pero ese atributo combinado con otros de sus nodos descendientes pudiera ser que sí contribuyese notablemente a hacer esa discriminación, de lo cual quizás no es posible darse cuenta hasta llegar a las hojas puras y luego hacer una postpoda. Por eso la mayoría de árboles hacen postpoda.

Los árboles utilizados en este trabajo, C4.5 y REPTree, admiten ambos la utilización de postpoda. La técnica de postpoda de REPTree es la que da su nombre al algoritmo *Reduced-Error Pruning* [93] (i.e., REP). En esta tesis REPTree se ha utilizado siempre sin poda, la razón se explicará en la sección 2.2.2, por lo que no tiene interés para este trabajo describir REP. Para una descripción de REP ver [117]. La postpoda para C4.5 se denomina *Error-Based Pruning* o *EBP*, y se describe a continuación.

La poda de C4.5 se lleva a cabo partiendo de las hojas y ascencendiendo en el árbol hasta la raíz. Al llegar a un nodo t el algoritmo puede tomar tres decisiones:

1. Dejar la rama que cuelga de t sin podar.
2. Sustituirla por una hoja hija.
3. Sustituirla por una rama completa correspondiente a uno de sus nodos hijos, o *injerto*. Esto exige volver a construir la propia rama hija, pues ahora le llegan más instancias de entrenamiento que cuando colgaba de t.

En cuanto al criterio para saber si hay que podar en un nodo, los algoritmos de poda se basan en comparar el error del árbol antes y después de una operación de poda, de manera que sólo se lleva a cabo en el caso de que esta diferencia resultase favorable a la poda. Naturalmente, el problema está en que el árbol sin podar tendrá un error de entrenamiento cero o muy próximo a cero (si hay instancias no discernibles no sería exactamente cero). Por tanto, sería muy optimista asumir que el error antes de la poda es el error de entrenamiento sin más. La solución más sencilla consistiría en retirar un pequeño número de instancias del conjunto de entrenamiento antes de construir el árbol, y hacer la estimación de los errores con dicho conjunto (como por ejemplo hacen los REPTrees al aplicar Reduced Error Pruning). Sin embargo, esta aproximación plantea un problema para aquellos conjuntos de datos que cuenten con pocas instancias de entrenamiento.

C4.5 evita este problema utilizando todas las instancias de entrenamiento de un subárbol para ver si ese nodo t se puede eliminar o no. C4.5 compensa el efecto optimista de utilizar el conjunto de entrenamiento haciendo una estimación pesimista del error (i.e., el límite superior del intervalo de confianza, dado un umbral del confianza especificado por parámetro).

Para que se produzca un injerto del nodo hijo t' sustituyendo al padre t, la estimación del error en ese hijo (E'_t) deberá ser menor o igual que la del padre (E_t). En el caso de que se produjera la conversión de t en hoja, habrá que obtener las estimaciones del error $E'_1, ...E'_d$ para cada uno de los d hijos de t, y hallar la media ponderada de las mismas E''_t (se pondera por el porcentaje de instancias de entrenamiento que encierre cada uno de los d subárboles). La conversión de t en hoja tendrá lugar si E''_t es menor o igual que E_t.

Por último, y dado que existen métodos multiclasificadores basados en cambiar los pesos de las instancias del conjunto de entrenamiento, conviene comentar que adaptar los árboles de decisión para que sean sensibles a los pesos de las instancias es inmediato. Basta con modificar en todos los razonamientos anteriores las operaciones de contar instancias, por operaciones de suma de pesos de las instancias. Por ejemplo, cuando se calcule la proporción de instancias de una clase en un subárbol, en lugar de contar cuantas hay, se sumaría el peso de las mismas.

2.2. Multiclasificadores

Un *Multiclasificador* es un conjunto de clasificadores cuyas predicciones individuales se combinan de alguna forma para así obtener una predicción final

conjunta. Según Dietterich [28] existen tres razones para preferir usar un esquema de combinación de clasificadores, antes que un solo clasificador:

1. La primera sería de tipo *estadístico*, pues elegir un solo clasificador de entre un conjunto es arriesgado. Aun cuando el error de entrenamiento de ese clasificador sea cero, no es conocida la respuesta que va a tener frente a datos que aún no le han sido presentados. En ese sentido, combinar de alguna forma varios clasificadores no es una solución tan buena como quedarse con un solo clasificador siendo éste el mejor posible, pero reduce el riesgo de tomar uno que esté lejos de serlo.

2. *Computacionalmente* los algoritmos de entrenamiento de muchos clasificadores dependen de algún elemento que los hace llegar a un mínimo local del espacio de posibles clasificadores para un conjunto de datos dado. La combinación de varios clasificadores puede paliar este efecto.

3. La tercera razón atiende a la dificultad o imposibilidad que pueda tener una familia de clasificadores para obtener un modelo que *represente* adecuadamente un determinado problema. Por ejemplo, un problema que no presenta separabilidad lineal, no es adecuado tratarlo con un modelo lineal, ya que no es posible encontrar un hiperplano que consiga separar las regiones de las clases en el espacio del problema. Sin embargo, es posible que la combinación de varios de estos clasificadores puedan aproximar una superficie de decisión que se adapte mucho mejor al problema.

Para Kuncheva [66] existen cuatro niveles de actuación en la construcción de multiclasificadores:

1. El nivel de combinación de los clasificadores base. A este nivel existen distintos modos de combinación de las predicciones individuales.

2. El nivel de los clasificadores base, seleccionando qué tipo de clasificadores base se van a utilizar.

3. El nivel de las características. Es posible obtener distintos clasificadores base quitando, añadiendo y/o modificando las características del conjunto de datos, de una manera distinta para cada clasificador base, aun cuando los clasificadores base sean del mismo tipo.

4. El nivel del conjunto de datos, haciendo que con algún criterio los conjuntos de datos de cada clasificador base sean distintos, aun cuando los clasificadores base sean del mismo tipo.

En esta tesis se abunda especialmente en la construcción de nuevas características, por lo que podría decirse que el nivel de actuación predominantes es el tercero de la lista.

Además Kuncheva recoge tres perspectivas que pueden servir para catalogar los multiclasificadores:

- *Fusión vs. Selección*: En la fusión cada clasificador base tiene conocimiento de todo el espacio correspondiente al conjunto de datos, mientras que en la selección, cada clasificador base se especializa en una parte del mismo.

- *Optimización de la Decisión vs. Optimización de la Cobertura*: Los métodos con optimización de la decisión se centran en cómo combinar la predicción de los distintos clasificadores base, mientras que los del grupo de optimización de la cobertura, asumiendo la existencia de una técnica de combinación, se centran en cómo entrenar los clasificadores base para que resulten diversos.

- *Entrenable vs. No-Entrenable*: Una vez obtenidos los clasificadores base, unos multiclasificadores necesitan su propio proceso de entrenamiento para adaptarse a los mismos, mientras que otros multiclasificadores siguen un esquema fijo de combinación y no necesitan ese entrenamiento adicional.

Un multiclasificador cualquiera puede ser visto desde las tres perspectivas, aunque ninguna de las tres divisiones por separado establece una frontera clara que permita encajar a todo multiclasificador a un lado u otro de la misma; pues hay casos en los que quedan a medio camino.

En lo concerniente a esta tesis los métodos presentados se encuadrarían dentro de los métodos de fusión y orientados a la optimización de la cobertura. En cuanto al entrenamiento, en general son sin entrenamiento del esquema de combinación, con la excepción de la Cascada que sí se puede considerar un esquema con entrenamiento, tal como se verá en el siguiente capítulo. El resto de esquemas que se presentan son claramente sin entrenamiento, debido a que se centran en hacer que los clasificadores base sean más diversos.

La *Diversidad* es una cualidad muy interesante de los clasificadores base de un multiclasificador. Es claro que el éxito de un multiclasificador reside en gran parte, en el propio éxito de sus clasificadores miembro; pero si todos coincidieran en sus predicciones, el multiclasificador resultante sería equivalente a tener uno solo de ellos. Por eso, es clave que las instancias en las que un clasificador base falle, no sean las mismas que las de los otros miembros del multiclasificador, compensando así los aciertos de unos, con los fallos de otros. Esta es la idea de diversidad. Cuando en esta tesis se ha explorado esta idea se ha llegado incluso a empeorar los resultados individuales de los clasificadores base de partida, para poder llegar así a clasificadores base más diversos. Por tanto, en el presente trabajo existen métodos que priorizan el aumento de la diversidad frente a la disminución del error en los clasificadores base.

Los métodos que se presentan en la tesis se han experimentado comparándolos con métodos existentes cuyo éxito es conocido. A continuación se describen todos ellos brevemente. En el siguiente sumario hay claramente dos grupos:

1. Por un lado: *Bagging*, *Random Forests*, *Random Subspaces* y *Boosting*, son métodos que utilizan un único algoritmo de construcción de clasificadores base. Por lo tanto, cada clasificador base se diferencia de los demás únicamente en que ha habido algún tipo de variación en las instancias de

entrenamiento que ha procesado (e.g., muestras con distintas instancias, distintos atributos etc.). Estos métodos se utilizarán en los capítulos 4 y 5.

2. Por otro lado: *Cascading*, *Stacking* y *Grading*, son métodos que combinan varios algoritmos de construcción de clasificadores para construir un único clasificador final. Estos métodos se utilizan en el capítulo 3.

2.2.1. Bagging

Bagging [7] es el acrónimo de *Bootstrap AGGregatING*, que viene a significar agregado de remuestreos. La idea es muy simple. El método construye N clasificadores base, cada uno de ellos utilizando el mismo algoritmo, pero utilizando distintos conjuntos de entrenamiento. Cada conjunto de entrenamiento se obtiene a partir de un remuestreo con reemplazamiento (*bootstrap*) de un determinado porcentaje de instancias del conjunto de entrenamiento original (habitualmente del 100 %), que es lo que da nombre al método. Al tratarse de un remuestreo con reemplazamiento, es normal que una misma instancia aparezca varias veces en el conjunto de datos utilizado para entrenar un mismo clasificador base.

El multiclasificador, una vez construido, hace sus predicciones a partir de las predicciones de sus clasificadores base, de forma que la clase con más votos es la que finalmente es tomada como resultado.

Por tanto, para elegir correctamente un buen clasificador base de Bagging se hace interesante que sea sensible a las pequeñas variaciones en el conjunto de entrenamiento que pueda introducir el remuestreo (i.e., el clasificador base sea inestable) como por ejemplo lo son los árboles de decisión.

Una ventaja evidente de Bagging es que es un multiclasificador cuyo entrenamiento es fácilmente paralelizable.

2.2.2. Random Forests

Los Random Forests según Breiman [11] consisten en «una colección de clasificadores estructurados como árboles $\{h(\mathbf{x}, \Theta_k), k = 1, ...\}$ donde $\{\Theta_k\}$ son vectores aleatorios independientes e idénticamente distribuidos, y cada árbol produce un voto de la clase más popular para una entrada $\mathbf{x}$». Los vectores aleatorios $\{\Theta_k\}$ representan un conjunto de números aleatorios que determinan la construcción de cada árbol. Por ejemplo, en Bagging cada $\{\Theta_k\}$ podría ser un multiconjunto de enteros indexando las instancias correspondientes a un remuestreo aleatorio, o en los Random Subspaces (ver la siguiente sección) otro conjunto de enteros representando los atributos seleccionados aleatoriamente para entrenar cada clasificador base. Por tanto, esta definición de Breiman es un marco que engloba a cualquier multiclasificador basado en árboles, en el que se variara algún parámetro de cada árbol de forma aleatoria.

En [11], se describen dos implementaciones de Random Forests, las cuales son variantes de Bagging de árboles sin poda:

1. *Forest-RI* (i.e., Random Input Selection), que es la implementación más sencilla y común. En ella cada árbol está compuesto de nodos en los que sólo se puede ramificar a partir de un subconjunto del conjunto de atributos de partida. Este subconjunto — salvo casualidad — es distinto para cada nodo y aleatorio en cuanto a su composición. El tamaño de los subconjuntos es fijo y se especifica como parámetro. El valor que utiliza Breiman es $\lfloor \log_2 n + 1 \rfloor$, donde n es el número de atributos del conjunto de entrenamiento de partida.

2. *Forest-RC* (i.e., Linear Combination of Inputs). Consiste en construir nuevas características en cada uno de los árboles. Estas nuevas características se obtienen como combinación lineal de L atributos del conjunto original. Los coeficientes de la combinación lineal son aleatorios, y se obtienen a partir de una distribución uniforme en el intervalo $[-1, 1]$.

En ambas implementaciones el entrenamiento sigue las mismas pautas que en Bagging (remuestreo aleatorio), así como la predicción (votación y predicción de la clase mayoritaria; si bien en la implementación utilizada [117] la predicción se obtiene a partir de la combinación de las estimaciones de las probabilidades de cada árbol).

En esta tesis se ha utilizado Forest-RI, y el número de atributos considerados por cada nodo ha sido el mismo que en [11]: $\lfloor \log_2 n + 1 \rfloor$.

Los árboles de decisión utilizados originalmente por Breiman en [11] para implementar Random Forests son una variante de los árboles CART (*Classification and Regression Trees*) [13, 119]. Esta variante se diferencia de los CART originales en que se ha añadido la limitación ya descrita del número de atributos aleatorios por los que ramificar en cada nodo, y además se ha eliminado la poda.

Sin embargo, los Random Forests de esta tesis son los de la implementación de WEKA [117], que utilizan como clasificadores base una variante de REPTree (ver Seccion 2.1.2). La variante de REPTree utilizada, mantiene los ingredientes fundamentales de los CARTs utilizados por Breiman, pues se diferencian de los REPTRees originales en que se ha añadido la limitación ya descrita del número de atributos aleatorios por los que ramificar en cada nodo, y además se ha eliminado la poda.

Las diferencias entre los CART que utilizaba Breiman para Random Forests, y los que utiliza WEKA, se pueden resumir en:

1. Los CART dan lugar siempre a ramificaciones binarias, pero los REPTrees cuando ramifican por un atributo nominal, crean tantas ramas como valores tenga dicho atributo.

2. Los CART utilizan como criterio de selección del atributo por el que ramificar el *Índice de Gini*, pero los REPTRees, como se recordará de la Seccion 2.1.2, utilizan la ganancia de información. El índice de Gini mide el error de clasificación que se comete en el nodo t si éste, aun no siendo hoja, asignase la clase de la instancia de forma aleatoria siguiendo la distribución de elementos de cada clase que hay en t. El índice de Gini para un nodo t puede calcularse como:

$$i(t) = \sum_{i \neq j}^{c} P_i P_j = 1 - \sum_{j}^{c} P_j^2 \tag{2.19}$$

Donde c es el número de clases y P_i es la estimación de la probabilidad de la clase i para las instancias que alcanzan el nodo.

Por tanto, índice de Gini y ganancia de información son ambas, medidas basadas en la impureza de la rama.

Estas diferencias no parecen muy sustanciales, en tanto los bosques en esta tesis han tenido siempre un tamaño relativamente grande (i.e., 50 iteraciones o árboles).

Cada uno de los árboles de un Random Forests, al tener limitados los atributos por los que pueden ramificarse, como clasificadores en solitario, son en principio clasificadores menos precisos que un árbol que no tenga esta limitación. Sin embargo, como miembro de un Random Forests cada árbol:

- Mantiene la diversidad que le viene dada por el remuestreo con reemplazamiento.
- Aumenta esa diversidad en cuanto que los atributos a considerar por cada nodo son distintos cada vez, según un criterio aleatorio.
- Al no estar podado provee la inestabilidad que según [7] requiere un buen clasificador base para Bagging.

Además es un algoritmo más rápido que Bagging, también paralelizable, y más robusto frente al ruido.

2.2.3. Random Subspaces

El método de los Random Subspaces [55] construye cada clasificador base utilizando sólo un subconjunto aleatorio del total de las características en el conjunto de entrenamiento original. La predicción se obtiene promediando las probabilidades de pertenencia a cada clase estimadas por los clasificadores base (i.e., árboles en [55]).

El número de atributos que cada clasificador base selecciona aleatoriamente es siempre el mismo y viene dado como parámetro del método. Ho sugiere en [55] que para obtener buenos resultados ese número debe de ser aproximadamente el 50 % del número de atributos del conjunto de entrenamiento de partida. Sin embargo, en esta tesis se han considerado dos configuraciones posibles en las que este parámetro toma respectivamente los valores del 50 % y 75 % de los atributos de partida. La razón es que con el 50 % se pierde a veces demasiada información. En [55] ya se indica que el método funciona mejor cuando hay un mayor número de atributos, estos presentan entre si cierta redundancia, y además el número de instancias de entrenamiento no es demasiado pequeño. Estas condiciones favorables no siempre se han dado en los numerosos conjuntos

de datos utilizados en las validaciones experimentales de esta tesis, por lo que utilizar el 75 % de los atributos en muchas ocasiones ha sido mejor opción que tomar el 50 %.

Random Subspaces es un método que presenta cierta robustez al ruido en los datos de entrenamiento. De hecho Ho cuando lo presenta para la construcción de multiclasificadores de árboles en [55], argumenta como ventaja principal que el método evita el dilema entre el sobre-entrenamiento y alcanzar la máxima tasa de acierto.

Aunque en [55] el método se presenta para clasificadores base árboles, su utilización con otros clasificadores base es directa.

2.2.4. Boosting

Se define Boosting en [42] como el problema de producir un clasificador muy preciso a partir de la combinación de otros más simples y moderadamente imprecisos. Dentro de los algoritmos de Boosting se encuentran los métodos de Boosting *adaptativos*, que son a los que en esta tesis, abusando del lenguaje, se les denominará genéricamente como métodos de Boosting, pues los métodos de Boosting adaptativos han terminado por ser los más populares. Los métodos de Boosting adaptativos se basan en dos ideas:

1. En entrenamiento, construir iterativamente los clasificadores base de manera que el clasificador base actual dé más importancia a las instancias del conjunto de entrenamiento mal clasificadas por el clasificador base de la iteración anterior.

2. En clasificación, hacer la predicción en base a un esquema de votación ponderado, de forma que aquellos clasificadores base con un mayor acierto sobre el conjunto de entrenamiento, tengan un mayor peso en la votación.

Los dos algoritmos adaptativos de Boosting utilizados en esta tesis como métodos de referencia son *AdaBoostM1* [41, 119] y *MultiBoosting* [116].

AdaBoost es quizás la variante más utilizada de Boosting. El nombre del algoritmo *AdaBoost* viene de *ADAptative BOOSTing*. La figura 2.1 presenta una versión simplificada del algoritmo [119], que es útil en cuanto permite razonar sobre su funcionamiento. Se trata de una versión para conjuntos de datos con sólo dos clases:

Como se puede ver, AdaBoost genera una hipótesis o clasificador base $h_t(x)$ en cada iteración t, para finalmente combinar sus predicciones linealmente a través de unos pesos α_t. De manera que la salida de AdaBoost es:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{2.20}$$

Por tanto, el problema es doble:

Figura 2.1: El algoritmo de entrenamiento de AdaBoost según [119].

AdaBoost
input : D: Conjunto de m ejemplos $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ con $y_i \in \{+1, -1\}$
L: Algortimo del clasificador base
T: Entero correspondiente al número de iteraciones
output: Hipótesis final: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

variables:
D_t: Vector $[1..m]$ de distribuciones de pesos de la iteración t
h: Vector de clasificadores base

begin
Inicializar $D_1(i) = 1/m$ para todo i ;
for $t = 1, ..., T$ **do**
`/* Obtener` h_t `con el algoritmo` L `y la distribución de pesos` D_t `*/`
$h_t \leftarrow L(D, D_t)$;
`/* Calcular el error de` h_t `*/`
$\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$;
if $\epsilon_t > 1/2$ **then**
$T \leftarrow t - 1$;
Salir del bucle ;
end
$\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{\epsilon_t}{1-\epsilon_t}\right)$;
`/* Actualizar la distribución` D_t `*/`
$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{Si } h_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t} & \text{Si } h_t(\mathbf{x}_i) \neq y_i \end{cases}$
Siendo Z_t es un factor de normalización (con el fin de que D_{t+1} sea un distribución);
end
end

---

1. Por un lado hay que generar las hipótesis h_t de forma que vayan fijándose en las instancias más difíciles de clasificar, cambiando adecuadamente la distribución de pesos de las instancias.

2. Por otro lado, hay que determinar los coeficientes α_t para ajustar el peso de cada hipótesis base en la votación final.

El objetivo final de estas dos partes del problema es minimizar el error del multiclasificador resultante. La forma que tiene Boosting de minimizar ese error es minimizar a su vez la pérdida exponencial. Se define dicha pérdida exponencial para un clasificador cualquiera $h(x)$ y una distribución del conjunto de datos $\mathcal{D}$ como:

$$\text{loss}_{\text{exp}}(h) = \mathbb{E}_{x\sim\mathcal{D},y}[e^{-yh(x)}] \tag{2.21}$$

Donde $\mathbb{E}$ es la esperanza matemática, y el producto $yh(x)$ es el *margen de clasificación* de la hipótesis, pudiendo valer -1 o $+1$ en función de si $h(x)$ respectivamente falla o acierta la predicción.

Sea $\epsilon = \mathbb{E}_{x\sim\mathcal{D}}[y \neq h(x)]$, el error de entrenamiento del clasificador base $h(x)$. Entonces, el peso óptimo de cada $h(t)$ que minimiza la pérdida exponencial de la predicción que haga el multiclasificador, es igual a α calculado como:

$$\alpha = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right) \tag{2.22}$$

Asimismo, la distribución de pesos de las instancias D_{t+1}, que minimiza esa pérdida exponencial es:

$$D_{t+1}(i) = D_t(i)e^{-\alpha yh(\mathbf{x}_i)}\ \forall i \tag{2.23}$$

Una demostración de ambos resultados puede encontrarse en [119, 43]. Para generalizar el algoritmo al caso multiclase habría que hacer que la predicción final fuese la clase más votada, según la votación ponderada por los α_t.

$$H(\mathbf{x}) = \arg\max_{y\in Y} \sum_{t:h_t(\mathbf{x})=y} \alpha_t \tag{2.24}$$

El algoritmo de AdaBoost de la figura 2.1 no es quizás el más célebre. Posiblemente el algoritmo AdaBoost.M1 de Freund y Schapire en [41] sea el más popular. AdaBoost.M1 se muestra en la figura 2.2.

Figura 2.2: El algoritmo de entrenamiento de AdaBoost.M1 según [41].

AdaBoost.M1

input : Conjunto de m ejemplos $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}, y_i \in Y$
$WeakLearn$: Algoritmo del clasificador base
T: Entero correspondiente al número de iteraciones

output: Hipótesis final: $H(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t:h_t(\mathbf{x})=y} \log \frac{1}{\beta_t}$

variables:
D: Vector $[1..m]$ de Pesos

begin
Inicializar $D_1(i) = 1/m$ para todo i ;
for $t = 1, 2, ..., T$ **do**
Entrenar $WeakLearn$ utilizando la distribución de pesos D_t ;
Obtener de dicho entrenamiento la hipótesis $h_t : X \to Y$;
Calcular el error de $h_t : \epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$;
if $\epsilon_t > 1/2$ **then**
$T \leftarrow t - 1$;
Salir del bucle ;
end
$\beta_t \leftarrow \epsilon_t / (1 - \epsilon_t)$;
Actualizar la distribución
$D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{Si } h_t(\mathbf{x}_i) = y_i \\ 1 & \text{en cualquier otro caso} \end{cases}$
siendo Z_t es un factor de normalización (con el fin de que D_{t+1} sea un distribución);
end
end

Esta última versión, en lugar de trabajar con α_t, trabaja con β_t, pero es un simple cambio de variable, dado que es fácil ver que $\alpha_t = 1/2 \log(1/\beta_t)$, y que las operaciones donde ahora aparece β_t son consistentes con las que antes se hacían con α_t.

Existen dos implementaciones de AdaBoost atendiendo a cómo se utilizan las distribuciones de pesos para entrenar cada clasificador base: con *Remuestreo* y con *Repesado*.

1. La variante con Repesado sólo se puede utilizar cuando el clasificador base es sensible a una distribución de pesos en las instancias (como es el caso de las implementaciones de C4.5 y las SVM utilizadas). En la variante con repesado inicialmente se parte de una distribución uniforme de pesos. Tras construir un clasificador base, éste se valida contra el conjunto de entrenamiento, cambiando los pesos de las instancias (e.g., según (2.23)), para que incrementen su peso las mal clasificadas, y lo decrementen las clasificadas correctamente.

2. En la variante con Remuestreo, como indica su nombre, el clasificador base de la iteracción t es entrenado con un conjunto de instancias seleccionadas aleatoriamente del conjunto de entrenamiento original según la distribución D_t. El número de instancias seleccionadas suele ser el mismo que el número de instancias del conjunto de datos original, por lo que algunas instancias podrán ser seleccionadas varias veces. Por lo tanto, esta variante no exige que los clasificadores base puedan manejar pesos.

En cualquiera de ambas aproximaciones, en la siguiente iteración, el nuevo clasificador base se centrará en clasificar correctamente las instancias más difíciles de clasificar por los clasificadores base anteriores.

Al igual que Bagging, Boosting necesita de clasificadores base inestables, para que se adapten a la nueva distribución del conjunto de entrenamiento, pero la diferencia es que Boosting toma el control de cómo se va a producir ese cambio en esa distribución, de manera que se centra en las instancias más difíciles de clasificar.

En [41] se compara experimentalmente AdaBoost y Bagging, con el fin de conocer si la potencia de Boosting emana de cómo cambia la distribución, o de combinar múltiples clasificadores cuya tasa de acierto sea al menos superior a la que se obtendría de un predictor aleatorio. La conclusión es que Boosting mejora sustancialmente a Bagging cuando los clasificadores base son muy simples o débiles, pero que no mejora tanto a Bagging cuando el clasificador base es más preciso, como en el caso de C4.5.

Otra diferencia resaltable entre Bagging y Boosting es el tipo de error que generan. Sea $\mathcal{T}$ un conjunto de distribuciones de un determinado tamaño fijo del conjunto de entrenamiento para un problema de clasificación dado. Si para dicho algoritmo de clasificación se obtuvieran los predictores correspondientes a cada elemento de $\mathcal{T}$, se podría calcular para cada instancia $\mathbf{x}$ con qué probabilidad es predicha cada clase por los clasificadores obtenidos a partir de ese algoritmo.

Se denomina *tendencia central* [116] de la instancia $\mathbf{x}$, para un determinado algoritmo y un determinado $\mathcal{T}$, a la clase mayoritaria predicha por todos esos predictores. En el fondo la tendencia central de $\mathbf{x}$ no es otra cosa que la predicción hecha por Bagging al combinar esos predictores entrenados por separado con cada miembro de $\mathcal{T}$. Esta tendencia central puede tomarse como base para descomponer el error de un clasificador en dos componentes que se conocen como *bias* y *varianza* [3] [8, 62, 64, 45, 116]:

1. El error debido al bias es el que se produce cuando el clasificador comete un error por predecir la tendencia central. Es un tipo de error que se debe a las propias limitaciones del clasificador. Por ejemplo, un clasificador lineal requiere regiones linealmente separables, si un grupo reducido de instancias está en el interior de una región en la que hay abundantes instancias de la clase contraria, las instancias de dicho grupo estarán normalmente en el lado incorrecto de cualquiera de los hiperplanos generados a partir de diferentes muestras del mismo conjunto de datos para ese mismo clasificador lineal. Por tanto, la tendencia central de ese clasificador será predecirla como perteneciente a la clase equivocada, y por tanto se genera un error debido a la componente bias. En este caso, la limitación del clasificador (i.e., requerir regiones linealmente separables) es la que está detrás del error.

2. Por el contrario, la componente varianza del error es la que se produce cuando el clasificador comete un error por predecir cualquier otra clase que no sea la tendencia central. Este error aumenta por tanto, si para una misma entrada $\mathbf{x}$ el mismo algoritmo de clasificación — entrenado cada vez con un conjunto de entrenamiento distinto — es capaz de generar distintos clasificadores, habiendo muchos capaces de hacer predicciones distintas a la tendencia central. Debido a que el único cambio que se ha introducido para que tengan lugar estas diferencias es el conjunto de entrenamiento, es claro que este error se manifiesta cuanto mayor es la sensibilidad del algoritmo a dicho cambio, y por tanto es una componente del error que aumenta cuando el algoritmo tiene en cuenta en exceso a las instancias que representan casos que se dan con poca frecuencia (i.e., *outliers*). Este error, es el habitual cuando el clasificador sufre un sobreentrenamiento.

Por ejemplo, un árbol de decisión sin podar aumentará la componente varianza del error respecto al caso podado. Por el contrario, una poda muy agresiva dará lugar a una alta componente de bias en el error.

Existen estudios [8, 62, 106, 116] que sugieren que AdaBoost reduce tanto la componente de bias como la de varianza, mientras que Bagging sólamente tiende a reducir el término varianza. Esta conclusión parece lógica, en tanto

[3] Existe otra componente que es el error *irreductible* que por ejemplo ocurre cuando varias instancias $\mathbf{x}$ con una descripción idéntica, tienen asignadas clases distintas en el mismo conjunto de datos. El error irreductible, además de ser difícil de determinar, afecta por igual a cualquier clasificador, luego cara a comparar Bagging con Boosting, no se ha tenido en cuenta, como bien se argumenta en [116].

«Bagging puede verse como un método que clasifica a partir de una estimación de la tendencia central» [116], mientras que el repesado de AdaBoost puede incrementar la flexibilidad de clasificadores débiles, los cuales son estables (i.e., con alta componente bias) [43].

Precisamente la otra variante de Boosting utilizada en esta tesis, MultiBoosting [116], lo que hace es intentar aunar los potenciales que tienen Bagging y Boosting por separado de reducir cada una de las dos componentes del error. Aunque Bagging y Boosting reducen el error al aumentar el número de clasificadores base, el impacto relativo de la adición de nuevos clasificadores base es cada vez menor. Por ello, la idea central de MultiBoosting es que cada cierto número de iteraciones los pesos vuelvan a reinicializarse de forma aleatoria.

La idea que sirve de base a MultiBoosting es *Wagging* [5], que consiste en crear un multiclasificador a partir de clasificadores base entrenados con pesos aleatorios. En Wagging se utiliza la distribución continua de Poisson para dar pesos aleatorios a las instancias. Esto se debe a que el remuestreo de Bagging puede ser modelado como una distribución «discreta» de Poisson, ya que la probabilidad de una instancia para ser seleccionada en una muestra de entrenamiento es muy pequeña, pero tiene muchas oportunidades para que ocurra (ley de los pequeños números). Por ello, la distribución «continua» de Poisson es la más adecuada para emular la política de Bagging a través del repesado. Es por ello, que en MultiBoosting también se utiliza esta distribución para reinicializar los pesos de las instancias aleatoriamente.

El algoritmo de MultiBoosting [116] puede verse en la figura 2.3. El esquema de voto ponderado y la actualización de pesos teniendo en cuenta las instancias mal clasificadas es la misma que hacía AdaBoost en la figura 2.2.

MultiBoosting divide el conjunto de clasificadores base h_t en varios subcomités. En la figura 2.3, cada subcomité puede tener un tamaño variable según los valores que tomara el vector I. En la implementación utilizada en esta tesis, sin embargo, todos los subcomités son siempre del mismo tamaño. El número de subcomités será, por tanto, un parámetro del algoritmo. Si este número fuese uno, los pesos no se reinicializarán con lo que el resultado será equivalente a hacer AdaBoost.M1. Por el contrario, si el número de subcomités coincide con el de iteraciones, el resultado será equivalente a hacer Wagging.

Como cabe esperar, MultiBoosting puede encontrarse igualmente en versión con repesado y versión con remuestreo.

2.2.5. Cascading

Cascade Generalization (también conocida como *Cascading*) [47] es una arquitectura con la que combinar clasificadores (ver figura 2.4), que normalmente presenta dos niveles. El nivel 1 se entrena con el conjunto de datos original, mientras que el nivel 2 se entrena con un conjunto de datos aumentado, el cual contiene las características del conjunto de datos original junto con la salida del clasificador del nivel 1. La salida del clasificador del nivel 1 es un vector conteniendo la distribución de probabilidad condicional $(p_1, ..., p_c)$, donde c es el número de clases del conjunto de datos original, y p_i es la estimación de pro-

Figura 2.3: El algoritmo de entrenamiento de MultiBoosting según [116].

MultiBoosting
input : S:Conjunto de m ejemplos $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}, y_i \in Y$
$BaseLearn$: Algortimo del clasificador base
T: Entero correspondiente al número de iteraciones
I: Vector de enteros, especificando la iteración $i \geq 1$ en la que debe de terminar cada subcomité
output: Clasificador final: $C^*(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t:C_t(\mathbf{x})=y} \log 1/\beta_t$

variables:
w: Array $[1..m]$ con el peso actual de cada instancia

begin
  $S' \leftarrow S$, todos los pesos de S' toman el valor 1 ;
  $k \leftarrow 1$;
  for $t = 1, ..., T$ **do**
    if $I_k = t$ **then**
      Resetear S' con pesos aleatorios usando la distribución continua de Poisson;
      Estandarizar S' para que los pesos sumen m;
    end
    `/* Entrenar C. Base */`
1   $C_t \leftarrow BaseLearn(S')$;
    $\epsilon_t = \sum_{\mathbf{x}_j \in S': C_t(\mathbf{x}_j) \neq y_j} w(\mathbf{x}_j)/m$;
    if $\epsilon_t > 1/2$ **then**
      Resetear S' con pesos aleatorios;
      Estandarizar S' para que los pesos sumen m;
      Incrementar k ;
      Ir a 1 ;
    else if $\epsilon_t = 0$ **then**
      $\beta_t \leftarrow 10^{-10}$;
      Resetear S' con pesos aleatorios;
      Estandarizar S' para que los pesos sumen m;
      Incrementar k ;
    else
      $\beta_t \leftarrow \epsilon_t/(1 - \epsilon_t)$;
      foreach $\mathbf{x}_j \in S'$ **do**
        if $C_t(\mathbf{x}_j) \neq y_j$ **then** $w(\mathbf{x}_j) \leftarrow \frac{w(\mathbf{x}_j)}{2\epsilon}$ **else** $w(\mathbf{x}_j) \leftarrow \frac{w(\mathbf{x}_j)}{2(1-\epsilon)}$
        if $w(\mathbf{x}_j) < 10^{-8}$ **then** $w(\mathbf{x}_j) \leftarrow 10^{-8}$
      end
    end
  end
end

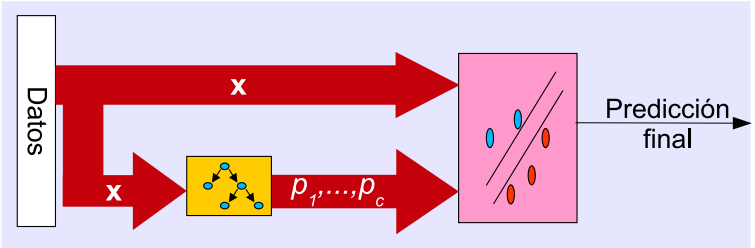


Figura 2.4: Funcionamiento de Cascading. El clasificador del nivel 1 se ha representado mediante un árbol, el de nivel 2 mediante un SVM.

babilidad calculada por el clasificador del nivel 1 de que la instancia pertenezca a la clase i. Cascading es una aproximación que se puede extender a más de dos niveles.

El entrenamiento de un clasificador A con la salida de otro B hace que A se vea influenciado notablemente por B, derivando en un esquema global sobreentrenado. Sin embargo, en Cascading se reduce este problema porque:

1. En cada nivel se utiliza un clasificador de naturaleza distinta al del otro.
2. Porque el clasificador del nivel 2 no se entrena únicamente con la salida del clasificador de nivel 1, sino que además tiene en cuenta las características originales.

Cascading persigue combinar dos clasificadores, uno con la componente bias del error con un valor bajo, y el otro con la componente varianza con valor bajo también, para así conseguir uno nuevo que tenga valores menores en ambas medidas. En [47] se prefiere que el clasificador con poca varianza esté en el nivel 1, mientras que el que tenga bias bajo esté en el nivel 2 porque «*seleccionando métodos con bajo bias en el nivel superior, es posible ajustarse a superficies de decisión más complejas, teniendo en cuenta las superficies 'estables' dibujadas por los clasificadores del nivel inferior*». La validación experimental en [47] sobre 26 conjuntos de datos del repositorio UCI da soporte a esta conclusión.

2.2.6. Stacking

Stacked Generalization, también conocida como *Stacking* [118] es otro multiclasificador que, como en el caso de Cascading, presenta cierto nivel de jerarquización entre sus clasificadores miembros. El nivel inferior, el que toma como entrada los atributos del conjunto original, es el que en [118] se denomina nivel 0, y el superior, el que toma como entrada las predicciones del nivel 0, lo denomina nivel 1. La aproximación podría también extenderse a cualquier número de niveles, aunque como en el caso de Cascading, lo usual es limitarlo a dos. En el nivel 0 normalmente hay varios clasificadores, cada uno de los cuales suele entrenarse

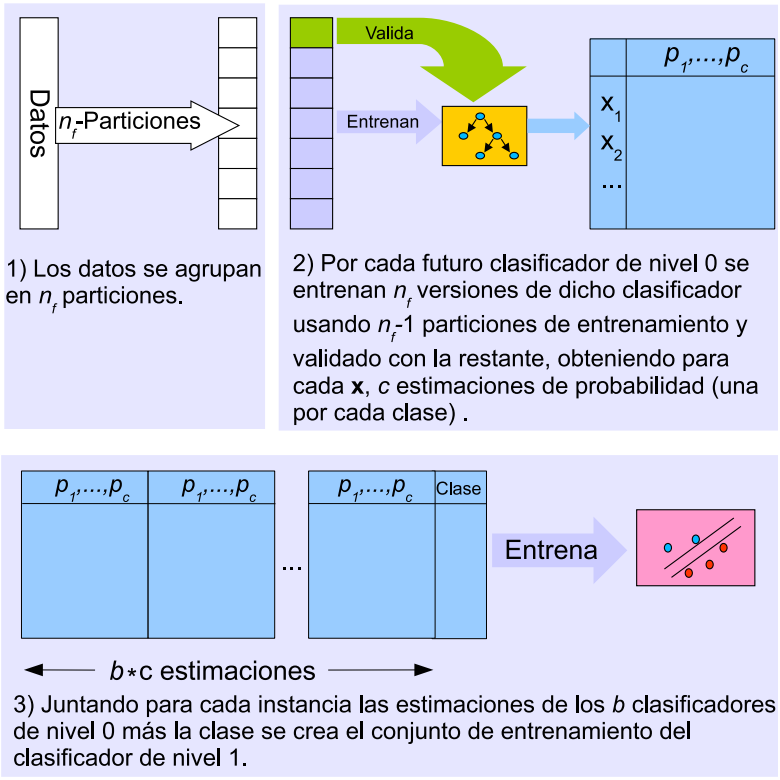


Figura 2.5: Funcionamiento de Stacking (I). El clasificador del nivel 0 se ha representado mediante un árbol, si bien podría haber b tipos de clasificadores distintos en dicho nivel (e.g. uno fuese un árbol, otro un k-NN, otro una red neuronal, etc.). El clasificador de nivel 1 se representa mediante un SVM.

mediante un algoritmo de aprendizaje distinto. Todos los clasificadores del nivel 0 se entrenan con conjuntos de entrenamiento que tienen el mismo espacio de características, que es el conjunto de entrenamiento de partida, mientras que un único clasificador de nivel 1 toma como entradas cada una de las salidas de los clasificadores del nivel inferior.

Stacking trabaja con los siguientes parámetros:

1. b: El número de clasificadores de nivel 0 que se desea obtener al final del algoritmo (los llamaremos clasificadores base *finales*).

2. n_f: Número de particiones disjuntas (o *folds*).

El algoritmo de entrenamiento de Stacking se muestra en las figuras 2.5 y 2.6. En una primera fase procede a entrenar $n_f \times b$ clasificadores base, que

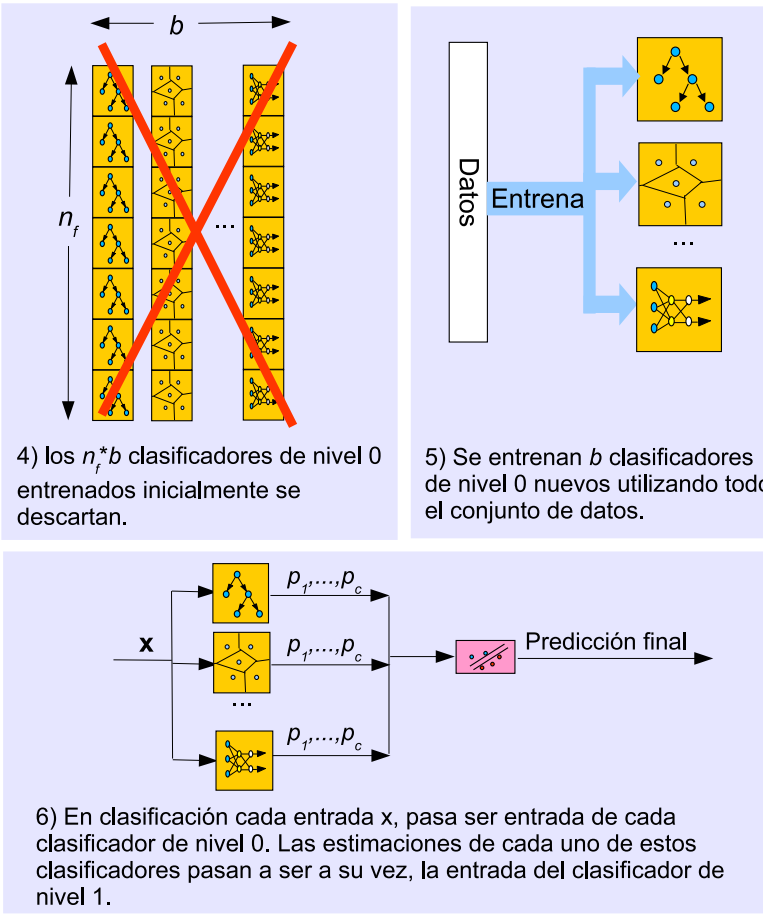


Figura 2.6: Funcionamiento de Stacking (II). Los b clasificadores del nivel 0 se han representado mediante tres: un árbol, un k-NN y una red neuronal, para remarcar que en general son distintos. El clasificador de nivel 1 se representa mediante un SVM.

llamaremos clasificadores base *iniciales*. Por tanto, se entrena un grupo de n_f clasificadores base iniciales por cada clasificador base que se desea obtener. Todos los clasificadores base iniciales pertenecientes al mismo grupo son del mismo tipo que uno de los clasificadores base finales. Cada clasificador base inicial se entrena con $n_f - 1$ particiones, validándose con la partición sobrante (validación cruzada). Por tanto, la partición de validación es diferente para cada uno de los n_f clasificadores de un grupo. Además, la misma instancia será usada para validar exactamente b veces (una vez por cada clasificador base final que se vaya a obtener).

Una vez superada esta fase, el clasificador del nivel 1 se entrena con un conjunto de datos compuesto por $c \times b$ atributos más la clase, donde c es el número de clases y b el número de clasificadores base finales. Para obtener este conjunto de datos, cada instancia original es tomada como entrada por los b clasificadores base finales que no la usaron para entrenarse. Cada uno devolverá un vector de probabilidad de dimensión c. Es por eso que el conjunto de entrenamiento que así se obtiene es de dimensión $c \times b$.

Finalmente, los $n_f \times b$ clasificadores base iniciales son descartados, calculándose los b clasificadores base finales a partir del conjunto de entrenamiento original por completo.

La influencia del clasificador del nivel 0 sobre el del nivel 1, y el correspondiente sobreentrenamiento se evita en esta ocasión porque:

1. Como en el caso de Cascading, también los clasificadores de ambos niveles son distintos.

2. La validación cruzada hace que el conjunto de entrenamiento de los dos niveles difiera, ya que los clasificadores de nivel 0 finales se entrenan con todo el conjunto, mientras que las particiones con las que se entrenaron los clasificadores base iniciales, sirven de punto de partida al conjunto de entrenamiento del nivel 1.

2.2.7. Grading

Grading [111] también es un multiclasificador de dos niveles que utiliza, como Stacking, n_f particiones. En [111] el nivel que toma directamente las entradas del conjunto de datos original se llama nivel base o nivel 0, mientras que el que procesa la salida del nivel base lo denomina nivel meta o nivel 1.

El algoritmo de entrenamiento de Grading se muestra en las figuras 2.7 y 2.8. Como en Stacking, Grading calcula previamente un conjunto de $n_f \times b$ clasificadores base *iniciales* con los que se hace validación cruzada. En el caso de Grading a cada clasificador base inicial se le añade un atributo binario que expresa si la predicción es correcta o no (*Graded Prediction*) para así formar el conjunto de entrenamiento del nivel 1 o meta. Como en el caso de Stacking, cada instancia es utilizada para entrenar $n_f - 1$ clasificadores base iniciales, pero sólo es validada por uno. Por tanto, para cada grupo de n_f clasificadores toda instancia es validada una sola vez, con lo que se obtiene un solo valor para el atributo binario *graded prediction*.

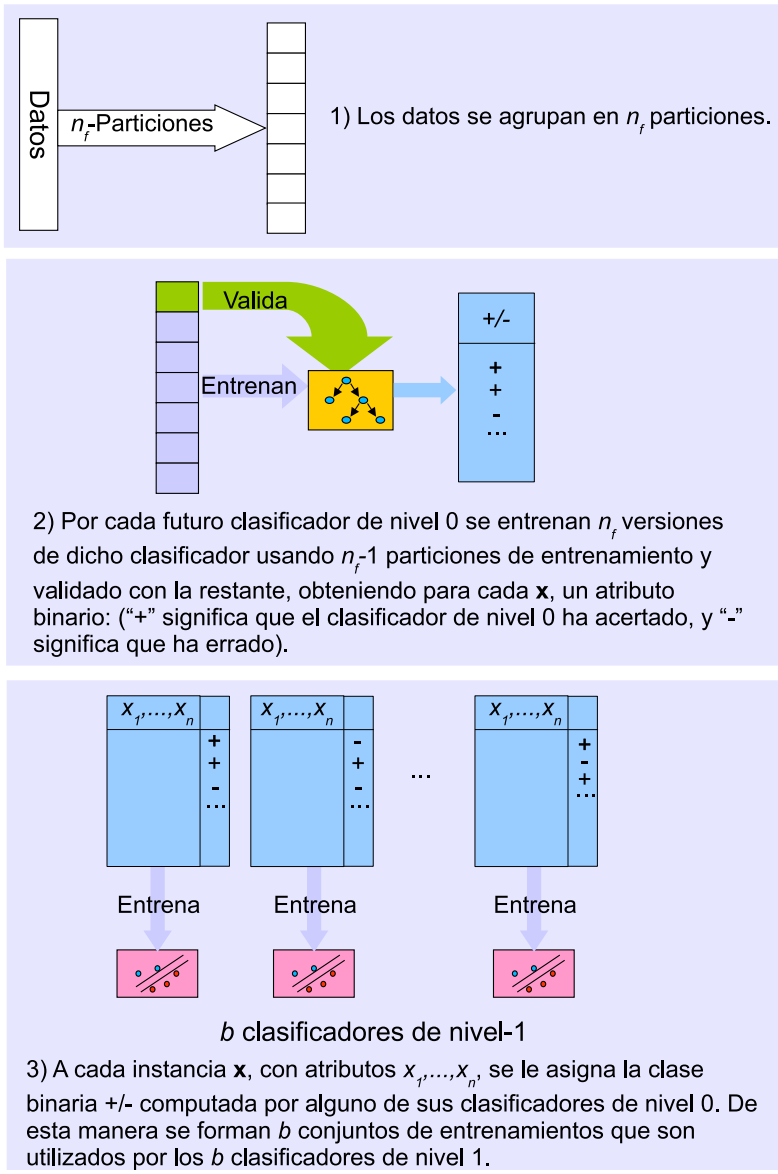


Figura 2.7: Funcionamiento de Grading (I). El clasificador del nivel 0 se ha representado mediante un árbol, si bien podría haber b tipos de clasificadores distintos en dicho nivel (e.g. uno fuese un árbol, otro un k-NN, y otro una red neuronal, etc.). Cada clasificador de nivel 1 se representa mediante un SVM. Los clasificadores de nivel 1 han de ser todos del mismo tipo.

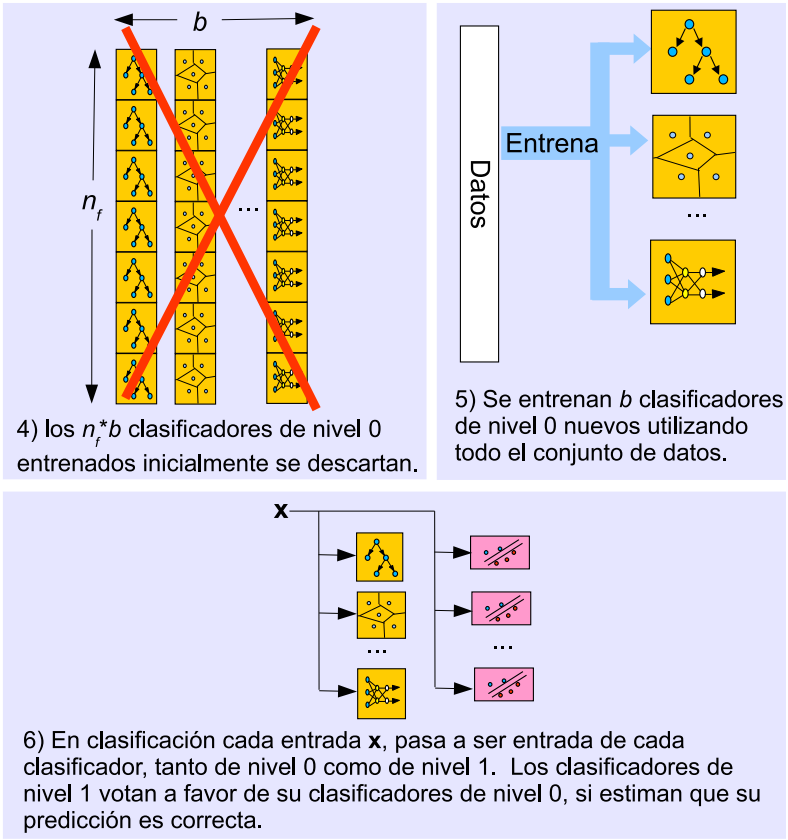


Figura 2.8: Funcionamiento de Grading (II). Los b clasificadores del nivel 0 se han representado mediante tres: un árbol, un k-NN y una red neuronal, para remarcar que en general son distintos. Cada clasificador de nivel 1 se representa mediante un SVM. Los clasificadores de nivel 1 han de ser todos del mismo tipo.

Posteriormente, se entrenan b clasificadores del nivel meta (todos del mismo tipo), cada uno de los cuales toma como entrada las características originales junto con el atributo binario que se obtienen de sus n_f clasificadores base iniciales. Dicho atributo binario es tomado como clase a predecir por los clasificadores del nivel meta.

La parte final del algoritmo de entrenamiento es muy similar a Stacking: Los $n_f \times b$ clasificadores base iniciales son también descartados, y se entrenan b nuevos clasificadores base finales utilizando todo el conjunto de datos original.

Grading predice mediante votación de sus clasificadores del nivel meta siguiendo el siguiente esquema:

1. Primero cada clasificador base hace su predicción para la entrada a evaluar.
2. Seguidamente los clasificadores de nivel superior deciden o predicen qué clasificadores de nivel inferior están en lo cierto.
3. Si hubiese un conflicto porque varios clasificadores del nivel base estén prediciendo clases distintas y sus respectivos clasificadores meta estén respaldándoles, el conflicto se resuelve por votación haciendo uso de la confianza en la predicción de los clasificadores base.
4. En el improbable caso de que ningún clasificador de nivel meta respalde la predicción de algún clasificador base, se hace una votación entre todos los clasificadores base, tomando como confianza de cada uno $1 - confianza$, de manera que se da más crédito a los más indecisos.
5. Si cualquiera de las votaciones diese lugar a empate, la predicción final será la clase más frecuente en el conjunto de datos.

La influencia del clasificador del nivel 0 sobre el del nivel 1, y el correspondiente sobreentrenamiento se evita en esta ocasión porque:

1. Nuevamente la naturaleza de los clasificadores en cada nivel es distinta.
2. El conjunto de entrenamiento del nivel meta es distinto al del base, pues en el nivel meta se ha cambiado la clase del conjunto original por la llamada *graded prediction*.
3. El nivel meta se entrena con las salidas producidas por validación cruzada, como ocurría en Stacking.

2.2.8. Otros métodos multiclasificadores

Los métodos multiclasificadores que hasta ahora se han incluido en es esta sección son los utilizados como métodos de referencia en esta tesis. Sin embargo, existen muchos más, algunos de los cuales se exponen brevemente a continuación. Los métodos que se presentan se han agrupado atendiendo a los siguientes bloques temáticos:

1. Métodos basados en Boosting: que incluye versiones de Boosting, métodos similares y métodos que en alguna fase lo utilizan.

2. Técnicas para el incremento de la diversidad de los clasificadores base.

3. Combinaciones multinivel y metalearning, en las que se incluyen principalmente formas de combinación de clasificadores en las que la salida de unos sirve de entradas a otros.

La gran variedad de métodos que existen hace que algunos de los métodos que se van a presentar podrían estar en varios de estos grupos simultáneamente.

Métodos basados en Boosting

Boosting, al ser uno de los métodos más estudiado en los últimos tiempos, es quizás es el método que más variantes presenta.

Real AdaBoost [43] es una generalización de AdaBoost para el caso en el que el clasificador base sea capaz de hacer predicciones en términos de números reales, cuyo significado es la probabilidad de que el clasificador pertenezca a una de las dos clases, ya que trabaja sólo con problemas binarios. La contribución de cada clasificador base al multiclasificador en Real AdaBoost es la transformación logística de la estimación de la probabilidad dividida por dos. En [43] Real AdaBoost con datos sintéticos obtiene mejores resultados que la versión discreta [4] con árboles de una sola decisión, pero los resultados de Real AdaBooost pueden ser superados por la versión discreta a medida que aumentan el número de iteraciones y la complejidad de los árboles utilizados como clasificadores base. Por ejemplo, con árboles de 100 nodos, la versión discreta parece funcionar mejor que la real a partir de 200 iteraciones.

En [43] AdaBoost se analiza desde una perspectiva estadística, viéndolo como un método de ajuste de los coeficientes de un modelo aditivo (i.e. la predicción final de AdaBoost, que no es más que una combinación lineal de las predicciones de los clasificadores base). Pero ese modelo aditivo se puede ajustar mediante otras funciones más simples y potentes. Así, *LogitBoost* [43] es una variante de AdaBoost que se basa en que éste puede aproximarse mediante una regresión logística aditiva que optimiza la pérdida exponencial. LogitBoost halla este modelo logístico directamente.

Gradient Boosting [44] construye un multiclasificador ajustando el clasificador base a los pseudoresiduos que se obtienen de aplicar mínimos cuadrados en cada iteración. Los pseudoresiduos son el gradiente de la función de pérdida que está siendo minimizada. Además los clasificadores base son entrenados con un conjunto de entrenamiento de tamaño menor que el original, el cual se obtiene por remuestreo aleatorio sin remplazo, lo cual ayuda a que el multiclasificador resultante sea más robusto y rápido de entrenar.

Una posible limitación de AdaBoost, es que combinan las predicciones de los clasificadores base de forma lineal. En [90] se presenta *Quadratic Boosting*.

[4] La versión originaria de AdaBoost [41] incialmente sólo trabaja con predicciones discretas (binarias) de los clasificadores base, y por eso Friedman en [43] la denomina *Discrete AdaBoost*.

Esta variante combina las predicciones de los clasificadores base a través de una función cuadrática, a diferencia de la función lineal que utiliza AdaBoost. Para optimizar los coeficientes de la función cuadrática el método recurre a optimizar los coeficientes de una función lineal en el que las variables se obtienen de multiplicar las salidas de un par de clasificadores base. En cada iteración de Boosting se calculan varios pares de clasificadores base a fin de obtener el mejor par posible. Para obtener los clasificadores base combina el repesado tradicional de Boosting con la asignación aleatoria de la clases a las instancias de entrenamiento.

AdaBoost.M1 ya permite trabajar con problemas multiclase, aunque necesita que los clasificadores base tengan tasas de acierto superiores al 50 %, lo cual es excesivo para clasificadores poco precisos en conjuntos de datos multiclase. Por ello, ésta es una de las líneas de mejora de AdaBoost. Existen varias aproximaciones a este fin:

- *AdaBoost.M2* [41] mantiene un peso por cada instancia y clase. Los clasificadores base de AdaBoost.M2 han de devolver un vector de probabilidades, con dimensión igual al número de clases, en el que cada componente indica la estimación de probabilidad de que la instancia pertenezca a una de las clases. AdaBoost.M2 utiliza el concepto de *pseudopérdida* que mide la bondad del clasificador base. En el cálculo de la pseudopérdida intervienen los pesos por instancia y clase de la iteración anterior, y el vector de estimaciones de probabilidad que devuelve el clasificador base. Los pesos se actualizan para minimizar la pseudopérdida del siguiente clasificador base siguiendo una regla muy similar a AdaBoostM1. El algoritmo garantiza que el error de entrenamiento irá decreciendo en cada error si la pseudopérdida es menor que 1/2, lo cual es factible incluso para clasificadores no muy precisos.

- *BoostMA* [35] es una adaptación más directa de AdaBoost.M1 a problemas multicalse. AdaBoost.M1 necesita para continuar con la siguiente iteración que la tasa de acierto del clasificador base sea superior al 50 %. Cuando se trata de un problema binario, esta exigencia quiere decir que el clasificador base debe predecir al menos algo mejor que aleatoriamente, pero si hay más de dos clases, la exigencia es demasido fuerte y puede hacer que el algoritmo pare prematuramente. BoostMA tiene en cuenta este hecho relajando la condición de parada a que el error del clasificador de entrenamiento base en curso sea superior al de predecir la clase más frecuente. Además BoostMA adapta la función por la que se pondera cada clasificador base haciendo que sea positiva si el error es menor que el de predecir la clase mayoritaria. Asimismo, la regla de actualización de pesos incrementa los pesos en las instancias en función del error respecto a predecir la clase mayoritaria.

- En *AdaBoost.MH* [43] un problema con J clases se convierte en J problemas binarios que deciden si la instancia pertenece a una de las clases o no. Para ello cada instancia se expande en J instancias, cada una de

ellas es la instancia original más un atributo que contiene uno de los J posibles valores de la clase. Los clasificadores base toman como entrada este conjunto extendido. AdaBoostMH, utiliza clasificadores base cuyas estimaciones son un número real, cuyo signo determina la clase y cuyo módulo la confianza, por lo que resuelve los problemas binarios mediante otra versión de AdaBoost (i.e., *Real AdaBoost* [43]) capaz de tratar con este tipo de salidas de los clasificadores base. Existen además variantes de AdaBoost.MH capaces de tratar el caso multietiqueta [107] (i.e., una instancia puede ser asociada a varias clases).

Uno de los problemas más conocidos de la técnicas de Boosting es que son muy sensibles al ruido. Existen numerosas variantes que tratan de combatir este problema. Freund [38] presenta *Boost-by-Majority* (BBM), que es una técnica que da poco peso a las instancias con mucho margen negativo (el margen de una instancia en un clasificador es negativo en caso de que la predicción del mismo sea incorrecta). Es decir, no intenta clasificar aquellas instancias mal clasificadas que al estar demasiado lejos de la superficie de decisión, nunca acabarían por ser clasificadas correctamente. Estas instancias se computan como parte del error de entrenamiento, que es un parámetro que determina la condición de parada. BMM no es un método de Boosting adaptativo porque los pesos de cada clasificador no se asignan según el error de entrenamiento, sino que todos los clasificadores base tienen el mismo peso. Sin embargo, consigue que el multiclasificador resultante sea menos sensible al ruido. *BrownBoost* [39] es una mejora sobre la aproximación anterior, también debida a Freund, que sí es adaptativa. Una evolución de estos algoritmos propuesta también por el propio Freund es *RobustBoost* [40], que en vez de minimizar el error de entrenamiento, minimiza el número de ejemplos cuyos márgenes normalizados son inferiores a una constante positiva que se especifica como parámetro.

En [114] se presentan otras variantes (i.e., *AdaBoostKL* y *AdaBoostNorm2*) que permiten reducir el sobreentrenamiento, y que son adecuadas para datos con ruido, dado que utilizan una función de penalización (distinta para cada una de las dos variantes) para prevenir que AdaBoost asigne demasiado peso a los ejemplos más difíciles de clasificar.

Otra variante de AdaBoost es *Local Boosting* [121]. Se trata de una variante de Boosting con remuestreo, luego la distribución de pesos representa la probabilidad de que cada instancia sea seleccionada para el conjunto de entrenamiento de la siguiente iteración. La distribución de pesos se actualiza teniendo en cuenta el *error local* de cada instancia $\mathbf{x}$ en la iteración actual. El error local de una instancia se calcula como el cociente resultante de dividir la suma de los pesos de las instancias vecinas de $\mathbf{x}$ en las que el clasificador base actual no acierta, entre la suma de los pesos de todas las instancias vecinas de $\mathbf{x}$. El tamaño del vecindario y el tipo de distancia utilizada son configurables. La distribución de pesos de la siguiente iteración no se calcula en función del error global, como en AdaBoost, sino en función del error local. Al igual que en AdaBoost, las instancias mal clasificadas aumentan su peso, y las correctamente clasificadas lo disminuyen, pero la magnitud de ese cambio en Local Boosting dependerá del

error que el clasificador base cometa con los vecinos de cada instancia. Local Boosting, además, tiene un parámetro β que actúa sobre la función de repesado, de manera que si toma valores muy grandes provoca que los cambios en los pesos de una iteración a otra sean muy leves, mientras que si toma valores muy pequeños los cambios son de mayor magnitud. Tanto β como el número de vecinos son parámetros que conviene sintonizar previamente. Experimentalmente se muestra como una buena sintonización de estos parámetros puede hacer que Local Boosting tenga mejores resultados que AdaBoost.

Existen variantes de Boosting para problemáticas concretas. Por ejemplo, *Ivoting* [9] es una variante para aprendizaje para grandes conjuntos de datos, que también sirve para aprendizaje en línea, que utiliza remuestreo para construir los conjuntos de entrenamiento de los clasificadores base, tomando en cada iteración una muestra de tamaño fijo compuesta con mayor probabilidad por instancias incorrectamente clasificadas en la iteracción anterior, que por instancias correctamente clasificadas. Es una aproximación más robusta al ruido que AdaBoost, y no requiere repesado. Ivoting tiene una versión paralelizable [20]. Otra versión paralelizable de AdaBoost es *P-AdaBoost* [84] que consta de dos fases, la primera es idéntica al funcionamiento estándar de AdaBoost y está limitada a un número de iteraciones; la segunda es la que es realmente paralelizable, y utiliza estimaciones de los pesos obtenidas a partir de la primera fase.

AdaCost [36], es una variante de AdaBoost que tiene en cuenta una distribución de costes asignadas a cada instancia, de manera que se penalice el error en la clasificación con coste mayor. Para ello, reajusta los pesos de Boosting según una función que tiene en cuenta los costes.

Non-Linear Boosting Projection (NLBP) [48] es un algoritmo basado en Boosting. Cada clasificador base de NLBP es entrenado con todas las instancias, pero las proyecta en un espacio generado a partir de la capa oculta de un perceptrón multicapa. NLBP también utiliza pesos en las instancias para que a medida que se vayan entrenando nuevos clasificadores base, estos se centren en las instancias más difíciles de clasificar. Pero a diferencia de Boosting, NBLP no usa los pesos directamente por los clasificadores base, sino únicamente por el perceptrón multicapa que hace la proyección.

Boosting suele dar lugar a clasificadores base más diversos que Bagging, pero pagando el precio de tener unos clasificadores base que en promedio son menos precisos. Según los autores de NBLP, esta técnica puede considerarse a medio camino entre Boosting y Bagging; pues tiende a aumentar la diversidad como Boosting, pero sin dañar tanto la precisión individual como hace éste.

Boosting puede también ser un paso intermedio en la construcción de un clasificador. En [2] Boosting sirve para estimar distancias entre instancias. En lugar de entrenar AdaBoost con instancias del conjunto de entrenamiento, se entrenan con la diferencia entre dos instancias, tomando como clase 1, si los vectores que se restan pertenecen a la misma clase, y 0 en caso contrario; de manera que la salida final de AdaBoost es la *similaridad* entre las instancias del conjunto de datos. Esa similaridad luego puede ser utilizada en un método clasificador cualquiera que necesite trabajar con distancias, como por ejemplo 1-NN que tomaría como vecino más cercano el que tuviera la similaridad más

elevada.

Otro ejemplo de la utilización de Boosting para construir otro tipo de clasificador puede encontrarse en [99] en el que en entranamiento se aplica AdaBoost con *decision stumps* (árboles de una sola decisión) como clasificadores base. Con los últimos r decision stumps, en cada iteración se forma un árbol de profundidad r, donde r es un parámetro (el nivel de reutilización); de manera que el decision stump de la iteración anterior se utiliza como criterio de bifurcación en todos los nodos de nivel r, el decision stump de la penúltima iteración se utiliza como criterio de bifurcación en todos los nodos de nivel $r-1$, y así sucesivamente. La estimación del error de de entrenamiento de AdaBoost se hace sobre el árbol correspondiente a cada iteración, pero lo que genera cada iteración es un nuevo decision stump, con el que es posible crear un árbol. Los árboles se utilizan en clasificación, ponderados por los pesos de boosting. El resultado es un multiclasificador que resulta mejor que AdaBoost con el mismo número de decision stumps.

Técnicas para el aumento de la diversidad

En dos de los capítulos de esta tesis se describen dos técnicas para el aumento de la diversidad. Es por ello, que parece obligado hacer un repaso de algunas otras aproximaciones que consiguen dicho efecto. Algunas de ellas, o son específicas, o sólo se han experimentado, para bosques (i.e., multiclasificadores que utilizan como clasificadores base árboles de decisión). Esto parece debido a que los árboles de decisión son bastante sensibles a los cambios en las condiciones de entrenamiento, normalmente por la introducción de algún elemento aleatorio.

En [58] se presenta una variante de Random Forests en la que el elemento aleatorio en la construcción de los árboles está en utilizar muestras de entrenamiento distintas para cada atributo del nodo cuando se determina la condición de bifurcación del mismo. Esto hace que cada árbol sea diferente. En [59] se utilizan histogramas para discretizar los atributos. La utilización de histogramas es adecuada para acelerar la construcción de árboles en grandes conjuntos de datos. Los puntos de bifurcación de los árboles cuando se utilizan histogramas son los extremos de cada intervalo que se ha definido en la discretización. En la técnica descrita en [59] en lugar de considerar los extremos de los intervalos, se toman puntos elegidos aleatoriamente que están cerca de esos extremos.

En [29] se presenta una técnica que consigue el aumento de la diversidad haciendo que cada nodo de un C4.5 [92] decida aleatoriamente el criterio por el que bifurcar a partir de los 20 mejores criterios candidatos. Un criterio puede ser un atributo, en el caso de los atributos nominales, pero también puede ser el valor umbral por el que bifurcan los atributos numéricos. Luego para el caso numérico el mismo atributo puede ser considerado en varios criterios de bifurcación.

La técnica de *Rotation Forests* [97] utiliza árboles de decisión como clasificadores base. Para cada árbol de decisión se hacen de forma aleatoria una serie de grupos disjuntos con cada atributo. Posteriormente cada grupo se proyecta por separado utilizando Análisis de Componentes Principales (PCA). El

conjunto de todas las características proyectadas en todos los grupos disjuntos forma el conjunto de entrenamiento de un árbol C4.5. La diversidad en este caso proviene de la generación aleatoria de grupos. Rotation Forests es un método que consigue generar clasificadores base más precisos que otros multiclasificadores basados en árboles como AdaBoost, Bagging o Random Forests. En [69] se prueban distintas variantes de Rotation Forests concluyendo que la división en particiones aleatorias disjuntas es el ingrediente más decisivo del algoritmo, por encima de la técnica de proyección utilizada. Rotation Forests puede hacer remuestreos en base al peso de las instancias mal clasificadas en la iteración anterior, tal y como hace AdaBoost, dando lugar a una técnica conocida como *RotBoost* [122]. Rotation Forests también ha sido aplicado a redes RBF [100].

En [98] se presenta el método de los *Árboles Injertados* o *Grafted Trees*. La idea es que cada clasificador base esté formado por dos niveles de árboles. El primer nivel se entrena con muy pocas instancias (e.g. el 10 % del conjunto de datos), de manera que se generan árboles muy diversos. El segundo nivel consta de un árbol por cada hoja del primer nivel. Cada árbol del segundo nivel se entrena únicamente con las instancias que llegan a la hoja del primer nivel de la que se deriva. Para clasificar una instancia, cada árbol utiliza primero el árbol de primer nivel, y según sea la hoja que la instancia alcance, entonces se utiliza el correspondiente árbol de segundo nivel. Como resultado los árboles injertados son más diversos que sin el injerto, pero tan precisos como sin él, pues al final cada uno utiliza todas las instancias de entrenamiento. Al utilizar estos clasificadores base en otros multiclasificadores, como por ejemplo Bagging [7], AdaBoost.M1 [41] o Random Forests [11], se aprecia una mejora.

En [95] se utilizan *Dicotomías Anidadas* o *Nested Dichotomies* como clasificadores base de bosques. Una dicotomía es un árbol de decisión que siempre hace bifurcaciones binarias. En cada nodo se distingue entre las instancias que pertenecen a un conjunto de cases A y las que pertenecen a otro conjunto de clases B, siendo ambos conjuntos disjuntos, y siendo la unión de A y B el conjunto de clases correspondiente al nodo padre. En [95] los conjuntos A y B se crean aleatoriamente a partir de las clases en el nodo padre pero intentando equilibrar ambos conjuntos (i.e., la diferencia en el número de clases que hay entre ambos conjuntos como máximo es uno). A medida que se desciende en la construcción del árbol los nodos agrupan menos clases, hasta llegar a las hojas, a las que sólo las corresponde una única clase. Este proceso de construcción hace además que haya por cada clase sólo una hoja, a diferencia de la mayoría de árboles donde una clase, en general, puede aparecer en varias hojas. En cada nodo que no es hoja hay un clasificador binario que es el encargado de hacer la distinción entre las instancias de las clases de A y de las clases de B. Este clasificador en principio puede ser cualquiera, con tal de que sus predicciones las haga en forma de estimación de probabilidades. Cada clasificador en un nodo no hoja se entrena únicamente con las instancias del conjunto de entrenamiento que pertenecen al conjunto de clases correspondientes a ese nodo. La dicotomía predice considerando todos los caminos desde la raíz a todas las hojas y estimando cuál es el más probable. Para hacer la estimación de probabilidad de cada camino, se multiplican todas las estimaciones que van haciendo los clasificadores en los

nodos de ese camino.

Lo que se muestra en [95] es que los clasificadores en los nodos podrían ser a su vez multiclasificadores. Se utilizan como multiclasificadores de referencia Bagging [7], Random Subspaces [55], AdaBoost.M1 [41], MultiBoosting [116] y Random Forests [11]. Pero también es posible utilizar estos multiclasificadores de referencia haciendo que el clasificador base sea una dicotomía que, a su vez, utilice árboles en sus nodos. La conclusión es que esta segunda forma de combinar los árboles da mejores resultados, y que además el uso de dicotomías de cualquiera de las dos formas, da mejores resultados que la aproximación convencional que utilizaría estos multiclasificadores de referencia directamente con árboles de decisión (i.e., con C4.5 o Random Trees). La razón que hace que los bosques de dicotomías funcionen mejor está precisamente en la aleatoriedad con la que se han construido los grupos de clases en las dicotomías, haciendo que estas sean diversas.

Los *Oráculos Aleatorios Lineales* o *Random Linear Oracles* [68] son también otra forma de construir clasificadores base diversos utilizando todo el conjunto de entrenamiento. En este caso, cada clasificador base se construye definiendo previamente un hiperplano aleatorio en el espacio del problema. Cada oráculo consta de dos clasificadores miembros además del hiperplano (árboles de decisión en [68]). Cada uno de los clasificadores miembros se entrena sólo con las instancias que quedan a un lado de dicho hiperplano. Para clasificar una instancia, primero el hiperplano determina qué clasificador le corresponde atendiendo a su posición relativa respecto a dicho hiperplano. Seguidamente, el clasificador miembro del oráculo al que ha sido derivada la instancia es el que realiza la clasificación. Los resultados experimentales demuestran una mejora generalizada en distintos tipos de multiclasificadores cuando se utilizan oráculos como clasificadores base. Los hiperplanos al ser aleatorios constituyen la fuente de diversidad, pero no son la única alternativa en la construcción de oráculos.

De hecho, en [96] se comparan los oráculos lineales con los *esféricos* utilizando clasificadores base *Naïve Bayes* [31, 52, 67, 94], que es un clasificador muy simple y efectivo basado en el teorema de Bayes, el cual computa las probabilidades condicionales de cada clase y predice para una entrada la clase más probable. En el caso de los oracles esféricos las instancias se dividen entre las que están dentro o fuera de una hiperesfera en un subespacio aleatorio. El subespacio aleatorio ha de tener al menos el 50 % de las características originales. El centro de la hiperesfera es una de las instancias de entrenamiento seleccionada aleatoriamente, el radio se computa como la mediana de las distancias a otras K instancias también seleccionadas aleatoriamente. Experimentalmente los oráculos esféricos resultan ser mejores que los lineales cuando el clasificador base es Naïve Bayes.

En [21] se presenta una técnica para forzar la obtención de clasificadores base diversos utilizando una matriz de pesos, en la que cada elemento $d_{i,j}$ representa el peso que tiene la instancia i en el clasificador base j. El cálculo de la matriz de pesos puede llevarse a cabo de varias formas, una de las que se proponen consiste en dividir el espacio del problema en regiones (una por clasificador base) y calcular los pesos en función de la distancia de la instancia i a la región

del clasificador j.

DECORATE [83] entrena el primer clasificador base con todo el conjunto de entrenamiento, pero en las iteraciones sucesivas utiliza una mezcla de instancias del conjunto de entrenamiento original con otras generadas artificialmente. Para generar estas instancias respeta la distribución de cada uno de los atributos, que asume son independientes; de forma que los atributos numéricos artificiales siguen una distribución gaussiana, y los nominales ocurren con la misma probabilidad que en el conjunto original. Para asignar valores a la clase en las instancias artificiales, primero la clasifica con la versión en curso del multiclasificador, y luego la da un valor aleatorio en base a la probabilidad inversa que se obtiene de dicha clasificación.

Existen técnicas de diversidad para multiclasificadores que se derivan de partir el conjunto de datos original en otros más pequeños [20, 22, 27]. Estas técnicas de particionamiento normalmente se aplican en los casos en los que el conjunto de entrenamiento sea demasiado grande como para tener un tiempo de entrenamiento aceptable mediante métodos multiclasificadores convencionales, pero además pueden mejorar la precisión en base al incremento de la diversidad.

Output Smearing y *Output Flipping* [10] son dos técnicas que consiguen aumentar la diversidad asignado etiquetas aleatorias a un subconjunto de las instancias utilizadas para entrenar los clasificadores base. Esta dos técnicas consiguen buenos resultados si el multiclasificador tiene un gran tamaño (i.e., en torno a cien clasificadores base).

1. Output Smearing es una técnica inicialmente pensada para regresión que añade ruido gaussiano a las etiquetas numéricas. Smearing se puede extender a clasificación; pues si se tienen J clases, el problema se transforma en J problemas de regresión con etiquetas igual a la suma de la etiqueta original binarizada (i.e., 1 indica que la instancia pertenece a la clase y 0 que no) más un valor aleatorio que tiene en cuenta la distribución de las instancias en cada una de las clases. La clase se predice como la salida de mayor valor de las J regresiones.

2. Output Flipping sustituye un porcentaje de las etiquetas originales, especificadas por el parámetro *flip rate*, por otras aleatorias manteniendo la proporción de instancias en cada clase. En [75] se presenta una variante de flipping en la que utilizando multiclasificadores todavía más grandes (i.e., en torno a mil clasificadores base) se consiguen mejorar los resultados. Para ello, la asignación de etiquetas aleatorias a la clase no tiene en cuenta la proporción de instancias de cada clase. En concreto, esta última técnica funciona bien para conjuntos de datos en los que la distribución de clases no esté equilibrada, pues aumentando el valor del flip rate es posible contrarrestar en cierta medida ese desequilibrio.

En la misma línea que Random Subspaces [55], *Attribute Bagging* [15] (AB) también selecciona aleatoriamente un número fijo de atributos para entrenar cada clasificador base. Una diferencia con Random Subspaces es que el número m de atributos a seleccionar lo calcula el propio método en un primer paso. Tras

entrenar varios clasificadores base con m atributos, hace un ranking con ellos y sólo toma los mejores para participar en el multiclasificador final.

En [25] se aplica Random Subspaces, pero en una segunda fase en cada clasificador base se prueba a ir sustituyendo un atributo por otro hasta que la precisión sobre un conjunto de validación no mejora. De esta forma se consigue clasificadores base más precisos que con Random Subspaces [55], pero sin el coste computacional de otras técnicas como por ejemplo, el uso de algoritmos genéticos que se hace con el mismo fin en [50], y sin el riesgo que comporta dicho uso de algoritmos genéticos en cuanto a disminución de la diversidad cuando se trata de conjuntos de datos con pocos atributos (i.e., menos de 35).

Input Decimation [89] (ID) es otra técnica orientada a la selección de características en problemas con más de dos clases. Para ello, se entrenan L clasificadores base, donde L es el número de clases del problema. En cada clasificador base se toman sólo las n_l características que mantienen por separado una mayor correlación con la clase, donde n_l es un parámetro. En clasificación, se toma como buena la predicción del clasificador que da una estimación de probabilidad más alta.

En [72] primero se agrupan las características, de manera que cada grupo se forma con características que están entre sí muy correlacionadas. En una segunda fase se construye para cada clasificador base un conjunto de entrenamiento formado a partes iguales por características de cada grupo. Con ello, se pretende alcanza mayor diversidad que si las características hubieran sido seleccionadas con un criterio puramente aleatorio como en los Random Subspaces [55].

Combinaciones multinivel y metalearning

En este apartado se revisan técnicas que tienen una cierta similitud con Cascading [47], Stacking [118] y Grading [111], en el sentido de que los clasificadores se agrupan en niveles (i.e., los de nivel superior utilizan en su entrenamiento las salidas de los de nivel inferior) y/o permiten combinar la salida clasificadores de distinta naturaleza.

Model Class Selection [14] (MCS), divide el conjunto de datos en subespacios y a través de reglas obtenidas empíricamente, asigna a cada subespacio un clasificador base de tres posibles (i.e., un árbol de decisión, una función discriminante o un clasificador basado en instancias). En la misma línea, y para redes neuronales, en *Mixture of Experts* (ME) [87] el espacio de entrada se divide en subespacios, que pueden estar solapados entre sí, de forma que un clasificador «experto» se especializa en cada subespacio, y es otro clasificador el que combina las salidas de los expertos. Una extensión de ME es *Hierarchical Mixture of Experts* [57] (HME), en el que los espacios se descomponen recursivamente a su vez en nuevos subespacios.

Esta última idea se asemeja a los *Arbiter Trees* [18], que se forman a partir de k particiones disjuntas del conjunto de entrenamiento, cada una entrena un clasificador base del primer nivel. Estos clasificadores base se emparejan de dos en dos, y por cada pareja se entrena otro de nivel superior que es el *arbiter*. El arbiter es del mismo tipo que los clasificadores del primer nivel, pero se entrena

con aquellas instancias pertenecientes a la unión de las particiones de la pareja de primer nivel en las que las predicciones de dicha pareja de clasificadores, bien no sea la misma, o sea dudosa según una regla de selección de dichas instancias. La regla de selección puede ser distinta dependiendo de la variante de arbiter tree de la que se trate (p.e. se pueden añadir también las instancias en las que se equivoque la pareja de clasificadores). Este esquema se extiende recursivamente hacia arriba, de manera que por cada pareja de arbiters se entrena uno de nivel superior, y así sucesivamente hasta llegar a un único arbiter ráiz, formando así el arbiter tree. Para clasificar una instancia todos los clasificadores votan, pero en caso de empate se aplica una regla de arbitraje que puede ser también dependiente de la implementación. En este tipo de reglas los clasificadores votan y los arbiters tienen un voto de mayor peso para evitar empates. Los arbiter trees han resultado ser un método interesante para grandes conjuntos de datos. Pueden verse otras variantes de este mismo método en [19].

Los *Combiner Trees* [18] mantienen cierta similitud con los Arbiter Trees. La diferencia principal es que los clasificadores que no son hojas se entrenan con las salida de los clasificadores del nivel anterior. El conjunto de entrenamiento de estos clasificadores que no son hojas se especifica mediante una «regla de composición», que puede ser distinta en cada implementación (e.g. la misma regla que utiliza Staking para alimentar el nivel meta, o la misma que utiliza Cascading).

NBTree [61] es una método que básicamente es un árbol de decisión en el que en sus nodos terminales hay un clasificador *Naïve Bayes* [31, 52, 67, 94]. Naturalmente, el árbol no se desarrolla hasta el final haciendo una especie de prepoda, con el objeto de que las hojas no sean puras y el clasificador Naïve Bayes pueda intervenir en la predicción final. Para ello, se calcula una magnitud llamada «utilidad» mediante validaciones cruzadas en cada nodo con Naïve Bayes, la cual sirve para saber si al dejar el nodo actual como hoja, la precisión del Naïve Bayes correspondiente es significativamente mejor que la que se obtendría si el árbol sigue ramificándose por ese nodo.

StackingC [109, 110] es una variante de Stacking motivada porque el funcionamiento de Stacking se degrada con datos multiclase. Esta degradación se manifiesta

1. En costes computacionales, ya que cuando el número de clases es elevado, el número de atributos en el conjunto de entrenamiento del nivel meta crece proporcionalmente, haciendo que el clasificador de nivel meta generalmente aumente sus tiempos de entrenamiento y consumo de memoria.

2. En precisión, ya que según Seewald [109, 110], el exceso de características en el clasificador meta impide obtener buenos modelos.

Por ello, en StackingC el nivel meta se utiliza un regresor por cada clase (típicamente Multiresponse Linear Regression (MLR)). Cada regresor toma como entrada sólo las estimaciones de probabilidad que hacen los clasificadores de nivel base correspondientes a una de las clases. Los regresores normalizan sus

salidas entre 0 y 1, para que así tomen forma de probabilidades. La predicción más probable es la que se toma como predicción del multiclasificador.

Otra aplicación interesante de este tipo de esquemas que combinan varios tipos de multiclasificadores es la que se deriva de *NeC4.5* [123], que primeramente hace Bagging [7] utilizando redes neuronales como clasificadores base. Seguidamente se forma un nuevo conjunto de entrenamiento de forma que a las instancias del conjunto de entrenamiento original se las asigna como clases las predicciones de dicho multiclasificador Bagging. A este conjunto además se le añaden aleatoriamente instancias del conjunto de entrenamiento original según lo que indique un parámetro (i.e., *extra data ratio*). Finalmente, se entrena un árbol C4.5 [92] con este conjunto, con lo que se obtiene una representación comprensible del conocimiento que habían extraído previamente las redes neuronales.

2.3. Técnicas de validación experimental

Para poder juzgar objetivamente si un determinado clasificador es mejor que otro para un conjunto de datos dado se necesita conocer cuál es su tasa de acierto (o alternativamente la tasa de error); para lo cual habrá que recurrir a hacer una estimación de cualquiera de estas dos medidas. En esta tesis siempre se han utilizado estimaciones de la tasa de acierto.

Lógicamente, la estimación del acierto no debe de hacerse sobre el mismo conjunto que se está utilizando para entrenamiento, pues la estimación así obtenida sería sumamente optimista, especialmente en aquellos clasificadores que son capaces de «memorizar» todo el conjunto de entrenamiento (como el caso de los árboles no podados, ya vistos en 2.1.2). Por ello, se requiere que la estimación se haga sobre un conjunto de instancias distintas a las de entrenamiento (i.e., conjunto de *test*).

Reservar parte de los datos para hacer un conjunto de test, tiene dos problemas:

1. En muchos casos el número de instancias del conjunto de datos no es demasiado grande, por lo que dividirlo en dos subconjuntos — uno de entrenamiento y otro de test — disminuye el número de instancias disponibles para entrenamiento, lo que puede dar lugar a un clasificador con resultados muy pobres y una estimación del acierto del clasificador que tenga poco que ver con la real.

2. La distribución de los ejemplos en los conjuntos de entrenamiento y test, puede dar lugar a conjuntos que no sean demasiado representativos del conjunto de datos de partida.

La solución más común al primero de estos problemas es utilizar *validación cruzada* $N \times M$. Esta técnica consiste en dividir de forma aleatoria el conjunto de datos en M particiones o *folds*, cada uno con igual número de instancias. Entonces se toman $M-1$ de estas particiones para entrenar el clasificador, y la

partición restante para estimar el acierto (i.e., partición de test), simplemente contando el número de aciertos y dividiéndole por el tamaño de la partición. Este proceso se repite otras $M-1$ veces tomando cada vez una partición distinta como partición de test, y el resto como particiones de entrenamiento. Todo ello, vuelve a repetirse N veces, de forma que cada una de estas N veces se habrá elegido otra división aleatoria en M particiones. El resultado final, es la construcción de $N \times M$ clasificadores, y por tanto $N \times M$ estimaciones del acierto del método que se esté probando.

El tratamiento del segundo de los problemas es hacer que cada una de las particiones de la validación cruzada sea *estratificada*. Esto quiere decir que el número de instancias de cada clase dentro de cada partición mantenga la misma proporción que el número de instancias de cada clase dentro del conjunto de datos. La estratificación unida a la repetición del test $N \times M$ veces debería de mitigar el riesgo de que las particiones no fuesen representativas.

En esta tesis se ha utilizado siempre validación cruzada estratificada 10×10, esto es $N = M = 10$, que provee un número total de cien tests de cada clasificador sobre cada conjunto de datos. La elección de 10 como número de particiones y repeticiones es usual en el ámbito de la minería de datos.

2.3.1. Tests estadísticos utilizados

Una vez obtenidas las tasas de acierto, para poder saber si la diferencia entre dos clasificadores es significativa, se utilizan tests estadísticos. En esta tesis se han utilizado varios que a continuación se describen.

Comparación de dos métodos en un solo conjunto de datos

Se ha utilizado la versión corregida del *Resampled t-test* [85] para probar cuando un método es significativamente mejor que otro en un conjunto de datos dado.

Este test está basado en el t-test o test de *Student*, que se emplea cuando hay que estimar la media de una población normalmente distribuida, en la que el tamaño de la muestra es pequeño, y en la que la desviación típica es desconocida y hay que estimarla a partir de los datos de la muestra.

Sean $x_1, \ldots x_M$ las estimaciones de las tasa de acierto para un clasificador y un conjunto de datos dado en los M tests de una repetición de la validación cruzada, e $y_1 \ldots y_M$ las mismas estimaciones para el otro clasificador. Además $d_i = x_i - y_i$ representa las diferencias entre los resultados de los dos clasificadores en la partición i-ésima. Estas diferencias d_i constituyen la población considerada por el test.

Dado que las M diferencias d_i se suponen independientes y correspondientes a una distribución normal, M es pequeño (i.e., 10) y no es conocida la desviación típica, en principio el t-test es adecuado.

Un parámetro del test son los grados de libertad, que se corresponde con el tamaño de la población menos uno, es decir 9 para $M = 10$.

Sea $\overline{d}$ la estimación de la media de las d_i, y σ_d^2 la estimación de la varianza. Entonces, la distribución se transforma en la correspondiente de media 0 y varianza 1, usando la ecuación 2.25:

$$t = \frac{\overline{d}}{\sqrt{\sigma_d^2/k}} \tag{2.25}$$

Donde k representa el número de muestras d_i del que se dispone, en este caso M. Se calcula, pues, para un nivel de significación dado (i.e., 5 % en esta tesis), cuál es el intervalo de confianza correspondiente a la distribución t, y si la media cae fuera del intervalo, la diferencia entre ambos clasificadores será significativa con el nivel de significación propuesto.

Repetir N veces la validación cruzada, en principio supondría hacer el t-test de forma similar, sólo que ahora se tienen $N \times M$ estimaciones de la tasa de acierto por cada uno de los dos clasificadores a comparar. Sin embargo, no es cierto que las tasas de acierto correspondientes a la repetición i-ésima de la validación cruzada sean totalmente independientes de las tasas de acierto que resulten de la repetición j-ésima. El efecto es que las N repeticiones aumentarán k en la ecuación 2.25, incrementando t. Esto supone que al no ser independientes las particiones, algunas diferencias acabarían por hacerse significativas cuando quizás no lo son.

Por ello, se hace necesario introducir una corrección en el test estadístico. Una de estas correcciones es la que precisamente introduce el test *Resampled t-test*, en la que t se calcula como:

$$t = \frac{\overline{d}}{\sqrt{(\frac{1}{k} + \frac{n_2}{n_1})\sigma_d^2}} \tag{2.26}$$

Donde n_1 representa el número de instancias de entrenamiento y n_2 el número de instancias de test (e.g., en validación cruzada 10×10, $k = 100$, $n_2/n_1 = 0{,}1/0{,}9$). En la ecuación 2.26 el incremento de k no influye de una forma tan decisiva como en la ecuación 2.25.

Comparación de dos métodos a través de una colección de conjuntos de datos

La versión corregida del *Resampled t-test* sirve, por tanto para valorar si un método se comporta mejor que otro en un determinado conjunto de datos. Sin embargo, la tesis está orientada más bien a comparar nuevos métodos que se proponen, contra métodos de referencia ya existentes, a través de una colección de conjuntos de datos.

Los conjuntos de datos utilizados en cada capítulo se ha intentado que siempre fueran los mismos. Excepcionalmente, en el Capítulo 3 por su temática se utilizaron únicamente 27 conjuntos de datos en los que todos los atributos eran nominales, procedentes de los repositorios UCI [3] y Statlib (todos los conjuntos

de Statlib utilizados, a su vez, proceden de [112]). El resto de capítulos trabajaron todos con una misma colección de 62 conjuntos de datos del repositorio UCI [3].

Una primera aproximación al problema de comparar dos métodos utilizando una colección de conjuntos de datos consistiría en utilizar el t-test y dar por mejor método al que tenga más victorias significativas sobre el otro. Pero esta forma de proceder plantea los siguientes problemas, ya constatados en [26], donde dice:

> Supóngase que se están comparando dos algoritmos usando mil conjuntos de datos distintos. En todos los casos el algoritmo A resulta mejor que el B, pero la diferencia nunca llega a ser significativa. Es cierto que para un caso en particular la diferencia entre ambos algoritmos puede atribuirse al azar, pero ¿qué probabilidad habría de que un algoritmo tan sólo hubiera tenido suerte en los mil experimentos, todos ellos independientes entre si?.
>
> En contra de la creencia popular, contar únicamente las victorias y derrotas significativas no hace a los tests más fiables, sino menos, ya que previamente se establece un umbral arbitrario $p < 0{,}05$ (se refiere al nivel de significación) entre lo que cuenta y lo que no.

Por ello, en [26] se propone el *Sign test* o test de signos como más adecuado para hacer este tipo de comparaciones. Este test se utiliza en estadística para probar que no hay diferencia entre las distribuciones de dos variables aleatorias, como ocurre con el problema de comparar los resultados de dos clasificadores a través de una colección de n conjuntos de datos. En este caso las dos variables aleatorias son el número de veces que cada uno de los dos clasificadores acierta más que el otro, y la hipótesis nula es que ambos no difieren. Por tanto, se espera que cada uno acierte por separado $n/2$ veces, y que la probabilidad de acierto de cada uno sea 0,5.

Al partir de una validación cruzada $N \times M$, las tasas de acierto que se han sometido al test son las medias aritméticas de los $N \times M$ resultados obtenidos para cada método y conjunto de datos.

En [26] se asume que para n pequeño (menor o igual a 25), en el test de signos el número de conjuntos de datos en que un clasificador es mejor que el otro sigue una distribución binomial, mientras que para colecciones de datos mayores seguiría una distribución normal con media $n/2$ y desviación típica $\sqrt{n}/2$. En esta tesis las colecciones de conjuntos de datos utilizadas siempre han sobrepasado esos 25 conjuntos.

Dados unos determinados resultados experimentales, el test sirve para calcular si la probabilidad de que se verifique la hipótesis nula es menor que un cierto umbral α . Este umbral α es el *nivel de significación* o *nivel de certeza del test* que es un parámetro del mismo, el cual, en esta tesis siempre ha tomado un valor del 5 %. Por tanto, el test en este caso sirve para determinar si la probabilidad de rechazar equivocadamente la hipótesis de que ambos clasificadores son similares es menor que el 5 %.

El test podría llevarse a cabo calculando de forma iterativa para qué valor i (para $i = n, n-1$, etc...) la probabilidad acumulativa de las dos colas de la distribución normal supera ese 5 %. Entonces, el valor $i-1$ determinará el número mínimo de veces que un clasificador ha de ganar a otro para poder decir que es mejor con el nivel de significación elegido.

Sin embargo, i se puede calcular directamente. Para el caso de $\alpha = 5\,\%$, se puede utilizar el z-test: $i = n/2 + 1{,}96\sqrt{n}/2$ [26]. Esta última expresión es la que finalmente se ha utilizado en esta tesis para llevar a cabo el test de signos. En el caso de que existiesen conjuntos de datos en los que los dos clasificadores tuviesen la misma tasa de acierto, lo que se hace es sumar media victoria a cada clasificador por cada uno de estos conjuntos.

Comparación de n métodos a través de una colección de conjuntos de datos

El test de Friedman [26] sirve para contrastar la hipótesis nula de que varios métodos tienen tasas de acierto semejantes frente una misma colección de conjuntos de datos. Para realizar este test, previamente hay que computar los rankings promedios. Este paso previo se detalla en la siguiente sección.

Si dos métodos tuviesen una tasa de acierto similar, sus rankings promedios coincidirían. Si esta hipótesis nula fuese cierta, la expresión

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_i R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2.27}$$

seguiría una distribución χ_F^2 con $k-1$ grados de libertad, donde n es el número de conjuntos de datos, k el número de métodos, y R_j el ranking promedio del método j-ésimo, siempre que n y k fuesen suficientemente grandes (i.e., $n > 10$ y $k > 5$).

Una vez descartada esa hipótesis, es posible aplicar el test de Nemenyi [26]. Este test indica que un clasificador es significativamente mejor que otro cuando sus rankings promedios difieren al menos en una diferencia crítica CD calculada como:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \tag{2.28}$$

donde α es el nivel de significación del test (5 %, como de costumbre), y q_α son una serie de valores basados en la estadística *Studentized Range* dividida por $\sqrt{2}$.

El test de Nemenyi resulta ser muy conservador, por lo que se utilizará menos que el test de los signos para ver si hay diferencias significativas entre dos clasificadores.

2.3.2. Ordenación de los métodos por su acierto

Esta tesis está orientada a descubrir nuevos métodos y nuevas variantes de métodos. Por ello, es habitual comparar varias de estas variantes con varios métodos de referencia, siendo importante tener una imagen de dónde se ubica cada método en un ranking o lista ordenada de los métodos en función de su comportamiento a lo largo de la colección de conjuntos de datos utilizada.

Una primera aproximación para confeccionar esta lista consiste en enfrentar por parejas a todos los métodos con cada conjunto de datos, y contar el número de victorias, empates y derrotas significativas de cada método a lo largo de todas estas comparaciones siguiendo — en este caso — el *Resampled t-test*. Finalmente, los métodos se ordenan por la diferencia entre victorias y derrotas significativas de mayor a menor, de forma que la lista es encabezada por los que más veces han ganado significativamente, y menos veces han perdido significativamente, y nos referiremos a él como *ranking de diferencias entre victorias y derrotas significativas*.

Sin embargo, esta opción plantea la misma problemática que utilizar el número de victorias en el t-test para valorar cuál de dos métodos es mejor en una colección de conjuntos de datos, y que motivó el uso del test de signos para tal fin.

Basta imaginar que si el número de conjuntos de datos en los que dos métodos tienen diferencias significativas es escaso, el ranking de diferencias que surgiría de quitar o sustituir esos escasos conjuntos permitiría llegar a resultados totalmente distintos. Por ello, es más aconsejable utilizar para este fin el *Ranking Promedio* [26]. En el apartado anterior se vio además que este ranking se utiliza en para calcular el test de Nemenyi.

Supuesto el número de conjuntos de datos es n, este ranking se obtiene haciendo previamente un ranking según la tasa de acierto de cada método para cada conjunto de datos. Nuevamente, al haber hecho validación cruzada $N \times M$, esas tasas de acierto han de ser calculadas como la media en esas repeticiones y particiones. Si varios métodos empatan en un conjunto de datos, se les asigna el ránking de promediar sus posiciones. Por ejemplo, si 3 métodos quedasen en quinta posición, se les asignaría el ranking $(5+6+7)/3$ a los tres. De esta forma se obtienen n rankings, uno por cada conjunto de datos.

Después, se calcula la posición promedio de cada método en estos n rankings (i.e., a lo largo de todos los conjuntos de datos). Esta posición promedio se conoce como *ranking promedio*. Finalmente, los métodos se ordenan por el ranking promedio.

Aún siendo el ranking de diferencias menos fiable que el ranking promedio, en esta tesis se han calculado ambos rankings en todos los experimentos que se han hecho. En algunas ocasiones los resultados son similares, y en otras no. Naturalmente, se ha dado más credibilidad al ranking promedio. La coincidencia de ambos rankings tiene un cierto interés en cuanto podría verse como una forma de ratificar los resultados del ranking promedio.

2.3.3. Gráficas para visualización de la diversidad

El éxito de algunos de los multiclasificadores presentados en esta tesis tiene que ver con un posible aumento de la diversidad en sus clasificadores base. Para analizar experimentalmente este hecho se ha recurrido a los *diagramas Kappa-Error* [74] y se han diseñado dos nuevos tipos de diagramas basados en los propios diagramas Kappa-Error, a saber: los *diagramas de Movimiento Kappa-Error* y los *diagramas de Movimiento Relativo Kappa-Error* [79, 80, 81]. Los dos últimos constituyen quizás por si mismos una de las aportaciones de esta tesis.

Los diagramas Kappa-Error están basados en la estadística Kappa [23], que sirve para medir cuánto son de diversos dos clasificadores. Dado un problema con L clases, se construye una matriz de contingencias C de dimensión $L \times L$, donde cada $C_{i,j}$ contiene el número de instancias que cumplen simultáneamente ser asignadas a la clase i por el primer clasificador y ser asignadas a la clase j por el segundo.

En la diagonal de C se contabilizan las ocasiones en que los dos clasificadores estuvieron de acuerdo. La probabilidad de que estén de acuerdo en la clase i es $C_{i,i}/n$, donde n es el número de instancias del conjunto, de datos. Por tanto, la probabilidad de que estén de acuerdo en alguna de las clases, es la suma de estas probabilidades:

$$\Theta_1 = \frac{\sum_{i=1}^{L} C_{i,i}}{n} \tag{2.29}$$

Por tanto, Θ_1 (i.e., la probabilidad de que ambos clasificadores estén de acuerdo) ya es por si mismo una medida de la coincidencia entre las predicciones de los dos clasificadores. Sin embargo, en un conjunto de datos en el que predominasen de forma clara las instancias de una clase sobre las de otras, ambos clasificadores tenderían a predecir a esa clase, obteniéndose siempre un valor de Θ_1 elevado cualquiera que fueran los clasificadores base comparados.

Por ello, es necesario introducir algún tipo de corrección que elimine el efecto de predecir fortuitamente una determinada clase por parte de cualquiera de los dos clasificadores.

Si el primer clasificador hiciera predicciones de forma aleatoria, la estimación de la probabilidad de que el primer clasificador prediga aleatoriamente la clase i es la proporción de veces que ha predicho dicha clase, esto es: $\sum_{j=1}^{L} \frac{C_{i,j}}{n}$.

De la misma forma, si el segundo clasificador también actuase de forma aleatoria, la estimación de la probabilidad de que prediga la clase i sería $\sum_{j=1}^{L} \frac{C_{j,i}}{n}$.

La probabilidad de que ambos predijesen de forma aleatoria y simultánea la misma clase i, sería el producto de ambas probabilidades. Sumando las probabilidades para cada una de las clases queda:

$$\Theta_2 = \sum_{i=1}^{L} \left(\sum_{j=1}^{L} \frac{C_{i,j}}{n} \times \sum_{j=1}^{L} \frac{C_{j,i}}{n} \right) \tag{2.30}$$

Donde Θ_2 representa la probabilidad de que de que ambos clasificadores coincidan en sus predicciones de forma fortuita dados los valores de la matriz C (i.e., si ambos clasificadores hiciesen predicciones de forma aleatoria conforme a las frecuencias registradas en C).

A partir de estos dos estimadores, se define κ como:

$$\kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \tag{2.31}$$

κ puede tomar valores entre -1 y 1. Cuando dos clasificadores obtienen los mismos resultados, dan lugar a una matriz C donde todas las celdas distintas de cero están en la diagonal, por lo que el sumatorio de los valores de la diagonal sería el número de instancias que tuviese el conjunto, esto es n. Por tanto, en ese caso $\Theta_1 = 1$, lo que a su vez hace $\kappa = 1$, que es el máximo valor que puede tomar.

A medida que los clasificadores se vayan diferenciando los elementos en la diagonal de C irán disminuyendo su valor, mientras los que están fuera de la diagonal irán incrementándose, de manera que Θ_1 se irá haciendo cada vez más pequeño.

κ se hace cero cuando Θ_1 y Θ_2 se igualan, lo que quiere decir que la probabilidad de acuerdo entre ambos clasificadores medida por Θ_1 coincide con la probabilidad de que dicho acuerdo sea fortuito.

Si el desacuerdo fuese mayor que el esperado por dos predictores aleatorios, el numerador $\Theta_1 - \Theta_2$ se hace negativo, haciendo que κ también lo sea. Pero esto, ocurrirá rara vez.

Los valores de κ pueden utilizarse para dibujar los diagramas Kappa-Error [74]. La figura 2.9 muestra unos ejemplos de estos diagramas. En esta figura se ven cuatro nubes, correspondientes a cuatro diagramas Kappa-Error para el conjunto de datos *letter* del repositorio UCI [3]. Cada nube se corresponde con los resultados de un método multiclasificador.

Para obtener, cada nube se dibuja un punto (x, y) por cada par de clasificadores base pertenecientes a un mismo multiclasificador, de manera que x es la medida de kappa para esos dos clasificadores, e y es el promedio del error de ambos. Por lo tanto, lo ideal es que cada par de clasificadores generara un punto lo más cercano posible a la esquina inferior izquierda, porque eso significaría que son precisos y a la vez diversos.

Los métodos $\mathcal{DN}$-Bagging y $\mathcal{DN}$-Random Subspaces son dos ejemplos de métodos desarrollados en este trabajo de tesis, y que se presentan en el capítulo 4. Se trata de métodos que pretenden ser una mejora de otro existente (i.e., $\mathcal{DN}$-Bagging pretende ser una mejora de Bagging, y $\mathcal{DN}$-Random Subspaces pretende ser una mejora de Random Subspaces).

La nube del método $\mathcal{DN}$-Bagging está un poco desplazada hacia la izquierda respecto a la nube de Bagging, lo que significa que para el conjunto de datos *letter* $\mathcal{DN}$-Bagging es más diverso. Lo mismo ocurre con la nube $\mathcal{DN}$-Random Subspaces, está también desplazada a la izquierda respecto a Random Subspaces, indicando una ganancia —en este caso, algo menor— de diversidad.

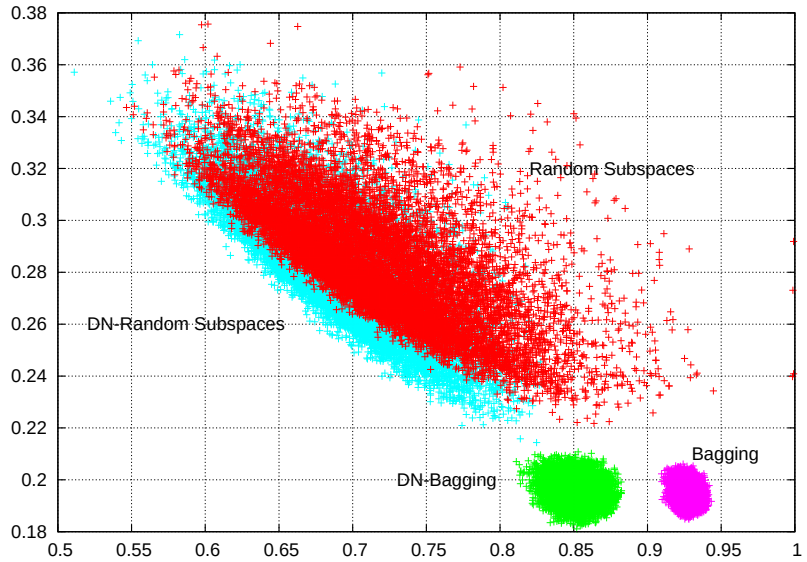


Figura 2.9: Ejemplos de diagramas Kappa-Error para el conjunto de datos *letter* y dos versiones de dos multiclasificadores. El eje horizontal representa la medida κ entre dos clasificadores base pertenecientes al mismo multiclasificador, mientras que el eje vertical representa el promedio del error de ambos.

Además cada $\mathcal{DN}$-nube está más o menos a la misma altura que la nube correspondiente a la versión pura del mismo método (i.e., sin $\mathcal{DN}$); lo que significa que los clasificadores base de la versión $\mathcal{DN}$ no empeoran respecto de los de la versión sin $\mathcal{DN}$. Este objetivo es difícil de conseguir, pues lógicamente, cuanto más diversos sean los clasificadores base, es normal que hagan peores predicciones, y que veamos como la nube de la variante más diversa se mueva hacia la parte superior del gráfico.

Dado que los métodos se validan contra una colección de conjuntos de datos, y los diagramas Kappa-Error están orientados a representar los resultados para un único conjunto de datos, es necesario acudir a otro tipo de representación que permita la visualización simultánea para todos los conjuntos de datos. Por ello, en el presente trabajo se han desarrollado dos métodos específicos que permiten tener una visión global a través de una colección de conjuntos de datos: los *diagramas de Movimiento Kappa-Error* y los *diagramas de Movimiento Relativo Kappa-Error* [79, 80, 81]. Ambos diagramas sirven para comparar dos multiclasificadores. El segundo se obtiene a partir del primero.

Para obtener el diagrama de Movimiento Kappa-Error, primero se calculan los centros de las nubes correspondientes a los diagramas Kappa-Error de cada conjunto de datos, y cada uno de los dos métodos a comparar, M_1 y M_2. Los centros se calculan promediando los valores de x e y de cada uno de los puntos de una nube. Una vez obtenidos esos centros se dibuja una flecha por cada conjunto de datos, de forma que su origen es el centro de la nube de M_1, y el fin es el centro de la nube M_2. La dirección mayoritaria de las flechas indicará si en general, teniendo en cuenta todos los conjuntos de datos, M_2 mejora (i.e., flechas con la componente x apuntando hacia la izquierda) o empeora (i.e., flechas con la componente x apuntando hacia la derecha) la diversidad de M_1, quizás a costa de aumentar el error (i.e., flechas con la componente y apuntando hacia arriba).

En el ejemplo de la figura 2.10 M_1 es Bagging y M_2 es $\mathcal{DN}$-Bagging. Se observa la tendencia de las flechas a apuntar hacia la izquierda, lo que significa una mejora de la diversidad de los métodos $\mathcal{DN}$-Bagging respecto de los Bagging puros. No hay una tendencia clara en el eje vertical, unas flechas suben un poco, otras bajan un poco, y otras se mantienen prácticamente horizontales, luego la mejora de la diversidad no ha afectado apreciablemente al acierto por separado de los clasificadores base.

El diagrama de Movimiento Relativo Kappa-Error, es un refinamiento del anterior, en el que se unifica el origen de todas las flechas en el origen de coordenadas. La información que visualizan ambos diagramas es la misma, pero quizás el efecto visual del diagrama de Movimientos Relativos evidencia aún más las diferencias entre los métodos que se estén comparando.

La figura 2.11 es el diagrama de Movimiento Relativo obtenido a partir del diagrama de Movimiento de la figura 2.10. La mejora de la diversidad y el equilibrio en las variaciones del error parecen más patentes que en el diagrama anterior.

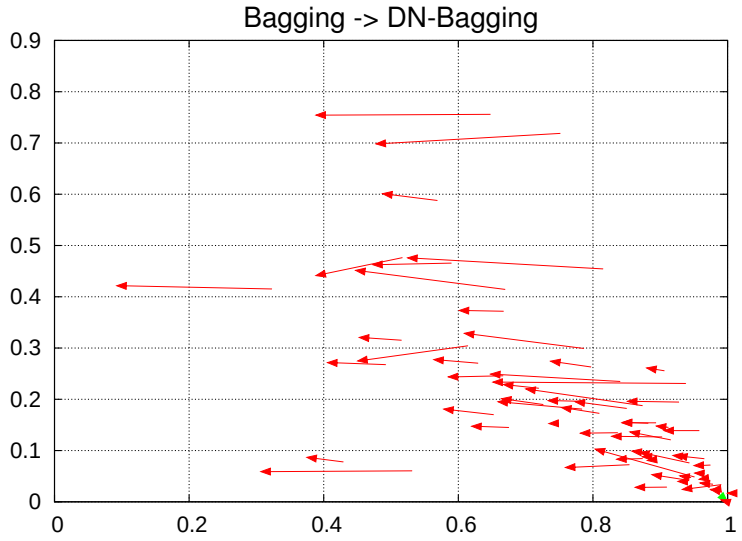


Figura 2.10: Ejemplo de diagrama de Movimiento Kappa-Error

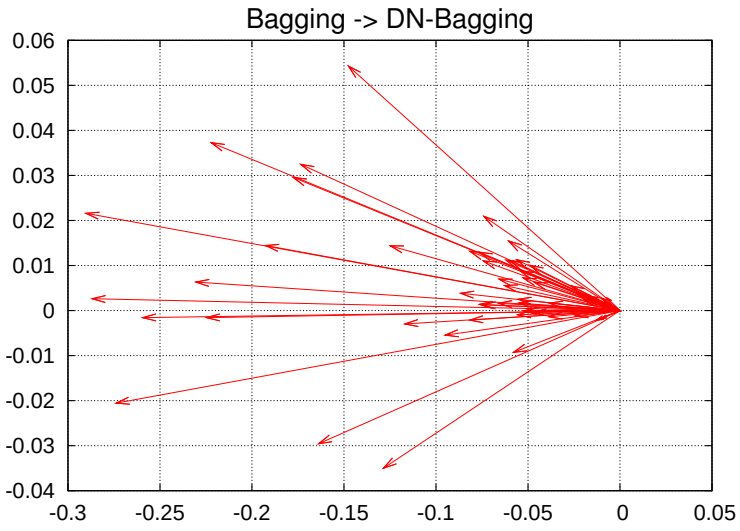


Figura 2.11: Ejemplo de diagrama de Movimiento Relativo de Kappa-Error

Capítulo 3

Cascadas para Datos Nominales

3.1. Introducción

Los datos que se utilizan en reconocimiento de patrones podrían dividirse en dos grupos: *numéricos* y *cuantitativos*. Los datos cuantitativos son aquellos que toman sus valores de un conjunto finito y predefinido. Si en estos valores no se supone que que haya orden alguno, diremos que los datos son *nominales* o *categóricos*.

Muchos métodos de clasificación trabajan únicamente con datos numéricos, por ello no son aplicables directamente a datos nominales. La forma más común de adaptar los datos nominales a métodos numéricos consiste en transformar cada característica nominal en n características binarias. Este método se conoce como NBF [49] (*n Binary Features*), donde n es el número de posibles valores que ese atributo nominal puede tomar. Según dicho método, cada valor nominal se representa mediante un grupo de atributos binarios, de forma que todos ellos valen cero, excepto el atributo correspondiente a ese valor nominal.

Una alternativa a NBF consiste en transformar los valores simbólicos en continuos. En [113] se presenta VDM (*Value Difference Metric*) como una medida de la distancia entre dos valores de tipo simbólico. VDM fue utilizado por Duch [33] para aplicar a un clasificador datos nominales previamente convertidos a numéricos mediante esta técnica. En [33] se prueba VDM utilizando redes neuronales FSM (*Feature Space Mapping*) [34] y k-NN como clasificadores. El resultado obtenido con VDM es parecido, y en algunos casos mejor, al que se obtiene con NBF.

VDM reemplaza cada valor nominal x de un atributo A, por un vector de probabilidades $\mathbf{v} = (v_1, \ldots, v_c)$, donde c es el número de clases y $v_i = P(clase = c_i | A = x)$. Es por ello que VDM acaba por aumentar de manera considerable la dimensión del conjunto de datos de entrada a medida que crece el número de clases del mismo. También NBF incrementa la dimensión del espacio de entra-

da, pero en este caso, el incremento se debe a la cardinalidad de los dominios correspondientes a los atributos nominales.

Como se vio en la sección 2.2.5, La *Cascada* o *Cascading* [47] es un multiclasificador de varios niveles, normalmente dos. En las cascadas de dos niveles se distingue un nivel *base* y otro *meta*. El clasificador del nivel base construye una extensión del conjunto de datos original al cual se han añadido nuevos atributos. Estos nuevos atributos se obtienen a partir de la distribución de probabilidades de que la instancia pertenezca a cada una de las clases. Esta estimación viene dada por el propio clasificador base. El clasificador del nivel meta toma esta extensión del conjunto de datos como datos de entrada. El clasificador base podría ser a su vez una Cascada, por lo que este esquema se puede extender de manera recursiva a más de dos clasificadores.

El presente capítulo ha dado lugar a los trabajos publicados en [76], [77] y [78]. En los mismos se presenta una mejora de los resultados de las SVM con función núcleo lineal utilizando una Cascada en la que la SVM es el clasificador de nivel meta mientras que en el nivel base utiliza un árbol de decisión. Los árboles de decisión son capaces de manejar directamente datos nominales, sin necesidad de ninguna transformación previa. Por tanto, este tipo de Cascada está orientada a utilizar datos nominales para obtener una serie de atributos correspondientes a las probabilidades. Estos valores, al ser continuos, pueden ser manejados directamente por un clasificador lineal.

No hay que perder de vista que una Cascada se limita a añadir c atributos al conjunto de datos original, donde c es el número de clases. Por lo tanto, la Cascada no reemplaza los atributos nominales por otros continuos, tan solo añade nuevas dimensiones continuas al espacio de entrada, por lo que si el clasificador del nivel meta requiere datos numéricos, seguirá siendo necesario aplicar alguna técnica de transformación (en [76] se utiliza NBF). Por ello, la dimensión del espacio de entrada seguirá tendiendo a explosionar por la influencia del método que finalmente se use para reemplazar valores nominales por numéricos.

Según los resultados que se muestran en este capítulo, la utilización de clasificadores lineales con datos nominales puede dar mejor resultado usando las características continuas que construya el clasificador de nivel base de la Cascada, que utilizando la transformación VDM, debido a que:

1. Si dos instancias tienen el mismo valor simbólico para un atributo nominal, pero pertenecen a clases diferentes, VDM calculará para ambas el mismo vector de probabilidad. Sin embargo, sería conveniente que tuvieran valores diferentes, como por ejemplo puede ocurrir al tener en cuenta los valores del resto de atributos de las instancias, tal y como ocurre con las características que construye el clasificador base de la Cascada. Esta cuestión se hace más crítica si se requiere separabilidad lineal.

2. La Cascada no reemplaza los atributos nominales originales, por lo que se requiere un método como NBF a tal efecto. Pero NBF (o cualquier otra técnica para transformar los atributos nominales en numéricos) puede conducir con relativa facilidad a una representación de los datos que no

Tabla 3.1: Ejemplo de conversión de datos nominales a binarios conducente a regiones que no son linealmente separables.

Instancias		**Instancias Binarizadas**
$(a_1, b_1, c_1)(a_1, b_2, c_2)$		$(1,0,0,1,0,0,c_1)(1,0,0,0,1,0,c_2)$
$(a_2, b_2, c_1)(a_3, b_1, c_2)$		$(0,1,0,0,1,0,c_1)(0,0,1,1,0,0,c_2)$
$(a_3, b_3, c_1)(a_2, b_3, c_2)$		$(0,0,1,0,0,1,c_1)(0,1,0,0,0,1,c_2)$
Puntos clase c_1	**Puntos clase** c_2	**Suma inecuaciones** c_1
$x_1 + x_4 + k > 0$	$x_1 + x_5 + k < 0$	$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + 3k > 0$
$x_2 + x_5 + k > 0$	$x_3 + x_4 + k < 0$	**Suma inecuaciones** c_2
$x_3 + x_6 + k > 0$	$x_2 + x_6 + k < 0$	$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + 3k < 0$
		$\Rightarrow$ **incompatibilidad**
		$\Rightarrow$ **no separables linealmente**

sea linealmente separable. En la tabla 3.1 se muestra un ejemplo en el que obtener un hiperplano de coeficientes x_i y término independiente k, que separe los puntos de la clase c_1 de los puntos de la clase c_2 resulta imposible. Sin embargo, al añadir nuevas dimensiones de entrada, las cuales son estimaciones de las probabilidades de pertenencia de esa instancia a una clase (tal y como hace la Cascada), contribuye a que se gane separabilidad lineal.

En cuanto a los resultados experimentales en las publicaciones a las que ha dado lugar este capítulo de la tesis, primeramente mostraron que Cascading aplicado a un Árbol de Decisión es un método interesante para datos nominales [76], sugiriendo que existe alguna combinación de VDM, Cascading y árboles que permite obtener un multiclasificador competitivo cuando se trata de este tipo de datos. Esta idea posteriormente fue experimentada más fondo, dando lugar a los resultados publicados en [77] y [78]. Este capítulo está basado principalmente en estos dos artículos. En los mismos se muestra experimentalmente que la configuración de cascada propuesta inicialmente en [76] se puede mejorar si se usan árboles de decisión binarios y si se usa VDM en lugar de NBF para la transformación de las características nominales.

El capítulo se estructura de la siguiente forma: la sección 3.2 describe el funcionamiento de los multiclasificadores de dos niveles utilizados en la validación experimental (Cascading, Stacking y Grading) y cómo se pueden aplicar al aprendizaje de datos nominales. La sección 3.3 analiza el efecto de VDM aplicado a los árboles de decisión y qué diferencia hay entre VDM aplicado a los árboles y utilizar directamente árboles de decisión binarios. La sección 3.4 deriva algunas equivalencias entre multiclasificadores aparentemente distintos con el fin de simplificar la la validación experimental, la cual se expone en la sección 3.5. La sección 3.6 expone de forma resumida las conclusiones del capítulo.

3.2. Multiclasificadores de dos niveles para datos nominales

Un multiclasificador de dos niveles consta de un nivel llamado *meta*, y de otro nivel llamado *base*. Cada nivel contiene clasificadores, en general, distintos. La idea en este tipo de arquitectura es que la salida del clasificador del nivel base alimente la entrada del clasificador del nivel meta. Se asume que las entradas procedentes del clasificador base que le llegan al clasificador meta serán numéricas, en tanto la salida del clasificador base sea un vector de probabilidades expresando la estimación de la pertenencia de la instancia a cada una de las clases. Esto permite transformar los datos nominales que eventualmente le puedan llegar al clasificador base en datos numéricos correspondientes a dichas estimaciones de probabilidad. Esta idea es especialmente útil cuando el clasificador meta no pueda trabajar directamente con datos nominales, utilizando para ello un clasificador base que carezca de este problema. Se han considerado tres esquemas de dos niveles: Cascading, Stacking y Grading.

Cascade Generalization (también conocida como *Cascading*)[47] es una arquitectura con la que combinar clasificadores, que normalmente presenta dos niveles. El nivel 1 (correspondiente al nivel base) se entrena con el conjunto de datos original, mientras que el nivel 2 (correspondiente al nivel meta) se entrena con un conjunto de datos aumentado, el cual contiene las características del conjunto de datos original junto las correspondientes a la salida del clasificador base. La salida del clasificador base es un vector conteniendo la distribución de probabilidad condicional $(p_1, ..., p_c)$, donde c es el número de clases del conjunto de datos original, y p_i es la estimación de probabilidad calculada por el clasificador base, de que la instancia pertenezca a la clase i.

Es posible que en ocasiones la conversión de nominal a numérico resulte en una representación de las instancias que no sea linealmente separable (como ocurría con la conversión NBF de la tabla 3.1). Con Cascading se puede resolver este problema, ya que el espacio de entrada es aumentado con nuevas dimensiones que en ocasiones serán capaces de transformar una representación no linealmente separable de los datos en una que sí lo sea.

En la aproximación que se presenta en esta tesis, estas nuevas dimensiones han sido calculadas por un clasificador capaz de trabajar directamente con datos nominales, como es el caso de los árboles de decisión.

Por otro lado, es conveniente notar que cuando: (i) el clasificador en el nivel meta de Cascading no es capaz de trabajar directamente sobre datos nominales, y (ii) parte de los atributos originales son nominales; ha de existir algún tipo de conversión de nominal a numérico asociada al clasificador del nivel meta (como por ejemplo NBF o VDM) que permita que dicho nivel pueda trabajar con esos atributos nominales. Esto es debido a que el nivel meta de Cascading toma los atributos originales junto con los de la salida del nivel base. Por tanto, si el clasificador meta no puede tratar directamente con datos nominales, el uso de Cascading con un clasificador base que trate directamente con datos nominales, no evita el uso de las técnicas convencionales de conversión a numérico.

VDM aumenta la dimensión del espacio de entrada de manera que cada atributo nominal se convierte en tantos atributos como clases tenga el problema. NBF la aumenta de manera que cada atributo nominal se convierte en tantos como valores posibles tenga ese atributo. Cascading añade a la dimensión del espacio de entrada del nivel meta un número fijo de c atributos correspondientes al vector de probabilidad, pero no evita el crecimiento debido a la aplicación de NBF o VDM al nivel meta. Luego no es una solución que presente como ventaja la reducción del número de características adicionales que usa el clasificador final. En todo caso, la ventaja será la mejora de la tasa de acierto de dicho clasificador, como se verá en la sección 3.5.

La utilización de un Árbol de Decisión en el nivel base para construir un conjunto de características numéricas podría ser implementada con otras aproximaciones de multiclasificadores de dos niveles. Una de esas posibles aproximaciones es *Stacked Generalization*, también conocida como *Stacking* [118] (ver la sección 2.2.6). En [118] los niveles de Stacking se numeran de distinta manera que en esta tesis. Para evitar confusiones, se seguirá usando la terminología meta/base para los niveles en lugar de hacer referencia a su numeración.

Normalmente Stacking utiliza más de un clasificador base, y estos clasificadores base suelen ser además distintos entre sí. Otro parámetro de Stacking es el número de particiones disjuntas (o *folds*) a utilizar en la parte de validación cruzada del entrenamiento.

Grading [111] (ver sección 2.2.7) también es un multiclasificador de dos niveles que, como Stacking, necesita hacer validación cruzada de los clasificadores base durante el entrenamiento. En [111] también se numeran lo niveles de distinta manera que en Cascading, por lo que nuevamente ambos niveles serán denotados como meta y base en el presente trabajo para evitar confusiones.

Stacking y Grading podrían utilizarse para clasificación de datos nominales de la misma manera que se ha propuesto para Cascading (i.e., usando un único clasificador en el nivel base capaz de tratar directamente con datos nominales, como por ejemplo un Árbol de Decisión). Es notable que Stacking y Grading requieren un proceso de validación cruzada que los hacen más costosos computacionalmente que Cascading incluso para el caso de un único clasificador base (que es como se van a usar en esta tesis).

Un detalle diferenciador a favor de Stacking, es que su nivel meta no necesita convertir a numéricos ningún dato de tipo nominal, pues toma como entrada únicamente el vector de probabilidad del clasificador base. Como consecuencia, la dimensión del espacio de entrada del nivel meta es fija e igual al número de clases que tenga el problema. Por el contrario, en Grading y Cascading la dimensión del espacio de entrada del clasificador del nivel meta es en general mucho mayor, debido a que no sólo toman como entrada la predicción del nivel base, sino las características correspondientes al conjunto de datos original, las cuales cuando sean nominales han de sufrir algún tipo de transformación a numérico, haciendo aumentar rápidamente el número total de atributos.

Lo habitual es ver configuraciones de Stacking y Grading con varios clasificadores base. Sin embargo, sólo se ha considerado el caso de un solo clasificador base a fin de poder comparar Cascading con otras combinaciones de un clasifi-

Para representar un multiclasificador de dos niveles, en el que el clasificador C1() es usado en el nivel meta, y el C2() en el base, se utilizará la notación

$$\text{MultiCl[M=C1();B=C2()]}(\mathbf{x})$$

donde el multiclasificador MultiCl puede ser:

- C, Cascading,
- S, Stacking,
- G, Grading,

y los clasificadores C1 y C2 pueden ser:

- DT, un Árbol de Decisión (Decision Tree),
- DTB, un Árbol de Decisión binario,
- SVM, una máquina de vectores soporte.

Figura 3.1: Notación utilizada para los multiclasificadores de dos niveles

cador en el nivel meta y otro en el base. En este capítulo se utilizarán combinaciones de una SVM en un nivel, y un Árbol de Decisión en el otro. Además se utilizará la notación que se muestra en la figura 3.1. La $\mathbf{x}$ representa una instancia a predecir. Por ejemplo, C[M=SVM(); B=DT()]($\mathbf{x}$) es una configuración de Cascading que utiliza una SVM en el nivel meta y un Árbol de Decisión en el base.

En aquellos casos en los que a un clasificador componente de estas configuraciones (ya sea meta o base) le llegue alguna característica de tipo nominal, y no sea capaz de procesarla directamente, supondremos que se hace implícitamente una transformación usando NBF. Por ejemplo:

- El SVM de C[M=SVM();B=DT()]($\mathbf{x}$) necesita NBF.
- El SVM de Stackig[M=SVM();B=DT()]($\mathbf{x}$) no necesita NBF.

3.3. Árboles de decisión binarios vs. VDM aplicado a árboles de decisión

Sea $(\mathbf{x}, y)$ una instancia del conjunto de datos, en la que $\mathbf{x}$ es un vector que actúa como entrada al clasificador, e y la variable de salida o clase. En este capítulo VDM($\mathbf{x}$) representa otro vector, tal que cada componente x_i de $\mathbf{x}$ pasa a transformarse en un grupo de componentes VDM(x_i); de manera que si x_i es nominal, VDM(x_i) se obtiene aplicando VDM a x_i; y si x_i fuese cuantitativa, VDM(x_i) es igual a x_i. Por lo tanto, la dimensión de VDM($\mathbf{x}$) es mayor o igual que la de $\mathbf{x}$.

DT(VDM($\mathbf{x}$)) representa un Árbol de Decisión que toma como entrada las características que surgen de la aplicación de VDM. En la validación experi-

mental de [76] se probó la configuración DT(VDM(**x**)) resultando ser bastante interesante, tanto por su resultado, como por su bajo coste computacional frente a otras alternativas presentadas en el estudio.

Al aplicar características nominales a los árboles de decisión, los nodos que bifurcan por dichas características se ramifican en tantas ramas como valores simbólicos posibles tenga ese atributo. Por tanto, es habitual que estos nodos tengan más de dos ramas. Sin embargo, al someter al conjunto de datos a la transformación VDM, todos los atributos se hacen numéricos, haciendo que el árbol siempre se bifurque en dos ramas: una correspondiente a las instancias que tiene un valor por debajo de un determinado umbral, y otra rama para el resto.

Cuando se trabaja con bifurcaciones no binarias, el número de ramas que parten de cada nodo aumenta haciendo que cada rama abarque un menor número de instancias de entrenamiento, por lo que para llegar a los nodos hoja basta atravesar unos pocos nodos de decisión. Una bifurcación binaria, por el contrario, tenderá a generar un árbol más profundo, dando oportunidad a que al analizar una instancia se evalúen más nodos, esto es, se tengan en cuenta más atributos. Si el número de instancias del conjunto de datos es pequeño este efecto puede verse reforzado. Por tanto, es de esperar que los árboles binarios obtengan mejores resultados que los que no lo sean. Por ejemplo, parece mejor usar DT(VDM(**x**)) que DT(**x**).

Es posible hacer unos pocos cambios en el algoritmo del Árbol de Decisión y así conseguir que todas las ramificaciones sean de tipo binario, incluso aunque se esté trabajando con datos nominales y sin VDM. En esta tesis se denotan los árboles de decisión binarios como DTB(**x**). En un árbol de este tipo, las ramificaciones binarias por un atributo nominal tendrán asociado un test lógico que compruebe si el valor del atributo es igual o distinto a un determinado valor simbólico. Sin embargo, una ramificación de tipo binario en un árbol con VDM (p.e. DT(VDM(**x**))) tiene asociado un test lógico que comprobará si algún valor numérico, proveniente de la transformación VDM, supera o no un determinado umbral. Por lo tanto, cabe esperar que los resultados de DTB(**x**) y DT(VDM(**x**)) sean ligeramente distintos.

Nótese que al transformar con VDM un conjunto de datos con el que se va a entrenar un árbol, todos los atributos pasan a ser numéricos, por lo que tanto se trate de un Árbol de Decisión no binario, como de uno binario, éste sólo tendrá nodos que efectúen comprobaciones del tipo «mayor que». De donde se deduce directamente la siguiente equivalencia:

$$\mathrm{DT}(\mathrm{VDM}(\mathbf{x})) \equiv \mathrm{DTB}(\mathrm{VDM}(\mathbf{x})) \tag{3.1}$$

Esta igualdad reduce considerablemente el número de combinaciones posibles con árboles de decisión en multiclasificadores de dos niveles.

Tabla 3.2: Ejemplos de equivalencias de multiclasificadores de dos niveles con VDM.

1.	S[M=SVM(); B=DT(VDM())](**x**)	S[M=SVM(); B=DTB(VDM())](**x**)
2.	C[M=DT(); B=DTB()](VDM(**x**))	C[M=DTB(); B=DT()](VDM(**x**))
3.	C[M=SVM(VDM()); B=DT(VDM())](**x**)	C[M=SVM(); B=DT()](VDM(**x**))
4.	S[M=SVM(); B=DT()](**x**)	S[M=SVM(VDM(()); B=DT()](**x**)
5.	S[M=DT(); B=SVM()](**x**)	S[M=DTB(); B=SVM()](**x**)
6.	S[M=DT(); B=SVM()](**x**)	S[M=DTB(VDM()); B=SVM()](**x**)

3.4. Equivalencias entre multiclasificadores de dos niveles

VDM se puede implementar como un filtro que mantiene intactos los atributos numéricos, mientras que transforma los nominales en un conjunto de probabilidades. Por ello, la notación C(VDM()) indica que se ha aplicado VDM a las entradas de un clasificador C.

Extendiendo esta misma notación, es posible indicar cuando se ha aplicado VDM a alguno de los niveles de un multiclasificador de dos niveles. Por ejemplo: C[M=SVM(VDM()); B=DT()](**x**) aplica VDM sólo a las entradas del clasificador meta SVM.

Si se tiene en cuenta la igualdad (3.1) de la sección anterior, es posible derivar un gran número de multiclasificadores de dos niveles equivalentes. En la tabla 3.2 se muestran ejemplos de multiclasificadores de dos niveles con VDM que resultan ser equivalentes. Los métodos en la segunda columna son equivalentes a los de la tercera.

Las filas primera y segunda en la tabla muestran ejemplos que se pueden derivar a partir de la equivalencia (3.1). La tercera fila ilustra que transformar los datos con VDM y luego aplicar esta transformación a una cascada, es lo mismo que aplicar por separado VDM a cada uno de los niveles de la misma. Esta equivalencia ocurre únicamente con Cascading. En el caso de Stacking y Grading, las estimaciones de probabilidad que hace VDM no son las mismas cuando se considera todo el conjunto de datos, que cuando se consideran los datos pertenecientes a las particiones de la validación cruzada. Por lo tanto, no es lo mismo aplicar VDM al conjunto de datos, y utilizar el resultado para hacer Stacking o Grading, que aplicar por separado el filtro VDM a cada uno de los dos niveles de estos multiclasificadores.

Además es posible derivar algunas equivalencias más, como consecuencia de que las entradas al nivel meta de Stacking son siempre probabilidades continuas:

1. Es lo mismo aplicar VDM al nivel meta de Stacking que no aplicarlo. Se puede ver un ejemplo en la cuarta fila de la tabla 3.2.

2. En el nivel meta de Stacking tampoco hay diferencia entre usar árboles de decisión binarios o no binarios. Puede verse un ejemplo de esta equivalencia en la quinta fila de la tabla 3.2.

Estas dos reglas se pueden combinar derivando nuevas equivalencias de forma transitiva. Por ejemplo, la sexta fila de la tabla 3.2.

Todas estas equivalencias van a servir para simplificar el número de configuraciones a tener en cuenta en el diseño de la validación experimental, descartando aquellas que sean redundantes.

3.5. Validación experimental

Para hacer la validación experimental se ha implementado Cascading y VDM en Java integrándolo en WEKA [117]. Se han probado los siguientes métodos, utilizando para ello 27 conjuntos de datos:

1. El Árbol de Decisión, tanto en su variante binaria, como en la que no es binaria. Para ello, se ha usado la implementación del árbol C4.5 de Quinlan [92] provista por WEKA, conocida como J.48. En las tablas correspondientes a la validación que aparecerán más adelante, se denota como J.48 la variante no binaria, y como J.48bin la variante binaria. Ambas variantes de J.48 serán aplicadas además a los multiclasificadores de dos niveles que usen árboles de decisión en alguno de sus niveles.

2. SMO [91], que es la implementación de SVM provista por WEKA. Se ha utilizado una función núcleo lineal. Como en el caso anterior, esta implementación ha sido utilizada también en los multiclasificadores de dos niveles que tienen una SVM en alguno de sus niveles.

3. J.48 (binario o no) con VDM, y SMO con VDM. En ambos casos, las características nominales fueron reemplazadas por la salida de VDM.

4. Cascading con un Árbol de Decisión en el nivel base y con un SVM en el meta, y la configuración invertida. A estas combinaciones se suman las variantes resultantes de aplicar VDM a los dos niveles, a uno de ellos, o a ninguno de ellos.

5. La implementación de Stacking de WEKA, con diez particiones o *folds*. Se han probado las combinaciones que surgen de aplicar J.48 (binario y no binario) en el nivel base, y SMO en el meta; así como la configuración inversa. Como en el caso de Cascading, a estas combinaciones hay que sumar las variantes resultantes de aplicar VDM a los dos niveles, a uno de ellos, o a ninguno de ellos.

6. La implementación de Grading que hace WEKA, también con diez particiones. Las configuraciones probadas se obtienen igual que en Stacking, esto es: aplicar J.48 (binario y no binario) en el nivel base, y SMO en el meta; así como la configuración inversa, más las variantes resultantes de aplicar VDM a los dos niveles, a uno de ellos, o a ninguno de ellos).

Sobre todas estas configuraciones iniciales se descartan aquellas que resultan redundantes por la aplicación de las equivalencias deducidas en la sección 3.4, de donde surgen un total de 57 métodos a probar. Se amplía la notación de la figura 3.1 con los valores SMO, J.48 y J.48bin para los clasificadores base y meta.

Se ha utilizado validación cruzada 10×10, y la versión corregida del *Resampled t-test* [85] ya comentada en la sección 2.3.1. El nivel de significación elegido para comparar los métodos fue del 5 %. Las entradas al test son los 100 resultados de aplicar la validación 10×10 a cada método y conjunto de datos.

La tabla 3.3 muestra los conjuntos de datos utilizados. En la tabla la marca (U) indica que el conjunto procede del repositorio UCI [3], mientras que la marca (S) indica que procede de Statlib. Todos los conjuntos de Statlib utilizados, a su vez, proceden de [112]. Todos los conjuntos de datos seleccionados no tienen ningún atributo numérico u ordinal [1]. Las únicas modificaciones que se hicieron a los conjuntos de datos fueron:

1. Suprimir los atributos que actuasen de clave primaria (i.e., en *Molecular biology promoters* y *Splice* datasets).

2. En *Monks-1* y *Monks-2*, sólo se ha tomado el conjunto de entrenamiento, ya que el de validación es un subconjunto del primero.

3. En *Monks-3* y *Spect*, se ha efectuado la unión de los conjuntos de entrenamiento y validación.

La tabla 3.4 ofrece una comparativa de las distintas configuraciones. Los métodos equivalentes han sido omitidos. La primera y tercera columna de la tabla 3.4 (V−D Rank y V−D, respectivamente) presentan los métodos ordenados según el test estadístico utilizado (ver *ranking de diferencias entre victorias y derrotas significativas* en la sección 2.3.2). El test se efectúa entre cada conjunto de datos y cada par de métodos, de manera que es capaz de estimar si no hay diferencia significativa entre ambos métodos o si uno es mejor que otro. Cada método tiene un número de victorias y derrotas asociadas que surgen al confrontarlo mediante el test contra el resto de métodos para todos los conjuntos de datos. Estas diferencias se muestran en la columna V−D de la tabla. Con esa diferencia es posible hacer el ranking por el que aparece ordenada la tabla (columna V−D Rank).

La segunda y cuarta columna de la tabla 3.4 (Avg Rank y Avg, respectivamente) muestran los métodos valorados según el *Ranking Promedio* [26], también

[1] En realidad, algunos de estos atributos sí que son ordinales, pero han sido tratados como nominales, tal y como pueden encontrarse en el sitio web de WEKA (`http://www.cs.waikato.ac.nz/ml/weka/`).

Tabla 3.3: Conjuntos de datos utilizados en la validación experimental del Capítulo 3. La marca (U) indica que el conjunto procede del repositorio UCI, mientras que la marca (S) indica que procede de Statlib. #I indica el número de instancias, #A el número de atributos incluyendo la clase, #C el número de clases.

Dataset	#I	#A	#C
Audiology (U)	226	70	24
Boxing1 (S)	120	4	2
Boxing2 (S)	132	4	2
Breast cancer (U)	286	10	2
Car (U)	1728	7	4
Dmft (S)	797	5	6
Fraud (S)	42	12	2
Kr-vs-kp (U)	3196	37	2
Mol Biol Prmtrs (U)	106	58	2
Monks-1 (U)	432	7	2
Monks-2 (U)	432	7	2
Monks-3 (U)	438	7	2
Mushroom (U)	8124	23	2
Nursery (U)	12960	9	5
Postop. patient (U)	90	9	3
Primary tumor (U)	339	18	21
Solar flare 1 C (U)	323	11	3
Solar flare 1 M (U)	323	11	4
Solar flare 1 X (U)	323	11	2
Solar flare 2 C (U)	1066	11	8
Solar flare 2 M (U)	1066	11	6
Solar flare 2 X (U)	1066	11	3
Soybean (U)	683	36	19
Spect (U)	267	23	2
Splice (U)	3190	61	3
Tic-tac-toe (U)	958	10	2
Vote (U)	435	17	2

comentado en la sección 2.3.2. Para ello, se hace previamente un ranking según la tasa de acierto de cada método para cada conjunto de datos. Después se calcula la posición promedio en estos rankings de cada método a lo largo de todos los conjuntos de datos, este valor se conoce como Ranking Promedio (columna Avg de la tabla). Finalmente, los métodos se ordenan por el Ranking Promedio. La columna Avg Rank de la tabla muestra las posiciones de cada método según este último ranking.

Tabla 3.4. Estudio de 57 métodos para datos nominales ordenados por su ranking promedio (se han omitido los métodos equivalentes).

V−D Rank	Avg Rank	V−D	Avg	Métodos
1	**1**	294	20.85	C[M=SMO(VDM());B=J.48bin()](**x**)
2	**2**	250	21.78	C[M=SMO();B=J.48bin()](**x**)
3	**7**	215	23.96	C[M=SMO();B=J.48()](VDM(**x**))
4	**8**	185	24.61	C[M=J.48(VDM());B=J.48bin()](**x**)
5	**3**	172	22.15	C[M=SMO();B=J.48()](**x**)
6	52	146	34.17	C[M=SMO();B=J.48(VDM())](**x**)
7.5	53	142	34.37	C[M=J.48bin();B=J.48(VDM())](**x**)
7.5	**6**	142	23.85	C[M=SMO(VDM());B=J.48()](**x**)
9	**9**	131	25.06	C[M=J.48();B=J.48bin()](VDM(**x**))
10	13	128	25.59	J.48bin(**x**)
11	15	127	26.02	C[M=J.48bin();B=SMO()](**x**)
12	4	126	23.44	C[M=J.48();B=J.48bin()](**x**)
13	25	118	28.26	S[M=J.48();B=J.48bin()](**x**)
14	14	111	25.85	C[M=J.48bin(VDM());B=J.48()](**x**)
15	5	107	23.83	C[M=J.48bin();B=J.48()](**x**)
16	57	102	37.28	C[M=J.48();B=J.48bin(VDM())](**x**)
17	20	99	27.22	G[M=J.48();B=J.48bin()](**x**)
18	16	94	26.39	J.48(VDM(**x**))
19	23	90	28.04	S[M=J.48bin();B=J.48(VDM())](**x**)
20	24	88	28.09	S[M=J.48bin();B=J.48()](VDM(**x**))
21	11	83	25.28	S[M=SMO();B=J.48bin()](**x**)
22.5	18	71	26.85	G[M=J.48();B=J.48bin(VDM())](**x**)
22.5	33	71	29.57	C[M=J.48();B=SMO()](VDM(**x**))
24	12	69	25.56	S[M=SMO();B=J.48()](VDM(**x**))
25	26	68	28.52	G[M=J.48(VDM());B=J.48bin()](**x**)
26	10	67	25.15	S[M=SMO();B=J.48(VDM())](**x**)
27	31.5	62	29.54	C[M=J.48bin();B=SMO(VDM())](**x**)
28	31.5	38	29.54	C[M=J.48(VDM());B=SMO()](**x**)
29	21	13	27.39	G[M=J.48(VDM());B=J.48bin(VDM())](**x**)
30	39	12	31.24	G[M=SMO(VDM());B=J.48bin()](**x**)
31	28	9	28.69	G[M=SMO();B=J.48bin()](**x**)
32	35	8	29.98	G[M=J.48bin();B=J.48(VDM())](**x**)
33	17	3	26.59	C[M=J.48();B=SMO()](**x**)
34	30	0	29.22	G[M=J.48bin();B=J.48()](VDM(**x**))
35	51	-23	33.81	G[M=SMO();B=J.48(VDM())](**x**)
36	45	-26	32.20	G[M=SMO(VDM());B=J.48(VDM())](**x**)
37	43	-53	32.07	G[M=SMO();B=J.48()](VDM(**x**))
38	19	-72	27.09	J.48(**x**)
39	47	-75	32.69	C[M=J.48();B=SMO(VDM())](**x**)
40	27	-95	28.59	G[M=J.48();B=SMO()](**x**)
41	29	-117	29.17	G[M=J.48bin();B=SMO()](**x**)

Tabla 3.4. Continúa de la página anterior.

V−D Rank	Avg Rank	V−D	Avg	Métodos
42	22	-126	27.87	S[M=SMO();B=J.48()](**x**)
43.5	46	-134	32.31	G[M=J.48();B=SMO(VDM())](**x**)
43.5	42	-134	32.02	S[M=J.48bin();B=J.48()](**x**)
44.5	34	-137	29.78	SMO(**x**)
44.5	38	-137	30.96	G[M=J.48(VDM());B=SMO()](**x**)
47	49	-173	32.89	SMO(VDM(**x**))
47.5	36	-174	30.09	G[M=J.48bin(VDM());B=J.48()](**x**)
47.5	37	-174	30.19	G[M=J.48bin();B=J.48()](**x**)
50	41	-179	31.85	G[M=J.48();B=SMO()](VDM(**x**))
51	50	-184	32.93	G[M=J.48(VDM());B=SMO(VDM())](**x**)
52	44	-185	32.17	G[M=SMO(VDM());B=J.48()](**x**)
53	48	-187	32.87	G[M=J.48bin();B=SMO(VDM())](**x**)
54	40	-210	31.59	G[M=SMO();B=J.48()](**x**)
55.5	56	-278	35.72	S[M=J.48();B=SMO()](VDM(**x**))
55.5	55	-278	35.15	S[M=J.48();B=SMO(VDM())](**x**)
57	54	-290	35.06	S[M=J.48();B=SMO()](**x**)

A la vista de estos rankings es posible hacer las siguientes observaciones:

1. J.48bin(**x**) parece funcionar mejor que J.48(VDM(**x**)), y ambos a su vez funcionan mejor que J.48(**x**).
2. SMO(**x**) no mejora usando VDM (i.e., SMO(VDM(**x**)).
3. Los métodos mejor posicionados en el ranking son configuraciones de Cascading que tienen un SMO en el nivel meta, y un Árbol de Decisión en el nivel base. Los dos mejores métodos en ambos rankings son C[M=SMO(VDM()); B=J.48bin()](**x**) and C[M=SMO(); B= J.48bin()](**x**). Por lo tanto, la utilización de J.48bin en el nivel base parece una mejora más relevante que el uso de VDM en el nivel meta.
4. Los métodos más costosos computacionalmente (i.e., Stacking y Grading) suelen funcionar peor que Cascading, y a veces incluso peor que un J.48 ó un SMO en solitario.
5. Cascading utilizando SMO como método base no es una mala configuración, pero suele funciona peor que Cascading usando un árbol como método base.

Este último punto aparentemente se contradice con [47], en cuanto a que en esta referencia se sugiere la elección del tipo de clasificador que ha de ir en cada uno de los dos niveles basándose en los siguientes puntos:

- *Combinar clasificadores que difieran desde la perspectiva de un análisis Bias-Varianza.*

- *En el nivel inferior utilizar algoritmos con poca varianza.*
- *En el nivel superior utilizar algoritmos con poco bias.*

Como ya se explicó en la sección 2.2.4, varianza y bias son ambas componentes del error de un clasificador [8, 62, 64, 45, 116]. Intuitivamente, dadas varias muestras del mismo conjunto de datos, el bias mide el error promedio del algoritmo de aprendizaje, mientras que la varianza mide como puede variar el error del clasificador obtenido de una muestra a otra. Por lo tanto, un algoritmo inestable, como por ejemplo una Red Neuronal o un Árbol de Decisión, tendrá una componente varianza alta, mientras que un algoritmo estable, como por ejemplo un SVM lineal o Bagging, tendrá varianza baja. Normalmente, el efecto de incrementar la componente varianza es una disminución del valor de la bias, y viceversa. Este efecto puede verse en la sintonización de parámetros de algunos clasificadores. La aplicación de Cascading usando las tres reglas anteriores, es por tanto, un intento de combinar dos clasificadores, uno con un bias bajo, y el otro con una varianza baja, para así conseguir uno nuevo que tenga valores menores en ambas medidas.

En [47] se prefiere una varianza baja en el nivel inferior, y un bias bajo en el superior, porque «*seleccionando métodos con bajo bias en el nivel superior, es posible ajustarse a a superficies de decisión más complejas, teniendo en cuenta las superficies 'estables' dibujadas por los clasificadores del nivel inferior*». La validación experimental en [47] sobre 26 conjuntos de datos del repositorio UCI da soporte a esta conclusión, pero en este experimento los conjuntos de datos tienen atributos tanto nominales como continuos.

Sin embargo, esas «*superficies estables*» es posible que no se dibujen de una manera apropiada cuando hay datos nominales, especialmente si el método del nivel inferior no puede tratar directamente con este tipo de datos, y por tanto necesita de algún tipo de conversión, como es el caso de SVM. Es por ello, que es factible que no exista realmente una contradicción entre los resultados en [47] y los obtenidos en este capítulo, pues la diferencia se explica en base a que los experimentos de ambos trabajos están enfocados a diferentes tipos de datos.

En los rankings presentados en la tabla 3.4 es obvio que existen muchos métodos no demasiado competitivos. Podría argumentarse que la presencia de dichos métodos puede estar distorsionando los resultados obtenidos. Para asegurarse de que no es así, se ha repetido el mismo estudio enfocándolo únicamente a los siguientes métodos:

1. Los siete métodos correspondientes a la intersección de los diez mejores métodos en ambos rankings (métodos con segunda columna en negrita en la tabla 3.4). Es notable que dicha intersección además contiene a los tres mejores métodos de cada ranking por separado.

2. Los métodos que sirven de componentes de los multiclasificadores funcionando en solitario (i.e., SMO($\mathbf{x}$), J.48($\mathbf{x}$), J.48bin($\mathbf{x}$)).

3. VDM aplicado a los métodos anteriores: SMO(VDM($\mathbf{x}$)) y J.48(VDM($\mathbf{x}$)). Nótese que J.48(VDM($\mathbf{x}$)) es equivalente a J.48bin(VDM($\mathbf{x}$)).

Igual que antes, se utiliza una validación cruzada 10×10 y el test corregido *Resampled t-test*, en esta ocasión para únicamente para estos doce métodos. Los resultados de los experimentos se muestran en las tablas 3.5 y 3.6. Se ha marcado en negrita el mejor método de ambas tablas para cada conjunto de datos. La mejor configuración según el ranking por la diferencia entre victorias y derrotas significativas está en la primera columna de la primera de las tablas (Cascading utilizando un SVM filtrado con VDM en el nivel meta, con un Árbol de Decisión binario en el nivel base). Dicho método se utiliza como referencia, de manera que el símbolo "∘" indica una victoria significativa sobre el método de referencia y el símbolo "•" indica una derrota significativa contra dicho método. La última fila de la tabla ofrece un resumen del total de Victorias/Empates/Derrotas significativas de cada método sobre el mencionado método de referencia.

Según la última fila de las tablas 3.5 y 3.6 el método de referencia (i.e., C[M=SMO(VDM()); B=J.48bin()](**x**)) es mejor método, aunque las diferencias entre este método y el resto de configuraciones de las tablas que usan Cascading no parecen muy importantes. Por otro lado, sólo los árboles de decisión binarios (i.e., J.48bin(**x**) and J.48(VDM(**x**))) tienen resultados comparables.

Nuevamente se hace un ranking de los métodos según la diferencia entre victorias y derrotas significativas (ver tabla 3.7) y según el ranking promedio (ver table 3.8).

C[M=SMO(VDM()); B=J.48bin()](**x**) es nuevamente el mejor método en ambos rankings. Una vez más, los multiclasificadores de tipo Cascading copan las mejores posiciones de los dos rankings. Los árboles de decisión binarios nuevamente parecen el mejor clasificador en solitario. Como en la anterior validación, la utilización de árboles de decisión binarios en el nivel base de las cascadas se manifiesta como la mejora que da el éxito a estos multiclasificadores en los rankings de la tabla 3.7. Sorprendentemente, SMO(VDM(**x**)) es el peor método en ambos rankings.

3.6. Conclusiones

Hay muchos clasificadores que requieren que los conjuntos de datos con los que trabajen tengan únicamente entradas de tipo numérico. La existencia de técnicas que permiten dar una representación numérica a los datos nominales permite aplicar clasificadores que requieren números a datos simbólicos. NBF y VDM son dos de estas técnicas. La transformación de nominal a numérico puede mejorar si se añaden nuevas características construidas por otro clasificador que sea capaz de trabajar directamente con datos nominales, como por ejemplo un Árbol de Decisión. Estas nuevas dimensiones, añadidas a las originales, pueden resultar en representaciones de los datos separables a través de algún tipo de superficie en el espacio n-dimensional (e.g. linealmente separables). Esta propiedad resulta muy interesante para bastantes algoritmos de aprendizaje numéricos. En ese capítulo se ha propuesto y probado la utilización de multiclasificadores de tipo Cascading para generar esas dimensiones extra, con el fin de mejorar el comportamiento de las SVM con función núcleo lineal sobre datos

Tabla 3.5: Acierto de los 12 métodos para datos nominales considerados (I). V/E/D son el número de victorias/empates/derrotas significativas del método de esa columna contra el método de la primera columna. Se ha marcado en negrita el mejor método de ambas tablas para cada conjunto de datos. El símbolo "∘" indica una victoria significativa sobre el método de la primera columna, mientras que el símbolo "•" indica una derrota significativa contra dicho método.

Conjunto de Datos	C[M=SMO(VDM()); B=J.48bin()](x)	C[M=SMO(); B=J.48bin()](x)		C[M=SMO(); B=J.48()](x)		C[M=SMO(VDM()); B=J.48()](x)		C[M=SMO(); B=J.48()](VDM(x))		C[M=J.48(VDM(); B=J.48bin()](x)	
Audiology	84.21	80.91	•	82.65		**84.74**		84.34		76.65	•
Boxing1	85.33	84.00		83.67		85.42		81.17		82.83	
Boxing2	79.77	80.98		80.44		78.76		79.91		79.41	
Breast cancer	70.50	70.40		74.14		**74.28**		70.92		70.43	
Car	96.72	96.76		95.14	•	95.03	•	97.32		97.21	∘
Dmft	20.10	20.22		19.81		19.89		20.18		19.72	
Fraud	70.75	68.85		73.60		75.65		85.50	∘	72.55	
Kr-vs-kp	**99.44**	**99.44**		**99.44**		**99.44**		99.35		**99.44**	
M.Biol.Prm	88.98	90.05		91.42		91.43		86.55		78.90	•
Monks-1	98.17	98.19		96.60		96.60		76.21	•	**99.51**	
Monks-2	94.79	94.89		67.14	•	67.14	•	89.70	•	**96.37**	
Monks-3	**98.63**	**98.63**		**98.63**		**98.63**		**98.63**		**98.63**	
Mushroom	**100.0**	**100.0**		**100.0**		**100.0**		**100.0**		99.99	
Nursery	99.36	99.36		98.29	•	98.25	•	99.42		**99.59**	∘
Post.patient	69.11	69.11		67.22		67.11		68.78		69.22	
Prim. tumor	43.13	43.25		44.31		43.93		43.55		40.61	
Solar f.1 C	89.70	**89.73**		88.95		88.92		88.18	•	89.61	
Solar f.1 M	89.24	89.55		89.67		89.33		88.99		89.64	
Solar f.1 X	**97.84**	**97.84**		**97.84**		**97.84**		**97.84**		**97.84**	
Solar f.2 C	82.59	82.67		82.91		82.77		82.76		82.70	
Solar f.2 M	**96.62**	**96.62**		**96.62**		**96.62**		**96.62**		**96.62**	
Solar f.2 X	**99.53**	**99.53**		**99.53**		**99.53**		**99.53**		**99.53**	
Soybean	93.91	93.78		**94.13**		93.95		93.83		92.43	
Spect	81.96	81.62		81.62		81.96		82.14		81.84	
Splice	94.93	93.96	•	93.55	•	95.33		94.80		94.32	•
Tic-tac-toe	93.81	94.16		97.35	∘	85.53	•	94.28		94.06	
Vote	96.75	96.69		96.69		96.75		96.75		**97.19**	
V/E/D		0/25/2		1/22/4		0/23/4		1/23/3		2/22/3	

Tabla 3.6: Acierto de los 12 métodos para datos nominales considerados (II). V/E/D son el número de victorias/empates/derrotas significativas del método en esa columna frente al método de la primera columna de la tabla anterior. Se ha marcado en negrita el mejor método de ambas tablas para cada conjunto de datos. El símbolo "○" indica una victoria significativa sobre el método de la primera columna de la tabla 3.5, mientras que el símbolo "•" indica una derrota significativa contra dicho método.

Conjunto de Datos	C[M=J.48(); B=J.48bin()](VDM(**x**))		SMO(**x**)		SMO(VDM(**x**))		J.48(**x**)		J.48(VDM(**x**))		J.48bin(**x**)	
Audiology	76.86	•	80.77	•	84.16		77.26	•	76.73	•	76.92	•
Boxing1	81.25		81.58		83.67		**87.00**		81.08		85.33	
Boxing2	79.75		**82.34**		79.15		80.44		79.91		79.62	
Breast cancer	70.96		69.52		68.97		**74.28**		70.88		70.50	
Car	**97.54**		93.62	•	93.20	•	92.22	•	97.30		96.63	
Dmft	19.72		**21.14**		20.73		19.60		20.06		19.82	
Fraud	86.40	○	76.10		73.10		63.05		**86.40**	○	66.10	
Kr-vs-kp	99.37		95.79	•	96.79	•	**99.44**		99.36		**99.44**	
Mol.Biol.Prm	76.82	•	91.01		**91.65**		79.04	•	76.22	•	79.09	•
Monks-1	76.12	•	74.86	•	75.00	•	96.60		76.28	•	98.33	
Monks-2	91.14		67.14	•	67.14	•	67.14	•	90.07	•	94.31	
Monks-3	**98.63**		96.12	•	95.89	•	**98.63**		**98.63**		**98.63**	
Mushroom	**100.0**		**100.00**		**100.00**		**100.00**		**100.00**		99.99	
Nursery	**99.59**	○	93.08	•	93.08	•	97.18	•	99.42		99.36	
Post. patient	69.33		67.33		67.11		69.78		69.33		**70.11**	
Primary tumor	40.56		**47.09**		42.69		41.39		41.22		41.19	
Solar flare1 C	88.15	•	88.49	•	88.19	•	88.95		88.15	•	**89.73**	
Solar flare1 M	89.58		89.70		89.33		**89.98**		89.61		89.76	
Solar flare1 X	**97.84**		**97.84**		**97.84**		**97.84**		**97.84**		**97.84**	
Solar flare2 C	82.90		82.91		82.77		**82.93**		82.89		82.72	
Solar flare2 M	**96.62**		**96.62**		**96.62**		**96.62**		**96.62**		**96.62**	
Solar flare2 X	**99.53**		**99.53**		**99.53**		**99.53**		**99.53**		**99.53**	
Soybean	92.75		93.10		93.38		91.78	•	92.77		92.30	
Spect	82.29		83.61		82.79		81.35		81.69		81.35	
Splice	94.24		92.88	•	**95.47**		94.17		94.28		94.36	•
Tic-tac-toe	94.49		**98.33**	○	73.90	•	85.28	•	94.28		93.79	
Vote	**97.19**		95.77		96.04		96.57		96.57		96.57	
V/E/D	2/21/4		1/17/9		0/19/8		0/20/7		1/21/5		0/24/3	

Tabla 3.7: Ranking de los 12 métodos por la diferencia entre victorias y derrotas significativas (V−D: Victorias−Derrotas, V: Victorias, D: Derrotas).

V−D	V	D	Métodos
44	52	8	C[M=SMO(VDM());B=J.48bin()](**x**)
32	46	14	C[M=SMO();B=J.48bin()](**x**)
25	44	19	C[M=SMO();B=J.48()](VDM(**x**))
19	42	23	C[M=J.48(VDM());B=J.48bin()](**x**)
14	42	28	C[M=SMO();B=J.48()](**x**)
7	34	27	J.48bin(**x**)
6	38	32	C[M=J.48();B=J.48bin()](VDM(**x**))
6	39	33	C[M=SMO(VDM());B=J.48()](**x**)
-3	32	35	J.48(VDM(**x**))
-40	15	55	J.48(**x**)
-55	20	75	SMO(VDM(**x**))
-55	21	76	SMO(**x**)

Tabla 3.8: Ranking promedio de los 12 métodos considerados.

Ranking Promedio	Métodos
5.80	C[M=SMO(VDM());B=J.48bin()](**x**)
5.80	C[M=SMO();B=J.48()](**x**)
5.93	C[M=SMO(VDM());B=J.48()](**x**)
5.94	C[M=SMO();B=J.48bin()](**x**)
6.00	C[M=SMO();B=J.48()](VDM(**x**))
6.43	C[M=J.48();B=J.48bin()](VDM(**x**))
6.74	C[M=J.48(VDM());B=J.48bin()](**x**)
6.83	J.48bin(**x**)
6.85	SMO(**x**)
6.87	J.48(VDM(**x**))
7.02	J.48(**x**)
7.80	SMO(VDM(**x**))

puramente nominales.

La validación experimental efectuada muestra, que para construir estas características adicionales, es muy adecuado utilizar árboles de decisión en el nivel base de un multiclasificador de tipo Cascading. Entre los resultados experimentales presentados en este capítulo destaca que, según el ranking de diferencias entre victorias y derrotas significativas, los árboles de decisión binarios parecen dar mejor resultado que los no binarios. Este comportamiento podría deberse a que los árboles de decisión no binarios tienden a expandirse más en anchura, generando prematuramente subárboles de profundidad reducida, lo que no favorece el comportamiento del árbol que finalmente resulta.

Las mejores configuraciones de Cascading obtenidas en los experimentos tienen el Árbol de Decisión en el nivel base y el SVM en el nivel meta. Sin embargo, según [47], sería más conveniente una configuración de Cascading intercambiando el nivel en el que aparecen cada uno de los clasificadores (colocando el clasificador con menor bias en el nivel meta, y el de menor varianza en el base). No obstante, esta conclusión en [47] se corresponde con una validación experimental para conjuntos de datos de cualquier tipo, en general, mientras que los resultados expuestos en este capítulo están centrados en experimentos con datos puramente nominales, donde puede interesar tener un clasificador capaz de trabajar con este tipo de datos en el nivel base aún cuando sea inestable, como en el caso de los árboles de decisión.

También han sido probados otros multiclasificadores de dos niveles (i.e., Stacking y Grading), pero sus resultados son peores que los obtenidos con Cascading, aun siendo algoritmos más costosos computacionalmente. Asimismo, se ha probado a utilizar VDM en alguno o en ambos niveles de Cascading, Stacking y Grading. Aunque VDM aparece en el método mejor posicionado en el ranking (Cascading con SVM filtrado con VDM en el nivel meta, mas un Árbol de Decisión binario en el nivel base), una segunda validación experimental conteniendo solamente los métodos componentes y los mejores multiclasificadores, revela que la diferencia de esta configuración con otras sin VDM, pero que tienen un Árbol de Decisión en el nivel base, no son muy significativas. Además, los experimentos muestran que SVM con VDM obtiene similares o peores resultados que SVM con NBF.

Capítulo 4

Disturbing Neighbors

4.1. Introducción

Un *multiclasificador* o *ensemble* consiste en un conjunto de clasificadores cuyas predicciones se combinan de alguna forma con el objeto de obtener una predicción conjunta más precisa. Para que la tasa de acierto de un multiclasificador sea mejor que la de un solo clasificador base, se requiere que los clasificadores base no predigan de forma incorrecta las mismas instancias. Por ello, es necesario que los clasificadores base sean *diversos*, a fin de que los errores de unos, a la hora de clasificar una determinada instancia, sean compensados por el acierto de otros, y en la predicción global del multiclasificador el resultado mejore.

Los multiclasificadores más populares (como por ejemplo Bagging, Random Forests, Random Subspaces o Boosting) aplican repetidamente a cada clasificador base el mismo algoritmo de entrenamiento. Por ello, cabe preguntarse si sus clasificadores base son capaces, y cómo, de proveer distintas salidas para las mismas entradas. La diversidad de los clasificadores base ha sido obtenida a través de distintas estrategias en cada caso, pero la mayoría de ellas se basan en hacer algún tipo de modificación en el conjunto de entrenamiento de cada uno de los clasificadores base.

En Bagging [7] la diversidad proviene de tomar distintos subconjuntos de instancias para entrenar cada clasificador base. En el método de los Random Subspaces [55] se toman distintos subconjuntos de atributos para entrenar cada clasificador base. Los Random Forests [11] pueden considerarse una variante de Bagging que utilizan Random Trees como clasificadores base. En estos Random Trees, la selección del atributo por el que se bifurca un nodo se hace teniendo en cuenta un subconjunto aleatorio de atributos que va cambiando para cada nodo.

Boosting [41] entrena de forma iterativa los clasificadores base modificando los pesos de las instancias que va a utilizar el siguiente clasificador base. Los nuevos pesos se computan a partir del error de entrenamiento del clasificador base de la última iteración, de manera que los siguientes clasificadores base cada vez

están más especializados en las instancias que han clasificado incorrectamente las iteraciones anteriores.

En este abanico de multiclasificadores es posible observar:

1. Que algunos de estos métodos están restringidos a tener que usar un determinado tipo de clasificador base (e.g., Random Trees), mientras que otros pueden usar cualquier clasificador base (e.g., Bagging o Random Subspaces).

2. Que en muchas ocasiones es posible mezclar métodos multiclasificadores exportando la estrategia con la que un multiclasificador obtiene diversidad a otro multiclasificador. Por ejemplo, el remuestreo que utiliza Bagging puede ser exportado a Boosting y hacer Boosting con remuestreo. La selección aleatoria de características en Random Subspaces también podría combinarse con otros métodos como Bagging o Boosting.

El método que se describe en este capítulo tiene el nombre de *Disturbing Neighbors*. Dos ventajas importantes de este método son que presenta las dos características anteriores, pues:

- Puede ser aplicado con cualquier método base.
- Su forma de obtener diversidad puede exportarse a cualquier multiclasificador existente.

De hecho, en este capítulo se validará el método haciendo pruebas con SVM y dos tipos de árboles como clasificadores base, así como con todos los métodos multiclasificadores mencionados en esta introducción.

Profundizando en la segunda de las ventajas:

1. Disturbing Neighbors no necesita tener en cuenta el esquema de combinación en el que va a ser utilizado. Es más, puede ser utilizado en cualquiera de los multiclasificadores mencionados en esta sección. Aunque estos métodos ya tengan su propia forma de obtener diversidad, la utilización adicional de Disturbing Neighbors dará lugar, generalmente, a multiclasificadores aún más diversos y más precisos.

2. La diversidad en estos métodos exportables a otros, frecuentemente, se adquiere a través de algún elemento aleatorio que interviene en el proceso de entrenamiento de los clasificadores base, (i.e., aleatoriedad en el remuestreo, aleatoriedad en la selección de características). El método que se presenta en este capítulo, Disturbing Neighbors también inserta ingredientes aleatorios que hacen que el mismo clasificador base sea construido de diferente manera cada vez.

   Disturbing Neighbors inserta esa componente aleatoria a través de la información que le suministra otro clasificador que no es necesario que sea muy preciso, pero sí que requiere que cada vez sea construido de diferente manera. Para ello, se ha utilizado un clasificador del tipo *Vecino más*

Cercano (i.e., NN ó *Nearest Neighbor*), el cual se construye a partir de un subconjunto muy pequeño del conjunto de entrenamiento. Este subconjunto es seleccionado de manera aleatoria cada vez que se aplica Disturbing Neighbors a un clasificador base. Con la información que suministra el clasificador NN a partir de una entrada de entrenamiento dada (i.e., la predicción de la clase y cuál es el vecino más cercano) se construirán nuevas características que servirán para crear un conjunto de entrenamiento aumentado que usará el clasificador base de nuestra elección. Estas nuevas características por lo general alterarán o «perturbarán» las predicciones que hubiera hecho ese clasificador base de haber sido entrenado directamente con el conjunto de entrenamiento de partida, y es por ello por lo que se ha elegido el nombre Disturbing Neighbors para este método.

El presente capítulo se organiza como sigue. La sección 4.2 describe el método Disturbing Neighbors, la sección 4.3 presenta los resultados experimentales correspondientes a la utilización de Disturbing Neighbors con SVM, la sección 4.4 valida el método experimentalmente, pero esta vez usando como clasificadores base árboles de decisión. La sección 4.5 analiza qué componentes del método resultan esenciales para su éxito, y qué partes del algoritmo pueden descartarse sin que ello suponga obtener resultados que sean significativamente peores. Finalmente, la sección 4.6 muestra las conclusiones.

4.2. Algoritmo

El algoritmo de Disturbing Neighbors ($\mathcal{DN}$) genera varios clasificadores diferentes. Para ello, añade nuevas características al conjunto de entrenamiento. Estas nuevas características son distintas cada vez que se genera un clasificador base, haciendo que dichos clasificadores base sean diversos entre si. El algoritmo se muestra en la figuras 4.1 y 4.2.

Esta última figura muestra una pequeña función que implemente el algoritmo de los vecinos más cercanos en los que se apoya el algoritmo principal. Se puede observar que dicha función tan solo devuelve un índice al vecino más cercano (no devuelve la clase predicha), y que la distancia se calcula tomando únicamente el subconjunto de dimensiones seleccionadas aleatoriamente a través de un vector de booleanos que actúa de máscara.

Dado un conjunto de datos de entrenamiento D, el método hace dos inicializaciones aleatorias para empezar a construir un clasificador 1-NN:

1. Selecciona m instancias de D aleatoriamente. Con esas instancias construye un clasificador 1-NN, donde m es un parámetro numérico de tipo entero. No se pretende que el clasificador 1-NN tenga una gran tasa de acierto, por lo que m tomará un valor relativamente pequeño.

2. También se seleccionan aleatoriamente más del 50 % de los atributos de D (i.e., los elementos de la máscara booleana que tomarán el valor *true*), de manera que las distancias euclídeas del clasificador 1-NN se computan

tomando en cuenta únicamente estas dimensiones. Es decir, las distancias se cacularán en una proyeccion aleatoria del espacio de entrada.

Figura 4.1: Entrenamiento de un clasificador base usando $\mathcal{DN}$. Función Principal.

Function $\mathcal{DN}$-BaseClassifierTrainer
input : D: Conjunto de entrenaiento con l características y n instancias,
m: Entero pequeño,
BC_T: Algoritmo de entrenamiento de un clasificadore tipo BC
output: Un clasificador entrenado utilizando la variante $\mathcal{DN}$ del método BC. Este clasificador resultante puede ser usado como clasificador base en cualquier multiclasificador

variables:
$RndDimensions$: Vector $[1..l]$ de Booleanos
$RndNeighbors$: Vector $[1..m]$ de instancias de D
D' : Conjunto de Entrenamiento Aumentado (inicialmente vacío)
begin
Tomar aletoriamente más de $l/2$ elementos de $RndDimensions$ dándoles el valor $True$ y poniendo el resto a $False$;
Llenar aleatoriamente $RndNeighbors$ con m instancias de D ;
$D' \leftarrow \emptyset$;
forall $x \in D$ **do**
$x' \leftarrow x$;
$i \leftarrow$ `NearestNeighbor(` x, $RndNeighbors$, $RndDimensions$`)` ;
Añadir m atributos booleanos a x', poniendo todos sus valores a falso excepto el correspondiente a la posición i ;
$p \leftarrow$ clase de $RndNeighbors\,[i]$;
Añadir p como una nueva característica de x' ;
Insertar x' en el conjunto de entrenamiento aumentado D' ;
end
Entrenar un clasificador de tipo BC utilizando el conjunto de entrenamiento D' y el algoritmo BC_T ;
Devolver BC;
end

Una vez hechas estas inicializaciones, se añaden $m+1$ características nuevas a cada instancia x del conjunto de entrenamiento D:

1. Una correspondiente a la clase que predice el clasificador 1-NN.
2. Otras m características booleanas, una por cada una de las m instancias pertenecientes al clasificador 1-NN. Todas estas características tomarán el valor *false* a excepción de la que indique cuál es el vecino más cercano a la instancia x.

Figura 4.2: Función 1-Nearest Neighbor utilizada en $\mathcal{DN}$.

Function NearestNeighbor
input : x:instancia del conjunto de entrenamiento,
$Neighbors$: Vector [1..m] de instancias,
$BooleanMask$: Vector [1..l] de Booleanos
output: i:entero que indica cuál es el vecino más cercano
begin
Calcular el Vecino más Cercano a x en $Neighbors$ vía distancia euclídea usando sólo dimensiones puestas a $True$ en $BooleanMask$;
Devolver el índice en $Neighbors$ que indica el 1-NearestNeighbor ;
end

La tabla 4.1 muestra algunas instancias del conjunto *iris* en las que ya aparecen los nuevos atributos construidos por $\mathcal{DN}$. Los atributos añadidos son los que empiezan por *Nearest*. El atributo *Nearest Class* representa la clase predicha por el vecino más cercano. Puede verse que m toma el valor 10. Cada atributo *Nearest* i se hace verdadero (T) únicamente cuando el i-ésimo vecino es el más cercano de entre esas diez instancias tomadas aleatoriamente. El resultado es un conjunto de datos aumentado que puede utilizarse para entrenar cualquier clasificador base de cualquier multiclasificador.

La figura 4.3 muestra el efecto de utilizar los atributos booleanos que indican cuál es el vecino más cercano en el conjunto de datos artificial *conus-torus* [65]. Este conjunto de datos se caracteriza por la dificultad para definir unas fronteras que separen adecuadamente las regiones correspondientes a cada clase. La parte izquierda de la figura muestra el conjunto de datos, mientras que en la parte derecha aparece el mismo conjunto, esta vez con las regiones de Voronoi que se derivan de utilizar 10 vecinos aleatorios.

En esa figura, además se puede ver que $\mathcal{DN}$ está proveyendo un aumento de la diversidad en tres sentidos:

1. Las regiones de Voronoi son diferentes cada vez que se construye un clasificador $\mathcal{DN}$. Como cada atributo booleano expresa si la instancia a clasificar, pertenece (o no) a su correspondiente región, la división en regiones del espacio provee una expresividad adicional al método base.

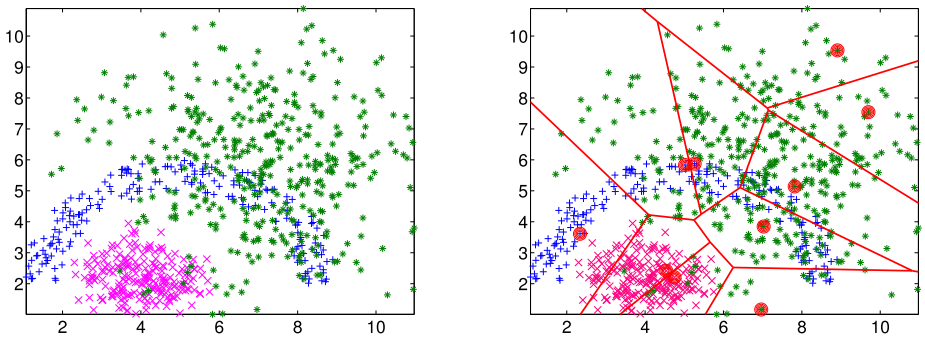
2. Aunque no es de esperar que la predicción del clasificador 1-NN sea muy precisa, probablemente tenga el acierto suficiente como para que sea considerada uno de los atributos más influyentes por el método base que lo use.

3. Además de los dos puntos anteriores, la selección aleatoria de características que se tiene en cuenta para calcular las distancias euclideas también contribuye a aumentar la diversidad. Esto es debido a que, incluso en

Tabla 4.1: Vista de algunas instancias del conjunto *iris* aumentadas al añadir nuevas dimensiones mediante $\mathcal{DN}$.

Nearest Class	Nearest 1	Nearest 2	Nearest 3	Nearest 4	Nearest 5	Nearest 6	Nearest 7	Nearest 8	Nearest 9	Nearest 10	sepallength	sepalwidth	petallength	petalwidth	class
setosa	F	F	F	F	F	F	F	**T**	F	F	5.1	3.5	1.4	0.2	setosa
setosa	F	F	F	F	F	F	F	**T**	F	F	4.9	3.0	1.4	0.2	setosa
setosa	F	F	F	F	F	F	F	**T**	F	F	4.7	3.2	1.3	0.2	setosa
versicolor	F	F	F	F	F	F	**T**	F	F	F	7.0	3.2	4.7	1.4	versicolor
versicolor	F	F	**T**	F	F	F	F	F	F	F	5.5	2.3	4.0	1.3	versicolor
virginica	**T**	F	F	F	F	F	F	F	F	F	5.9	3.2	4.8	1.8	versicolor
virginica	F	F	F	**T**	F	F	F	F	F	F	7.4	2.8	6.1	1.9	virginica
virginica	F	F	F	F	F	F	F	F	**T**	F	6.1	2.6	5.6	1.4	virginica
versicolor	F	F	F	F	F	F	**T**	F	F	F	4.9	2.5	4.5	1.7	virginica
							...								

Figura 4.3: Regiones de Voronoi para el conjunto de datos *conus-torus*.



el caso en el que dos clasificadores base contuvieran un conjunto de m vecinos muy similar, tanto la salida de ambos 1-NN, como los atributos booleanos correspondientes a las regiones de Voronoi serían diferentes, al ser distintos los atributos utilizados en el cálculo de las distancias.

En las secciones 4.2.1 y 4.2.2 se desarrollan estos puntos uno para el caso de dos clasificadores base concretos: SVM y árboles de decisión.

Por lo tanto, el clasificador 1-NN altera (*molesta*) las condiciones normales bajo las que se construiría un clasificador base en esas tres formas descritas. Un clasificador base entrenado con el conjunto de datos aumentado por $\mathcal{DN}$ probablemente no funcione mejor que cuando se entrena con el conjunto de datos original, pero cuando se utiliza $\mathcal{DN}$ en un conjunto de clasificadores base pertenecientes a un multiclasificador, toda la aleatoriedad que inyecta $\mathcal{DN}$ resulta en un conjunto de clasificadores diverso, haciendo que generalmente mejore la tasa de acierto del multiclasificador del que forman parte, como se ve en las secciones 4.3 y 4.4.

Finalmente, otra ventaja de $\mathcal{DN}$ es que aquellos multiclasificadores que son paralelizables computacionalmente, como Bagging, Random Subspaces o Random Forests, siguen manteniendo esta propiedad cuando se usen con clasificadores base tipo Disturbing Neighbors. En los experimentos que se muestran en este capítulo el parámetro m siempre toma el valor diez, por lo que el incremento de coste computacional no crece significativamente por la utilización de la variante $\mathcal{DN}$ en los clasificadores base utilizados (SVM y árboles de decisión).

4.2.1. El efecto del algoritmo en SVM

En capítulos anteriores se ha visto que las máquinas de vectores soporte [115], o SVM, son clasificadores bastante estables. Pero la estabilidad de SVM los hace problemáticos como clasificadores base. Si un clasificador base resulta ser muy estable, es decir, poco sensible a cambios en el conjunto de entrenamiento, es difícil obtener un conjunto de dichos clasificadores base que sean distintos.

Más adelante, en este mismo capítulo se verifica experimentalmente que Disturbing Neighbors sirve para hacer multiclasificadores de SVM más diversos y más precisos. Este aumento de diversidad por la aplicación de $\mathcal{DN}$ puede verse en el ejemplo de la tabla 4.2, que muestra los coeficientes de los hiperplanos SVM y $\mathcal{DN}$-SVM para el conjunto *iris*.

Los hiperplanos $\mathcal{DN}$-SVM no sólo se diferencian de los hiperplanos SVM en que tienen más coeficientes correspondientes a la predicción hecha por su clasificador NN (los tres coeficientes $N_{class} = ...$), y al vecino más cercano (los diez coeficientes N_i), sino que los coeficientes de las cuatro clases originales (sepal length/width y petal length/width) son también distintos.

Los coeficientes de la tabla han sido todos escalados para que los correspondientes a estas cuatro últimas dimensiones sean comparables. Esto es, el cuadrado de los mismos suma uno en todos los casos [1].

[1] El término independiente en la tabla tiene el signo correspondiente a encontrarse en el lado de la igualdad contrario al resto de términos de la ecuación del hiperplano.

Tabla 4.2: Coeficientes de los hiperplanos resultantes de computar los SVM y los $\mathcal{DN}$-SVM ($m = 10$) para el conjunto *iris*.

Coeficientes del hiperplano	Iris-setosa vs. Iris-versicolor		Iris-setosa vs. Iris-virginica		Iris-versicolor vs. Iris-virginica	
	SVM	$\mathcal{DN}$-SVM	SVM	$\mathcal{DN}$-SVM	SVM	$\mathcal{DN}$-SVM
N_{class}=setosa		-1.74		-1.19		0.00
N_{class}=versicolor		0.90		0.51		-0.16
N_{class}=virginica		0.84		0.67		0.16
N_1		0.00		0.06		0.11
N_2		0.00		0.14		0.19
N_3		-0.53		-0.39		0.00
N_4		0.34		0.17		-0.10
N_5		-0.59		-0.36		0.00
N_6		0.39		0.17		-0.32
N_7		0.46		0.29		0.14
N_8		0.56		0.34		-0.06
N_9		0.00		0.00		0.04
N_{10}		-0.62		-0.43		0.00
sepallength	0.20	0.22	0.25	0.30	0.06	0.24
sepalwidth	-0.45	-0.27	-0.25	-0.18	-0.17	-0.21
petallength	0.65	0.65	0.70	0.68	0.59	0.72
petalwidth	0.57	0.68	0.63	0.64	0.79	0.62
t. independiente	0.21	-0.16	0.49	0.17	0.89	0.96

Por tanto, existirán instancias que pueden ser clasificadas de manera distintas por ambos conjuntos de tres hiperplanos, consiguiendo así la diversidad que se persigue.

Al aplicar $\mathcal{DN}$ a un clasificador lineal como SVM, las nuevas características que provee el vector de booleanos son al final nuevas dimensiones en el espacio de entrada. Estas nuevas dimensiones representan la pertenencia de cada instancia a una de las regiones de Voronoi definidas por los m vecinos. En general, cualquiera de estas regiones Voronoi puede contener una mezcla de instancias de los dos tipos a discriminar. Sin embargo, en algunas de ellas es posible que predomine notablemente la población de instancias correspondiente a una de las dos clases sobre la otra. Si fuese así, la pertenencia a estas regiones puede ser utilizada en alguna medida por el clasificador «molestado» para predecir a qué clase pertenece la instancia a clasificar.

Los m valores booleanos de estas m dimensiones son, para cualquier instancia, todos cero (o falso) menos uno que vale uno (o verdadero). Por ello, en el caso de que el clasificador molestado sea un SVM ($\mathcal{DN}$-SVM), al sustituir estos valores en la ecuación del hiperplano, la contribución de estas nuevas dimensiones se corresponde únicamente con el valor del coeficiente correspondiente a la región Voronoi a la que pertenece la instancia. Por tanto, si el hiperplano se sirve de esta nueva dimensión para clasificar, aumentará el valor absoluto de ese coeficiente para que así pueda influir en mayor medida en que la predicción tenga un determinado signo: el signo de las instancias de la región.

4.2.2. El efecto del algoritmo en árboles de decisión

Las dimensiones booleanas correspondientes a los vecinos influyen en la expresividad de los árboles de decisión. Si se aplica $\mathcal{DN}$ a un Árbol de Decisión, los nodos internos no sólo pueden hacer comparaciones por el valor de uno de los atributos del conjunto de datos original, sino que ahora cabe la posibilidad de hacer comparaciones por la pertenencia a alguna de estas regiones, como muestra la figura 4.4.

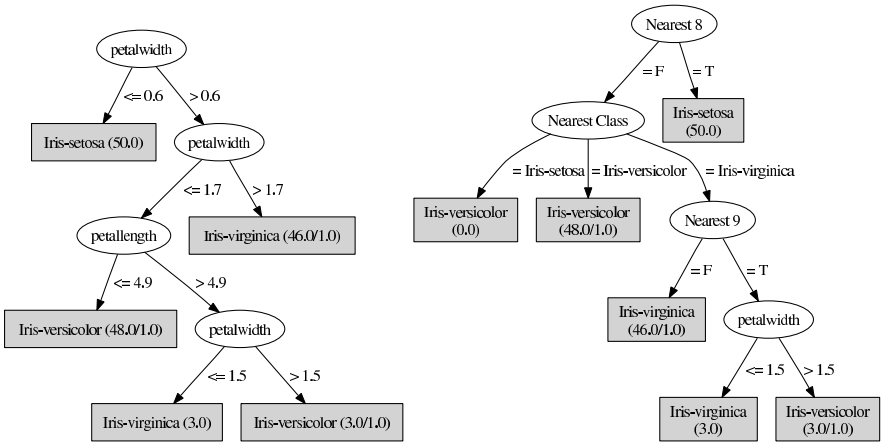
En la parte izquierda de la figura se muestra un árbol C4.5 para el conjunto de datos *iris*, mientras que a la derecha un árbol $\mathcal{DN}$-C4.5 para el mismo conjunto de datos. Puede observarse que algunos de los nodos internos del $\mathcal{DN}$-C4.5 usan atributos del tipo *Nearest Neighbor* y *Nearest Class* (predicción del Nearest Neighbor).

Es razonable pensar que si una región contuviera instancias que mayoritariamente pertenecieran a una determinada clase, probablemente la pertenencia a esa región se utilizaría como comparación en un nodo interno del árbol.

Como cada vez que se construye un árbol usando $\mathcal{DN}$ se eligen vecinos distintos, también se producirán regiones Voronoi distintas que darán lugar a atributos booleanos diferentes, haciendo el que conjunto de árboles utilizados en un multiclasificador sean diversos.

En cuanto a la predicción hecha por los vecinos, a diferentes vecinos, diferentes predicciones, que nuevamente beneficiarían el aumento de la diversidad entre los clasificadores base. La figura 4.4 muestra como es probable que exista

Figura 4.4: Un árbol C4.5 y otro $\mathcal{DN}$-C4.5 para el conjunto de datos *iris*.



un nodo interno cercano a la raíz que utilice la predicción de la clase del 1-NN (ver el nodo *Nearest Class*).

4.3. Resultados de $\mathcal{DN}$ con SVM

La validación experimental se ha llevado a cabo implementando el método a validar (Disturbing Neighbors) en Java e integrándolo dentro de WEKA [117]. El número de vecinos ha sido siempre $m = 10$. $\mathcal{DN}$ ha sido validado utilizando las siguientes implementaciones de multiclasificadores que ya provee WEKA (a no ser que se indique otra cosa, siempre se han utilizado los parámetros por defecto de WEKA):

1. Bagging [7].

2. Boosting: Se ha utilizado tanto AdaBoost [41] como MultiBoost [116]. En ambas versiones de Boosting se han considerado tanto la variante de repesado (*reWeighting*), como la de remuestreo (*reSampling*), las cuales se han denotado respectivamente como (W) y (S) en las tablas.

3. Random Subspaces [55]: Se ha utilizado dos configuraciones, tomando respectivamente el 50 % y el 75 % de las dimensiones del problema original.

El tamaño del multiclasificador es siempre cincuenta en todos los experimentos. La implementación de SVM utilizada como método base es el método SMO (*Sequential Minimal Optimization*) [91] de WEKA, el cual se ha ejecutado con función núcleo lineal.

También se ha incluido en el estudio un multiclasificador con cincuenta clasificadores base del tipo $\mathcal{DN}$-SVM cuya salida es un vector de probabilidades

calculado como el promedio de las cincuenta probabilidades de los clasificadores base. Este multiclasificador se añade para comprobar si $\mathcal{DN}$-SVM es capaz de funcionar por si mismo bien, sin necesidad de ningún esquema de combinación más o menos sofisticado. Se dará el nombre de $\mathcal{DN}$-Ensemble a dicho método.

Finalmente, también se quiere averiguar:

1. Si el método 1-NN es el responsable del buen comportamiento de los multiclasificadores con $\mathcal{DN}$. Por ello, se incluye en la prueba IBk (la implementación que hace WEKA de k-NN [1]). Las configuraciones que se han probado de este método incluyen $k = 1$ y k variable. En esta última configuración WEKA optimiza el valor de k para cada conjunto de datos mediante validación cruzada.

2. Si el acierto de un solo $\mathcal{DN}$-SVM es significativamente mejor que el de un solo SVM. Esta prueba sirve para ver si la mejora en los multiclasificadores con $\mathcal{DN}$ proviene del incremento de la diversidad o del aumento de la precisión de los clasificadores base. Por ello, se ha añadido un $\mathcal{DN}$-SVM al test.

El método de los vecinos más cercanos suele ser muy robusto frente a posibles variaciones en el conjunto de datos, por lo que cuando se combina con multiclasificadores como los que se propone en esta validación, no suele dar lugar a clasificadores que mejoren el comportamiento de un sólo k-NN [30]. Esta es la razón por la que no se ha incluido en la validación multiclasificadores que utilicen k-NN como clasificador base. En particular, es conocido que Bagging con 1-NN como clasificador base es equivalente a 1-NN [17], y además según [5], Bagging puede degradar levemente su rendimiento cuando se utilizan algoritmos estables como clasificadores base, como es el caso de k-NN.

Para hacer la validación se han utilizado 62 conjuntos de datos de la UCI [3], que pueden verse en la tabla 4.3. Como en ocasiones anteriores, se ha utilizado validación cruzada estratificada 10×10. Las tasas de acierto del experimento se plasman en las tablas de la A.1 a la A.5.

La tabla 4.4 muestra el resultado del ranking promedio [26] de los métodos considerados sobre los 62 conjuntos (ver sección 2.3.2). Como en capítulos anteriores, se asigna un número por cada método y conjunto de datos, correspondiente a la posición que ocupa ese método en el ranking correspondiente a ese conjunto de datos (en el caso de que hubiese empates, se toma el promedio de los métodos empatados). Con estos valores se calcula la posición promedio de cada uno de los métodos considerando esta vez todos los conjuntos de datos (véase primera columna de la tabla 4.4). Finalmente, los métodos se ordenan utilizando esos valores.

La tabla 4.6 ordena los métodos por la diferencia entre victorias y derrotas significativas usando la versión corregida del *Resampled t-test* [85] con nivel de significación del 5 % (ver Secciones 2.3.1 y 2.3.2).

Es observable en ambas tablas que todas las versiones de multiclasificadores sin $\mathcal{DN}$ son siempre superadas por la correspondiente versión con $\mathcal{DN}$. La figura 4.5 muestra ese aumento de precisión en cada conjunto de datos para

Tabla 4.3: Lista de los conjuntos de datos utilizados en los experimentos para $\mathcal{DN}$.

id	Conjunto	#N	#D	#I	#C	id	Conjunto	#N	#D	#I	#C
1	abalone	7	1	4177	28	32	lymphography	3	15	148	4
2	anneal	6	32	898	6	33	mushroom	0	22	8124	2
3	audiology	0	69	226	24	34	nursery	0	8	12960	5
4	autos	15	10	205	6	35	optdigits	64	0	5620	10
5	balance-scale	4	0	625	3	36	page	10	0	5473	5
6	breast-w	9	0	699	2	37	pendigits	16	0	10992	10
7	breast-y	0	9	286	2	38	phoneme	5	0	5404	2
8	bupa	6	0	345	2	39	pima	8	0	768	2
9	car	0	6	1728	4	40	primary	0	17	339	22
10	credit-a	6	9	690	2	41	promoters	0	57	106	2
11	credit-g	7	13	1000	2	42	ringnorm	20	0	300	2
12	crx	6	9	690	2	43	sat	36	0	6435	6
13	dna	0	180	3186	3	44	segment	19	0	2310	7
14	ecoli	7	0	336	8	45	shuttle	9	0	58000	7
16	glass	9	0	214	6	46	sick	7	22	3772	2
16	heart-c	6	7	303	2	47	sonar	60	0	208	2
17	heart-h	6	7	294	2	48	soybean	0	35	683	19
18	heart-s	5	8	123	2	49	soybean-small	0	35	47	4
19	heart-statlog	13	0	270	2	50	splice	0	60	3190	3
20	heart-v	5	8	200	2	51	threenorm	20	0	300	2
21	hepatitis	6	13	155	2	52	tic-tac-toe	0	9	958	2
22	horse-colic	7	15	368	2	53	twonorm	20	0	300	2
23	hypo	7	18	3163	2	54	vehicle	18	0	846	4
24	ionosphere	34	0	351	2	55	vote1	0	15	435	2
25	iris	4	0	150	3	56	voting	0	16	435	2
26	krk	6	0	28056	18	57	vowel-context	10	2	990	11
27	kr-vs-kp	0	36	3196	2	58	vowel-nocontext	10	0	990	11
28	labor	8	8	57	2	59	waveform	40	0	5000	3
29	led-24	0	24	5000	10	60	yeast	8	0	1484	10
30	letter	16	0	20000	26	61	zip	256	0	9298	10
31	lrd	93	0	531	10	62	zoo	1	15	101	7

#N: Atributos Numéricos, #D: Atributos Discretos,
#I: Número de Instancias, #C: Número de Clases

Tabla 4.4: Ranking promedio de la validación experimental de $\mathcal{DN}$ con clasificadores base SVM.

Ranking Promedio	Método
6.69	$\mathcal{DN}$-MultiBoost (S)
6.78	$\mathcal{DN}$-Bagging
6.87	$\mathcal{DN}$-MultiBoost (W)
7.78	$\mathcal{DN}$-Ensemble
8.15	$\mathcal{DN}$-Subspaces (75 %)
9.31	k-Nearest Neighbors
9.36	Bagging
9.40	MultiBoost (W)
9.50	MultiBoost (S)
9.57	$\mathcal{DN}$-AdaBoost (W)
10.07	$\mathcal{DN}$-AdaBoost (S)
10.90	$\mathcal{DN}$-Subspaces (50 %)
11.14	SMO
11.46	Subspaces (75 %)
11.71	AdaBoost (W)
12.31	1-Nearest Neighbor
12.56	$\mathcal{DN}$-SVM
12.77	AdaBoost (S)
13.66	Subspaces (50 %)

Tabla 4.5: Comparación de las posiciones de los multiclasificadores con $\mathcal{DN}$-SVM vs. SVM en el ranking promedio.

Posición	Método	Posición	Método
1	$\mathcal{DN}$-MultiBoost (S)	7	Bagging
2	$\mathcal{DN}$-Bagging	8	MultiBoost (W)
3	$\mathcal{DN}$-MultiBoost (W)	9	MultiBoost (S)
5	$\mathcal{DN}$-Subspaces (75 %)	14	Subspaces (75 %)
10	$\mathcal{DN}$-AdaBoost (W)	15	AdaBoost (W)
11	$\mathcal{DN}$-AdaBoost (S)	18	AdaBoost (S)
12	$\mathcal{DN}$-Subspaces (50 %)	19	Subspaces (50 %)

Tabla 4.6: Ranking de diferencias entre victorias y derrotas significativas de la validación experimental de $\mathcal{DN}$ con clasificadores base SVM (V−D: Victorias−Derrotas, V: Victorias, D: Derrotas).

V−D	V	D	Método
273	334	61	$\mathcal{DN}$-MultiBoost (S)
254	318	64	$\mathcal{DN}$-MultiBoost (W)
181	266	85	$\mathcal{DN}$-AdaBoost (S)
178	257	79	$\mathcal{DN}$-AdaBoost (W)
176	282	106	$\mathcal{DN}$-Bagging
127	305	178	k-Nearest Neighbor
126	245	119	$\mathcal{DN}$-Ensemble
63	220	157	$\mathcal{DN}$-Subspaces (75 %)
-1	153	154	MultiBoost (S)
-22	145	167	MultiBoost (W)
-48	141	189	Bagging
-82	106	188	AdaBoost (S)
-94	269	363	1-Nearest Neighbor
-103	106	209	AdaBoost (W)
-127	93	220	$\mathcal{DN}$-SMO
-149	141	290	$\mathcal{DN}$-Subspaces (50 %)
-151	88	239	SMO
-201	107	308	Subspaces (75 %)
-400	61	461	Subspaces (50 %)

Tabla 4.7: Comparación de las posiciones de los multiclasificadores con $\mathcal{DN}$-SVM vs. SVM en el ranking de diferencias entre victorias y derrotas significativas.

Posición	Método	Posición	Método
1	$\mathcal{DN}$-MultiBoost (S)	9	MultiBoost (S)
2	$\mathcal{DN}$-MultiBoost (W)	10	MultiBoost (W)
3	$\mathcal{DN}$-AdaBoost (S)	11	Bagging
4	$\mathcal{DN}$-AdaBoost (W)	12	AdaBoost (S)
5	$\mathcal{DN}$-Bagging	14	AdaBoost (W)
8	$\mathcal{DN}$-Subspaces (75 %)	18	Subspaces (75 %)
16	$\mathcal{DN}$-Subspaces (50 %)	19	Subspaces (50 %)

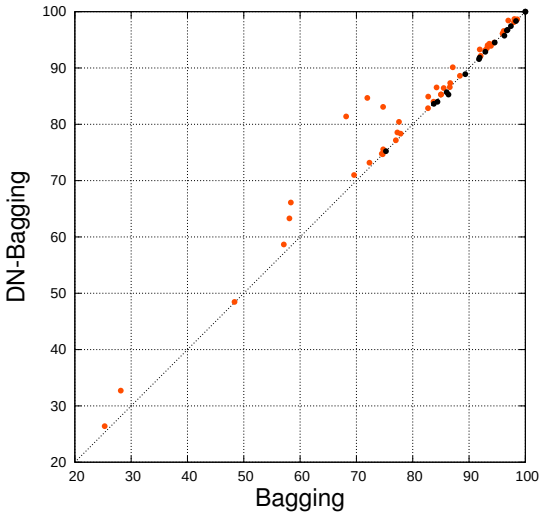


Figura 4.5: Bagging vs. $\mathcal{DN}$-Bagging.

el caso de Bagging [2]. Cada eje representa la precisión de cada método. Cada punto de color claro sobre la diagonal representa un conjunto de datos en el que $\mathcal{DN}$-Baging es mejor que Bagging.

Las tablas 4.5 y 4.7 muestran que el orden relativo entre las versiones de multiclasificador con $\mathcal{DN}$ es muy similar al de las versiones sin $\mathcal{DN}$, cualquiera que sea el ranking considerado. La mayoría de las posiciones relativas coinciden, y cuando excepcionalmente no es así, el descuadre es por una o dos posiciones. Esto permite pensar en $\mathcal{DN}$ como en una mejora de un método ya existente, y que el peso de la tarea de clasificación la tiene el método de partida. Aunque por otro lado, $\mathcal{DN}$-Ensemble ocupa el cuarto puesto en el ranking promedio, y el séptimo en el ranking de diferencias entre victorias y derrotas significativas, luego parece también que con únicamente hacer que los clasificadores base sean $\mathcal{DN}$-SVM es suficiente como para obtener multiclasificadores de SVM bastante competitivos.

Según el *Sign test* [26] (ver sección 2.3.1), para 62 conjuntos de datos, un método es mejor que otro con un nivel de significación del 5 %, si el número de victorias más la mitad del número de empates es mayor o igual que 39. Por tanto, a la vista de la segunda columna de la de la tabla 4.8, en todos los multiclasificadores la mejora de usar $\mathcal{DN}$ respecto a no usarlo es significativa.

La tercera columna de la misma tabla permite ver las victorias, empates y derrotas según la versión corregida del *Resampled t-test*. Estas victorias significativas fueron ya marcadas con $\circ$ en las tablas de la A.2 a la A.5 del anexo, mientras que las derrotas se marcaron con $\bullet$. Puede verse que en todos los casos las victorias superan a las derrotas, y que un multiclasificador con $\mathcal{DN}$ no pierde

[2] Se ha elegido Bagging por ser el multiclasificador sin $\mathcal{DN}$ mejor posiciando en el ranking promedio.

Tabla 4.8: Comparación de los métodos basados en SVM con y sin $\mathcal{DN}$ (V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
Bagging	44-5-14	17-45-0
Subspaces (50 %)	48-4-11	19-43-0
Subspaces (75 %)	46-6-11	17-45-0
AdaBoost (W)	41-3-19	17-45-0
AdaBoost (S)	39-5-19	18-44-0
MultiBoost (W)	42-5-16	18-44-0
MultiBoost (S)	47-3-13	19-43-0
SMO	22-4-37	5-56-1

de forma significativa en ninguno de los conjuntos de datos considerados. Por ello, nuevamente se puede concluir que todos los métodos mejoran con $\mathcal{DN}$.

Esta tabla 4.8 también permite analizar el comportamiento de $\mathcal{DN}$-SMO en solitario, el cual no solo no mejora frente a SMO sino que acumula más derrotas que victorias, aunque si sólo tenemos en cuenta las victorias y derrotas significativas, $\mathcal{DN}$-SMO vence en 5 ocasiones a SMO y sólo pierde 1. La conclusión es que no son métodos base muy distintos en cuanto a precisión.

La segunda columna de la tabla 4.9 permite ver que según el mismo *Sign test* todos los multiclasificadores son significativamente mejores que 1-NN, con la excepción de Random Subspaces 50 % y 75 %. Este último a falta de una sola victoria para ser significativamente mejor que 1-NN. Por lo que tampoco parece que los multiclasificadores con $\mathcal{DN}$ mejoren a los que no son $\mathcal{DN}$ por incorporar en su interior un pequeño clasificador 1-NN.

Random Subspaces 50 % es un método que está prácticamente empatado en cuanto a precisión con 1-NN, de hecho en la columna de la derecha de la tabla 4.8 puede verse que sólo gana significativamente a 1-NN 17 veces, mientras que es derrotado 18 veces. Este resultado no contradice demasiado la conclusión de que la fuerza de los multiclasificadores con $\mathcal{DN}$ no procede de su pequeño clasificador 1-NN, pues Random Subspaces 50 % es el multiclasificador más débil de todos, y en el que los clasificadores base pierden más información.

Por tanto, si (i) los multiclasificadores con $\mathcal{DN}$ son mejores que los que no utilizan $\mathcal{DN}$ (ii) la mejora no proviene de una mejora del clasificador base ($\mathcal{DN}$-SMO vs. SMO), y (iii) la mejora tampoco no proviene del aumento de precisión que pudiera aportar el pequeño clasificador 1-NN que incorporan las versiones con $\mathcal{DN}$, parece entonces que el origen de la mejora de los multiclasificadores sólo puede estar en el aumento de la diversidad que proveen los $\mathcal{DN}$-SMO combinado con que no pierden demasiada precisión frente a los clasificadores base SMO puros.

Tabla 4.9: Comparativa de los multiclasificadores que usan $\mathcal{DN}$ contra 1-NN (V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
$\mathcal{DN}$-Ensemble	39-3-21	24-24-14
$\mathcal{DN}$-Bagging	38-3-22	22-26-14
$\mathcal{DN}$-Subspaces (50 %)	32-3-28	17-27-18
$\mathcal{DN}$-Subspaces (75 %)	36-4-23	23-25-14
$\mathcal{DN}$-AdaBoost (W)	40-3-20	18-31-13
$\mathcal{DN}$-AdaBoost (S)	39-2-22	16-33-13
$\mathcal{DN}$-MultiBoost (W)	41-3-19	25-24-13
$\mathcal{DN}$-MultiBoost (S)	41-3-19	25-24-13
$\mathcal{DN}$-SMO	36-3-24	17-28-17

Tabla 4.10: Comparativa de los métodos que usan $\mathcal{DN}$ con el clasificador k-NN (V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
$\mathcal{DN}$-Ensemble	29-3-31	11-37-14
$\mathcal{DN}$-Bagging	31-3-29	11-37-14
$\mathcal{DN}$-Subspaces (50 %)	26-3-34	9-31-22
$\mathcal{DN}$-Subspaces (75 %)	28-3-32	12-34-16
$\mathcal{DN}$-AdaBoost (W)	26-3-34	10-38-14
$\mathcal{DN}$-AdaBoost (S)	24-2-37	11-38-13
$\mathcal{DN}$-MultiBoost (W)	33-3-27	14-35-13
$\mathcal{DN}$-MultiBoost (S)	34-2-27	14-37-11
$\mathcal{DN}$-SMO	26-3-34	7-35-20

Finalmente, el método k-NN ocupa la sexta posición en ambos rankings (tablas 4.4 y 4.6). Un punto importante en contra de los multiclasificadores con $\mathcal{DN}$-SVM es que si se comparan con k-NN, este último resulta ser un método más ligero de entrenar y bastante competitivo [3]. La tabla 4.10 incide en este hecho, pues según el *Sign test*, no hay ningún multiclasificador con $\mathcal{DN}$ que sea significativamente mejor que k-NN, incluso $\mathcal{DN}$-AdaBoost (S) está a punto de ser peor significativamente; y la mayoría de los métodos con $\mathcal{DN}$ sufren más derrotas significativas que victorias al ser comparados con k-NN. Por tanto, aunque el aumento de la diversidad ha conseguido la difícil tarea de construir multiclasificadores diversos de SVM, no ha sido capaz de hacerlo de forma que mejore otras opciones computacionalmente menos costosas.

4.3.1. Análisis de la diversidad en multiclasificadores con $\mathcal{DN}$-SVM

Una de las conclusiones del apartado anterior es que la mejora de los multiclasificadores con $\mathcal{DN}$-SVM no proviene de una mejora en la precisión de $\mathcal{DN}$-SVM respecto a SVM, sino en que $\mathcal{DN}$permite obtener unas SVM más diversas sin que esto haga mella considerable en la precisión de las mismas.

Para soportar esta afirmación también se ha testado la mejora en la diversidad de los métodos con $\mathcal{DN}$-SVM utilizando para ello la estadística Kappa [74] y los diagramas asociados a la misma que se presentaron en la sección 2.3.3. Kappa sirve para medir cuánto son de diversos dos clasificadores, pudiendo tomar valores entre -1 y 1. Un valor de κ igual a 1 quiere decir que ambos clasificadores están de acuerdo en todas las instancias, un valor igual a 0 significa que los clasificadores no están de acuerdo más allá de lo que por azar cabría esperar, mientras que los valores negativos de κ indican desacuerdo entre los clasificadores. Para poder obtener los diagramas Kappa-Error de los 62 conjuntos de datos se ha optado por hacer validación cruzada 5×2, menos costosa computacionalmente.

Los valores de Kappa pueden utilizarse para dibujar los diagramas Kappa-Error [74]. Las figuras 4.6 y 4.7 muestran dos ejemplos de estos diagramas para los métodos Bagging y Random Subspaces 75 % con el conjunto de datos *letter* del repositorio UCI. Para obtenerlos, se dibuja un punto (x, y) por cada par de clasificadores base pertenecientes a un mismo multiclasificador, de manera que x es la medida de κ para esos dos clasificadores, e y es el promedio del error de ambos. Por lo tanto, lo ideal es que cada par de clasificadores generara un punto lo más cercano posible a la esquina inferior izquierda, porque eso significaría que son precisos y a la vez diversos. En las figuras 4.6 y 4.7 se ve como las $\mathcal{DN}$-nubes están un poco desplazadas hacia la izquierda respecto a las nubes correspondientes a los multiclasificadores sin $\mathcal{DN}$, lo cual significa que los métodos con $\mathcal{DN}$ son más diversos.

En la figura 4.8 se muestran los *diagramas de Movimiento Kappa-Error* [79,

[3] No obstante k-NN es más costoso en cuanto al tiempo que necesita para llevar a cabo una predicción.

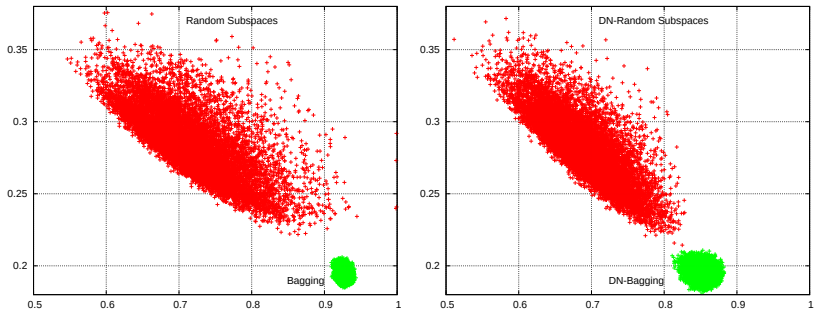


Figura 4.6: Error vs. Kappa para Bagging y Subspaces(75 %) en el conjunto de datos *letter*. Vista separada.

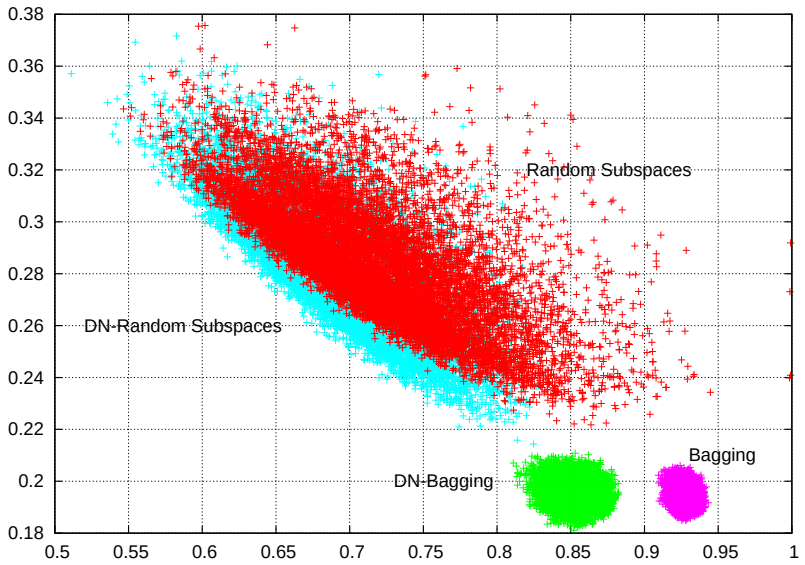


Figura 4.7: Error vs. Kappa para Bagging y Subspaces(75 %) en el conjunto de datos *letter*. Vista conjunta.

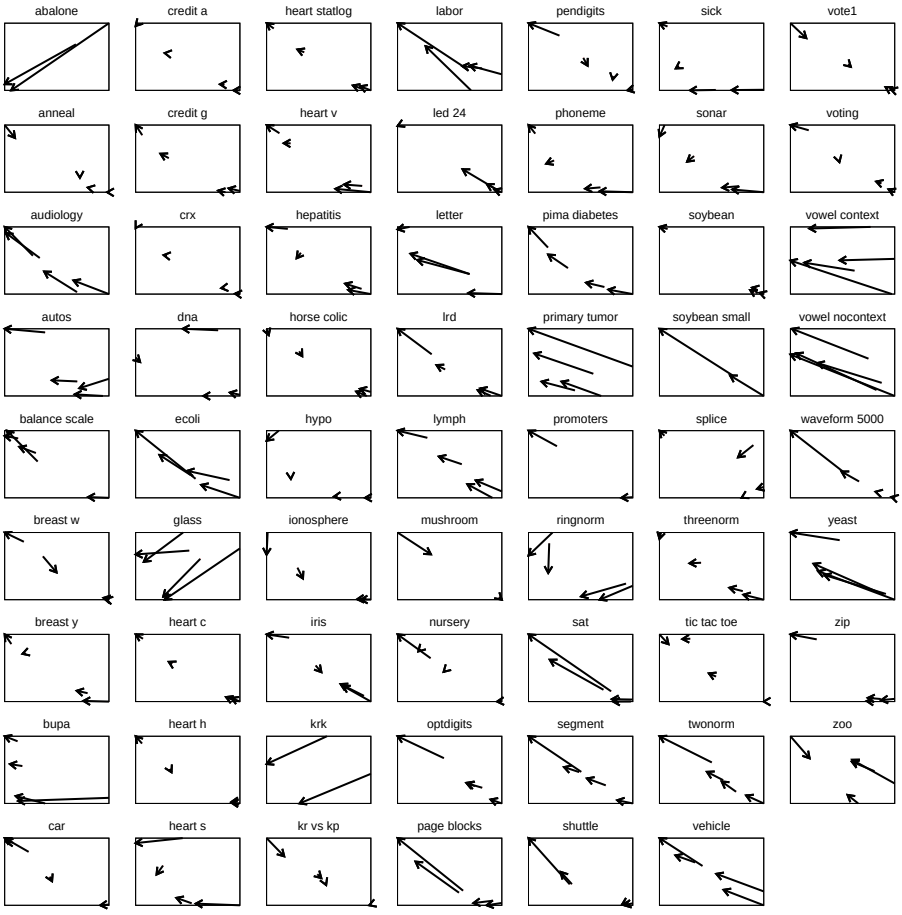


Figura 4.8: Diagramas de movimiento κ-Error para $\mathcal{DN}$ con SVM en los 62 conjuntos de datos.

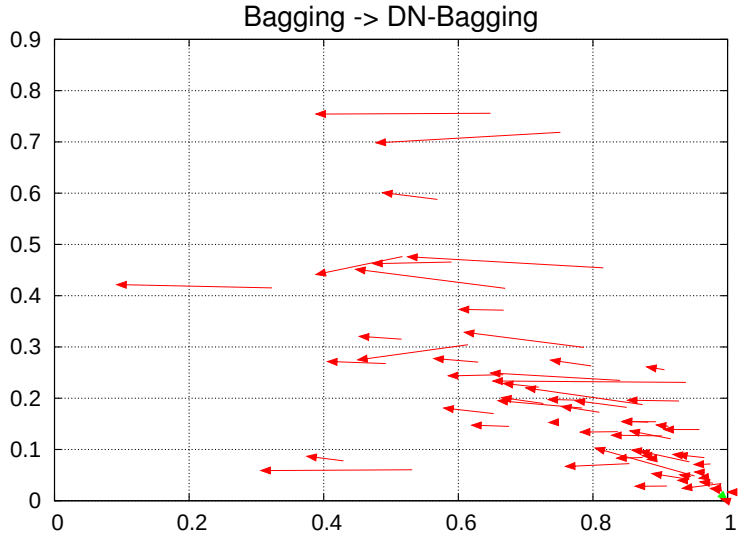


Figura 4.9: Diagrama de Movimiento κ-Error para Bagging de SVM.

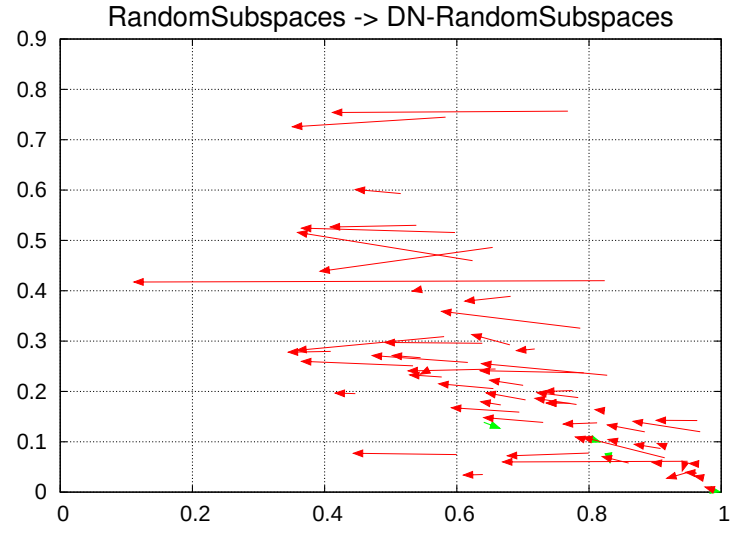


Figura 4.10: Diagrama de Movimiento κ-Error para Subspaces (75 %) de SVM.

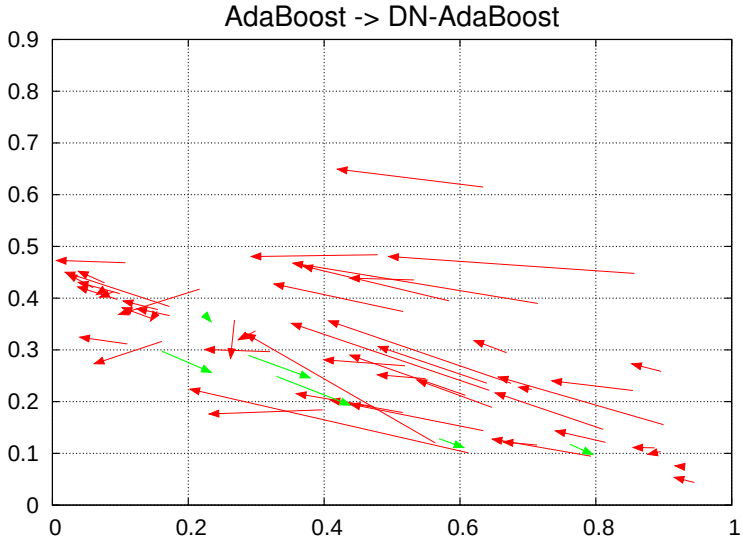


Figura 4.11: Diagrama de Movimiento κ-Error para AdaBoost(S) de SVM.

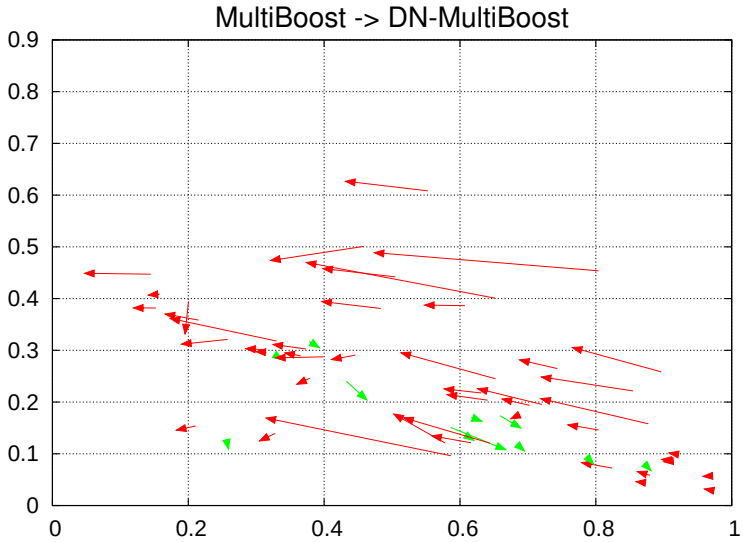


Figura 4.12: Diagrama de Movimiento κ-Error para MultiBoost(S) de SVM.

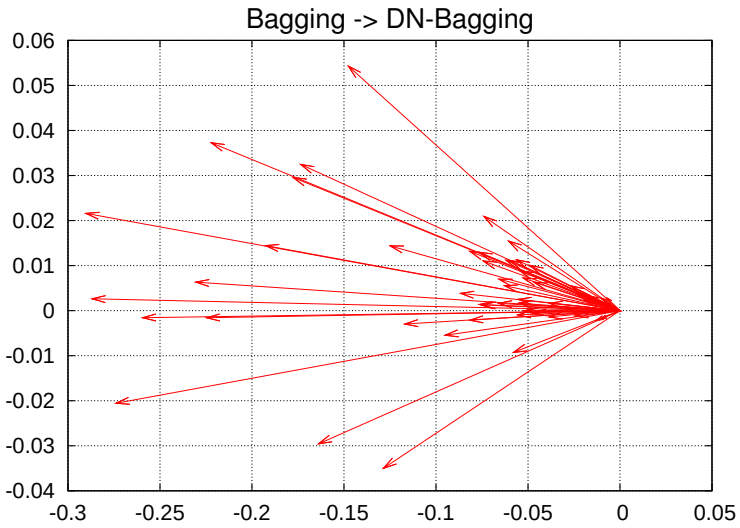


Figura 4.13: Diagrama de Movimiento Relativo de κ-Error para Bagging de SVM.

80, 81] presentados en la sección 2.3.3. Estos diagramas unen con flechas los centros de las nubes de los diagramas Kappa-Error. Estos centros se calculan como el valor promedio de las coordenadas κ y error.

Los métodos que se han considerado en los diagramas de Movimiento Kappa-Error son Bagging, Random Subspaces 75 %, y las dos variantes de Boosting con remuestreo. Por ello, cada rectángulo normalmente tiene 4 flechas correspondientes a estos cuatro métodos. Las subfiguras en las que parece que hay menos de 4 flechas es porque o hay varias solapadas, o efectivamente falta alguna porque coinciden los centros de las dos nubes. En esta figura 4.8 se aprecia que la mayoría de flechas apuntan hacia la izquierda, indicando una mejora de la diversidad, aunque también hay bastantes que apuntan hacia arriba, indicando que esa mejora en la diversidad de los clasificadores base, es a costa de aumentar su imprecisión.

Para poder ver mejor este efecto se han agrupado todas las flechas correspondientes a un mismo método en un sólo diagrama, de donde surgen los cuatro diagramas de movimiento Kappa-Error, uno por cada método considerado (figuras de la 4.9 a la 4.12). Nuevamente, se aprecia que casi todas las flechas apuntan a la izquierda indicando el aumento de diversidad, cuanto más larga es la flecha mayor es este incremento.

Finalmente, las figuras de la 4.13 a la 4.16 muestran los *diagramas de Movimiento Relativo Kappa Error* [79, 80, 81] para cada uno de los multiclasificadores (ver sección 2.3.3). Estos diagramas se obtienen al juntar todas las flechas correspondientes a un mismo método de la figura 4.8, trasladando su punto de partida al origen de coordenadas. Nuevamente, la gran mayoría de las flechas apuntan a la izquierda, indicando el aumento de la diversidad. También se ve

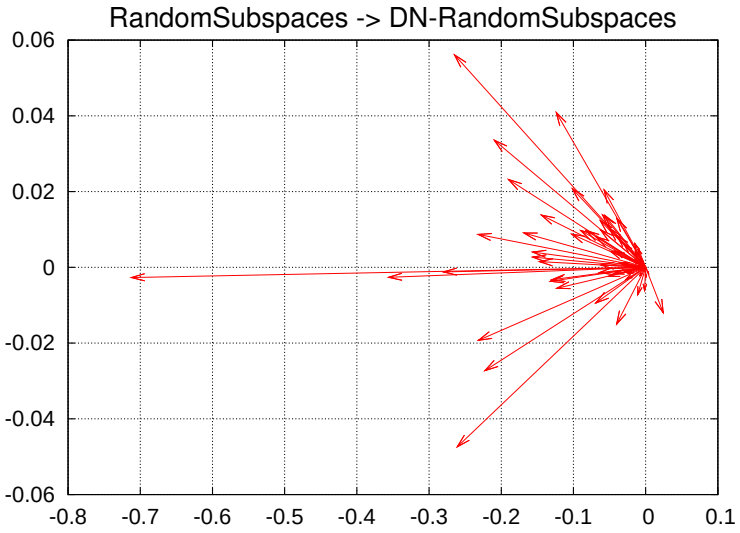


Figura 4.14: Diagrama de Movimiento Relativo de κ-Error para Subspaces (75 %) de SVM.

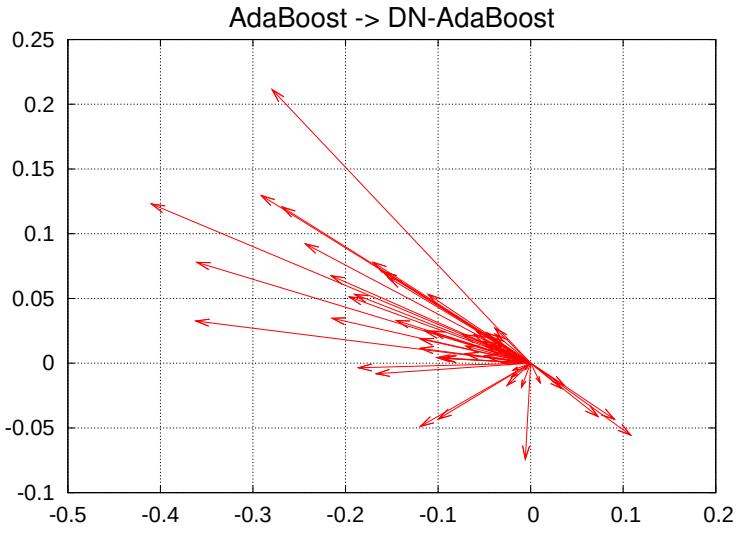


Figura 4.15: Diagrama de Movimiento Relativo de κ-Error para AdaBoost(S) de SVM.

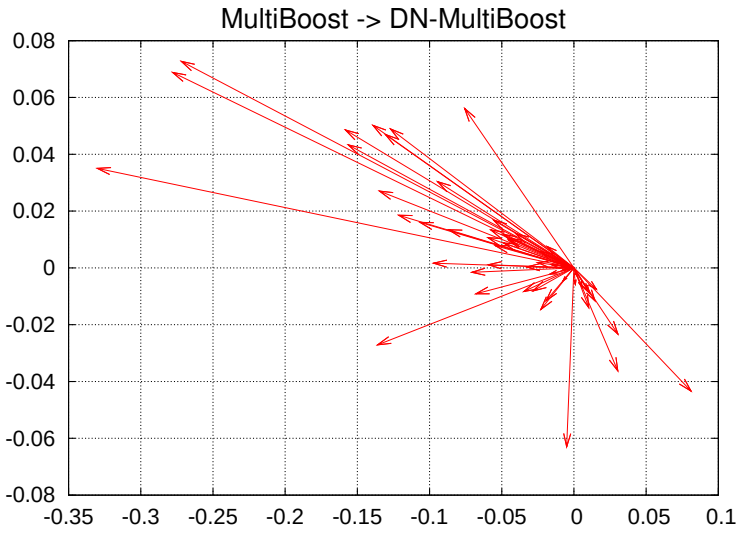


Figura 4.16: Diagrama de Movimiento Relativo de κ-Error para MultiBoost(S) de SVM.

como bastantes flechas apuntan hacia arriba, dejando entrever que el aumento de diversidad es a costa de la pérdida de acierto de los clasificadores base.

4.4. Resultados de $\mathcal{DN}$ con árboles

En el apartado anterior $\mathcal{DN}$ fue probado con clasificadores base SVM. En el presente apartado se hará un análisis paralelo al anterior pero utilizando como clasificadores base árboles de decisión.

Nuevamente, la validación experimental ha sido hecha con WEKA [117]. Se ha utilizado la implementación de $\mathcal{DN}$ para WEKA utilizada en el apartado anterior, en la que el número de vecinos ha sido siempre $m = 10$. Los parámetros de cada uno de los métodos son siempre los parámetros que trae por defecto WEKA, salvo que se indique lo contrario. En la validación se han testado los siguiente métodos:

1. Bagging [7].

2. Random Forests [11].

3. Boosting: Se ha utilizado tanto Adaboost [41] como Multiboost [116]. En el caso de Multiboost el tamaño de los subcomités es cinco. En ambas versiones de Boosting se ha probado tanto con la variante de *remuestreo* (*reSampling*), como con la de *repesado* (*reWeighting*), que se han denotado respectivamente como (S) y (W) en las tablas.

4. Random Subspaces [55]. Se han testado dos configuraciones, tomando respectivamente el 50 % y el 75 % de las dimensiones del problema original.

En todos los multiclasificadores se han construido cincuenta clasificadores base. Los clasificadores base utilizados han sido:

- En el caso de Boosting, Bagging y Random Subspaces, se han utilizado clasificadores base J.48 (la implementación de los árboles de decisión C4.5 de Quinlan [92] que hace WEKA). Se han hecho pruebas tanto con árboles J.48 sin usar $\mathcal{DN}$, como usando $\mathcal{DN}$ para perturbarlos ($\mathcal{DN}$-Decision Trees).

- Para el caso de Random Forests, lógicamente, sólo cabe utilizar Random Trees como clasificadores base. También se han probado estos multiclasificadores tanto con Random Trees sin $\mathcal{DN}$, como con su variante perturbada por $\mathcal{DN}$ ($\mathcal{DN}$-Random Trees).

Como en el caso de las SVM, también interesa probar en la validación, si los $\mathcal{DN}$-Decision Trees son ya de por sí, clasificadores base suficientemente potentes como para obtener buenas tasas de acierto sin tener que combinarlos con ninguno de los sofisticados esquemas que se han enumerado anteriormente. Por ello, se ha añadido a la prueba el multiclasificador formado por cincuenta $\mathcal{DN}$-Decision Trees, de manera que la predicción de dicho multiclasificador se computa simplemente haciendo la media de las probabilidades de las predicciones de cada uno de los clasificadores base. Tal y como se hizo en la sección 4.3, se dará el nombre de $\mathcal{DN}$-Ensemble a dicho método.

De manera análoga a como se hizo en la sección 4.3, en la que se analizaban los resultados para $\mathcal{DN}$-SVM, se ha incluido el método k-NN en el test, tanto en su versión de un sólo vecino (1-NN), como la versión k-NN de WEKA que da a k el valor óptimo para cada conjunto, para así conocer si la tasa de acierto que pueda tener un clasificador k-NN es en el fondo la causa por la que pudiera mejorar, en su caso, el rendimiento de los multiclasificadores con $\mathcal{DN}$. Como en la sección 4.3, tampoco se han añadido multiclasificadores de vecinos más cercanos al test por las mismas razones por las que no se hizo con SVM.

Se han utilizado los mismos 62 conjuntos de datos de la UCI [3] de la sección 4.3 para $\mathcal{DN}$-SVM, (ver tabla 4.3), y validación cruzada estratificada 10×10. Los resultados para cada método y conjunto de datos se muestran en las tablas de la A.6 a la A.10.

La tabla 4.11 muestra los distintos multiclasificadores ordenados por los rankings promedios según [26] (ver sección 2.3.2). Puede verse que todos los multiclasificadores que utilizan $\mathcal{DN}$ mejoran su versión sin $\mathcal{DN}$.

También se ha aplicado la versión corregida del *Resampled t-test* [85] (ver sección 2.3.1) sobre los resultados de la validación cruzada 10×10. Como en ocasiones anteriores, el nivel de significación elegido es del 5 %. El test se ha utilizado para confeccionar el ranking de victorias menos derrotas siginificativas descrito en la sección 2.3.2 (ver tabla 4.13). La conclusión es la misma que en el ranking anterior:

Tabla 4.11: Ranking promedio de la validación experimental de $\mathcal{DN}$ con clasificadores base árboles.

Ranking Promedio	Método
7.19	$\mathcal{DN}$-MultiBoost (W)
7.47	$\mathcal{DN}$-Random Forest
7.48	$\mathcal{DN}$-MultiBoost (S)
8.02	$\mathcal{DN}$-AdaBoost (S)
8.42	$\mathcal{DN}$-AdaBoost (W)
8.65	MultiBoost (S)
8.77	MultiBoost (W)
8.89	$\mathcal{DN}$-Subspaces (50 %)
9.30	Random Forest
9.71	$\mathcal{DN}$-Subspaces (75 %)
9.78	AdaBoost (S)
9.87	AdaBoost (W)
10.01	$\mathcal{DN}$-Bagging
11.37	Subspaces (50 %)
11.61	k-Nearest Neighbor
12.18	Bagging
13.30	Subspaces (75 %)
13.40	$\mathcal{DN}$-Ensemble
14.59	1-Nearest Neighbor

Tabla 4.12: Posiciones de los multiclasificadores con $\mathcal{DN}$-árboles vs. árboles puros en el ranking promedio.

Posición	Método	Posición	Método
1	$\mathcal{DN}$-MultiBoost (W)	6	MultiBoost (S)
2	$\mathcal{DN}$-Random Forest	7	MultiBoost (W)
3	$\mathcal{DN}$-MultiBoost (S)	9	Random Forest
4	$\mathcal{DN}$-AdaBoost (S)	11	AdaBoost (S)
5	$\mathcal{DN}$-AdaBoost (W)	12	AdaBoost (W)
8	$\mathcal{DN}$-Subspaces (50 %)	14	Subspaces (50 %)
10	$\mathcal{DN}$-Subspaces (75 %)	15	Bagging
13	$\mathcal{DN}$-Bagging	16	Subspaces (75 %)

Tabla 4.13: Ranking de diferencias entre victorias y derrotas significativas de la validación experimental de $\mathcal{DN}$ con clasificadores base árboles (V−D: Victorias−Derrotas, V: Victorias, D: Derrotas).

V−D	V	D	Método
208	246	38	$\mathcal{DN}$-MultiBoost (W)
205	262	57	$\mathcal{DN}$-AdaBoost (W)
203	240	37	$\mathcal{DN}$-MultiBoost (S)
201	250	49	$\mathcal{DN}$-AdaBoost (S)
171	232	61	MultiBoost (W)
170	231	61	MultiBoost (S)
159	229	70	AdaBoost (S)
144	230	86	AdaBoost (W)
100	208	108	$\mathcal{DN}$-Random Forest
80	193	113	Random Forest
-33	165	198	$\mathcal{DN}$-Subspaces (50 %)
-67	153	220	$\mathcal{DN}$-Bagging
-76	145	221	$\mathcal{DN}$-Subspaces (75 %)
-111	136	247	Subspaces (50 %)
-155	109	264	Bagging
-220	146	366	k-Nearest Neighbor
-260	85	345	Subspaces (75 %)
-280	93	373	$\mathcal{DN}$-Ensemble
-439	113	552	1-Nearest Neighbor

Tabla 4.14: Posiciones de los multiclasificadores con $\mathcal{DN}$-árboles vs. árboles puros en el Ranking de diferencias entre victorias y derrotas significativas.

Posición	Método	Posición	Método
1	$\mathcal{DN}$-MultiBoost (W)	5	MultiBoost (W)
2	$\mathcal{DN}$-AdaBoost (W)	6	MultiBoost (S)
3	$\mathcal{DN}$-MultiBoost (S)	7	AdaBoost (S)
4	$\mathcal{DN}$-AdaBoost (S)	8	AdaBoost (W)
9	$\mathcal{DN}$-Random Forest	10	Random Forest
11	$\mathcal{DN}$-Subspaces (50 %)	14	Subspaces (50 %)
12	$\mathcal{DN}$-Bagging	15	Bagging
13	$\mathcal{DN}$-Subspaces (75 %)	17	Subspaces (75 %)

Todos los multiclasificadores que utilizan $\mathcal{DN}$ mejoran la versión de partida que no utiliza $\mathcal{DN}$.

Las tablas 4.12 y 4.14 sirven para ilustrar como en ambos rankings el orden relativo entre los métodos sin y con $\mathcal{DN}$ apenas varía. Si tomamos una fila cualquiera de esas tablas, se puede ver que la versión con $\mathcal{DN}$ en la mayoría de las ocasiones está en la misma fila que la versión sin $\mathcal{DN}$; y cuando no es así ambas versiones distan una fila o muy excepcionalmente dos. Esto indica que la mejora en el uso de $\mathcal{DN}$ muestra cierta independencia respecto al esquema de combinación en el que se utilice. Por ello, parece lógico pensar en Disturbing Neighbors como una *mejora de un método ensemble existente*. Esta hipótesis es también sustentada por las posiciones tan bajas que muestra $\mathcal{DN}$-Ensemble en las tablas 4.11 y 4.13. Lo cual, ilustra que (a diferencia de las $\mathcal{DN}$-SVM) la mera utilización de un multiclasificador con clasificador base $\mathcal{DN}$ sin tener en cuenta cómo se van a combinar éstos, no garantiza el mejor resultado posible.

La mejora en el utilización de las versiones $\mathcal{DN}$ de los multiclasificadores se cuantifica en la tabla 4.15.

- La segunda columna muestra las victorias, empates y derrotas de las versiones con $\mathcal{DN}$ de los multiclasificadores, frente a las versiones sin $\mathcal{DN}$. Según el *Sign test* [26] (ver sección 2.3.1), para 62 conjuntos de datos, un método es significativamente mejor que otro si el número de victorias más la división entre dos del número de empates es mayor o igual que 39. Por lo tanto, para todos los métodos de la tabla 4.15 las versiones con $\mathcal{DN}$ son significativamente mejores.

- La tercera columna muestra las victorias, empates y derrotas de las versiones con $\mathcal{DN}$ frente a las versiones sin $\mathcal{DN}$ que sean significativas según la versión corregida del *Resampled t-test*. Estas victorias significativas fueron ya marcadas con $\circ$ en las tablas de la A.7 a la A.10 del anexo, mientras que las derrotas se marcaron con $\bullet$. Puede verse que en todos los casos las victorias superan a las derrotas, y que sólo en el caso de los Random Forest hay alguna derrota significativa, pero que en el resto no. Por ello, nuevamente se podría concluir que todos los métodos mejoran con $\mathcal{DN}$.

  Existen algunos métodos en el que el número de victorias significativas es más grande, como por ejemplo Bagging y Subspaces(75 %), pero en el resto de métodos el número de victorias es reducido. Esto es posible que se deba precisamente a que al tratarse de una mejora de un método existente, el papel protagonista en la tarea de clasificación lo sigue ejerciendo el método multiclasificador de partida.

Las tablas 4.11 y 4.13 muestran que las posiciones en el ranking de los métodos 1-NN y k-NN no son muy buenas. Por lo que podría pensarse que la causa por la que los multiclasificadores con $\mathcal{DN}$ son mejores no es precisamente tanto por la bondad del algoritmo k-NN, sino por la diversidad que induce la selección aleatoria de las m instancias que forman parte de cada 1-NN.

Tabla 4.15: Comparación de los métodos basados en árboles con y sin $\mathcal{DN}$(V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
Bagging	48-3-11	11-51-0
Subspaces (50 %)	47-5-10	3-59-0
Subspaces (75 %)	53-3-6	16-46-0
AdaBoost (W)	43-2-17	3-59-0
AdaBoost (S)	45-5-12	2-60-0
MultiBoost (W)	43-2-17	1-61-0
MultiBoost (S)	40-4-18	1-61-0
Random Forest	40-2-20	3-57-2

Las tablas 4.16 y 4.17 sirven para comprobar esta hipótesis al comparar cada método con $\mathcal{DN}$ respectivamente contra los métodos 1-NN y k-NN.

Nuevamente, puede observarse que 1-NN es significativamente peor que todos los métodos según el *Sign test* [26] (columna central de la tabla 4.16), obteniéndose en todos los casos un número de victorias significativas muy superior al de derrotas (columna derecha de la tabla 4.16), aun cuando 1-NN utiliza todas las instancias de cada conjunto de datos, frente a las $m = 10$ que usan los algoritmos $\mathcal{DN}$ testados.

Es más, incluso optimizando el número k del método k-NN para cada conjunto de datos no es suficiente para derrotar significativamente a ninguno de los métodos $\mathcal{DN}$. La tabla 4.17 muestra esta comparación contra k-NN con un k óptimo. La columna central muestra que tan solo $\mathcal{DN}$-Bagging, $\mathcal{DN}$-Subspaces(75 %) y $\mathcal{DN}$-Ensemble no consiguen ser mejores significativamente usando el *Sign test*. En el caso de los dos primeros no se alcanza el umbral de las 39 victorias por muy poco.

No obstante, $\mathcal{DN}$-Ensemble es un método que se ha introducido en el test tan sólo para descartar que el uso de un clasificador base con $\mathcal{DN}$ sin tener en cuenta el esquema de combinación, no garantizaba un resultado satisfactorio. Además la columna derecha de la tabla 4.17 muestra también que en cualquier caso el número de victorias significativas sobre k-NN siempre supera al de derrotas, y que salvo en los tres métodos anteriormente mencionados esta diferencia es muy grande (las victorias quintuplican las derrotas).

Es importante notar en las tablas 4.11, 4.13 y 4.17, que a diferencia de los resultados con $\mathcal{DN}$-SVM, el optimizar el valor de k en el método k-NN no genera un clasificador que esté a la altura de un multiclasificador que use $\mathcal{DN}$-árboles.

Tabla 4.16: Comparativa de los multiclasificadores de árboles que usan $\mathcal{DN}$ contra 1-NN (V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
$\mathcal{DN}$-Ensemble	41-1-20	25-27-10
$\mathcal{DN}$-Bagging	46-1-15	30-23-9
$\mathcal{DN}$-Subspaces (50 %)	45-1-16	29-23-10
$\mathcal{DN}$-Subspaces (75 %)	44-1-17	29-26-7
$\mathcal{DN}$-AdaBoost (W)	51-0-11	34-25-3
$\mathcal{DN}$-AdaBoost (S)	51-1-10	33-26-3
$\mathcal{DN}$-MultiBoost (W)	49-0-13	37-20-5
$\mathcal{DN}$-MultiBoost (S)	50-0-12	36-21-5
$\mathcal{DN}$-Random Forest	50-2-10	37-20-5

Tabla 4.17: Comparativa de los multiclasificadores de árboles que usan $\mathcal{DN}$ contra k-NN (V-E-D: Victorias-Empates-Derrotas).

Método	V-E-D	V-E-D Significativas
$\mathcal{DN}$-Ensemble	29-1-32	14-36-12
$\mathcal{DN}$-Bagging	37-1-24	17-36-9
$\mathcal{DN}$-Subspaces (50 %)	40-1-21	16-34-12
$\mathcal{DN}$-Subspaces (75 %)	38-1-23	17-37-8
$\mathcal{DN}$-AdaBoost (W)	40-0-22	25-32-5
$\mathcal{DN}$-AdaBoost (S)	39-1-22	25-33-4
$\mathcal{DN}$-MultiBoost (W)	41-0-21	27-30-5
$\mathcal{DN}$-MultiBoost (S)	42-0-20	27-30-5
$\mathcal{DN}$-Random Forest	42-2-18	26-31-5

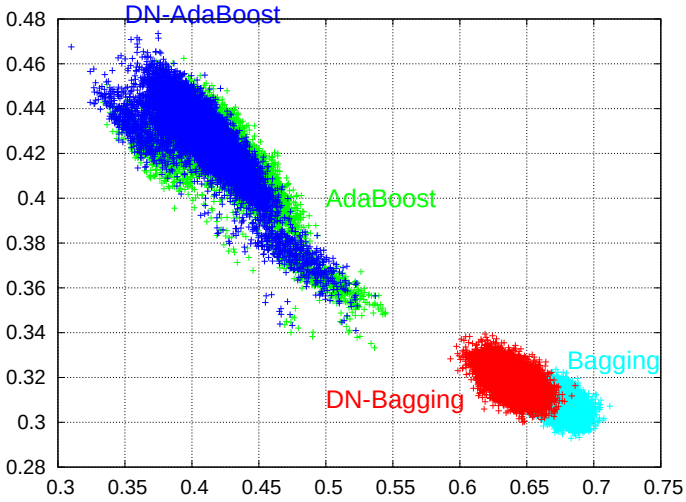


Figura 4.17: Error vs. Kappa para Boosting y Bagging en el conjunto de datos *krk*.

4.4.1. Análisis de la diversidad en multiclasificadores con $\mathcal{DN}$-árboles

Como en la sección 4.3, también se ha comprobado la mejora de la diversidad de los $\mathcal{DN}$-árboles mediante la estadística Kappa [74]. Los valores de Kappa se han utilizado para dibujar los diagramas de Kappa-Error [74]. La figura 4.17 muestra un ejemplo de diagrama Kappa-Error para el conjunto de datos *krk* con los métodos Bagging y AdaBoost.

En la figura 4.17 pueden observarse que las nubes correspondientes a los métodos con $\mathcal{DN}$ están algo desplazadas a la izquierda de las nubes correspondientes a los métodos sin $\mathcal{DN}$. Esto significa que los métodos con $\mathcal{DN}$ son más diversos. La figura 4.18 muestra el *diagrama de Movimiento Kappa-Error* para los métodos Bagging, Adaboost con remuestreo, Multiboost con remuestreo, Random Forests y la variante de Random Subspaces con el 50 % de los atributos. Los diagramas se han obtenido de idéntica manera que en la sección 4.3. Pueden observarse gran cantidad de flechas apuntando hacia la izquierda, lo que significa una mejora generalizada de la diversidad. Cuanto más larga es la flecha, mayor es la diferencia relativa.

Finalmente, las figuras que van de la 4.19 a la 4.23 muestran los *diagramas de Movimiento Relativo de Kappa-Error* para cada uno de los multiclasificadores.

Estos diagramas son ideales para poder observar el efecto conjunto de los resultados de la figura 4.18. Se puede observar que la mayor parte de las flechas apuntan a la izquierda, lo cual es un indicador de diversidad. Muchas de las flechas además apuntan hacia arriba, poniendo de manifiesto que generalmente esa ganancia de diversidad es a costa de una pérdida en el acierto de cada uno

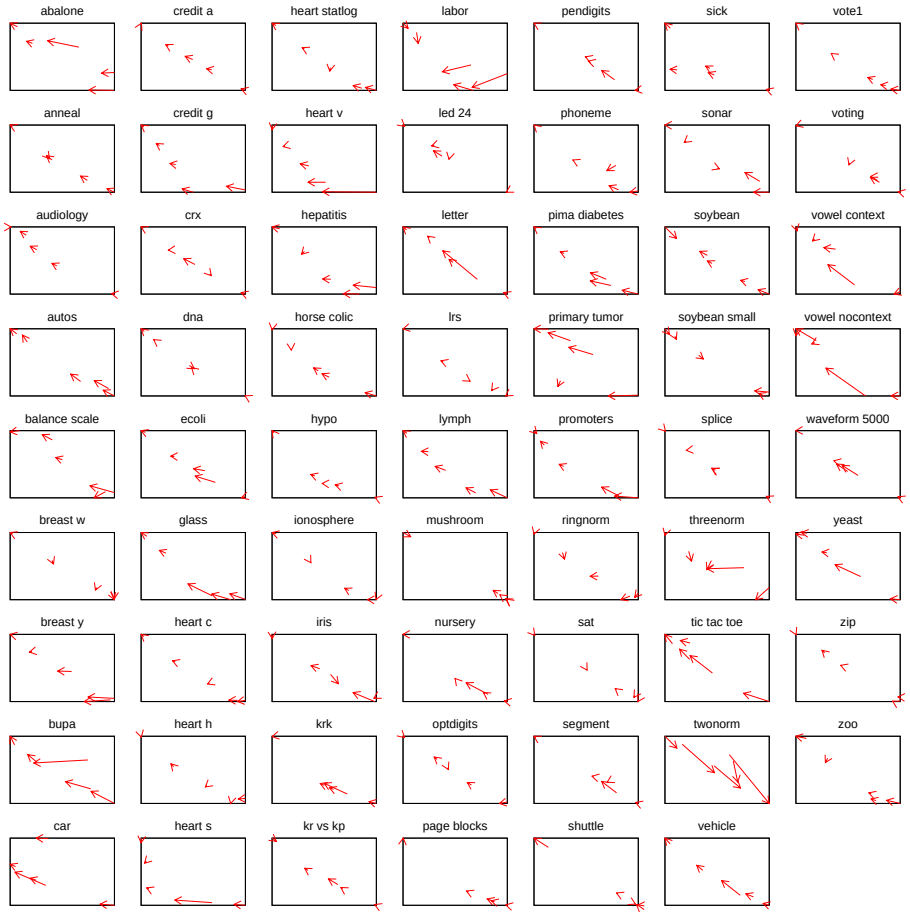


Figura 4.18: Diagramas de movimiento κ-Error para $\mathcal{DN}$ con árboles en los 62 conjuntos de datos.

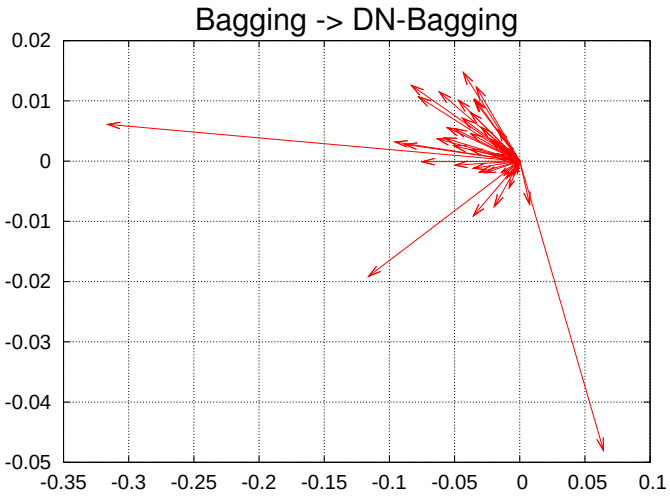


Figura 4.19: Diagrama de Movimiento Relativo de κ-Error para Bagging de árboles

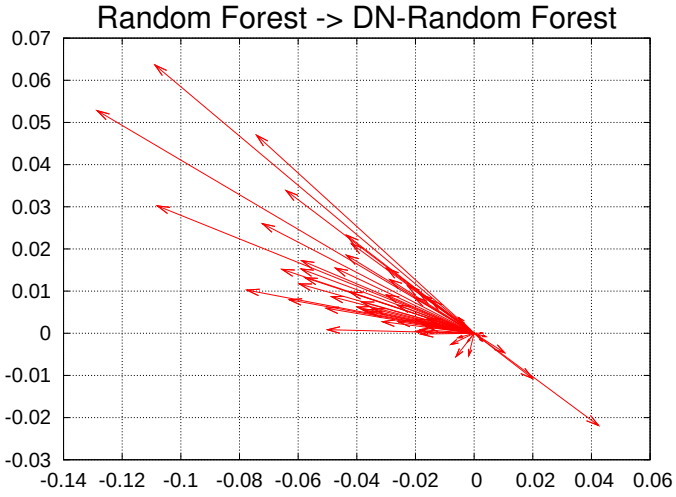


Figura 4.20: Diagrama de Movimiento Relativo de κ-Error para Random Forests

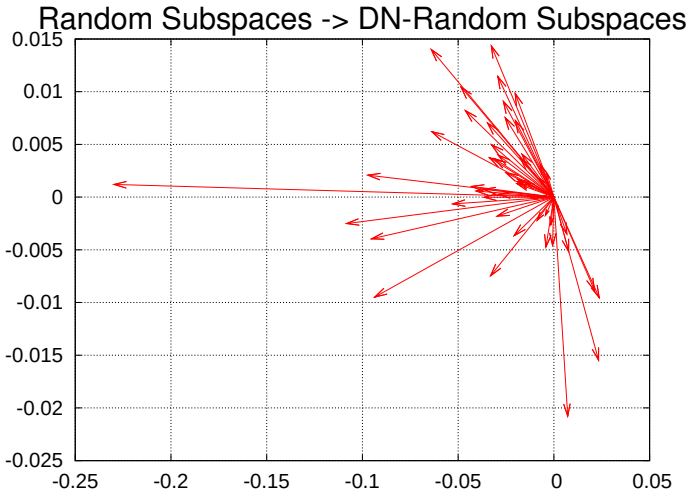


Figura 4.21: Diagrama de Movimiento Relativo de κ-Error para Random Subspaces (50 %) de árboles

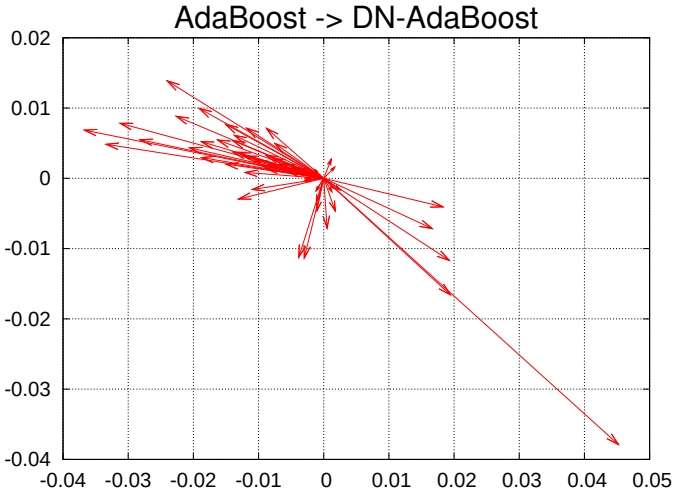


Figura 4.22: Diagrama de Movimiento Relativo de κ-Error para AdaBoost(S) de árboles

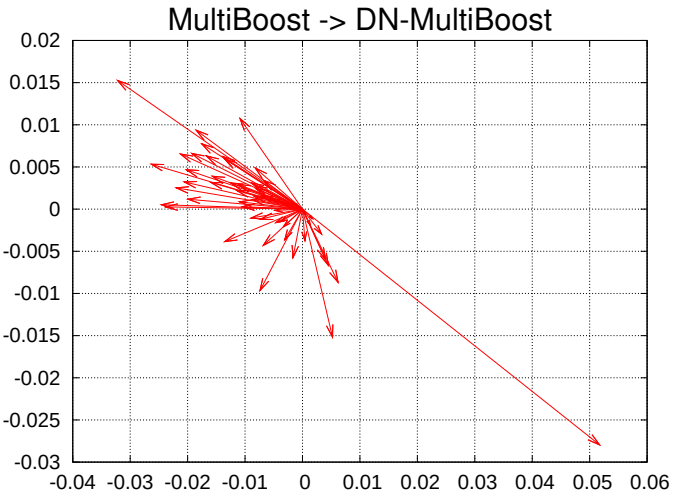


Figura 4.23: Diagrama de Movimiento Relativo de κ-Error para MultiBoost(S) de árboles

de los clasificadores base.

4.5. Estudio de lesiones

Existen tres elementos en la construcción de clasificadores base con $\mathcal{DN}$:

1. La aplicación de una selección aleatoria de características en base a las cuales se calculan las distancias del clasificador 1-NN.

2. La incorporación de m características binarias que indican cuál es el vecino más cercano.

3. La incorporación de otra característica adicional que toma el valor de la predicción del clasificador 1-NN.

Parece bastante interesante hacer un estudio experimental que desvele cuáles de estos tres elementos resultan esenciales a la hora de construir multiclasificadores con $\mathcal{DN}$, y cuáles de estos tres elementos son prescindibles.

Para ello, se han creado cinco variantes nuevas del método $\mathcal{DN}$:

- $\mathcal{DN}_P$: Esta variante no aplica ni la selección aleatoria de características, ni incorpora las m características binarias que indican cuál es el vecino más cercano. Tan solo toma en cuenta la predicción hecha por el clasificador 1-NN. Esta variante $\mathcal{DN}_P$ podría interpretarse como un tipo de Cascading [47] en el que el clasificador 1-NN juega el papel de clasificador del nivel 1 o base, mientras que el árbol de decisión toma el papel

de clasificador del nivel 2 o meta. La diferencia con un Cascading típico es que mientras este último utilizaría todo el conjunto de entrenamiento para entrenar el clasificador del nivel 1, $\mathcal{DN}$ tan sólo utiliza m instancias.

- $\mathcal{DN}_V$: Es una variante que tampoco aplica la selección aleatoria de características y que no tiene en cuenta la predicción de la clase hecha por el 1-NN. Tan solo utiliza los m atributos binarios que indican cuál es el vecino más cercano.

- $\mathcal{DN}_{VP}$: Es una variante que no aplica la selección aleatoria de características, pero sí que considera tanto la predicción hecha por el 1-NN, como los m atributos binarios.

- $\mathcal{DN}_{PA}$: Es una variante muy similar a $\mathcal{DN}_P$, ya que en ambas se tiene en cuenta la predicción del clasificador 1-NN. La diferencia entre ambas estriba en que $\mathcal{DN}_{PA}$ computa las distancias en el clasificador 1-NN utilizando la selección aleatoria de características.

- $\mathcal{DN}_{VA}$: Es una variante parecida a $\mathcal{DN}_V$ debido a que ambas utilizan las m características binarias. La diferencia entre ambas consiste también en que $\mathcal{DN}_{VA}$ computa las distancias del clasificador 1-NN utilizando la selección aleatoria de características.

Siguiendo esta misma nomenclatura, el propio método $\mathcal{DN}$ podría denotarse como $\mathcal{DN}_{VPA}$, ya que en él se aplican los tres elementos mencionados anteriormente (**V**: se utilizan las m características que indican cuál es el **V**ecino más cercano, **P**: se utiliza la **P**redicción de la clase hecha por el 1-NN, y **A**: se hace la selección **A**leatoria de características).

Los métodos multiclasificadores considerados en la sección 4.4 se han agrupado en las nueve familias que se muestran a continuación:

1. Bagging.
2. Random Forest.
3. Random Subspaces utilizando el 50 % de los atributos originales.
4. Random Subspaces utilizando el 75 % de los atributos originales.
5. AdaBoost en la variante de repesado.
6. AdaBoost en la variante de remuestreo.
7. MultiBoost en la variante de repesado.
8. MultiBoost en la variante de remuestreo.
9. $\mathcal{DN}$-Ensembles.

Tabla 4.18: Rankings para las distintas variantes de ($\mathcal{DN}$-)Bagging (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
2.84	$\mathcal{DN}_{VP}$-Bagging
3.28	$\mathcal{DN}$-Bagging
3.48	$\mathcal{DN}_{V}$-Bagging
3.89	$\mathcal{DN}_{VA}$-Bagging
3.94	$\mathcal{DN}_{P}$-Bagging
4.98	$\mathcal{DN}_{PA}$-Bagging
5.59	Bagging

V−D	V	D	Método
24	26	2	$\mathcal{DN}$-Bagging
23	25	2	$\mathcal{DN}_{VA}$-Bagging
14	18	4	$\mathcal{DN}_{VP}$-Bagging
12	16	4	$\mathcal{DN}_{V}$-Bagging
-8	7	15	$\mathcal{DN}_{P}$-Bagging
-17	6	23	$\mathcal{DN}_{PA}$-Bagging
-48	0	48	Bagging

Todas las familias a excepción de $\mathcal{DN}$-Ensembles, constan de siete versiones: el multiclasificador con clasificadores base árboles sin ningún tipo de $\mathcal{DN}$, el multiclasificador con $\mathcal{DN}$ (o si se quiere, con $\mathcal{DN}_{VPA}$), y las cinco variantes de $\mathcal{DN}$ descritas anteriormente. Excepcionalmente, la familia de los $\mathcal{DN}$-Ensembles sólo consta de séis métodos debido a que no existe la versión «sin» $\mathcal{DN}$.

Nuevamente, se han computado los rankings promedios con los mismos 62 conjuntos de datos de la UCI, y también con la validación cruzada estratificada 10×10, pero esta vez todos estos cálculos se han llevado a cabo para cada familia de forma separada.

Según [26], si se utiliza el test pareado de Nemenyi (ver sección 2.3.1), para 62 conjuntos de datos, se puede concluir que un clasificador es significativamente mejor que otro, con un nivel de significación del 5 %, si la diferencia entre sus rankings promedios es mayor que un valor crítico de 1.144 para el caso en el que sean siete los métodos comparados (tal y como ocurre en todas las familias a excepción de la familia $\mathcal{DN}$-Ensemble), o de 0.958 en el caso en el que sean seís métodos (como ocurre en la familia del $\mathcal{DN}$-Ensemble).

Los resultados de cada variante y conjunto de datos aparecen reflejados en las tablas de la A.11 a la A.19 del anexo.

Las tablas de la 4.18 a la 4.26 muestran los resultados para cada familia. Cada una de estas tablas tiene dos subtablas: la subtabla de la izquierda representa los rankings promedios y la de la derecha el ranking por diferencias entre victorias y derrotas significativas.

En los rankings promedios los métodos agrupados por las líneas verticales a la izquierda resultan no ser significativamente peores que el que está en primera posición, mientras que los métodos agrupados por las líneas verticales de la derecha no son significativamente mejores que el último.

Se observa que en todas las familias, cualquiera de las variantes de $\mathcal{DN}$ siempre quedan en el grupo de cabeza, mientras que el método puro (sin ningún tipo de $\mathcal{DN}$) siempre queda en el grupo de cola; a excepción de los $\mathcal{DN}$-Ensembles,

Tabla 4.19: Rankings para las distintas variantes de ($\mathcal{DN}$-)Random Forest (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
3.60	$\mathcal{DN}_V$-R Forest
3.60	$\mathcal{DN}$-R Forest
3.60	$\mathcal{DN}_{VA}$-R Forest
3.90	$\mathcal{DN}_{VP}$-R Forest
4.18	$\mathcal{DN}_{PA}$-R Forest
4.31	$\mathcal{DN}_P$-R Forest
4.81	R Forest

V−D	V	D	Método
12	12	0	$\mathcal{DN}_P$-R Forest
10	13	3	$\mathcal{DN}_{PA}$-R Forest
-3	5	8	$\mathcal{DN}$-R Forest
-3	6	9	$\mathcal{DN}_{VP}$-R Forest
-3	2	5	$\mathcal{DN}_{VA}$-R Forest
-6	2	8	$\mathcal{DN}_V$-R Forest
-7	8	15	R Forest

Tabla 4.20: Rankings para las distintas variantes de $\mathcal{DN}$-Ensemble (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
2.23	$\mathcal{DN}_{VP}$-Ensemble
2.74	$\mathcal{DN}$-Ensemble
3.07	$\mathcal{DN}_V$-Ensemble
3.08	$\mathcal{DN}_{VA}$-Ensemble
4.63	$\mathcal{DN}_P$-Ensemble
5.24	$\mathcal{DN}_{PA}$-Ensemble

V−D	V	D	Método
33	40	7	$\mathcal{DN}_{VP}$-Ensemble
32	42	10	$\mathcal{DN}$-Ensemble
29	40	11	$\mathcal{DN}_{VA}$-Ensemble
29	38	9	$\mathcal{DN}_V$-Ensemble
-48	7	55	$\mathcal{DN}_P$-Ensemble
-75	0	75	$\mathcal{DN}_{PA}$-Ensemble

Tabla 4.21: Rankings para las distintas variantes de ($\mathcal{DN}$-)Random Subspaces 50 % (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
2.88	$\mathcal{DN}_{VP}$-Subespaces (50 %)
3.42	$\mathcal{DN}_V$-Subspaces (50 %)
3.52	$\mathcal{DN}$-Subspaces (50 %)
3.60	$\mathcal{DN}_{VA}$-Subspaces (50 %)
4.06	$\mathcal{DN}_P$-Subspaces (50 %)
4.74	$\mathcal{DN}_{PA}$-Subspaces (50 %)
5.77	Subspaces (50 %)

V−D	V	D	Método
7	7	0	$\mathcal{DN}_{VP}$-Subspaces (50 %)
7	7	0	$\mathcal{DN}_V$-Subspaces (50 %)
6	6	0	$\mathcal{DN}$-Subspaces (50 %)
5	5	0	$\mathcal{DN}_{VA}$-Subspaces (50 %)
-3	3	6	$\mathcal{DN}_P$-Subspaces (50 %)
-5	2	7	$\mathcal{DN}_{PA}$-Subspaces (50 %)
-17	0	17	Subspaces (50 %)

Tabla 4.22: Rankings para las distintas variantes de ($\mathcal{DN}$-)Random Subspaces 75 % (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
2.90	$\mathcal{DN}$-Subspaces (75 %)
2.90	$\mathcal{DN}_{VP}$-Subspaces (75 %)
3.24	$\mathcal{DN}_{VA}$-Subspaces (75 %)
3.32	$\mathcal{DN}_{V}$-Subspaces (75 %)
4.49	$\mathcal{DN}_{P}$-Subspaces (75 %)
4.90	$\mathcal{DN}_{PA}$-Subspaces (75 %)
6.26	Subspaces (75 %)

V−D	V	D	Método
26	26	0	$\mathcal{DN}$-Subspaces (75 %)
26	26	0	$\mathcal{DN}_{VP}$-Subspaces (75 %)
25	25	0	$\mathcal{DN}_{VA}$-Subspaces (75 %)
24	24	0	$\mathcal{DN}_{V}$-Subspaces (75 %)
-9	9	18	$\mathcal{DN}_{P}$-Subspaces (75 %)
-24	5	29	$\mathcal{DN}_{PA}$-Subspaces (75 %)
-68	0	68	Subspaces (75 %)

Tabla 4.23: Rankings para las distintas variantes de ($\mathcal{DN}$-)AdaBoost(W) (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
3.45	$\mathcal{DN}$-AdaBoost (W)
3.54	$\mathcal{DN}_{VP}$-AdaBoost (W)
3.59	$\mathcal{DN}_{V}$-AdaBoost (W)
3.73	$\mathcal{DN}_{VA}$-AdaBoost (W)
4.19	$\mathcal{DN}_{PA}$-AdaBoost (W)
4.32	$\mathcal{DN}_{P}$-AdaBoost (W)
5.19	AdaBoost (W)

V−D	V	D	Método
5	5	0	$\mathcal{DN}$-AdaBoost (W)
4	4	0	$\mathcal{DN}_{VP}$-AdaBoost (W)
4	4	0	$\mathcal{DN}_{VA}$-AdaBoost (W)
3	3	0	$\mathcal{DN}_{V}$-AdaBoost (W)
-2	1	3	$\mathcal{DN}_{PA}$-AdaBoost (W)
-2	1	3	$\mathcal{DN}_{P}$-AdaBoost (W)
-12	0	12	AdaBoost (W)

Tabla 4.24: Rankings para las distintas variantes de ($\mathcal{DN}$-)AdaBoost(S) (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
3.34	$\mathcal{DN}$-AdaBoost (S)
3.35	$\mathcal{DN}_{VA}$-AdaBoost (S)
3.48	$\mathcal{DN}_{VP}$-AdaBoost (S)
3.76	$\mathcal{DN}_{V}$-AdaBoost (S)
4.31	$\mathcal{DN}_{PA}$-AdaBoost (S)
4.45	$\mathcal{DN}_{P}$-AdaBoost (S)
5.32	AdaBoost (S)

V−D	V	D	Método
4	4	0	$\mathcal{DN}_{VA}$-AdaBoost (S)
3	3	0	$\mathcal{DN}$-AdaBoost (S)
1	1	0	$\mathcal{DN}_{PA}$-AdaBoost (S)
1	1	0	$\mathcal{DN}_{V}$-AdaBoost (S)
0	1	1	$\mathcal{DN}_{VP}$-AdaBoost (S)
-1	1	2	$\mathcal{DN}_{P}$-AdaBoost (S)
-8	0	8	AdaBoost (S)

Tabla 4.25: Rankings para las distintas variantes de ($\mathcal{DN}$-)MultiBoost(W) (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
3.07	$\mathcal{DN}_{VA}$-MultiBoost (W)
3.14	$\mathcal{DN}_{VP}$-MultiBoost (W)
3.68	$\mathcal{DN}$-MultiBoost (W)
4.02	$\mathcal{DN}_{V}$-MultiBoost (W)
4.56	$\mathcal{DN}_{PA}$-MultiBoost (W)
4.56	$\mathcal{DN}_{P}$-MultiBoost (W)
4.98	MultiBoost (W)

V−D	V	D	Método
6	6	0	$\mathcal{DN}$-MultiBoost (W)
6	6	0	$\mathcal{DN}_{VA}$-MultiBoost (W)
2	2	0	$\mathcal{DN}_{PA}$-MultiBoost (W)
0	2	2	$\mathcal{DN}_{P}$-MultiBoost (W)
-2	2	4	$\mathcal{DN}_{V}$-MultiBoost (W)
-4	2	6	$\mathcal{DN}_{VP}$-MultiBoost (W)
-8	1	9	MultiBoost (W)

Tabla 4.26: Rankings para las distintas variantes de ($\mathcal{DN}$-)MultiBoost(S) (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

Ranking Promedio	Método
3.02	$\mathcal{DN}_{V}$-MultiBoost (S)
3.35	$\mathcal{DN}_{VP}$-MultiBoost (S)
3.82	$\mathcal{DN}$-MultiBoost (S)
3.84	$\mathcal{DN}_{VA}$-MultiBoost (S)
4.24	$\mathcal{DN}_{P}$-MultiBoost (S)
4.73	$\mathcal{DN}_{PA}$-MultiBoost (S)
5.00	MultiBoost (S)

V−D	V	D	Método
5	5	0	$\mathcal{DN}$-MultiBoost (S)
5	5	0	$\mathcal{DN}_{VA}$-MultiBoost (S)
5	5	0	$\mathcal{DN}_{PA}$-MultiBoost (S)
-2	1	3	$\mathcal{DN}_{VP}$-MultiBoost (S)
-2	1	3	$\mathcal{DN}_{P}$-MultiBoost (S)
-5	2	7	$\mathcal{DN}_{V}$-MultiBoost (S)
-6	1	7	MultiBoost (S)

ya que estos no presentan un método puro.

En todas las familias, las variantes que tienen en cuenta las m características binarias que indican cuál es el vecino más cercano (que denotaremos como $Variantes_V$) siempre quedan en el grupo de cabeza, y tan solo una vez quedan en el grupo de métodos de cola (i.e., $\mathcal{DN}_V$-MultiBoost(W)). Es más:

- Las $Variantes_V$ siempre toman los cuatro primeros lugares en todas las familias.
- Las $Variantes_V$ están siempre en el grupo de métodos de cabeza (métodos que no son significativamente mejores que el primero).

Pero además, siempre resulta haber una variante bien del tipo $\mathcal{DN}_{PA}$, bien del tipo $\mathcal{DN}_P$ en penúltimo y antepenúltimo lugar, menos en el caso de $\mathcal{DN}$-Ensemble, en el que estas dos variantes ocupan las dos últimas posiciones. En todas las familias, excepto $\mathcal{DN}$-Ensemble, la versión pura del ensemble ocupa el último lugar, y muchas de las variantes $\mathcal{DN}_{PA}$ y $\mathcal{DN}_P$ incluso no suponen una mejora significativa respecto de la variante pura.

En todas las familias la variante $\mathcal{DN}_V$ queda mejor colocada que la variante $\mathcal{DN}_P$, y en seis de las nueve familias el método en la primera posición es una $Variante_P$ (i.e., tienen en cuenta la predicción del 1-NN), pero siempre se trata de variantes $Variante_P$ que son además $Variante_V$.

En cuanto a la selección aleatoria de características, en cuatro familias la variante $\mathcal{DN}$ ocupa mejores posiciones en el ranking que la variante $\mathcal{DN}_{VP}$, pero en otras cinco ocurre todo lo contrario.

En la misma línea, se observa que predominan las familias en las que la variante $\mathcal{DN}_V$ ocupa mejor posición en el ranking que $\mathcal{DN}_{VA}$ (6 ocasiones frente a 3), y que en proporción 5 a 4 también predominan las familias en la que la variante $\mathcal{DN}_P$ ocupa mejores posiciones que la $\mathcal{DN}_{PA}$. Por todo ello, parece que la selección de características aleatorias para el cálculo de las distancias no aporta mucho. Sin embargo, esta selección reduce la dimensión del espacio de entrada, por lo que puede ser una opción interesante en conjuntos de datos grandes, ya que contribuiría algo a reducir los tiempos de computación, sin incurrir por ello en una disminución significativa en la tasa de acierto.

Por lo tanto, se puede concluir que la utilización de las m características binarias que indican cual es el vecino más cercano, es el elemento común en las familias que obtienen mejores resultados, mientras que tanto la predicción hecha por el clasificador 1-NN, como la selección aleatoria de características, no parecen ser esenciales.

Si se hace el mismo análisis a través del ranking por la diferencia entre victorias y derrotas significativas, los resultados apuntan en la misma dirección pero con menos claridad. En estos rankings, que encuentran en la parte derecha de las tablas de la 4.18 a la 4.26, es observable en dichas tablas que:

1. Salvo, lógicamente, en el caso de la familia $\mathcal{DN}$-Ensemble, el multiclasificador puro (i.e., sin $\mathcal{DN}$) es siempre el peor método.

2. Que en cinco de las nueve familias consideradas (i.e., AdaBoost(W), Bagging, $\mathcal{DN}$-Ensemble, Subespaces(50 %) y Subespaces(75 %)), las cuatro variantes de cabeza son las cuatro $Variantes_V$.

3. Salvo en los Random Forests, en el resto de familias los dos primeros métodos de cada ranking son siempre $Variantes_V$.

Pero por otro lado, habría algunos indicadores que contradirían la idea de que es el vector de booleanos el elemento más influyente en el algoritmo $\mathcal{DN}$. Así, aunque en todos los rankings promedios, y en la mayoría de los rankings por diferencias entre victorias y derrotas significativas las peores variantes de $\mathcal{DN}$ son las que no son $Variantes_V$, es observable que en los rankings por diferencias:

1. En Random Forests, las variantes $\mathcal{DN}_P$ y $\mathcal{DN}_{PA}$ son las dos mejores,

2. En AdaBoost(S), MultiBoost(W) y MultiBoost(S), la variante $\mathcal{DN}_{PA}$ llega a alcanzar una tercera posición.

Parece, por tanto, que el ranking de diferencia entre victorias y derrotas concede en general más crédito a las $Variantes_V$ que a las $Variantes_P$. Sin embargo, no lo hace con la misma rotundidad que el ranking promedio. No obstante, el ranking de diferencias suelen tomarse como menos fiable que el ranking promedio, en tanto el primero sólo toma en cuenta aquellos resultados en los que hay una diferencia según un nivel de significación prefijado, obviando el resto de resultados [26] (ver discusión en la sección 2.3.2).

Si se observan las tablas donde figuran las tasas de acierto para cada una de las variantes en cada conjunto de datos (tablas de la A.11 a la A.19), en la gran mayoría de los conjuntos de datos, las variantes $\mathcal{DN}$ resultan en empates frente a la versión pura del multiclasificador (sin $\mathcal{DN}$). Es por tanto patente en este caso la fragilidad del ranking de diferencias, pues basta quitar o sustituir esos escasos conjuntos de datos donde hay diferencias significativas para llegar a resultados totalmente distintos.

La tabla 4.27 muestra los rankings promedios para todas las variantes de los métodos considerados. La columna *Benef* (beneficio) en las filas correspondientes a variantes de $\mathcal{DN}$, se computa como la diferencia entre el ranking promedio que se obtiene al usar la correspondiente variante $\mathcal{DN}$ menos el ranking promedio del mismo multiclasificador sin ninguna variante de $\mathcal{DN}$. Por tanto, el beneficio indica la ganancia de la versión con $\mathcal{DN}$ respecto al multiclasificador puro. En esta tabla se pueden observar ciertas cuestiones:

1. Por un lado las familias van apareciendo en un orden que guarda bastante relación con el de la tabla 4.11. Primero aparecen entremezcladas las distintas variantes de MultiBoost y Random Forests; en un segundo grupo las variantes de AdaBoost, luego las de Subspaces(50 %), otro grupo en el que se entremezclan variantes de Subspaces(75 %) y Bagging, y finalmente los $\mathcal{DN}$-Ensemble junto con los 1-NN y k-NN.

Tabla 4.27: Rankings promedios de todas las $\mathcal{DN}$-variantes.

Ranking Promedio	Benef	Método
22.01	7.93	$\mathcal{DN}_V$-MultiBoost (S)
22.56	7.61	$\mathcal{DN}_{VP}$-MultiBoost (W)
22.73	7.21	$\mathcal{DN}_{VP}$-MultiBoost (S)
22.87	7.30	$\mathcal{DN}_{VA}$-MultiBoost (W)
24.65	5.52	$\mathcal{DN}$-MultiBoost (W)
25.16	6.65	$\mathcal{DN}_{VP}$-Random Forest
25.22	6.59	$\mathcal{DN}_V$-Random Forest
25.39	6.42	$\mathcal{DN}$-Random Forest
25.40	4.54	$\mathcal{DN}$-MultiBoost (S)
25.67	4.50	$\mathcal{DN}_V$-MultiBoost (W)
25.96	3.98	$\mathcal{DN}_P$-MultiBoost (S)
26.06	3.88	$\mathcal{DN}_{VA}$-MultiBoost (S)
26.57	5.23	$\mathcal{DN}_{VA}$-Random Forest
26.61	3.56	$\mathcal{DN}_P$-MultiBoost (W)
27.51	6.29	$\mathcal{DN}_{VP}$-AdaBoost (S)
27.75	6.05	$\mathcal{DN}$-AdaBoost (S)
27.77	2.40	$\mathcal{DN}_{PA}$-MultiBoost (W)
27.93	6.15	$\mathcal{DN}_{VP}$-AdaBoost (W)
28.13	5.67	$\mathcal{DN}_{VA}$-AdaBoost (S)
28.57	1.36	$\mathcal{DN}_{PA}$-MultiBoost (S)
28.61	5.47	$\mathcal{DN}_V$-AdaBoost (W)
28.75	5.33	$\mathcal{DN}$-AdaBoost (W)
28.78	5.02	$\mathcal{DN}_V$-AdaBoost (S)
29.33	9.80	$\mathcal{DN}_{VP}$-Subspaces (50 %)
29.42	4.66	$\mathcal{DN}_{VA}$-AdaBoost (W)
29.77	2.03	$\mathcal{DN}_P$-Random Forest
29.85	3.94	$\mathcal{DN}_P$-AdaBoost (S)
29.94		MultiBoost(S)
29.94	1.86	$\mathcal{DN}_{PA}$-Random Forest
30.17		MultiBoost (W)
30.68	3.40	$\mathcal{DN}_{PA}$-AdaBoost (W)
30.91	8.22	$\mathcal{DN}_V$-Subspaces (50 %)
31.17	7.96	$\mathcal{DN}$-Subspaces (50 %)
31.31	2.49	$\mathcal{DN}_{PA}$-AdaBoost (S)
31.34	2.74	$\mathcal{DN}_P$-AdaBoost(W)
31.68	10.07	$\mathcal{DN}_{VP}$-Bagging
31.76	13.73	$\mathcal{DN}_{VP}$-Subspaces (75 %)
31.81		Random Forest
32.13	7.00	$\mathcal{DN}_{VA}$-Subspaces (50 %)
32.19	6.94	$\mathcal{DN}_P$-Subspaces (50 %)
32.56	12.93	$\mathcal{DN}$-Subspaces (75 %)
33.80		AdaBoost (S)
34.08		AdaBoost (W)
34.19	4.94	$\mathcal{DN}_{PA}$-Subspaces (50 %)
34.35	7.40	$\mathcal{DN}$-Bagging
34.40	7.35	$\mathcal{DN}_P$-Bagging
34.61	7.14	$\mathcal{DN}_V$-Bagging
34.64	10.85	$\mathcal{DN}_{VA}$-Subspaces (75 %)
34.89	10.60	$\mathcal{DN}_V$-Subspaces (75 %)
36.56	8.92	$\mathcal{DN}_P$-Subspaces (75 %)
36.77	4.98	$\mathcal{DN}_{VA}$-Bagging
39.10	2.65	$\mathcal{DN}_{PA}$-Bagging
39.13		Subspaces (50 %)
39.23	6.25	$\mathcal{DN}_{PA}$-Subspaces (75 %)
39.40		k-NN
41.75		Bagging
42.23		$\mathcal{DN}_{VP}$-Ensemble
45.48		$\mathcal{DN}$-Ensemble
45.48		Subspaces (75 %)
45.85		$\mathcal{DN}_V$-Ensemble
46.84		$\mathcal{DN}_{VA}$-Ensemble
48.52		1-NN
49.19		$\mathcal{DN}_P$-Ensemble
52.91		$\mathcal{DN}_{PA}$-Ensemble

Tabla 4.28: Posiciones relativas en la familia de cada $\mathcal{DN}$-variante usando el ranking promedio de la Tabla 4.27.

Familia	VP	VPA	V	VA	P	PA	Método sin $\mathcal{DN}$
Bagging	1	2	4	5	3	6	7
Random Forest	1	3	2	4	5	6	7
$\mathcal{DN}$-Ensemble	1	2	3	4	5	6	-
Subspaces(50 %)	1	3	2	4	5	6	7
Subspaces(75 %)	1	2	4	3	5	6	7
AdaBoost(W)	1	3	2	4	6	5	7
AdaBoost(S)	1	2	4	3	5	6	7
MultiBoost(W)	1	3	4	2	5	6	7
MultiBoost(S)	2	3	1	5	4	6	7
Suma	10	23	26	34	43	53	56

2. Si tomamos la tabla 4.27 seleccionando por separado los miembros de cada familia y establecemos entre ellos un ranking relativo, se obtiene la tabla 4.28 [4]. En ella se observa con gran nitidez la fortaleza de las $Variantes_V$, ya que:

   - Salvo, lógicamente, en el caso de la familia $\mathcal{DN}$-Ensemble, el multiclasificador puro (i.e., sin $\mathcal{DN}$) es siempre el peor método.
   - La variante PA es siempre la peor variante $\mathcal{DN}$ salvo en el caso de AdaBoost(W).
   - La variante P salvo en dos ocasiones, suele ser la penúltima o última entre las versiones con $\mathcal{DN}$.
   - La variante VP es la que obtiene el mejor resultado en cada familia salvo en una sola ocasión (i.e., $\mathcal{DN}_V$-MultiBoost(S)).
   - Las variantes VPA, V y VA salvo en dos ocasiones que hacen quinto y una que hace primero, vagan entre los puestos segundo al cuarto.

3. Las diferencias significativas entre métodos que resultan de las tablas de la 4.18 a la 4.26 no se ven contradichas por ninguna permutación en el orden de los métodos de la tabla 4.27, con la salvedad de que en la familia MultiBoost(S) se había obtenido que $\mathcal{DN}_{VA}$-MultiBoost(S) era significativamente mejor que $\mathcal{DN}_P$-MultiBoost(S), sin embargo las tablas 4.27 y 4.28 muestran a $\mathcal{DN}_P$-MultiBoost(S) ligeramente por delante de $\mathcal{DN}_{VA}$-MultiBoost(S).

   Todo ello ratifica los resultados obtenidos en el análisis por familias en las tablas de la 4.18 a la 4.26.

[4] En la tabla VPA es la variante de $\mathcal{DN}$ pura; como por ejemplo $\mathcal{DN}$-Bagging, que es lo mismo que $\mathcal{DN}_{VPA}$-Bagging.

Tabla 4.29: Rankings de los beneficios computados en la Tabla 4.27.

Beneficio	Método	Es una $Variante_V$
13.73	$\mathcal{DN}_{VP}$-Subspaces (75 %)	•
12.93	$\mathcal{DN}$-Subspaces (75 %)	•
10.85	$\mathcal{DN}_{VA}$-Subspaces (75 %)	•
10.60	$\mathcal{DN}_{V}$-Subspaces (75 %)	•
10.07	$\mathcal{DN}_{VP}$-Bagging	•
9.80	$\mathcal{DN}_{VP}$-Subspaces (50 %)	•
8.92	$\mathcal{DN}_{P}$-Subspaces (75 %)	
8.22	$\mathcal{DN}_{V}$-Subspaces (50 %)	•
7.96	$\mathcal{DN}$-Subspaces (50 %)	•
7.93	$\mathcal{DN}_{V}$-MultiBoost (S)	•
7.61	$\mathcal{DN}_{VP}$-MultiBoost (W)	•
7.40	$\mathcal{DN}$-Bagging	•
7.35	$\mathcal{DN}_{P}$-Bagging	
7.30	$\mathcal{DN}_{VA}$-MultiBoost (W)	•
7.21	$\mathcal{DN}_{VP}$-MultiBoost (S)	•
7.14	$\mathcal{DN}_{V}$-Bagging	•
7.00	$\mathcal{DN}_{VA}$-Subspaces (50 %)	•
6.94	$\mathcal{DN}_{P}$-Subspaces (50 %)	
6.65	$\mathcal{DN}_{VP}$-Random Forest	•
6.59	$\mathcal{DN}_{V}$-Random Forest	•
6.42	$\mathcal{DN}$-Random Forest	•
6.29	$\mathcal{DN}_{VP}$-AdaBoost (S)	•
6.25	$\mathcal{DN}_{PA}$-Subspaces (75 %)	
6.15	$\mathcal{DN}_{VP}$-AdaBoost (W)	•
6.05	$\mathcal{DN}$-AdaBoost (S)	•
5.67	$\mathcal{DN}_{VA}$-AdaBoost (S)	•
5.52	$\mathcal{DN}$-MultiBoost (W)	•
5.47	$\mathcal{DN}_{V}$-AdaBoost (W)	•
5.33	$\mathcal{DN}$-AdaBoost (W)	•
5.23	$\mathcal{DN}_{VA}$-Random Forest	•
5.02	$\mathcal{DN}_{V}$-AdaBoost (S)	•
4.98	$\mathcal{DN}_{VA}$-Bagging	•
4.94	$\mathcal{DN}_{PA}$-Subspaces (50 %)	
4.66	$\mathcal{DN}_{VA}$-AdaBoost (W)	•
4.54	$\mathcal{DN}$-MultiBoost (S)	•
4.50	$\mathcal{DN}_{V}$-MultiBoost (W)	•
3.98	$\mathcal{DN}_{P}$-MultiBoost (S)	
3.94	$\mathcal{DN}_{P}$-AdaBoost (S)	
3.88	$\mathcal{DN}_{VA}$-MultiBoost (S)	•
3.56	$\mathcal{DN}_{P}$-MultiBoost (W)	
3.40	$\mathcal{DN}_{PA}$-AdaBoost (W)	
2.74	$\mathcal{DN}_{P}$-AdaBoost(W)	
2.65	$\mathcal{DN}_{PA}$-Bagging	
2.49	$\mathcal{DN}_{PA}$-AdaBoost (S)	
2.03	$\mathcal{DN}_{P}$-Random Forest	
2.40	$\mathcal{DN}_{PA}$-MultiBoost (W)	
1.86	$\mathcal{DN}_{PA}$-Random Forest	
1.36	$\mathcal{DN}_{PA}$-MultiBoost (S)	

Si se ordenan las entradas con beneficios de la tabla 4.27 por dicha columna, se obtiene la tabla 4.29. Los círculos rellenos de la tabla indican que la variantes de la fila contienen el vector de booleanos indicando cuál es el vecino más cercano. Es inmediato apreciar a través de esos puntos como las $Variantes_V$ tienden a escapar del fondo de la tabla, mientras que aquellas que no lo son, copan los puestos de cola. Esta representación, nuevamente, sirve para mostrar de manera gráfica que la presencia del vector de booleanos indicando cuál es el vecino más cercano es presumiblemente el origen de esos beneficios, y por tanto el elemento fundamental del algoritmo de $\mathcal{DN}$.

4.6. Conclusiones

En este capítulo se ha presentado un método para la mejora de la diversidad en los clasificadores base de un multiclasificador cualquiera. El método presentado construye previamente un conjunto de características que se añaden al conjunto de entrenamiento. Estos nuevos atributos serán diferentes para cada clasificador base, haciendo que los clasificadores base también lo sean, y provocando a la postre que el multiclasificador al que pertenecen sea diverso.

Estas características que hacen diferente a cada clasificador:

1. Alteran o «perturban» el resultado del proceso de entrenamiento, y
2. Provienen de la utilización de un clasificador 1-NN.

Es por estas dos razones por las que el método se ha dado en llamar Disturbing Neighbors ($\mathcal{DN}$). Las instancias con las que se construye el clasificador 1-NN, son un pequeño subconjunto del conjunto de entrenamiento, y son elegidas de forma aleatoria para cada clasificador base. La aleatoriedad presente en esta elección es la que finalmente proporciona la diversidad del multiclasificador.

Para probar que el método supone una mejora en la tasa de acierto de un multiclasificador, se han provisto sendas validaciones experimentales, la primera sobre multiclasificadores de SVM y la segunda sobre multiclasificadores de árboles de decisión. Ambas validaciones ponen de manifiesto que:

1. Las versiones de multiclasificadores que utilizan $\mathcal{DN}$ mejoran significativamente a las que no las usan.
2. Al probar todos los multiclasificadores $\mathcal{DN}$ contra 1-NN, se ve que siempre son significativamente mejores, salvo los $\mathcal{DN}$-Subspaces para SVM, que son mejores pero no significativamente, luego no hay una mejora del multiclasificador debida a la influencia del pequeño 1-NN empotrado en los clasificadores base $\mathcal{DN}$.
3. Tanto en los multiclasificadores de SVM, como en los de árboles, el orden relativo de las versiones con $\mathcal{DN}$ es muy similar al orden relativo a las versiones puras, por lo que $\mathcal{DN}$ no es el ingrediente principal de cada uno de estos algoritmos, sino que simplemente es una mejora.

4. Corroborando la afirmación anterior, se ha comprobado que combinar varios clasificadores base $\mathcal{DN}$ haciendo que la predicción final sea el promedio de las predicciones de cada uno ($\mathcal{DN}$-Ensemble), en principio no es suficiente para garantizar un multiclasificador competitivo respecto al resto de esquemas analizados. Aunque en el caso de los SVM tuvieron un resultado aceptable, en el caso de los árboles de decisión los $\mathcal{DN}$-Ensemble no dieron buen resultado.

5. En el caso de SVM, la mejora que se alcanza con $\mathcal{DN}$, no es suficiente como para justificar su utilización frente otras alternativas más ligeras computacionalmente, como por ejemplo k-NN optimizando el k en cada conjunto.

Todo ello muestra a $\mathcal{DN}$ como una mejora de métodos ya existentes, y esa mejora es debida al aumento de la diversidad de los clasificadores base. Para comprobar esta hipótesis, se han presentado nuevos diagramas basados en la estadística Kappa (diagramas de Movimiento Kappa-Error y diagramas de Movimiento Relativo Kappa-Error) que muestran claramente la mejora de la diversidad.

Se ha añadido un estudio de lesiones con el fin de clarificar cuáles son los ingredientes esenciales de los $\mathcal{DN}$, resultando que tanto la predicción hecha por el clasificador 1-NN, como la selección aleatoria de características para el cálculo de las distancias por si solos, no son los elementos que proveen la mejora significativa presente en el algoritmo. Sin embargo, la utilización del conjunto de características binarias indicando cuál es el vecino más cercano parece la clave del éxito del método. Por ello, el clasificador 1-NN en $\mathcal{DN}$ no parece reemplazable por otro clasificador cualquiera que se limite a predecir la clase, sino que además es necesario que construya una serie de atributos que representen una cierta división del espacio, tal y como ocurre con el método de los vecinos más cercanos.

Finalmente, aunque las técnicas de regresión quedan fuera del alcance de esta tesis, cabe resaltar que *Disturbing Neighbors* ha sido también utilizado para tareas de regresión en [101]. En este caso los regresores base utilizados fueron árboles, y los resultados experimentales sobre 61 conjuntos de datos muestran también, que cuando se aplican *Disturbing Neighbors* a los regresores base hay una mejora generalizada respecto de cuando no se utilizan en métodos que combinan regresores como Random Subspaces [55], Bagging [7], Iterated Bagging [12], y AdaBoost.R2 [32].

Capítulo 5

Random Feature Weights

5.1. Introducción

Los multiclasificadores son combinaciones de varios clasificadores base, que mejoran sus resultados respecto a cuando actúan por separado [66]. Gran parte de los multiclasificadores del estado del arte son homogéneos, en tanto utilizan clasificadores base construidos con el mismo algoritmo. Este tipo de multiclasificadores modifican el conjunto de entrenamiento original para obtener así distintos clasificadores base a partir del mismo método.

En capítulos anteriores ya se han descrito multiclasificadores que siguen esta estrategia. Así, Bagging [7] entrena cada clasificador base a partir de una muestra aleatoria del conjunto de entrenamiento; Random Subspaces [55] entrena cada clasificador base con todas las instancias, pero utilizando un subconjunto de los atributos; los métodos de la familia de Boosting [42], entre los cuales el más popular es AdaBoost, se basan en utilizar una distribución de pesos dependiente del error del clasificador base en la iteración anterior para las instancias de entrenamiento del próximo clasificador base, concentrándose cada vez más en las instancias más difíciles de clasificar. MultiBoost [116] es una combinación de AdaBoost y Bagging.

Los árboles de decisión son uno de los métodos que más se usan como clasificadores base, ya que son eficientes e inestables. Esta última propiedad supone que con cambios relativamente pequeños en el conjunto de datos de entrenamiento, es posible obtener clasificadores muy diferentes. Esta propiedad es muy deseable para un clasificador base, en tanto un buen multiclasificador necesita que dichos clasificadores base sean diversos.

Una estrategia para la obtención de multiclasificadores con clasificadores base diversos, es la introducción de algún tipo de aleatoriedad en dichos métodos base. Algunas de estas fórmulas son específicas de los árboles de decisión [29]. Random Forest [11] combina Bagging con Random Trees. En estos árboles aleatorios, sólo se considera un subconjunto aleatorio de atributos para cada nodo. En [4] se presenta una comparación muy exhaustiva de multiclasificadores con

árboles de decisión.

En este capítulo se propone un método perteneciente a esta familia de multiclasificadores de árboles de decisión que usan alguna estrategia para introducir aleatoriedad en la fase de construcción de los árboles: *Random Feature Weights* ($\mathcal{RFW}$).

Como en el caso de los Random Forests, el método se basa en inyectar aleatoriedad en el proceso de construcción del árbol de decisión. La principal diferencia con los Random Forests, es que para cada nodo del árbol se siguen utilizando todos los atributos, en lugar de un subconjunto de los mismos.

La fuente de diversidad proviene de un peso que se asocia a cada atributo. Todos los nodos pertenecientes al mismo árbol usarán los mismos pesos, pero en general, estos pesos serán distintos a los de otros árboles del mismo *bosque* (i.e., multiclasificador en el que los clasificadores base son árboles de decisión). Por ello, la importancia que se da a cada uno de los atributos será distinta en cada árbol, y hará que la construcción de cada árbol del bosque sea distinta.

En este capítulo el método se compara con Bagging, Random Forests, Random Subspaces, AdaBoost y MultiBoost, obteniendo resultados favorables para el método propuesto, especialmente para el caso en el que se usen datos con ruido. $\mathcal{RFW}$ puede combinarse con todos estos métodos multiclasificadores; y en general, dicha combinación produce mejores resultados que el método original.

Los diagramas de movimiento de Kappa-Error utilizados ya en el capítulo anterior, también se utilizarán en el presente para analizar la relación entre la precisión de los clasificadores base y su diversidad.

El resto del capítulo se organiza como sigue: el algoritmo es descrito en la sección 5.2, los resultados experimentales en la sección 5.3, en la que la comparativa con otros multiclasificadores, es extendida a conjuntos con ruido. Además esta misma sección incluye el análisis de la incidencia de $\mathcal{RFW}$ sobre el posible aumento de la diversidad, usando para ello diagramas basados en la estadística Kappa-Error. La sección 5.4 discute la influencia del único parámetro que tiene el método sobre el resultado. Finalmente, la sección 5.5 resume las propiedades de este nuevo método a la luz del estudio aportado.

5.2. Algoritmo

Cuando se construye descendentemente un árbol de decisión, cada vez que se llega a un nodo, hay que elegir cuál es el atributo en base al cual se hará la ramificación (ver sección 2.1.2). Para ello, se utiliza alguna función, para evaluar el «mérito» de cada uno de los atributos en ser el elegido para ese cometido. Ejemplos conocidos de estas funciones son *Information Gain* (Ganancia de Información) [117], *Gain Ratio* [92] (Ratio de Ganancia) y el *Gini Index* [13] (Índice Gini).

Sea D el conjunto de datos de entrenamiento para el nodo que, en un determinado paso de construcción de un árbol, se esté analizando. Sean $a_1, \ldots, a_n$ los atributos, y $f(a_i, D)$ el resultado de la función que calcula el mérito para el atributo a_i con el conjunto D. Es importante darse cuenta de que D es un sub-

conjunto del conjunto de entrenamiento original. Sólo coincide con el conjunto de entrenamiento original en el caso de que se estén seleccionando los atributos correspondientes al nodo raíz del árbol.

Lo que hace el método que se propone en este capítulo es modificar la función f que calcula el mérito, asociando un peso w_i a cada atributo a_i. La función resultante será:

$$f_{\mathbf{w}}(a_i, D) = w_i f(a_i, D)$$

Por tanto, los atributos que tengan más peso tienen más posibilidades de ser seleccionados en la construcción del árbol, como atributos por los que ramificar.

Esta función $f_{\mathbf{w}}$ es la misma para todos los nodos de un mismo árbol. Por ello, se puede intuir que los atributos con más peso tendrán tendencia a aparecer en los nodos más cercanos al raíz, condicionando fuertemente la elección de los atributos de los nodos más alejados a dicha raíz.

En definitiva, la utilización de los pesos introduce un *bias* (condiciona el método de inducción que usa el árbol de decisión) haciendo que exista una preferencia por algunos atributos frente a otros.

Cuando se construyen multiclasificadores, conviene perseguir dos objetivos para que éstos tengan éxito: uno es que los clasificadores base sean diversos entre sí, el otro que los clasificadores base sean precisos. Son dos objetivos contrapuestos, pues si los clasificadores son muy precisos, sus predicciones no diferirán mucho unas de otras. Como se verá en la siguiente sección, $\mathcal{RFW}$ permite obtener árboles muy distintos entre sí.

$\mathcal{RFW}$ puede entenderse de una manera más amplia, esto es, como un marco conceptual que permite describir otros multiclasificadores existentes como Random Subspaces [55] y Random Forest [11], pensando en estos métodos como un caso especial (o al menos parecido) de $\mathcal{RFW}$s en los que los pesos restringen sus valores a $\{0, 1\}$, ya que:

- En los Random Subspaces cada árbol se construye mediante un subconjunto de atributos que podría expresarse dando peso 1 a los atributos seleccionados y 0 a los descartados. Esos atributos son seleccionados aleatoriamente, y son los que se utilizan a lo largo de la construcción de todo el árbol, luego el peso no varía durante todo este proceso, como ocurre con $\mathcal{RFW}$.

- Con los Random Forests el parecido no es tan estrecho. En este caso también se selecciona un conjunto de atributos aleatoriamente, y esa elección también se puede expresar mediante unos y ceros. Sin embargo, los atributos seleccionados van variando de nodo a nodo a lo largo de la construcción del árbol. Es decir, los pesos no se conservan constantes durante todo el entrenamiento del árbol, como ocurre en $\mathcal{RFW}$.

Tanto en Random Subspaces como en Random Forests existe un parámetro común, que es el número de atributos a considerar, bien para cada árbol en los subespacios, bien en cada nodo en los Random Forests. Este parámetro podría expresarse en $\mathcal{RFW}$ como el número de pesos que han de ser uno.

Input

- p, exponente
- D, conjunto de entrenamiento
- $f(a, D)$, función que evalúa el mérito

Output

- Árbol de Decisión

Method

- Para cada atributo $a_i \in D$ hacer
  - $u \leftarrow$ valor aleatorio in $[0, 1]$
  - $w_i \leftarrow u^p$
- Entrenar un árbol de decisión usando como función que evalúa el mérito $f_{\mathbf{w}}(a_i, D) = w_i f(a_i, D)$

Figura 5.1: Algoritmo de construcón de un árbol $\mathcal{RFW}$.

Por tanto, es claro que $\mathcal{RFW}$ puede entenderse como una generalización de los Random Subspaces, ya que cada árbol usa en todos sus nodos el mismo conjunto de pesos, distinto al de los otros árboles del mismo bosque. La única diferencia con Random Subspaces, es que los pesos aleatorios de éstos están restringidos a los valores $\{0, 1\}$, mientras que en $\mathcal{RFW}$ se utilizan pesos que pueden ser cualquier número real, si bien la implementación que se ha hecho restringe sus valores al intervalo $[0, 1]$. El método toma el nombre $\mathcal{RFW}$ del acrónimo *Random Feature Weights*, ya que se basa en obtener estos pesos reales de forma aleatoria.

Es posible utilizar diferentes estrategias para dar valores a los pesos aleatorios. Inicialmente se pensó en tomarlos de una distribución uniforme en el intervalo $[0, 1]$. Sin embargo, se observó que al hacerlo de esta manera, la preferencia debida a los pesos, de un atributo sobre otro no era lo suficientemente grande, con lo que a menudo, varios árboles del mismo multiclasificador resultaban ser idénticos. Con el fin de dar más fuerza a los atributos elegidos por los pesos aleatorios, se introdujo un parámetro p en el método. Este parámetro se utiliza como exponente de cada peso, de manera que finalmente los pesos se obtiene de la distribución uniforme $[0, 1]$, pero se elevan cada uno a p antes de ser utilizados.

La figura 5.1 muestra el algoritmo de construcción de un árbol $\mathcal{RFW}$. En este capítulo se presenta el multiclasificador $\mathcal{RFW}$-Ensemble, que es un multiclasificador formado por árboles $\mathcal{RFW}$, y que para calcular su predicción utiliza el promedio de las predicciones de cada $\mathcal{RFW}$-árbol miembro. En la siguiente sección se verá que este multiclasificador es competitivo frente a los multiclasificadores de árboles más populares.

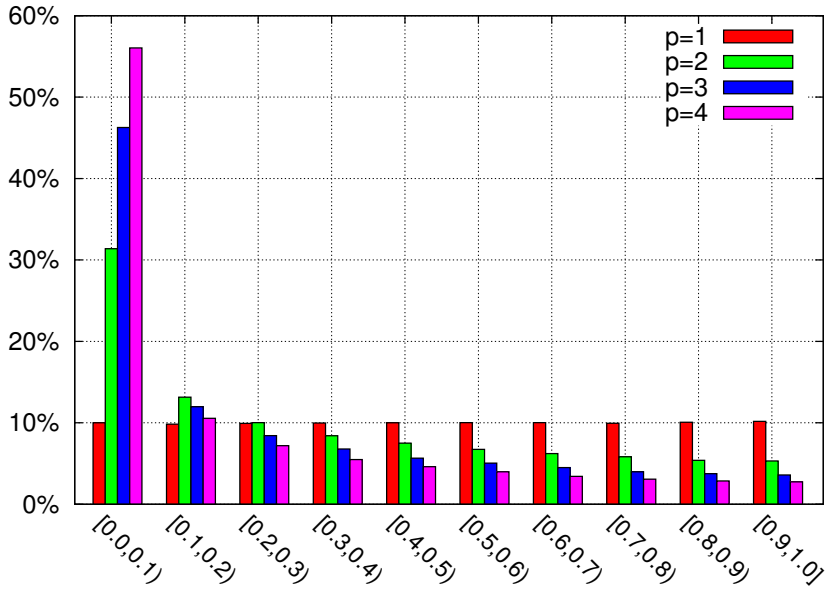


Figura 5.2: Distribución de los pesos en $\mathcal{RFW}$.

5.2.1. Distribución de los pesos aleatorios

En el algoritmo de la figura 5.1 muestra que los pesos se obtienen a través de una distribución uniforme, para luego elevarse a un exponente p. Por tanto, los pesos se corresponden a la distribución $\beta\left(\frac{1}{p}, 1\right)$.

La figura 5.2 muestra los histogramas que se obtienen de los pesos dando diferentes valores de p. Puede verse que a medida que p crece, hay una mayor proporción de pesos que tienen valores pequeños (e.g., para $p = 4$, más del 50 % de los pesos son menores que 0,1), y decree la proporción de pesos con valores grandes (i.e., cercanos a 1). Los atributos con mayor peso tendrán más ocpiones de ser elegidos como atributos por los que bifurcar un nodo del árbol. La abundancia de atributos con pesos pequeños contribuye a la diversidad, ya que cuando a un atributo se le asigna un peso pequeño, tiene más posibilidades de no ser utilizado en la construcción del árbol, incluso aunque la función de mérito lo señale como un buen atributo candidato.

5.3. Resultados experimentales

Como en el resto de capítulos, el método propuesto ha sido implementado en WEKA [117]. Las implementaciones del resto de algoritmos (árboles de decisión y multiclasificadores) utilizados en el experimento, se corresponden con las existentes en esta biblioteca. Para implementar el $\mathcal{RFW}$-árbol, se ha modificado el clasificador J.48 de WEKA.

El número de árboles de todos los multiclasificadores es 50. Los resultados se han obtenido haciendo validación cruzada estratificada 10×10. Se han utilizado los mismos 62 conjuntos de datos de la UCI [3] del capítulo anterior (ver tabla 4.3).

Se ha comparado el método $\mathcal{RFW}$-Ensemble con Bagging [7], Random Forests [11], Random Subspaces [55], AdaBoost [42] y MultiBoost [116] con las siguientes configuraciones:

- Random Subspaces nuevamente ha sido ejecutado utilizando las dos configuraciones correspondientes a la utilización del 50 % y el 75 % de los atributos.
- En Random Forests el número de atributos que se toman en cuenta en cada nodo, es el que toma por defecto WEKA, esto es, el logaritmo en base dos del número de atributos.
- En AdaBoost y MultiBoost, como de costumbre, se incluyen las mismas dos variantes que en anteriores capítulos: la variante con repesado (denotada por (W)), y la variante con remuestreo (denotada por (S)).

Para todas estas configuraciones de multiclasificadores se probaron tanto la versión podada *(P)* como la no podada *(U)* de los clasificadores base, salvo en el caso de Random Forests, ya que en este método nunca se utiliza poda [11]. Es por eso que la implementación en WEKA de este método, no permite la utilización de árboles podados.

El método que se propone en este capítulo, $\mathcal{RFW}$, tiene un único parámetro, que es el exponente de los pesos, para el cual se han usado los valores 1, 2, 3 y 4. Las tablas A.20 y A.23, muestran las tasas de acierto para $\mathcal{RFW}$ usando estos cuatro valores exponente. La tabla A.20 muestra los resultados para árboles podados, y la tabla A.23 para árboles sin podar. Las tablas A.21, A.22, A.24 y A.25 muestran las tasas de acierto para los multiclasificadores contra los que compite $\mathcal{RFW}$. Las tablas A.21 y A.22 ofrecen los resultados para el caso de árboles podados, y las tablas A.24 y A.25 para los no podados.

A su vez, las tablas 5.1 y 5.2 comparan las versiones de los multiclasificadores $\mathcal{RFW}$ contra el resto de multiclasificadores considerados. La tabla 5.1 analiza los multiclasificadores con árboles podados, y la tabla 5.2 los no podados. Cada celda de estas tablas muestra el número de victorias, empates y derrotas de cada versión de $\mathcal{RFW}$ al comparar en cada conjunto de datos el multiclasificador de la fila con el $\mathcal{RFW}$ de la columna. Se han marcado en negrita los casos en que $\mathcal{RFW}$ tiene más victorias que el otro método contra el que se compara. Las columnas $\mathcal{RFW}i$ representan la versión de $\mathcal{RFW}$ con exponente $p = i$.

A la luz de estas tablas, $\mathcal{RFW}$ obtiene más victorias que derrotas con Bagging y las versiones de Random Subspaces; mientras que al compararlo con AdaBoost y MultiBoost, ya depende del exponente elegido para los pesos: $\mathcal{RFW}$ podado con exponente 3 o 4 obtiene más victorias que derrotas contra estos métodos de Boosting, mientras que $\mathcal{RFW}$ sin podar obtiene más victorias que los métodos de Boosting cuando maneja exponentes 2, 3 y 4.

Como en capítulos anteriores, se ha intentado valorar si estos resultados son significativos desde el punto de vista estadístico utilizando dos técnicas:

1. Por un lado se ha utilizado el *Sign test* [26] (sección 2.3.1), como en ocasiones anteriores, con 62 conjuntos de datos, es posible decir que un método es mejor que otro con un nivel de significación del %5 si el número de victorias mas la mitad del número de empates alcanza la cifra crítica de 39.

   En las tablas 5.1 y 5.2 se ha distinguido mediante el símbolo «•» aquellos casos en los que según este criterio un método es superior a otro. El símbolo aparece a la izquierda (junto al número de victorias) cuando la versión de $\mathcal{RFW}$ es superior, y a la derecha (junto al número de derrotas) cuando el método contra el que se compara $\mathcal{RFW}$ es el que tiene resultados significativamente mejores.

   En el caso de $\mathcal{RFW}$ con árboles podados se observa que ninguna versión de $\mathcal{RFW}$ llega a ser significativamente peor que cualquiera de los métodos con los que es comparada. Es notable que $\mathcal{RFW}1$ es el método que peor funciona, pues el número de derrotas frente a las versiones de Boosting utilizadas se acerca al valor 39. Según el test, todas las versiones de $\mathcal{RFW}$ son significativamente mejores que Bagging y las dos configuraciones de Random Subspaces.

   En el caso de $\mathcal{RFW}$ con árboles sin podar, sólo hay una sola ocasión en la que $\mathcal{RFW}$ es significativamente peor ($\mathcal{RFW}1$ vs. AdaBoost(S)). Nuevamente, los resultados con $\mathcal{RFW}1$ son los peores, pues también el número de derrotas frente al resto de métodos de Boosting quedan cerca de las 39 que marca el test. Todas las versiones de $\mathcal{RFW}$ son mejores significativamente que Bagging, y — salvo en el caso Subspaces(50 %) vs. $\mathcal{RFW}1$ — también mejores que las dos versiones de Random Subspaces. A medida que el exponente de $\mathcal{RFW}$ sube el método mejora, y en los valores 3 y 4 es incluso capaz de superar significativamente alguna versión de Boosting (MultiBoost(W)) o de quedarse muy cerca de las 39 victorias.

   En la sección anterior ya se comentó que $\mathcal{RFW}$ con exponente uno no proporciona diversidad suficiente, por el contrario, facilita la aparición de árboles similares. En este sentido, los resultados obtenidos son consistentes con este hecho.

2. Por otro lado, se han contabilizado las victorias, empates y derrotas significativas mediante la versión corregida del *Resampled t-test* [85] (sección 2.3.1), también para un nivel de significación del 5 %. Las tablas A.21, A.22, A.24 y A.25 muestran a la derecha de la tasa de acierto de cada método hasta cuatro signos «•» ó «◦» correspondientes a los exponentes $p = 1 \ldots 4$ utilizados en las configuraciones de $\mathcal{RFW}$. Los signos «◦» indican que el método de la columna pierde significativamente contra la configuración de $\mathcal{RFW}$ correspondiente al valor de p en el que esté colocado. Por el contrario, «•» indica que el método gana significativamente a esa configuración de $\mathcal{RFW}$.

Tabla 5.1: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	• **46**-1-15	• **47**-1-14	• **45**-1-16	• **44**/1/17
Subspaces (50 %)	• **38**-2-22	• **46**-2-14	• **46**-3-13	• **44**-2-16
Subspaces (75 %)	• **51**-2-9	• **54**-2-6	• **52**-3-7	• **52**-2-8
AdaBoost (W)	26-1-**35**	**31**-1-30	**32**-1-29	**33**-2-27
AdaBoost (S)	26-1-**35**	30-2-30	**31**-2-29	**31**-1-30
MultiBoost (W)	23-1-**38**	29-1-**32**	**31**-2-29	**31**-2-29
MultiBoost (S)	24-2-**36**	30-1-**31**	**32**-1-29	29-2-**31**

Tabla 5.2: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	• **44**-1-17	• **51**-1-10	• **51**-1-10	• **49**-2-11
Random Forest	27-3-**32**	**36**-4-22	• **40**-3-19	• **41**-3-18
Subspaces (50 %)	**31**-1-30	• **43**-1-18	• **48**-2-12	• **47**-2-13
Subspaces (75 %)	• **50**-1-11	• **56**-1-5	• **55**-1-6	• **55**-2-5
AdaBoost (W)	24-2-**36**	**32**-2-28	**32**-2-28	**35**-1-26
AdaBoost (S)	22-2-**38** •	**33**-2-27	**33**-3-26	**35**-1-26
MultiBoost (W)	23-2-**37**	**35**-2-25	• **40**-3-19	• **39**-1-22
MultiBoost (S)	24-2-**36**	**32**-3-27	**35**-2-25	**36**-1-25

Las tablas 5.3 y 5.4 resumen estos resultados, contabilizando el número de victorias, empates y derrotas significativas del método $\mathcal{RFW}$ en la columna contra el método de la fila. Se han remarcado en negrita las celdas en las que $\mathcal{RFW}$ acumula un número superior de victorias significativas que de derrotas.

En el caso de los árboles podados se que observa $\mathcal{RFW}$ gana siempre contra Bagging y las dos configuraciones de Random Subspaces, y nunca con las cuatro configuraciones de Boosting. En el caso de los árboles sin podar, $\mathcal{RFW}$ también gana siempre contra Bagging y los Random Subspaces, pero además gana siempre contra Random Forests, aunque en este último caso predominan los empates.

Hay algunas configuraciones de Boosting que perderían en este cómputo contra algunas configuraciones de $\mathcal{RFW}$. Estas configuraciones tendrían siempre exponente superior a uno.

Tabla 5.3: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados y el *Resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**13-46-3**	**13-47-2**	**15-44-3**	**13-44-5**
Subspaces (50 %)	**12-49-1**	**11-51-0**	**11-50-0**	**13-48-1**
Subspaces (75 %)	**18-44-0**	**23-39-0**	**21-41-0**	**21-40-1**
AdaBoost (W)	7-45-10	9-44-9	8-44-10	8-43-11
AdaBoost (S)	5-47-10	8-45-9	7-45-10	8-42-12
MultiBoost (W)	2-53-7	4-51-7	5-48-9	3-50-9
MultiBoost (S)	1-55-6	3-52-7	3-50-9	1-52-9

Tabla 5.4: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados y el *Resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**14-48-0**	**16-46-0**	**19-43-0**	**16-46-0**
Random Forests	**5-55-2**	**6-55-1**	**8-52-2**	**5-55-2**
Subspaces (50 %)	**13-49-0**	**12-50-0**	**13-48-1**	**11-50-1**
Subspaces (75 %)	**16-46-0**	**23-39-0**	**23-39-0**	**22-39-1**
AdaBoost (W)	6-49-7	**9-47-6**	**8-47-7**	**8-47-7**
AdaBoost (S)	6-49-7	**10-45-7**	**7-49-6**	7-48-7
MultiBoost (W)	3-55-4	**6-52-4**	4-53-5	3-53-6
MultiBoost (S)	2-55-5	4-54-4	3-54-5	2-54-6

Se han calculado, además, los rankings promedios [26] (sección 2.3.2) correspondientes a los 23 métodos considerados (podados, junto no podados).

Los resultados se muestran en la tabla 5.5. Puede verse que:

1. Los cuatro primeros métodos, se corresponden con configuraciones de $\mathcal{RFW}$. Este punto es especialmente remarcable dada la simplicidad del método.

2. De entre estas cuatro configuraciones, las tres primeras utilizan árboles no podados. Siempre que el exponente ha sido mayor que uno la configuración sin podar ha ocupado mejor posición que la configuración podada. Este resultado era en cierta medida predecible debido a que, en general, los árboles no podados incrementan la diversidad [4]. De hecho, un método como Random Forests, que se apoya en una idea similar a la de $\mathcal{RFW}$, usa únicamente árboles sin podar. Sin embargo, Bagging y Random Subspaces (75 %) no mejoran con la versión sin podar, si bien las configuraciones de estos dos métodos, los cuales ocupan las cuatro últimas posiciones, presentan escasas diferencias entre la versión podada y la sin podar.

3. El exponente que mejor funciona en $\mathcal{RFW}$ es el 3, seguido del 4, 2 y 1.

El test de Nemenyi [26] (sección 2.3.1) permite usar los rankings promedios para determinar qué métodos son significativamente distintos. Según este test, cuando se tienen 62 conjuntos de datos y 23 métodos, ha de existir una diferencia por el valor crítico de 4.405 en el ranking promedio para poder decir que un método es mejor que otro con un un nivel de significación del 5 %.

En la tabla 5.5 una línea horizontal separa los métodos que según este test no son significativamente peores que el que ocupa la primera posición. Se observa que hay gran cantidad de métodos por encima de dicha línea, lo cual es lógico, en cuanto este test es «*en general conservador y podría tener poca potencia*» [26]. Es más, este test ajusta ese valor crítico de 4.405 pensando en comparar cada par posible de clasificadores, en este caso supondría $23 \times 22/2 = 253$ comparaciones. Por ello, a medida que crece el número de métodos, es difícil apreciar diferencias mediante el mismo.

Existen tests más potentes orientados a comparar cada clasificador con un clasificador de control, en lugar de compararlos entre sí por pares. Pero no se ha recurrido a este tipo de tests, porque tampoco hay razones objetivas que permitan determinar cuál sería ese clasificador de referencia.

En cualquier caso, es interesante observar que por debajo de esta línea horizontal están las configuraciones de Bagging, Random Subspaces y $\mathcal{RFW}$ con exponente uno, lo cual es congruente con todos los resultados anteriores.

Finalmente, en la tabla 5.6 aparece el ranking por la diferencia entre victorias y derrotas significativas según la versión corregida del *Resampled t-test* [85] (sección 2.3.2). Si bien su validez es discutible [26], se aprecian algunos efectos que respaldan algunas de las conclusiones ya extraídas:

1. Bagging y las configuraciones de Random Subspaces siguen ocupando los peores lugares.

Tabla 5.5: Ranking promedio de todos los métodos considerados en la validación de $\mathcal{RFW}$.

	Ranking Promedio	Método
1	8.02	$\mathcal{RFW}3$ Ensemble *(U)*
2	8.65	$\mathcal{RFW}4$ Ensemble *(U)*
3	8.80	$\mathcal{RFW}2$ Ensemble *(U)*
4	10.11	$\mathcal{RFW}3$ Ensemble *(P)*
5	10.16	MultiBoost-S *(P)*
6	10.18	MultiBoost-W *(P)*
7	10.37	$\mathcal{RFW}4$ Ensemble *(P)*
8	10.45	$\mathcal{RFW}2$ Ensemble *(P)*
9	10.50	MultiBoost-S *(U)*
10	10.93	AdaBoost-S *(P)*
11	11.19	AdaBoost-W *(P)*
12	11.35	AdaBoost-S *(U)*
13	11.39	Random-Forest
14	11.55	MultiBoost-W *(U)*
15	11.95	AdaBoost-W *(U)*
16	12.80	$\mathcal{RFW}1$ Ensemble *(P)*
17	12.82	$\mathcal{RFW}1$ Ensemble *(U)*
18	12.94	Random-Subspaces-50 % *(U)*
19	14.64	Random-Subspaces-50 % *(P)*
20	15.86	Bagging *(P)*
21	16.06	Bagging *(U)*
22	17.60	Random-Subspaces-75 % *(P)*
23	17.68	Random-Subspaces-75 % *(U)*

2. Las versiones de $\mathcal{RFW}$ no podadas ocupan mejores posiciones que las podadas.

3. Todavía es posible encontrar dos métodos $\mathcal{RFW}$ entre los cuatro mejores.

Sin embargo, en este ranking da la impresión de que los exponentes que mejor funcionan en $\mathcal{RFW}$ son 2, seguido de 3, 1 y 4.

5.3.1. Robustez

Una propiedad importante para un método clasificador, es su comportamiento ante datos con ruido. A fin de testar esta propiedad, se ha introducido ruido artificial en el atributo de la clase de la misma forma que en [29]. Para cada conjunto de datos, se selecciona aleatoriamente un porcentaje de instancias de entrenamiento, en las cuales se cambia el valor de la clase. En estos conjuntos

Tabla 5.6: Ranking por la diferencia entre victorias y derrotas significativas utilizando todos los métodos considerados en la validación de $\mathcal{RFW}$ (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

	V−D	V	D	Método
1	164	210	46	$\mathcal{RFW}2$ Ensemble *(U)*
2	147	198	51	MultiBoost-S *(P)*
3	143	202	59	MultiBoost-W *(P)*
4	134	194	60	$\mathcal{RFW}3$ Ensemble *(U)*
4	134	193	59	MultiBoost-S *(U)*
6	125	218	93	AdaBoost-S *(P)*
7	109	179	70	MultiBoost-W *(U)*
8	100	218	118	AdaBoost-W *(P)*
9	97	165	68	$\mathcal{RFW}1$ Ensemble *(U)*
10	96	210	114	AdaBoost-S *(U)*
11	87	164	77	$\mathcal{RFW}4$ Ensemble *(U)*
12	77	162	85	$\mathcal{RFW}2$ Ensemble *(P)*
12	77	203	126	AdaBoost-W *(U)*
14	44	158	114	$\mathcal{RFW}3$ Ensemble *(P)*
15	19	136	117	$\mathcal{RFW}1$ Ensemble *(P)*
16	16	147	131	Random-Forest
17	1	142	141	$\mathcal{RFW}4$ Ensemble *(P)*
18	-148	112	260	Random-Subspaces-50 % *(U)*
19	-187	98	285	Random-Subspaces-50 % *(P)*
20	-240	85	325	Bagging *(U)*
21	-252	84	336	Bagging *(P)*
22	-337	56	393	Random-Subspaces-75 % *(U)*
23	-406	46	452	Random-Subspaces-75 % *(P)*

Tabla 5.7: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados, con un error artificial del 10 %, y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**35**-1-26	• **46**-1-15	• **45**-1-16	• **44**/1/17
Subspaces (50 %)	**37**-1-24	• **49**-2-11	• **50**-1-11	• **45**/1/16
Subspaces (75 %)	• **53**-3-6	• **56**-1-5	• **57**-1-4	• **54**/1/7
AdaBoost (W)	• **54**-0-8	• **55**-0-7	• **53**-2-7	• **53**/0/9
AdaBoost (S)	• **52**-0-10	• **55**-0-7	• **55**-0-7	• **54**/1/7
MultiBoost (W)	• **45**-0-17	• **47**-0-15	• **49**-0-13	• **48**/0/14
MultiBoost (S)	• **47**-0-15	• **51**-0-11	• **52**-0-10	• **48**/0/14

de datos modificados se ha procedido a hacer la misma validación experimental que se hizo para los datos sin ruido.

Las tablas de la A.26 a la A.37 muestran las tasas de acierto para todos los métodos considerados. El grupo de tablas de la A.26 a la A.31 se corresponden con los resultados para tasas de error del 10 %, mientras que el grupo de tablas de la A.32 a la A.37 son las tasas de acierto para un error artificial del 20 %.

En estos grupos de tablas se repite el mismo esquema seguido para las tablas de tasas de acierto con los conjuntos de datos sin error. Es decir, para cada uno hay dos subgrupos de tres tablas, el primero dedicado a los multiclasificadores con métodos base podados, y el segundo a los no podados. Dentro de cada subgrupo, la primera tabla se dedica a los métodos $\mathcal{RFW}$ la segunda a Bagging y Random-Subspaces, y la tercera a Boosting.

Estas tablas están a su vez resumidas en las tablas 5.7, 5.8, 5.9, y 5.10, que muestran las victorias y derrotas correspondientes a cada versión de $\mathcal{RFW}$ sobre cada método de referencia. En estas tablas, al igual que en las tablas para el caso sin ruido, la negrita marca el método ganador, y el símbolo «•», si esa victoria es significativa (nivel de significación del 5 %) usando el criterio de las 39 victorias para 62 conjuntos de datos que marcaba el *Sign test* [26].

Tabla 5.8: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados, con un error artificial del 10 %, y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**37**-0-25	• **48**-0-14	• **49**-0-13	• **51**-0-11
Random Forest	**35**-0-27	• **41**-0-21	• **46**-0-16	• **44**-1-17
Subspaces (50 %)	27-1-**34**	**37**-0-25	**37**-0-25	**33**-0-29
Subspaces (75 %)	• **47**-0-15	• **51**-1-10	• **54**-0-8	• **53**-0-9
AdaBoost (W)	• **52**-0-10	• **57**-0-5	• **59**-0-3	• **58**-0-4
AdaBoost (S)	• **51**-0-11	• **54**-1-7	• **55**-0-7	• **57**-0-5
MultiBoost (W)	• **44**-1-17	• **54**-0-8	• **55**-1-6	• **54**-0-8
MultiBoost (S)	• **44**-0-18	• **48**-0-14	• **50**-0-12	• **50**-1-11

Tabla 5.9: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados, con un error artificial del 20 %, y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	• **39**-0-23	• **42**-0-19	• **46**-0-15	• **45**-0-16
Subspaces (50 %)	**33**-0-29	• **42**-0-19	• **40**-2-19	• **43**-0-18
Subspaces (75 %)	• **48**-1-13	• **53**-1-7	• **51**-1-9	• **49**-1-11
AdaBoost (W)	• **60**-1-1	• **60**-0-1	• **61**-0-0	• **61**-0-0
AdaBoost (S)	• **59**-0-3	• **59**-0-2	• **60**-0-1	• **60**-0-1
MultiBoost (W)	• **51**-1-10	• **56**-0-5	• **58**-0-3	• **56**-0-5
MultiBoost (S)	• **50**-0-12	• **54**-0-7	• **55**-0-6	• **54**-2-5

Tabla 5.10: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados, con un error artificial del 20 %, y el *Sign test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**33**-0-29	• **42**-1-18	• **48**-0-13	• **43**-0-18
Random Forest	• **41**-1-20	• **45**-0-16	• **44**-0-17	• **41**-0-20
Subspaces (50 %)	22-0-**40** •	**31**-0-30	28-0-**33**	30-1-30
Subspaces (75 %)	• **39**-0-23	• **45**-0-16	• **48**-0-13	• **49**-0-12
AdaBoost (W)	• **58**-0-4	• **58**-0-3	• **58**-0-3	• **59**-0-2
AdaBoost (S)	• **56**-0-6	• **58**-0-3	• **59**-0-2	• **60**-0-1
MultiBoost (W)	• **51**-1-10	• **55**-1-5	• **58**-0-3	• **56**-1-4
MultiBoost (S)	• **44**-0-18	• **52**-0-9	• **54**-0-7	• **52**-0-9

Para un error del 10 %:

- Cuando se trata de árboles podados (tabla 5.7) $\mathcal{RFW}$ siempre gana, y ademas salvo con $\mathcal{RFW}1$ contra Bagging y Random-Subspaces 50 %, estas victorias son siempre significativas. Incluso en estos dos casos, el número de victorias mas la mitad de los empates de $\mathcal{RFW}1$ es muy próximo a 39.
- Cuando se trata de árboles no podados (tabla 5.8) los resultados son algo peores, $\mathcal{RFW}1$ pierde únicamente con Random-Subspaces 50 %, pero no significativamente. Las victorias son siempre significativas salvo, $\mathcal{RFW}1$ contra Bagging y Random Forests, y cualquier version de $\mathcal{RFW}$ contra Random-Subspaces 50 %.

Tabla 5.11: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados, con un error artificial del 10 %, y el *Resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**11-49-2**	**14-47-1**	**12-49-1**	**11-50-1**
Subspaces (50 %)	**11-51-0**	**11-51-0**	**10-50-1**	**10-51-1**
Subspaces (75 %)	**18-44-0**	**21-41-0**	**20-41-1**	**20-40-2**
AdaBoost (W)	**29-32-1**	**31-31-0**	**33-28-1**	**30-31-1**
AdaBoost (S)	**32-30-0**	**33-29-0**	**34-28-0**	**31-30-1**
MultiBoost (W)	**20-41-1**	**21-40-1**	**20-41-1**	**21-40-1**
MultiBoost (S)	**18-42-2**	**22-39-1**	**20-41-1**	**19-42-1**

Tabla 5.12: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados, con un error artificial del 10 %, y el *resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**13-46-3**	**17-44-1**	**17-45-0**	**10-52-0**
Random Forests	**13-48-1**	**15-47-0**	**16-46-0**	**16-46-0**
Subspaces (50 %)	**10-47-5**	**10-50-2**	**10-49-3**	**9-50-3**
Subspaces (75 %)	**20-38-4**	**22-38-2**	**26-34-2**	**22-39-1**
AdaBoost (W)	**28-34-0**	**30-32-0**	**30-32-0**	**27-35-0**
AdaBoost (S)	**26-36-0**	**29-33-0**	**29-33-0**	**28-34-0**
MultiBoost (W)	**18-44-0**	**21-41-0**	**22-40-0**	**20-42-0**
MultiBoost (S)	**16-46-0**	**19-43-0**	**19-43-0**	**16-46-0**

Por tanto:

- Comparándolo con los otros métodos, se comporta relativamente mejor con el 10 % de ruido que sin ruido.
- El exponente $p = 1$ sigue siendo la peor opción.
- En el caso de los no podados Random-Subspaces 50 %, es el único método que no obtiene siempre resultados significativamente peores que $\mathcal{RFW}$.

Tabla 5.13: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base podados, con un error artificial del 20 %, y el *Resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**12-47-3**	**14-46-2**	**13-47-2**	**11-49-2**
Subspaces (50 %)	**12-50-0**	**12-49-1**	**11-50-1**	**9-51-2**
Subspaces (75 %)	**17-45-0**	**17-45-0**	**16-45-1**	**13-48-1**
AdaBoost (W)	**40-22-0**	**39-23-0**	**41-21-0**	**40-22-0**
AdaBoost (S)	**39-23-0**	**41-21-0**	**43-19-0**	**42-20-0**
MultiBoost (W)	**29-33-0**	**31-31-0**	**27-35-0**	**25-37-0**
MultiBoost (S)	**26-35-1**	**27-34-1**	**26-36-0**	**23-39-0**

Cuando el error es del 20 %:

- Para el caso de árboles podados (tabla 5.9), $\mathcal{RFW}$ también gana siempre, pero sólo hay una única victoria no significativa, contra Random-Subspaces 50 %.
- Para no podados (tabla 5.10), Random-Subspaces 50 % logra ganar dos veces a métodos $\mathcal{RFW}$ y en uno de los casos significativamente (contra $\mathcal{RFW}1$), empata con $\mathcal{RFW}4$, y casi empata con $\mathcal{RFW}2$. Bagging pierde de manera no sifnificativa contra $\mathcal{RFW}1$. En el resto de casos $\mathcal{RFW}$ siempre gana significativamente.

Por ello, parece que a medida que va aumentando el error, el método de los Random Subspaces 50 %, el que más información pierde en entrenamiento, va ganando terreno. Si bien este método sólo es competitivo con $\mathcal{RFW}$ en presencia de ruido. Lo contrario ha ocurrido con las ocho configuraciones de Boosting analizadas, las cuales eran competitivas con $\mathcal{RFW}$ cuando no había ruido, pero sin embargo pierden de manera significativa cuando sí que lo hay. En definitiva:

> *$\mathcal{RFW}$ destaca por su gran robustez frente al resto de métodos clasificadores considerados.*

Además, $\mathcal{RFW}$ con exponente uno parece la peor opción tanto para árboles podados como sin podar.

Tabla 5.14: Comparación de $\mathcal{RFW}$ con otros métodos, utilizando clasificadores base no podados, con un error artificial del 20 %, y el *Resampled t-test*.

Método	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
Bagging	**12-47-3**	**14-47-1**	**13-48-1**	**14-47-1**
Random Forests	**16-43-3**	**19-42-1**	**20-42-0**	**20-42-0**
Subspaces (50 %)	6-49-7	**7-50-5**	**7-50-5**	6-50-6
Subspaces (75 %)	**18-39-5**	**20-39-3**	**17-41-4**	**16-44-2**
AdaBoost (W)	**33-29-0**	**36-26-0**	**37-25-0**	**37-25-0**
AdaBoost (S)	**33-29-0**	**37-25-0**	**36-26-0**	**39-23-0**
MultiBoost (W)	**23-39-0**	**27-35-0**	**27-35-0**	**26-36-0**
MultiBoost (S)	**18-44-0**	**23-39-0**	**23-39-0**	**22-40-0**

Las tablas 5.11, 5.12, 5.13 y 5.14 muestran el mismo análisis pero considerando únicamente las diferencias significativas según la versión corregida del *Resampled t-test* [85] con nivel de significación del 5 %. En las tablas se han marcado en negrita las celdas en las que el número de victorias significativas supera al de derrotas. Las tablas muestran que tanto para un error del 10 % como del 20 % todas las versiones de $\mathcal{RFW}$ obtienen más victorias significativas que derrotas sobre cualquiera de los métodos comparados, tanto si se usan árboles podados, como no podados, salvo únicamente dos casos que aparecen en la tabla 5.14, que son Random-Subspaces 50 % contra $\mathcal{RFW}1$, en el que $\mathcal{RFW}1$ pierde por un conjunto, y $\mathcal{RFW}4$ también contra Random-Subspaces 50 %, que da lugar a un empate.

Para ver en qué conjuntos de datos ocurren esas diferencias significativas las tablas A.27, A.28, A.30, A.31, A.33, A.34, A.36 y A.37 utilizan los signos «○» y «●». El signo «○» indica que la versión de $\mathcal{RFW}$ con el exponente p correspondiente a la columna, gana significativamente al método de la fila; mientras que «●» indica que pierde significativamente.

Aunque es más difícil de observar que con las tablas correspondientes al *Sign test*, las tablas de diferencias significativas respaldan las conclusiones a las que ya se había llegado. Esto es, en presencia de error:

1. Las diferencias con las cuatro configuraciones de Boosting a favor de $\mathcal{RFW}$ se hacen más grandes.

2. Las diferencias con Random Subspaces 50 % van desapareciendo o incluso invirtiéndose.

Sin embargo, no está claro que el peor valor del exponente p sea uno.

Para finalizar el estudio de la robustez de $\mathcal{RFW}$, se realizan los mismos dos rankings que en el caso sin ruido, es decir:

1. Ordenando los métodos por los valores del ranking promedio (tabla 5.15 para un error igual al 10 %, y tabla 5.17 para un error igual al 20 %). Como apreciación global a ambas tasas de error, conviene notar que al introducir ruido se marcan las diferencias entre los métodos, alargándose el intervalo en el que varían los rankings promedios. Se pasa de [8.02, 17.68] en el caso sin ruido (tabla 5.5) a [5.83, 19.10] para el caso del 10 % (tabla 5.16), y a [4.51, 20.45] en el caso del 20 % (tabla 5.18). Es más, se reduce el número de métodos que no son significativamente peores que el mejor. En concreto, los ocho métodos de Boosting empeoran mucho en presencia de ruido.

2. Ordenando los métodos por la diferencia entre victorias y derrotas significativas según el *t-test* (tabla 5.16 para error igual al 10 %, y tabla 5.18 para error igual al 20 %).

En el caso del error del 10 % el ranking promedio (tabla 5.15) muestra que los siete primeros métodos son versiones de $\mathcal{RFW}$. Además la línea separando

Tabla 5.15: Ranking promedio de todos los métodos considerados al analizar $\mathcal{RFW}$ con un error artificial del 10 %.

	Ranking Promedio	Método
1	5.83	$\mathcal{RFW}3$ Ensemble *(P)*
2	6.24	$\mathcal{RFW}2$ Ensemble *(P)*
3	6.73	$\mathcal{RFW}4$ Ensemble *(P)*
4	7.50	$\mathcal{RFW}3$ Ensemble *(U)*
5	8.06	$\mathcal{RFW}4$ Ensemble *(U)*
6	8.43	$\mathcal{RFW}2$ Ensemble *(U)*
7	8.63	$\mathcal{RFW}1$ Ensemble *(P)*
8	10.34	Random-Subspaces-50 % *(U)*
9	10.73	Bagging *(P)*
10	10.89	Random-Subspaces-50 % *(P)*
11	11.13	Random-Forest
12	11.63	$\mathcal{RFW}1$ Ensemble *(U)*
13	12.12	MultiBoost-W *(P)*
14	12.22	MultiBoost-S *(P)*
15	13.42	MultiBoost-S *(U)*
16	13.58	Bagging *(U)*
17	13.84	Random-Subspaces-75 % *(P)*
18	14.86	MultiBoost-W *(U)*
19	16.28	Random-Subspaces-75 % *(U)*
20	18.03	AdaBoost-S *(U)*
21	18.17	AdaBoost-S *(P)*
22	18.26	AdaBoost-W *(P)*
23	19.10	AdaBoost-W *(U)*

Tabla 5.16: Ranking por la diferencia entre victorias y derrotas significativas de todos los métodos considerados al analizar $\mathcal{RFW}$ con un error artificial del 10 % (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

	V−D	V	D	Método
1	389	408	19	$\mathcal{RFW}2$ Ensemble *(P)*
2	362	394	32	$\mathcal{RFW}3$ Ensemble *(P)*
3	340	368	28	$\mathcal{RFW}1$ Ensemble *(P)*
4	314	374	60	$\mathcal{RFW}4$ Ensemble *(P)*
5	257	319	62	$\mathcal{RFW}3$ Ensemble *(U)*
6	257	320	63	$\mathcal{RFW}2$ Ensemble *(U)*
7	228	289	61	$\mathcal{RFW}4$ Ensemble *(U)*
8	180	277	97	$\mathcal{RFW}1$ Ensemble *(U)*
9	122	269	147	Bagging *(P)*
10	93	247	154	Random-Subspaces-50 % *(P)*
11	90	253	163	Random-Subspaces-50 % *(U)*
12	-46	172	218	MultiBoost-W *(P)*
13	-53	167	220	MultiBoost-S *(P)*
14	-57	181	238	Random-Forest
15	-57	220	277	Random-Subspaces-75 % *(P)*
16	-65	172	237	Bagging *(U)*
17	-145	124	269	MultiBoost-S *(U)*
18	-187	121	308	MultiBoost-W *(U)*
19	-210	162	372	Random-Subspaces-75 % *(U)*
20	-439	38	477	AdaBoost-S *(P)*
21	-446	36	482	AdaBoost-S *(U)*
22	-447	43	490	AdaBoost-W *(P)*
23	-480	31	511	AdaBoost-W *(U)*

Tabla 5.17: Ranking promedio de todos los métodos considerados al analizar $\mathcal{RFW}$ con un error artificial del 20 %.

	Ranking Promedio	Método
1	4.51	$\mathcal{RFW}3$ Ensemble *(P)*
2	4.84	$\mathcal{RFW}2$ Ensemble *(P)*
3	5.20	$\mathcal{RFW}4$ Ensemble *(P)*
4	7.02	$\mathcal{RFW}1$ Ensemble *(P)*
5	7.97	Random-Subspaces-50 % *(P)*
6	7.98	Bagging *(P)*
7	8.61	$\mathcal{RFW}3$ Ensemble *(U)*
8	8.93	$\mathcal{RFW}4$ Ensemble *(U)*
9	9.27	$\mathcal{RFW}2$ Ensemble *(U)*
10	9.53	Random-Subspaces-50 % *(U)*
11	11.35	Random-Subspaces-75 % *(P)*
12	11.86	Random-Forest
13	12.13	$\mathcal{RFW}1$ Ensemble *(U)*
14	12.24	Bagging *(U)*
15	13.24	MultiBoost-S *(P)*
16	13.92	MultiBoost-W *(P)*
17	14.68	MultiBoost-S *(U)*
18	15.39	Random-Subspaces-75 % *(U)*
19	16.56	MultiBoost-W *(U)*
20	20.06	AdaBoost-S *(P)*
21	20.08	AdaBoost-S *(U)*
22	20.20	AdaBoost-W *(P)*
23	20.45	AdaBoost-W *(U)*

Tabla 5.18: Ranking por la diferencia entre victorias y derrotas significativas de todos los métodos considerados al analizar $\mathcal{RFW}$ con un error artificial del 20 % (V−D: Victorias−Derrotas significativas, V: Victorias significativas, D: Derrotas significativas).

	V−D	V	D	Método
1	484	495	11	$\mathcal{RFW}2$ Ensemble *(P)*
2	471	482	11	$\mathcal{RFW}1$ Ensemble *(P)*
3	451	471	20	$\mathcal{RFW}3$ Ensemble *(P)*
4	402	437	35	$\mathcal{RFW}4$ Ensemble *(P)*
5	252	372	120	Random-Subspaces-50 % *(P)*
6	246	343	97	$\mathcal{RFW}2$ Ensemble *(U)*
7	242	359	117	Bagging *(P)*
8	233	333	100	$\mathcal{RFW}3$ Ensemble *(U)*
9	220	326	106	$\mathcal{RFW}4$ Ensemble *(U)*
10	179	310	131	Random-Subspaces-50 % *(U)*
11	149	296	147	$\mathcal{RFW}1$ Ensemble *(U)*
12	125	315	190	Random-Subspaces-75 % *(P)*
13	-40	226	266	Bagging *(U)*
14	-95	202	297	Random-Forest
15	-113	174	287	MultiBoost-S *(P)*
16	-128	165	293	MultiBoost-W *(P)*
17	-137	192	329	Random-Subspaces-75 % *(U)*
18	-223	132	355	MultiBoost-S *(U)*
19	-303	104	407	MultiBoost-W *(U)*
20	-590	36	626	AdaBoost-W *(P)*
21	-591	31	622	AdaBoost-S *(P)*
22	-613	21	634	AdaBoost-W *(U)*
23	-621	21	642	AdaBoost-S *(U)*

los métodos que no son significativamente peores que el primero engloba precisamente esos siete métodos. La única versión de $\mathcal{RFW}$ que no queda entre los métodos de cabeza tiene exponente uno (que es el exponente más débil según las comparaciones hechas a través de los rankings promedios) con árboles no podados (que son más sensibles al ruido).

Se observa que los dos peores $\mathcal{RFW}$s son los $\mathcal{RFW}1$. En el resto de $\mathcal{RFW}$s las versiones no podadas quedan detrás de las podadas, menos sensibles al ruido. Conviene observar que, en el ranking obtenido en la sección anterior para el caso sin ruido (tabla 5.5), los $\mathcal{RFW}$ no podados ocupaban mejores posiciones que los podados. Luego, parece que la poda es un parámetro que conviene ajustar en función de ruido que parezca tener el conjunto. El mejor exponente de los $\mathcal{RFW}$s es tres, tanto en $\mathcal{RFW}$ podados, como en no podados.

Los Random Subspaces 50 % y Bagging *(P)* son los métodos que quedan inmediatamente por debajo de la línea. Estos métodos, cuando no había ruido estaban cerca de la cola del ranking.

Los métodos de Boosting en general descienden en el ranking con la presencia de ruido. Los métodos AdaBoost quedan al final del todo. Los métodos AdaBoost, al no reinicializar periódicamente los pesos, tienden a ser más sensibles al ruido, que los MultiBoost, lo cual explicaría por qué los métodos MultiBoost no han quedado en posiciones tan bajas.

En la tabla 5.16, al considerar sólo diferencias significativas, los resultados anteriores se refuerzan. Las ocho configuraciones de $\mathcal{RFW}$ ocupan las ocho posiciones de cabeza. La cuatro primeras configuraciones son podadas y las cuatro segundas no podadas. $\mathcal{RFW}1$ *(U)* sigue siendo la peor configuración de $\mathcal{RFW}$. Random Subspaces 50 % y Bagging siguen siendo los métodos que más se acercan a los $\mathcal{RFW}$s, y los AdaBoosts copan la cola de la clasificación. Los $\mathcal{RFW}3$ son el mejor método no podado, y el segundo mejor método podado. Por tanto, esta vez los resultados entre ambos rankings coinciden bastante.

En el caso del error del 20 % el ranking promedio (tabla 5.17) muestra una mayor penetración en los métodos de cabeza de Bagging y Random Subspaces 50 % en sus versiones no podadas, las menos sensibles al ruido, de manera que quedan por encima de la línea separadora que los equipara en términos de significación con el método de cabeza. El resto de métodos por encima de esa línea son $\mathcal{RFW}$ podados, junto con $\mathcal{RFW}3$ *(U)*. El exponente tres, parece por tanto el mejor como ocurría tanto cuando no había error, como con el error del 10 %. Las versiones de AdaBoost siguen ocupando los cuatro últimos lugares, mientras que MultiBoost se acerca un poco más a las últimas posiciones que cuando habia un 10 % de error.

Al utilizar el ranking basado en el *t-test* (tabla 5.18) para el caso del error del 20 % se aprecian tambien bastantes coincidencias con el ranking promedio. Las cuatro primeras configuraciones son $\mathcal{RFW}$*(P)* ; por otro lado Random Subspaces 50 % *(P)* y Bagging *(P)* penetran hasta las posiciones 5 y 7 respectivamente. La peor configuración de $\mathcal{RFW}$ tiene exponente uno, y las cuatro versiones de AdaBoost cierran el ranking. Las versiones de MultiBoost también descienden respecto a sus posiciones con error del 10 %.

Tabla 5.19: Comparación mediante el *sign test* de las versiones con/sin $\mathcal{RFW}$ de los multiclasificadiores de referencia considerados. V=Victorias, E=Empates, D=Derrotas.

Método	V	E	D
Bagging *(P)*	•**50**	1	11
Bagging *(U)*	•**54**	2	6
Random-Subspaces-50 % *(P)*	•**39**	3	20
Random-Subspaces-50 % *(U)*	•**44**	1	17
Random-Subspaces-75 % *(P)*	•**52**	2	8
Random-Subspaces-75 % *(U)*	•**55**	1	6
AdaBoost-W *(P)*	**33**	2	27
AdaBoost-W *(U)*	**37**	3	22
AdaBoost-S *(P)*	**32**	1	29
AdaBoost-S *(U)*	**30**	3	29
MultiBoost-W *(P)*	•**41**	3	18
MultiBoost-W *(U)*	•**44**	5	13
MultiBoost-S *(P)*	**36**	4	22
MultiBoost-S *(U)*	•**39**	3	20

5.3.2. Árboles $\mathcal{RFW}$ como clasificadores base

Es claro que cualquier multiclasificador que sea capaz de utilizar árboles de decisión como clasificadores base, podría también utilizar los árboles de un $\mathcal{RFW}$ con el mismo fin. En el análisis que se presenta en esta sección, por tanto, los métodos considerados antes como adversarios pasan a ser potenciales aliados.

El valor óptimo del exponente de un clasificador base $\mathcal{RFW}$ puede depender tanto del método multiclasificador en el que se integre, como también del conjunto de datos. A fin de poder tener un número manejable de combinaciones, se decidió utilizar un solo valor para este parámetro. El valor elegido fue uno, que es el valor que peores resultados dio en los experimentos de las secciones anteriores. La razón de usar este exponente está en reducir la posibilidad de que el *bias* introducido por el repesado de los atributos fuera suficientemente grande como para eclipsar el procedente del método multiclasificador.

La tabla 5.19 compara cada multiclasificador usando árboles de decisión puros (i.e., que no son tipo $\mathcal{RFW}$), con la versión usando árboles $\mathcal{RFW}$. Cada fila representa Victorias-Empates-Derrotas totales de la versión con $\mathcal{RFW}1$ sobre el mismo método con árboles de decisión puros. En negrita aparece el resultado mayoritario. Los símbolos «•» marcan las diferencias significativas según el *Sign test*.

En todos los casos los resultados con $\mathcal{RFW}$ son mejores que con árboles de decisión puros. Además las victorias de los multiclasificadores $\mathcal{RFW}$ resultan ser significativas según el *Sign test*, con excepción de las cuatro versiones de AdaBoost y MultiBoost-S *(P)*, quizás debido al fuerte bias que introduce

Tabla 5.20: Comparación mediante el *t-Test* de las versiones con/sin $\mathcal{RFW}$ de los multiclasificadiores de referencia considerados. V=Victorias significativas, E=Empates (i.e. no hay diferencias significativas), D=Derrotas significativas.

Método	V	E	D
Bagging *(P)*	**15**	47	0
Bagging *(U)*	**17**	45	0
Random-Subspaces-50 % *(P)*	**2**	59	1
Random-Subspaces-50 % *(U)*	1	60	1
Random-Subspaces-75 % *(P)*	**19**	43	0
Random-Subspaces-75 % *(U)*	**19**	43	0
AdaBoost-W *(P)*	**4**	57	1
AdaBoost-W *(U)*	**3**	59	0
AdaBoost-S *(P)*	**2**	60	0
AdaBoost-S *(U)*	**2**	60	0
MultiBoost-W *(P)*	**5**	57	0
MultiBoost-W *(U)*	**3**	59	0
MultiBoost-S *(P)*	**2**	60	0
MultiBoost-S *(U)*	**2**	60	0

Boosting.

La tabla 5.20 hace la misma comparación pero teniendo en cuenta únicamente las diferencias significativas según el *t-test*. Cada fila representa Victorias-Empates-Derrotas significativas de la versión con $\mathcal{RFW}1$ sobre el mismo método con árboles de decisión puros. Nuevamente, se ha marcado en negrita el resultado mayoritario.

En esta última tabla también se observa que siempre hay más victorias significativas a favor de los multiclasificadores que usan $\mathcal{RFW}$ como clasificadores base que derrotas, salvo en un único caso en el que hay empate: Random Subspaces 50 % *(U)*. La diferencia con la otra versión de Random Subspacs 50 %, tampoco es muy grande. Quizás Random Subspaces 50 % pierde demasiada información de entrenamiento, condicionando el resultado final y haciendo que haya poco margen para la mejora. En el caso de Boosting, los ocho métodos de Boosting tampoco acumulan un gran número de victorias significativas, tal y como ocurría en la tabla 5.19, el bias de Boosting puede con el de los árboles $\mathcal{RFW}$ con exponente uno.

Quizás lo más destacable de esta tabla 5.20 sea que entre todos los métodos y conjuntos de datos, los multiclasificadores de árboles $\mathcal{RFW}$ sólo pierden tres veces contra los multiclasificadores de árboles de decisión. Esto quiere decir que utilizar este nuevo tipo de árboles normalmente no conllevará el riesgo de perder acierto significativamente, sino en todo caso mantenerse en unas tasas de error parecidas, y en algunas ocasiones mejorar significativamente.

Para ver los conjuntos de datos en los que ocurren las diferencias significa-

tivas de la tabla 5.20 se pueden consultar las tablas A.38 y A.39.

Finalmente se incluyen en las tablas 5.21 y 5.22 sendos rankings de los 37 métodos testados en esta sección, es decir:

- Bagging, las dos versiones de Random Subspaces y las cuatro de Boosting con árboles de decisión podados y no podados.
- Bagging, las dos versiones de Random Subspaces y las cuatro de Boosting con árboles $\mathcal{RFW}$ de exponente uno podados y no podados.
- Los multiclasificadores $\mathcal{RFW}$ con los cuatro $p = 1 \ldots 4$, también para árboles podados y no podados.
- Random Forests.

La tabla 5.21 muestra a los métodos ordenados por el valor de su ranking promedio, mientras que la tabla 5.22 muestra a los métodos ordenados por la diferencia entre victorias y derrotas significativas. En la primera de estas tablas se ha calculado la columna *beneficio* para aquellos multiclasificadores que aparecen tanto con la versión con árboles $\mathcal{RFW}$ como con la versión con árboles de decisión. El beneficio es la diferencia en el valor del ranking promedio entre usar un tipo de clasificador base u otro. Los beneficios positivos indican que el multiclasificador mejora su ranking promedio al utilizar árboles $\mathcal{RFW}$. Se observa que todos los beneficios son positivos menos el de $\mathcal{RFW}$ AdaBoost *(P)*, y aún en este caso el valor absoluto apenas llega a uno. El resto de versiones de AdaBoost aun con un beneficio positivo, no tienen un beneficio muy grande, siendo Bagging y Random Subspaces 75 % los métodos donde se aprecian beneficios mayores, en congruencia todo ello con las tablas hasta ahora obtenidas.

La mejor configuración en la tabla 5.21 surge de la aplicación de $\mathcal{RFW}$ al mejor método con árboles de decisión puros (i.e., $\mathcal{RFW}$ MultiBoost-W *(P)*). El resto de posiciones de cabeza de esta tabla se reparten entre distintas configuraciones de $\mathcal{RFW}$ Ensemble y $\mathcal{RFW}$ MultiBoost. En las de $\mathcal{RFW}$ Ensemble los métodos no podados quedan por encima de los podados y los exponentes mejores son el 3 y el 4. Estas últimas conclusiones ya se extrajeron de la tabla 5.5.

Al considerar la tabla 5.22 que ordena los métodos por la diferencia entre victorias y derrotas significativas, se observa que todos los métodos que utilizan árboles de decisión mejoran con $\mathcal{RFW}$. En esta tabla, nuevamente, la mejor configuración es $\mathcal{RFW}$ MultiBoost-W *(P)*, siendo las cuatro primeras posiciones versiones de $\mathcal{RFW}$-MultiBoost. Los $\mathcal{RFW}$ Ensembles sin podar nuevamente quedan por delante de los no podados, aunque a diferencia del ranking anterior no quedan tan a la cabeza, y además los exponentes se ordenan de mejor a peor como 2, 3, 1, 4, frente a 3, 4, 2, 1 que se tenía antes. Por tanto, teniendo en cuenta todas las tablas $p = 3$ sigue pareciendo la mejor opción.

5.3.3. Diagramas Kappa-Error

En el capítulo anterior ya se utilizaron los diagramas Kappa-Error para analizar el posible incremento de la diversidad con los $\mathcal{DN}$s.

Tabla 5.21: Ranking promedio de los métodos considerados tomando como clasificadores base árboles puros o árboles $\mathcal{RFW}$.

	Ranking Promedio	Método	Beneficio
1	12.34	$\mathcal{RFW}$ MultiBoost-W *(P)*	5.25
2	14.02	$\mathcal{RFW}3$ Ensemble *(U)*	
3	14.80	$\mathcal{RFW}4$ Ensemble *(U)*	
4	14.85	$\mathcal{RFW}$ MultiBoost-W *(U)*	5.12
5	15.02	$\mathcal{RFW}$ MultiBoost-S *(P)*	2.65
6	15.60	$\mathcal{RFW}2$ Ensemble *(U)*	
7	15.77	$\mathcal{RFW}$ MultiBoost-S *(U)*	2.69
8	16.16	$\mathcal{RFW}$ Random-Subspaces-75 % *(U)*	12.70
9	16.29	$\mathcal{RFW}$ Bagging *(U)*	10.15
10	16.75	$\mathcal{RFW}$ AdaBoost-W *(P)*	1.98
11	16.84	$\mathcal{RFW}$ Bagging *(P)*	9.21
12	17.20	$\mathcal{RFW}3$ Ensemble *(P)*	
13	17.45	$\mathcal{RFW}4$ Ensemble *(P)*	
14	17.55	$\mathcal{RFW}$ AdaBoost-W *(U)*	2.25
15	17.59	MultiBoost-W *(P)*	
16	17.65	$\mathcal{RFW}$ Random-Subspaces-50 % *(U)*	4.62
17	17.68	MultiBoost-S *(P)*	
18	18.10	$\mathcal{RFW}2$ Ensemble *(P)*	
19	18.45	MultiBoost-S *(U)*	
20	18.50	$\mathcal{RFW}$ Random-Subspaces-75 % *(P)*	9.87
21	18.70	$\mathcal{RFW}$ AdaBoost-S *(U)*	0.35
22	18.73	AdaBoost-W *(P)*	
23	18.83	AdaBoost-S *(P)*	
24	19.06	AdaBoost-S *(U)*	
25	19.26	Random-Forest	
26	19.68	$\mathcal{RFW}$ AdaBoost-S *(P)*	-0.85
27	19.80	AdaBoost-W *(U)*	
28	19.97	MultiBoost-W *(U)*	
29	20.29	$\mathcal{RFW}$ Random-Subspaces-50 % *(P)*	4.27
30	21.76	$\mathcal{RFW}1$ Ensemble *(U)*	
31	21.78	$\mathcal{RFW}1$ Ensemble *(P)*	
32	22.27	Random-Subspaces-50 % *(U)*	
33	24.56	Random-Subspaces-50 % *(P)*	
34	26.05	Bagging *(P)*	
35	26.44	Bagging *(U)*	
36	28.37	Random-Subspaces-75 % *(P)*	
37	28.86	Random-Subspaces-75 % *(U)*	

Tabla 5.22: Ranking por la diferencia entre victorias (V) y derrotas (D) significativas de los métodos considerados tomando como clasificadores base árboles puros o árboles $\mathcal{RFW}$.

	V - D	V	D	Método
1	307	348	41	$\mathcal{RFW}$ MultiBoost-W *(P)*
2	271	315	44	$\mathcal{RFW}$ MultiBoost-S *(P)*
3	261	311	50	$\mathcal{RFW}$ MultiBoost-W *(U)*
4	256	295	39	$\mathcal{RFW}$ MultiBoost-S *(U)*
5	223	357	134	$\mathcal{RFW}$ AdaBoost-W *(P)*
6	189	288	99	$\mathcal{RFW}2$ Ensemble *(U)*
7	173	339	166	$\mathcal{RFW}$ AdaBoost-W *(U)*
8	157	314	157	$\mathcal{RFW}$ AdaBoost-S *(P)*
9	147	259	112	MultiBoost-S *(P)*
10	143	267	124	$\mathcal{RFW}3$ Ensemble *(U)*
11	142	264	122	MultiBoost-W *(P)*
12	142	275	133	$\mathcal{RFW}$ Bagging *(U)*
13	129	295	166	AdaBoost-S *(P)*
14	126	247	121	MultiBoost-S *(U)*
15	109	305	196	$\mathcal{RFW}$ AdaBoost-S *(U)*
16	100	234	134	MultiBoost-W *(U)*
17	90	294	204	AdaBoost-W *(P)*
18	86	229	143	$\mathcal{RFW}1$ Ensemble *(U)*
19	75	225	150	$\mathcal{RFW}4$ Ensemble *(U)*
20	73	277	204	AdaBoost-S *(U)*
21	70	235	165	$\mathcal{RFW}2$ Ensemble *(P)*
22	51	217	166	$\mathcal{RFW}$ Random-Subspaces-75 % *(U)*
23	51	257	206	$\mathcal{RFW}$ Bagging *(P)*
24	46	267	221	AdaBoost-W *(U)*
25	8	225	217	$\mathcal{RFW}3$ Ensemble *(P)*
26	-21	198	219	$\mathcal{RFW}1$ Ensemble *(P)*
27	-41	203	244	Random-Forest
28	-45	200	245	$\mathcal{RFW}$ Random-Subspaces-75 % *(P)*
29	-50	204	254	$\mathcal{RFW}4$ Ensemble *(P)*
30	-193	171	364	$\mathcal{RFW}$ Random-Subspaces-50 % *(U)*
31	-254	149	403	$\mathcal{RFW}$ Random-Subspaces-50 % *(P)*
32	-299	143	442	Random-Subspaces-50 % *(U)*
33	-355	131	486	Random-Subspaces-50 % *(P)*
34	-443	129	572	Bagging *(U)*
35	-443	138	581	Bagging *(P)*
36	-585	95	680	Random-Subspaces-75 % *(U)*
37	-696	83	779	Random-Subspaces-75 % *(P)*

En estos diagramas la posición ideal de las nubes es la esquina inferior (poco error) izquierda (mucha diversidad), lo cual es un objetivo contradictorio, en tanto que dos clasificadores con poco error tenderán a coincidir en sus predicciones (ver sección 2.3.3).

Los diagramas Kappa-Error generados corresponden a diez configuraciones, todas ellas con árboles no podados. A saber:

1. Bagging y Random Subspaces 50 %, con árboles de decisión y con árboles $\mathcal{RFW}$.

2. Las versiones con remuestreo de AdaBoost y MultiBoost, también, con árboles de decisión y con árboles $\mathcal{RFW}$.

3. Random Forests.

4. Un bosque $\mathcal{RFW}$ con exponente 3.

Los diagramas se calculan a partir de validación cruzada 5×2. Las figuras 5.3, 5.4, y 5.5 muestran las nubes correspondientes a los conjuntos de datos *segment*, *sick* y *splice* del repositorio UCI a modo ilustrativo. Los asteriscos que se han añadido a los diagramas marcan los centros de cada nube.

Al igual que en el capítulo anterior, se han calculado los centros de las nubes obtenidas a partir de los 62 conjuntos de datos. Con esos centros se han dibujado:

1. El *diagrama de Movimiento Kappa-Error* (figura 5.6), en el que cada flecha representa el cambio de posición del centro de una nube. Cada flecha corresponde, por tanto, a un conjunto de datos. En ese diagrama se han pintado de rojo las flechas que apuntan a la izquierda, y de verde las que apuntan a la derecha. Por tanto, las rojas marcan un aumento de la diversidad, mientras que las verdes lo contrario.

2. El *diagrama de Movimiento Relativo Kappa-Error* (figura 5.7), que se obtiene a partir del anterior trasladando los orígenes de las flechas al origen de coordenadas.

Cuando se comparan los métodos contra $\mathcal{RFW}3$, la dirección de las flechas es distinta en cada caso. Con Bagging las flechas tienden a apuntar hacia la izquierda, indicando que los clasificadores de $\mathcal{RFW}3$ son más diversos. Sin embargo, las flechas de AdaBoost y MultiBoost apuntan mayoritariamente en dirección contraria, indicando que los clasificadores base de los métodos de Boosting son más diversos que los de $\mathcal{RFW}3$. En la comparación con Random Forests y Random Subspaces no hay una tendencia clara en la dirección de las flechas. Si se tiene presente que $\mathcal{RFW}3$ *(U)* no ganaba significativamente a tres de los cuatro métodos de Boosting sin poda, y que sí que ganaba significativamente a las versiones sin poda de Bagging, Random Forests y Random Subspaces, podría pensarse que cuando la diversidad de $\mathcal{RFW}3$ no empeora frente a la del método comparado, $\mathcal{RFW}3$ da mejor resultado.

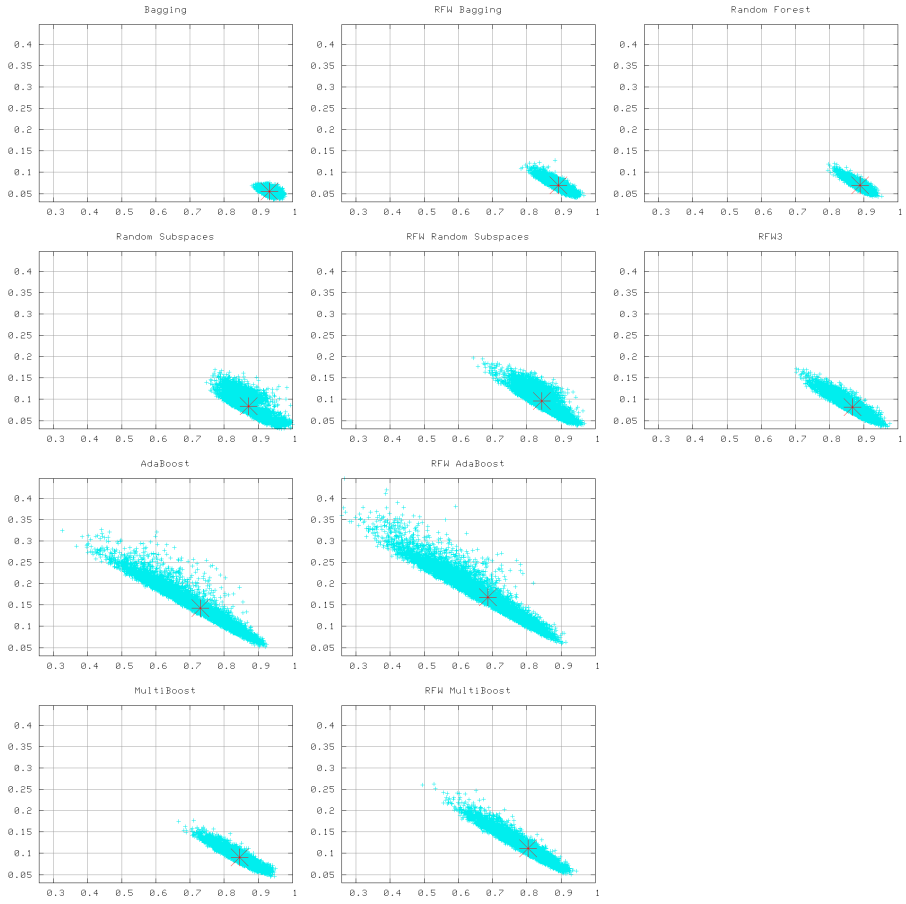


Figura 5.3: Diagramas κ-error correspondientes al estudio de los $\mathcal{RFW}$s para el conjunto *segment*.

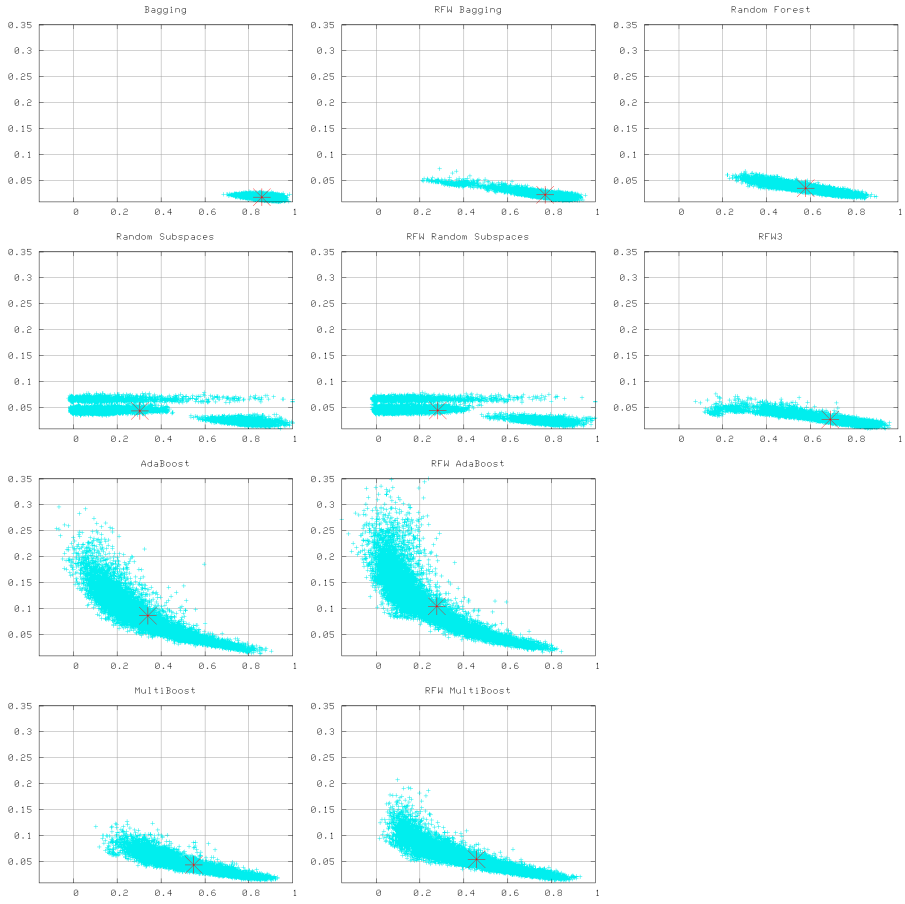


Figura 5.4: Diagramas κ-error correspondientes al estudio de los $\mathcal{RFW}$s para el conjunto *sick*.

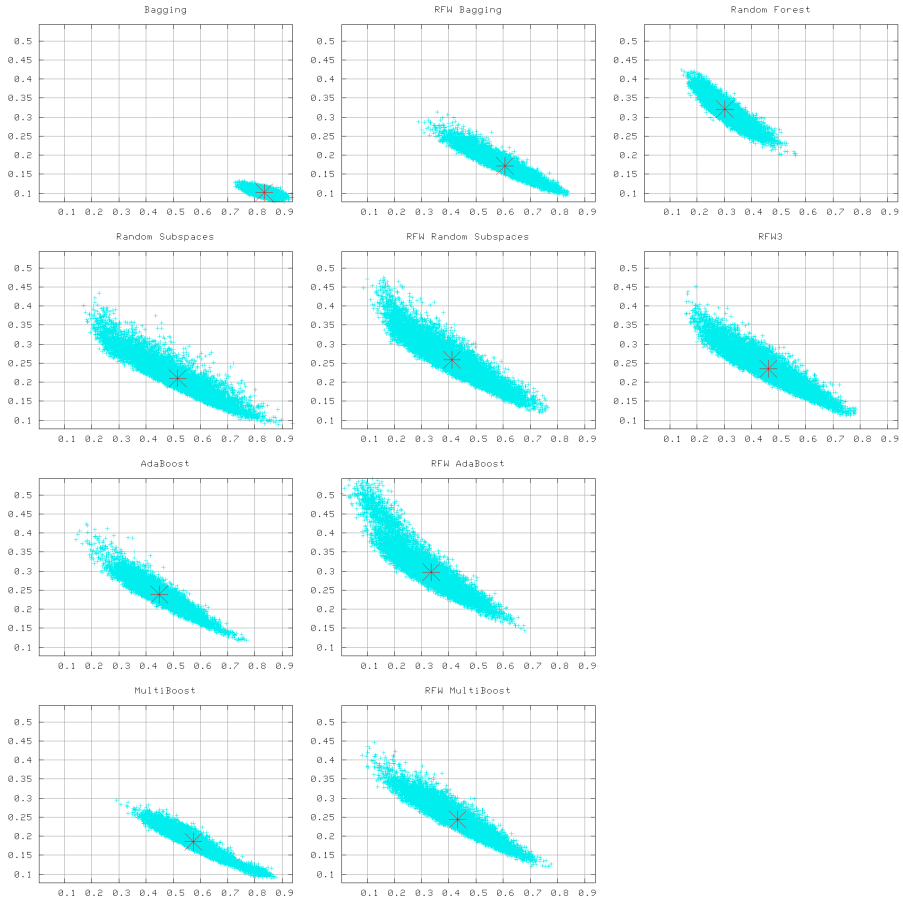


Figura 5.5: Diagramas κ-error correspondientes al estudio de los $\mathcal{RFW}$s para el conjunto *splice*.

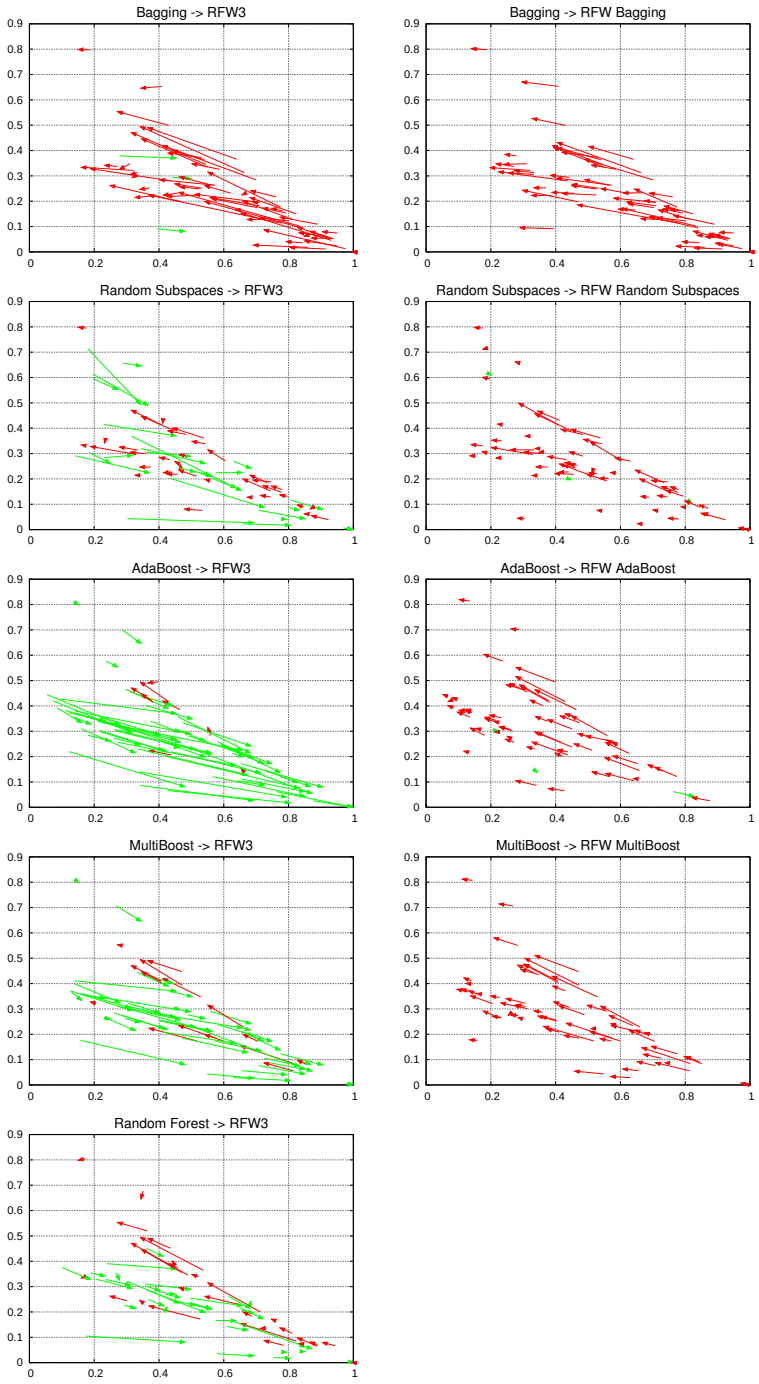


Figura 5.6: Diagramas de movimiento κ-Error correspondientes para los $\mathcal{RFW}$s.

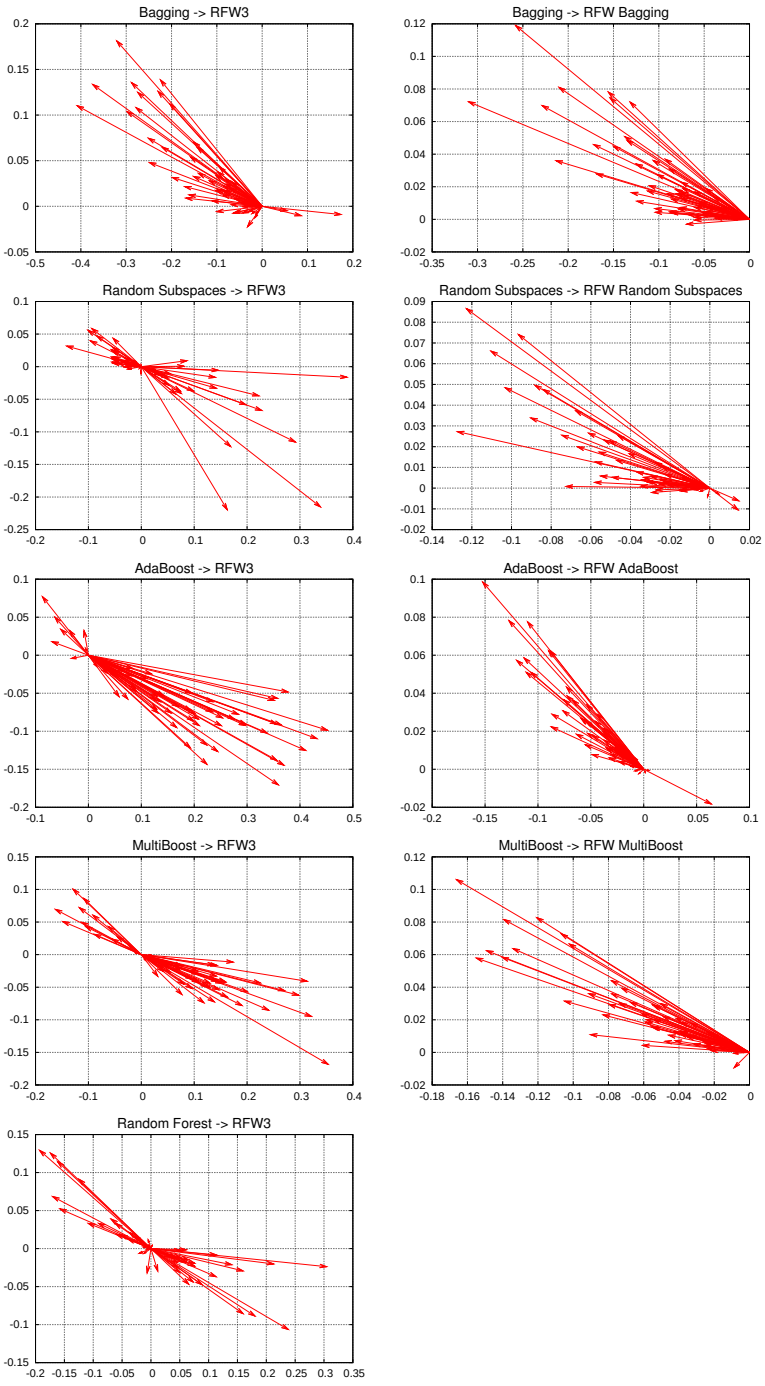


Figura 5.7: Diagramas de movimiento relativo κ-Error para los $\mathcal{RFW}$s.

Sin embargo, cuando se comparan dos versiones de un mismo multiclasificador, una con árboles de decisión puros contra otra con árboles $\mathcal{RFW}$, si que hay en todos los casos una tendencia abrumadora a que las flechas apunten a la izquierda. Lo que significa que el método gana en diversidad al sustituir los clasificadores base por árboles $\mathcal{RFW}$. Lógicamente, el aumento de diversidad ocasiona una pérdida de precisión que se nota en que las flechas además de apuntar a la izquierda, también apuntan hacia arriba. Esa mejora en la diversidad parece estar detrás de la mejora en la precisión final del multiclasificador cuando utiliza este nuevo tipo de árboles.

5.4. Influencia del parámetro

En esta pequeña sección está dedicada a experimentar la influencia del parámetro p (i.e., el exponente al que se elevan los pesos) en los multiclasificadores obtenidos.

La figura 5.8 muestra la influencia de p en el error de los multiclasificadores. Los valores considerados para el parámetro se han tomado en el intervalo $[0, 6]$, con un paso de 0,2. Los multiclasificadores de los diagramas son siempre del tipo $\mathcal{RFW}$-p y los árboles no están podados. El tamaño de los multiclasificadores es 50, y los resultados se han obtenido mediante validación cruzada 5×2.

Puede verse que un incremento del valor inicial $p = 0$, normalmente hace disminuir el error. Por ejemplo, para 61 de los 62 conjuntos de datos el error para $p = 1$ es menor que para $p = 0$. Esto es debido a que al elevar los pesos a $p = 0$ tomarán el valor 1, haciendo que todos los árboles sean idénticos.

En bastantes conjuntos de datos (e.g., *audiology*) se alcanza un mínimo en el error, y a partir de ese punto vuelve a aumentar. Para cada conjunto de datos ese mínimo es distinto. Por tanto, el exponente es un parámetro que conviene sintonizar si se quieren obtener los mejores resultados posibles.

La figura 5.9 muestra la influencia del exponente mediante diagramas de movimiento relativo kappa-error para exponentes $p = 1, \ldots, 4$. Es claro que al incrementar el valor del exponente se incrementa la diversidad. También es visible que ese aumento de la diversidad viene acompañado de un aumento del error de los clasificadores base en la mayoría de los casos. Esto confirma la necesidad de buscar un valor óptimo para el parámetro. Con $p = 1$, la diversidad de los clasificadores base es pequeña y el multiclasificador puede mejorarse aumentando la diversidad mediante el aumento del valor de p. Sin embargo, cuando la diversidad ya es suficiente, seguir aumentándola, y por tanto, aumentando también el error de los clasificadores base, deja de tener un efecto positivo en la precisión global del multiclasificador. Es decir, ha de existir algún valor intermedio de p donde el aumento del error de los clasificadores base deje de poder compensarse vía aumento de la diversidad.

El valor óptimo de p también depende del número de clasificadores base utilizados. La figura 5.10 es una gráfica que para cada valor de p (i.e., $p = 1, \ldots, 4$), muestra el porcentaje de conjuntos de datos que tienen el menor error en función del tamaño del multiclasificador.

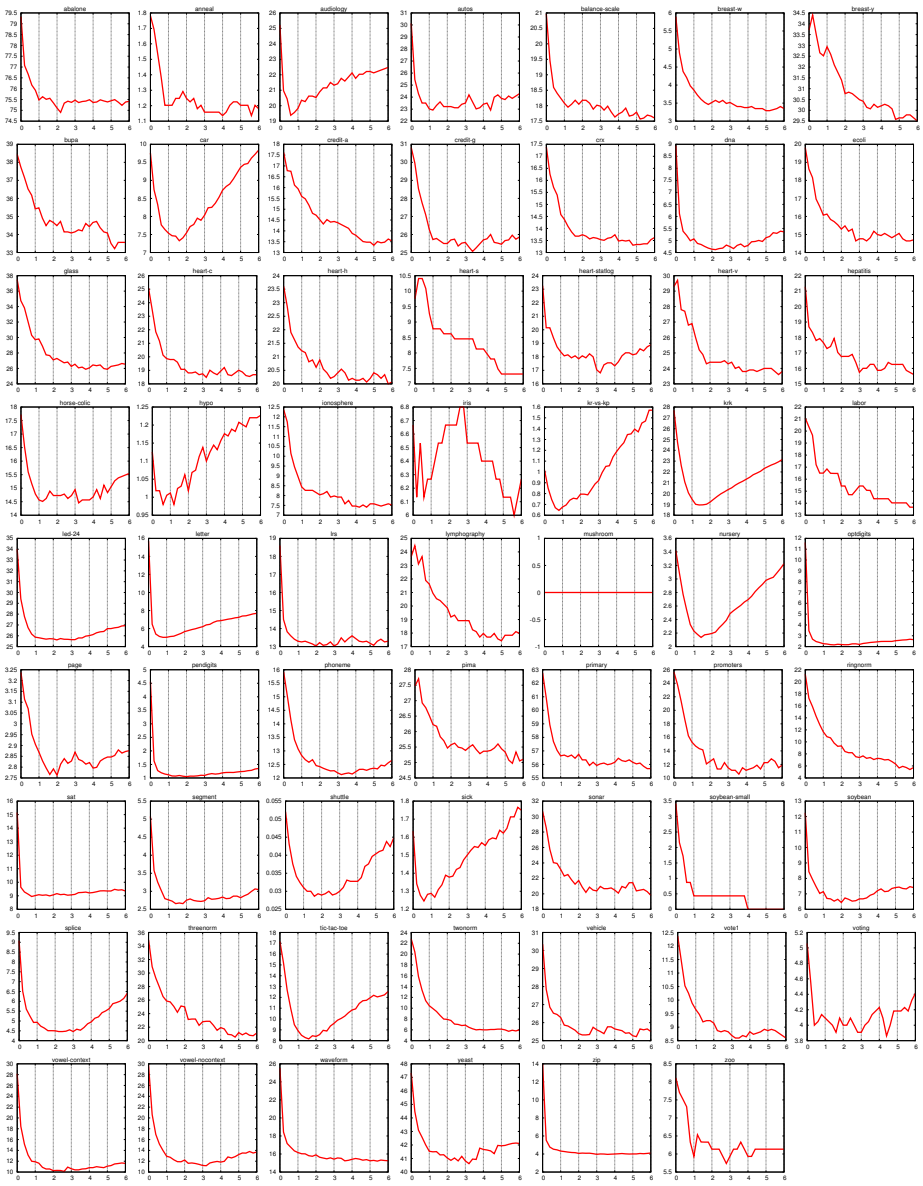


Figura 5.8: Influencia del parámetro p en el error. En cada conjunto de datos se ha hecho una gráfica en la que en el eje x representa el valor del parámetro p, y el eje y el valor del error.

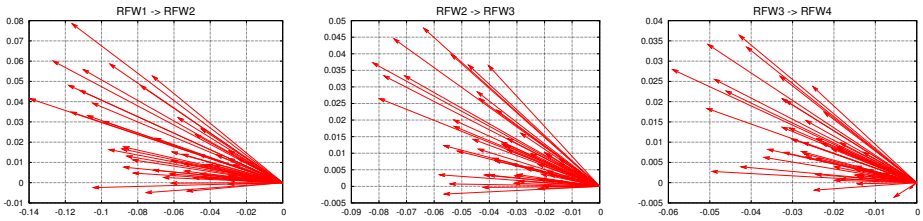


Figura 5.9: Influencia del parámetro p en los diagramas kappa-error.

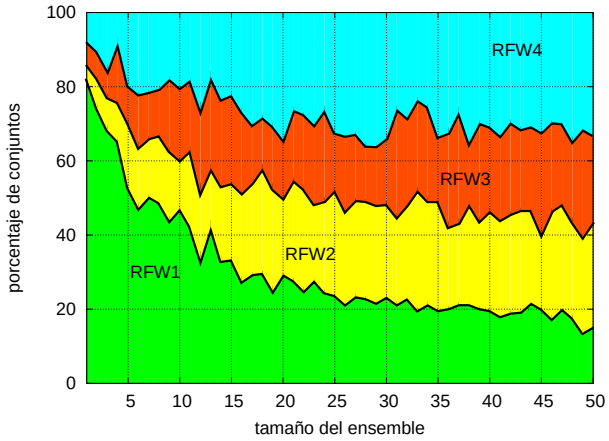


Figura 5.10: Diagrama de porcentajes para diferentes valores del parámetro p.

Por ejemplo, para el caso de que sólo haya un clasificador base, $\mathcal{RFW}1$ es la mejor configuración posible para más del 80 % de los conjuntos de datos. Sin embargo, para 50 clasificadores base, esta configuración sólamente es mejor en menos del 16 % de los conjuntos de datos, y es $\mathcal{RFW}4$ la que alcanza un porcentaje mayor [1].

Aunque es posible optimizar el valor de p para un conjunto de datos dado, es notable que en general con valores cualquiera mayores que 1 los experimentos de la sección 5.3 los resultados eran favorables a $\mathcal{RFW}$ (ver tabla 5.2).

5.5. Conclusiones

En este capítulo se ha presentado un método para la construcción de multiclasificadores usando un nuevo tipo de árbol. A pesar de su simplicidad, ha resultado ser un método competitivo con otros multiclasificadores de referencia (Bagging, Random Forests, Random Subspaces, AdaBoost y MultiBoost).

[1] En la sección 5.3 parecía algo mejor $p = 3$ para 50 árboles no podados y conjuntos de datos sin ruido, pero en ese experimento intervienen muchos más métodos que los cuatro $\mathcal{RFW}$s.

Al utilizarlo con conjuntos de datos con ruido, el método resulta aun más ventajoso. Para utilizarlo en conjuntos con ruido es mejor podar los árboles, mientras que para utilizarlo con los conjuntos de datos originales, es mejor no hacerlo.

El método tiene un único parámetro que es el exponente al que son sometidos los pesos aleatorios. Se ha aportado un estudio experimental que muestra que el aumento del exponente aumenta la diversidad de los clasificadores base, pero también aumenta el error en los mismos, por lo que aumentar el exponente por encima de 1 da buenos resultados, pero aumentarlo demasiado los degradaría. El valor óptimo del exponente dependerá del conjunto de datos y del número de clasificadores base. Así, parece que cuanto mayor sea el número de clasificadores base, mejor resultado da un exponente alto. En la validación experimental presentada es notable que los resultados obtenidos han sido en general favorables para valores exponente superiores a 1.

Al utilizar el clasificador base de este nuevo método con otros multiclasificadores, estos normalmente mejoran respecto de cuando usan árboles de decisión tradicionales, y en muchos casos las diferencias son significativas.

Al hacer los diagramas Kappa-Error y los diagramas de Movimiento Kappa-Error se aprecia una cierta relación entre las diferencia de diversidad de los métodos y el comportamiento global del multiclasificador, por lo que podría ser el incremento de diversidad la causa de la mejora.

Capítulo 6

Conclusiones y Trabajos Futuros

En el presente trabajo de tesis se han desarrollado distintas técnicas de multiclasificadores.

Cronológicamente el trabajo partía de la construcción de características, con lo que se llegó a una aproximación de construcción de características mediante Cascading para poder tratar adecuadamente conjuntos de datos nominales mediante clasificadores orientados a entradas numéricas.

Las características construidas de esta forma necesitan de un clasificador que pueda trabajar con datos nominales directamente y devolver una salida numérica, que son las probabilidades de las predicciones de cada clase; para lo cual se han utilizado árboles de decisión. El clasificador numérico con el que se probaron las características construidas mediante árboles fue un SVM.

Las pruebas se hicieron sobre 27 conjuntos de datos de la UCI y StatLib con atributos exclusivamente nominales y 57 configuraciones distintas de métodos de clasificación, incluyendo:

1. Técnicas de transformación nominal a numérico como NBF y VDM.
2. Árboles de decisión binarios o no.
3. Clasificadores SVM de función núcleo lineal.
4. Arquitecturas de dos niveles que combinarán del mayor número de formas posibles estos ingredientes, lo cual incluye —además de Cascading— Stacking y Grading.

Los mejores resultados se encontraron con las configuraciones de Cascading que utilizaban árboles de decisión binarios para construir características que luego eran usadas por SVM en el nivel superior. Es notable que estos resultados fuesen mejores que los obtenidos con otras aproximaciones computacionalmente más costosas, como Stacking y Grading.

El hecho de que los árboles fueran binarios resultó según la validación un elemento fundamental, pues el análisis que hacen los árboles no binarios de los datos nominales es tendente a resultar en árboles menos profundos, los cuales suelen tomar decisiones de ramificación que se podrían calificar de «prematuras» comparadas con las ramificaciones de un árbol binario.

Siguiendo con la construcción de características se llegó a una aproximación que construía un clasificador 1-NN distinto por cada clasificador base de un multiclasificador (i.e., Disturbing Neighbors o $\mathcal{DN}$). Este clasificador 1-NN provee una serie de nuevas características que pueden ser utilizadas en el entrenamiento del clasificador base cualquiera que sea, perteneciente a un multiclasificador, cualquiera que sea éste también. En concreto, cada $\mathcal{DN}$ tiene tres ingredientes que los diferencian entre si:

1. Un vector de booleanos indicando a qué región de Voronoi pertenece la instancia que se esté clasificando.

2. Una máscara booleana indicando qué dimensiones se han de tener en cuenta para calcular las distancias del clasificador 1-NN.

3. La predicción del propio clasificador 1-NN.

Cada $\mathcal{DN}$ en un multiclasificador suele ser distinto, porque la máscara booleana para el calculo de las distancias es aleatoria, y por tanto generalmente distinta para cada clasificador base, y porque las instancias que se utilizan para construir el 1-NN son muy pocas (10 instancias en los experimentos) y también son seleccionadas aleatoriamente.

Esta técnica está orientada a potenciar la diversidad de los clasificadores base de un multiclasificador genérico, sin que ello parezca que disminuya la tasa de acierto individual de los mismos.

Para probar esta idea se utilizaron 62 conjuntos de datos de la UCI y diversas configuraciones de los multiclasificadores de referencia. Se hicieron dos estudios para dos tipos de clasificadores base. Por un lado, con clasificadores base muy estables (e.g., SVM con función núcleo lineal), y por otro con clasificadores base más inestables (e.g., árboles de decisión). Los resultados en ambos casos apuntan a una mejoría en los multiclasificadores que usan $\mathcal{DN}$ respecto de las versiones sin $\mathcal{DN}$. Descartada experimentalmente que la mejora pudiera venir del acierto del pequeño clasificador 1-NN, la única explicación encontrada es que dicha mejora provenga del mencionado aumento de la diversidad.

Para sintetizar y condensar los resultados experimentalmente de mejora de la diversidad se diseñaron dos nuevos diagramas basados en los diagramas Kappa-Error. Estos dos nuevos diagramas se han denominado *diagramas de Movimiento Kappa-Error* y *diagramas de Movimiento Relativo Kappa-Error*, y en si mismos constituyen a nuestro juicio unas de las aportaciones interesantes de esta tesis.

Estos diagramas visualizan flechas representando la variación de la diversidad (medida mediante la estadística kappa) y el error de los clasificadores base en cada conjunto de datos. Los diagramas obtenidos confirman el aumento de la

diversidad con $\mathcal{DN}$, sin que ello conlleve por lo general un aumento apreciable del error individual de cada clasificador base.

El estudio de $\mathcal{DN}$ finalizó con un análisis de lesiones para encontrar cuál de los tres ingredientes de $\mathcal{DN}$ es el más importante, siendo el vector de booleanos que establecía la pertenencia a regiones de Voronoi el ingrediente esencial. Esto hace que la técnica $\mathcal{DN}$ esté muy ligada a que el clasificador que proporcione la diversidad sea del tipo vecinos más cercanos, en tanto es más importante que haga una división del espacio de entrada, que la propia predicción que pueda computar.

Finalmente, siguiendo con la idea de mejorar la diversidad en multiclasificadores con clasificadores base árboles, se ha presentado otra técnica llamada *Random Feature Weights* ($\mathcal{RFW}$) basada en asociar pesos aleatorios a los atributos del conjunto de entrenamiento de cada uno de los árboles de un *bosque* (i.e., un multiclasificador con clasificadores base árboles). Estos pesos son iguales para cada árbol, pero normalmente son distintos para cada atributo.

$\mathcal{RFW}$ utiliza estos pesos para condicionar la preferencia de los atributos sobre los que un árbol de decisión ramifica. Este condicionamiento se hace en base a multiplicar el mérito de cada atributo por su peso aleatorio elevado a un exponente que es un parámetro del método (el mérito es la función que obtiene un valor numérico en función de cuya magnitud se decide por qué atributo ramificar). A mayor exponente, mayor es el condicionamiento que los pesos aleatorios ejercen sobre la función de mérito. De hecho, con exponente 1 aumenta el riesgo de generar árboles idénticos.

$\mathcal{RFW}$ se ha probado desde las siguientes perspectivas:

1. Construyendo un multiclasificador que se denota como $\mathcal{RFW}$-p (o bien $\mathcal{RFW}p$-Ensemble) en el que todos sus árboles miembros se computan a través de esta técnica a base de pesos aleatorios en el que p es el exponente al que se elevan los pesos. $\mathcal{RFW}$-p calcula su predicción como el promedio de las probabilidades predichas por sus árboles miembros. Las versiones de $\mathcal{RFW}$-p con árboles podados y sin podar fueron probadas contra otros multiclasificadores de referencia: AdaBoost, MultiBoost, Bagging, Random Subspaces y Random Forests, este último sólo para árboles sin podar.

2. Probando ese mismo $\mathcal{RFW}$-p contra los mismos multiclasificadores que en el punto anterior, pero introduciendo un 10 % y un 20 % de ruido artificial a los conjuntos de datos, a fin de experimentar la robustez del método.

3. Introduciendo estos árboles de los $\mathcal{RFW}$s dentro de los multiclasificadores de referencia del estado del arte y comparándoles con su versión con árboles sin pesos aleatorios.

Todas estas pruebas se hicieron con los mismos 62 conjuntos de la UCI que se utilizaron para $\mathcal{DN}$ obteniendo resultados favorables. En los dos primeros casos se probaron con exponentes 1, 2, 3, y 4; y tanto con árboles sin podar como podados.

Las conclusiones principales para el caso de $\mathcal{RFW}$-p con datos sin ruido son que:

1. $\mathcal{RFW}$ es un método muy simple que es competitivo con los métodos de referencia.
2. Generalmente funciona mejor con árboles no podados, lo cual es razonable porque favorecen aún más la diversidad de los clasificadores base.
3. El exponente $p = 1$ es el peor, los exponentes más elevados (e.g 3 ó 4) suelen dar mejores resultados. Es razonable que sea así porque se condiciona más al clasificador base, lo que también favorece la diversidad.

Para el caso de $\mathcal{RFW}$-p con datos con ruido, sin embargo:

1. El método $\mathcal{RFW}$ no sólo suele ser mejor, sino que suele marcar diferencias más significativas con respecto a los clasificadores de referencia cuanto mayor es el ruido.
2. En este caso suelen dar mejor resultado los árboles podados, lo cual es razonable porque los árboles sin podar serán más sensibles al ruido.
3. La preferencia de un exponente sobre otro no está tan clara.

Al utilizar árboles de $\mathcal{RFW}$ como clasificadores base de los multiclasificadores de referencia, sólo se probaron árboles con exponente uno, para no condicionar en exceso el propio *bias* del método de referencia. También se probó con clasificadores base podados y sin podar.

La principal conclusión obtenida en este último caso es que en ninguno de los métodos experimentados, el multiclasificador con árboles con pesos aleatorios obtuvo peores resultados que sin pesos aleatorios. Además, en muchos casos las mejoras fueron significativas.

Como en el caso de $\mathcal{DN}$, se hicieron los diagramas de Movimiento y de Movimiento Relativo Kappa-Error. Los resultados de los diagramas al comparar los árboles con pesos aleatorios frente a los árboles sin pesos aleatorios en los multiclasificadores de referencia, es que efectivamente los árboles con pesos aleatorios aumentan la diversidad. Este aumento de la diversidad ocasiona la lógica pérdida de precisión individual de los clasificadores miembro, y este comportamiento se acentúa cuanto mayor es el exponente.

Sin embargo, al comparar los multiclasificadores de referencia con $\mathcal{RFW}3$ (el método $\mathcal{RFW}$-p con el mejor p según los experimentos), los resultados no son claros, pues hay conjuntos de datos en los que sí hay ese aumento de la diversidad y hay conjuntos en los que no.

En cuanto a los trabajos futuros:

- Lo más inmediato es la aplicación de las técnicas de aumento de la diversidad a multiclasificadores en regresión. De hecho se están obteniendo buenos resultados en la aplicación de $\mathcal{DN}$ a multiclasificadores con regresores base árboles [101]. $\mathcal{DN}$ podría utilizarse también con SVM para regresión, y $\mathcal{RFW}$ podría ser aplicado a árboles de regresión.

- En cuanto a la utilización de cascadas para datos nominales, quedan por probar el comportamiento de otros clasificadores tanto en el nivel base, como en el nivel meta. En el nivel base sería interesante disponer de otros métodos que proveyesen de estimaciones de probabilidad menos groseras que las que se obtienen de C4.5. En el nivel meta, cabe cambiar la función núcleo del SVM por otra más sofisticado y ajustar sus parámetros, o incluso cabe probar a utilizar en el nivel meta otros métodos que necesiten entradas numéricas.

- En el caso concreto de Disturbing Neighbors para SVM también parece interesante probar el método en clasificación utilizando otros tipos de función núcleo, e incluso ver si las mejoras siguen siendo importantes aún cuando los parámetros de los SVM que actúan como clasificadores base estén optimizados.

- Una línea muy interesante de trabajo es profundizar en las causas que hacen que el método $\mathcal{RFW}$ produzca buenos resultados. Esta linea incluye entre otros aspectos analizar las componentes bias y varianza del error dependiendo del número de clasificadores base, del porcentaje de error en el conjunto de datos y del valor del exponente utilizado para los pesos aleatorios.

- Otra línea que se está siguiendo es investigar el posible aumento de la diversidad en clasificadores base a través de proyecciones aleatorias. Esta idea se ha experimentado ya utilizando SVM lineales como clasificadores base [82]. Los resultados obtenidos son buenos, pero parecen indicar que la mejora en la diversidad no proviene tanto de las propias proyecciones aleatoras como de aplicar varias de estas proyecciones en cada clasificador base. Cada proyección se aplicaría sobre un conjunto de atributos distintos seleccionados aleatoriamente, siguiendo una estrategia similar a los *Rotation Forests* [69].

- Finalmente, se espera poder utilizar algunos de los métodos desarrollados en la tesis en problemas reales. Por ejemplo, se están comenzando a utilizar las técnicas de aumento de la diversidad en conjuntos de datos para detección de roturas en fresadoras. En este caso concreto $\mathcal{RFW}$ es un buen candidato dado que los datos disponibles contienen un importante porcentaje de ruido.

Apéndice A

Tablas con las Tasas de Acierto

A.1. Tasas de acierto para $\mathcal{DN}$ con SVM

Tabla A.1: Tasas de acierto para $\mathcal{DN}$-Ensemble, 1-NN y k-NN.

Conjunto	$\mathcal{DN}$-Ensemble	1-NN	k-NN
abalone	26.52	19.97	24.56
anneal	97.97	99.13	99.13
audiology	80.81	78.43	78.43
autos	72.90	74.55	74.27
balance-scale	90.18	86.72	90.19
breast-w	96.75	95.28	96.52
breast-y	69.92	72.85	72.11
bupa	63.15	62.22	62.07
car	94.29	93.05	93.04
credit-a	85.09	81.57	86.13
credit-g	75.15	71.88	73.55
crx	85.01	81.39	86.19
dna	93.87	74.55	85.20
ecoli	86.16	80.66	86.31
glass	65.43	69.95	68.14
heart-c	84.16	76.06	81.14
heart-h	83.58	78.33	81.75
heart-s	91.82	91.01	93.14
heart-statlog	83.59	76.15	80.81
heart-v	75.50	70.70	78.05
hepatitis	85.82	81.40	84.35
horse-colic	82.15	79.11	82.25
hypo	97.43	97.08	97.15
ionosphere	88.67	87.10	89.77
iris	95.53	95.40	95.80
kr-vs-kp	96.46	96.12	96.50
krk	32.60	62.08	77.78
labor	93.20	84.30	90.53
led-24	74.51	56.06	72.34
letter	84.88	96.01	96.01
lrd	88.76	86.24	86.40
lymphography	85.41	81.69	81.65
mushroom	100.00	100.00	100.00
nursery	93.49	98.39	98.39
optdigits	98.65	98.70	98.75
page	93.55	96.08	95.93
pendigits	98.69	99.37	99.34
phoneme	78.46	90.28	90.28
pima	77.07	70.62	73.68
primary	47.56	39.91	46.78
promoters	91.56	79.71	76.87
ringnorm	82.13	61.53	68.93
sat	87.25	90.33	90.84
segment	93.17	97.15	97.15
shuttle	98.43	99.93	99.93
sick	93.88	96.10	96.14
sonar	79.86	86.17	85.44
soybean-small	100.00	100.00	100.00
soybean	93.22	91.20	91.24
splice	93.43	74.43	85.15
threenorm	86.33	74.60	84.03
tic-tac-toe	98.33	98.98	98.98
twonorm	96.93	93.23	95.67
vehicle	74.00	69.59	69.32
vote1	91.63	89.61	89.49
voting	95.97	92.58	92.67
vowel-context	81.47	99.05	99.05
vowel-nocontext	81.20	98.84	98.84
waveform	86.61	73.41	82.37
yeast	58.69	52.61	57.91
zip	95.83	96.89	96.89
zoo	93.10	96.05	94.86

Tabla A.2: Tasas de acierto para las configuraciones de SVM y Bagging. ∘ indica una victoria significativa del algoritmo $\mathcal{DN}$ respecto de la versión sin $\mathcal{DN}$, mientras que • indica una derrota.

Conjunto	SMO	$\mathcal{DN}$-SMO	Bagging	$\mathcal{DN}$-Bagging
abalone	25.12	25.05	25.29	26.40
anneal	97.46	97.90	97.97	98.14
audiology	80.77	80.37	77.85	78.34
autos	71.29	70.94	72.31	73.19
balance-scale	87.57	88.27	87.11	90.13 ∘
breast-w	96.75	96.54	96.71	96.68
breast-y	69.52	70.01	69.59	70.99
bupa	57.98	59.14	58.10	63.29 ∘
car	93.62	94.06	93.61	94.31 ∘
credit-a	84.88	84.97	85.01	85.28
credit-g	75.09	75.61	75.24	75.22
crx	84.93	84.87	85.03	85.30
dna	93.41	93.16	94.55	94.52
ecoli	83.48	80.41	84.22	86.55
glass	57.46	58.32	58.35	66.12 ∘
heart-c	83.86	83.06	84.39	83.99
heart-h	82.74	82.46	83.69	84.02
heart-s	91.65	91.08	92.07	92.15
heart-statlog	83.89	83.52	83.74	83.63
heart-v	75.75	74.75	74.75	75.55
hepatitis	85.77	85.05	86.02	85.69
horse-colic	82.66	81.58	82.72	82.85
hypo	97.44	97.40	97.43	97.42
ionosphere	88.07	87.98	88.38	88.61
iris	96.27	91.47	96.27	95.73
kr-vs-kp	95.79	96.37	96.15	96.56 ∘
krk	28.08	30.59 ∘	28.15	32.70 ∘
labor	92.97	91.57	91.90	93.30
led-24	74.55	74.47	74.67	74.68
letter	82.34	82.94 ∘	82.75	84.93 ∘
lrd	89.00	88.06	89.32	88.89
lymphography	86.48	85.49	86.35	85.29
mushroom	100.00	100.00	100.00	100.00
nursery	93.08	93.26	93.18	93.50 ∘
optdigits	98.25	98.23	98.56	98.65
page	92.84	93.53 ∘	93.35	94.01 ∘
pendigits	97.98	98.22 ∘	98.04	98.70 ∘
phoneme	77.34	76.79	77.27	78.55 ∘
pima	76.80	76.17	77.00	77.16
primary	47.09	45.46	48.35	48.47
promoters	91.01	90.65	91.77	91.58
ringnorm	73.27	75.80	74.73	83.10 ∘
sat	86.76	86.79	86.65	87.32 ∘
segment	92.92	92.42	92.88	93.14
shuttle	96.96	97.66 ∘	96.97	98.42 ∘
sick	93.87	94.02	93.89	93.92
sonar	76.60	76.98	77.55	80.44
soybean-small	100.00	100.00	100.00	100.00
soybean	93.10	93.00	92.90	92.88
splice	92.88	92.74	94.46	94.50
threenorm	85.50	84.20	85.50	86.43
tic-tac-toe	98.33	98.33	98.33	98.33
twonorm	96.67	96.43	96.83	96.77
vehicle	74.08	71.46	74.50	74.77
vote1	91.61	91.38	91.89	91.84
voting	95.77	95.84	95.97	96.11
vowel-context	70.63	71.11	71.93	84.69 ∘
vowel-nocontext	68.84	66.20	68.17	81.38 ∘
waveform	86.48	86.41	86.59	86.59
yeast	56.96	54.96 •	57.10	58.66 ∘
zip	95.21	95.22	95.96	96.11
zoo	93.68	93.67	93.29	93.99

Tabla A.3: Tasas de acierto para las configuraciones de Random Subspaces. ∘ indica una victoria significativa del algoritmo $\mathcal{DN}$ respecto de la versión sin $\mathcal{DN}$. No hay derrotas significativas de las versiones con $\mathcal{DN}$.

Conjunto	Subspaces(50 %)	$\mathcal{DN}$-Subspaces(50 %)	Subspaces(75 %)	$\mathcal{DN}$-Subspaces(75 %)
abalone	24.76	26.55 ∘	24.73	26.57 ∘
anneal	94.61	95.11	97.78	98.18
audiology	79.27	78.95	80.76	80.27
autos	65.58	71.14 ∘	69.62	71.43
balance-scale	78.31	86.32 ∘	83.47	87.09 ∘
breast-w	96.77	96.91	96.78	96.88
breast-y	69.50	72.12 ∘	68.83	71.13
bupa	57.98	62.97 ∘	57.95	62.80 ∘
car	70.11	70.30	90.28	91.70
credit-a	85.25	85.46	84.94	85.10
credit-g	72.39	72.56	75.18	75.44
crx	85.00	85.41	84.97	85.10
dna	96.07	96.10	95.91	95.94
ecoli	79.76	81.90	82.68	84.46
glass	55.47	66.86 ∘	56.99	65.80 ∘
heart-c	83.99	83.96	83.96	83.99
heart-h	82.50	83.32	82.24	83.25
heart-s	93.04	92.88	92.06	92.56
heart-statlog	84.30	84.19	84.22	84.04
heart-v	75.60	75.00	75.85	75.80
hepatitis	83.96	83.89	84.20	85.32
horse-colic	82.81	83.45	83.66	83.66
hypo	96.36	96.50	97.26	97.29
ionosphere	88.69	90.18	88.47	89.64
iris	94.13	94.60	95.80	95.40
kr-vs-kp	92.34	92.89	95.02	95.29
krk	24.49	28.76 ∘	27.72	31.95 ∘
labor	92.90	95.30	93.83	95.03
led-24	73.52	74.46	74.61	74.77
letter	77.02	80.43 ∘	81.31	83.68 ∘
lrd	88.50	88.00	88.80	88.59
lymphography	84.02	84.48	86.07	85.49
mushroom	100.00	100.00	100.00	100.00
nursery	85.44	87.01	91.41	91.75
optdigits	97.93	98.13	98.37	98.50
page	91.77	92.80 ∘	92.74	93.58 ∘
pendigits	95.80	97.23 ∘	97.65	98.47 ∘
phoneme	73.51	77.77 ∘	75.41	77.95 ∘
pima	71.48	74.27	76.73	76.69
primary	43.60	44.61	47.20	47.49
promoters	92.26	91.08	91.67	91.77
ringnorm	74.73	89.87 ∘	74.93	84.77 ∘
sat	85.95	86.78 ∘	86.36	87.04 ∘
segment	91.29	92.50 ∘	92.54	92.76
shuttle	95.85	98.72 ∘	96.77	98.42 ∘
sick	93.88	93.88	93.88	93.88
sonar	77.65	80.70	77.89	80.73
soybean-small	100.00	100.00	100.00	100.00
soybean	93.35	93.41	93.59	93.64
splice	96.35	96.32	96.00	96.00
threenorm	84.90	85.90	84.97	85.63
tic-tac-toe	70.71	74.09 ∘	74.43	77.44 ∘
twonorm	97.30	96.63	97.33	96.87
vehicle	70.00	69.40	72.89	72.78
vote1	91.17	91.20	91.84	92.23
voting	95.15	95.22	95.61	95.54
vowel-context	57.82	76.72 ∘	66.58	80.65 ∘
vowel-nocontext	61.32	78.70 ∘	66.58	80.70 ∘
waveform	86.42	86.47	86.65	86.56
yeast	37.73	50.77 ∘	54.28	57.07 ∘
zip	96.15	96.29	95.98	96.14 ∘
zoo	92.61	93.48	92.91	93.40

Tabla A.4: Tasas de acierto para las configuraciones de AdaBoost. ∘ indica una victoria significativa del algoritmo $\mathcal{DN}$ respecto de la versión sin $\mathcal{DN}$. No hay derrotas significativas de las versiones con $\mathcal{DN}$.

Conjunto	AdaBoost(S)	$\mathcal{DN}$-AdaBoost(S)	AdaBoost(W)	$\mathcal{DN}$-AdaBoost(W)
abalone	24.92	24.87	25.12	25.27
anneal	99.31	99.55	99.24	99.31
audiology	80.86	80.86	80.65	80.71
autos	74.88	76.49	73.86	75.91
balance-scale	87.32	88.83	87.54	89.33
breast-w	96.58	96.05	96.72	96.27
breast-y	67.14	65.39	68.52	67.15
bupa	64.65	67.50	65.99	67.60
car	94.67	98.35 ∘	94.42	98.32 ∘
credit-a	83.86	84.23	83.74	84.16
credit-g	74.45	73.36	75.14	72.92
crx	83.72	84.12	83.49	83.04
dna	93.23	93.57	92.55	92.98
ecoli	86.04	84.43	84.69	84.59
glass	56.94	62.76	59.55	64.01
heart-c	82.38	80.72	83.89	81.71
heart-h	82.76	81.57	83.21	81.74
heart-s	90.91	89.28	91.56	90.50
heart-statlog	82.74	80.04	83.59	80.22
heart-v	71.05	71.65	72.65	71.05
hepatitis	82.07	80.88	82.35	83.38
horse-colic	78.02	80.06	78.54	80.24
hypo	97.11	98.00 ∘	96.97	97.69 ∘
ionosphere	88.52	90.34	89.38	89.92
iris	96.73	94.73	97.93	95.73
kr-vs-kp	97.63	98.81 ∘	97.64	98.97 ∘
krk	28.09	30.34 ∘	28.08	30.47 ∘
labor	92.93	93.97	89.57	91.60
led-24	74.52	74.49	74.55	74.62
letter	82.20	82.64 ∘	82.34	82.86 ∘
lrd	89.81	89.21	89.64	89.59
lymphography	85.35	86.74	83.00	85.81
mushroom	99.99	99.99	100.00	100.00
nursery	93.04	94.72 ∘	93.07	96.85 ∘
optdigits	98.19	98.48 ∘	98.08	98.43 ∘
page	93.28	95.10 ∘	92.84	94.40 ∘
pendigits	98.17	99.23 ∘	98.14	99.23 ∘
phoneme	77.17	82.07 ∘	77.46	82.02 ∘
pima	77.02	74.53	76.80	74.89
primary	43.86	44.66	47.09	45.61
promoters	87.67	88.76	91.01	91.48
ringnorm	74.80	91.43 ∘	74.57	92.10 ∘
sat	86.65	87.32 ∘	86.76	87.17
segment	92.69	96.20 ∘	92.92	94.62 ∘
shuttle	96.95	99.78 ∘	96.96	99.58 ∘
sick	94.22	96.71 ∘	94.46	96.42 ∘
sonar	78.86	82.14	78.57	83.08
soybean-small	100.00	100.00	100.00	100.00
soybean	92.81	93.00	92.88	93.15
splice	93.52	93.69	92.80	93.14
threenorm	83.53	83.60	85.10	86.27
tic-tac-toe	98.04	97.48	97.95	97.99
twonorm	95.20	95.23	95.33	95.73
vehicle	73.70	74.20	74.10	74.37
vote1	89.46	88.83	90.39	89.37
voting	95.40	95.40	95.51	94.92
vowel-context	83.57	96.35 ∘	80.86	95.43 ∘
vowel-nocontext	68.45	91.15 ∘	68.84	88.70 ∘
waveform	86.25	86.11	86.48	86.37
yeast	56.16	55.84	56.96	55.99
zip	95.26	95.61 ∘	94.86	95.43 ∘
zoo	96.74	96.35	96.73	94.75

Tabla A.5: Tasas de acierto para las configuraciones de MultiBoost. ∘ indica una victoria significativa del algoritmo $\mathcal{DN}$ respecto de la versión sin $\mathcal{DN}$. No hay derrotas significativas de las versiones con $\mathcal{DN}$.

Conjunto	MultiBoost(S)	$\mathcal{DN}$-MultiBoost(S)	MultiBoost(W)	$\mathcal{DN}$-MultiBoost(W)
abalone	24.92	24.87	25.12	25.27
anneal	99.53	99.58	99.52	99.49
audiology	80.73	80.89	80.99	81.11
autos	75.42	76.90	74.40	76.24
balance-scale	87.63	92.06 ∘	87.65	92.43 ∘
breast-w	96.71	96.75	96.77	96.80
breast-y	68.95	70.00	68.61	69.48
bupa	64.93	68.17	64.02	68.36
car	94.61	96.53 ∘	94.64	96.31 ∘
credit-a	85.54	85.88	85.46	85.72
credit-g	74.59	74.92	75.13	75.37
crx	85.38	85.81	85.28	85.75
dna	93.85	93.96	93.36	93.63
ecoli	86.12	86.91	85.11	87.15
glass	57.08	65.20 ∘	59.88	64.77
heart-c	83.66	83.40	83.64	83.89
heart-h	84.81	83.54	84.61	83.75
heart-s	92.53	91.50	92.38	92.38
heart-statlog	83.70	82.81	83.78	83.04
heart-v	74.90	74.95	74.70	75.35
hepatitis	84.02	84.08	84.00	84.00
horse-colic	81.39	81.91	81.63	81.45
hypo	97.65	98.10 ∘	97.59	97.82
ionosphere	89.23	90.29	89.09	90.06
iris	96.80	95.33	96.33	95.33
kr-vs-kp	97.63	98.21 ∘	97.65	98.16 ∘
krk	28.09	30.34 ∘	28.08	30.47 ∘
labor	91.67	94.10	91.43	91.33
led-24	74.56	74.55	74.52	74.62
letter	82.31	85.01 ∘	82.34	84.35 ∘
lrd	90.19	89.74	90.15	89.49
lymphography	86.63	87.01	86.02	85.69
mushroom	99.99	99.99	100.00	100.00
nursery	93.06	94.02 ∘	93.10	93.94 ∘
optdigits	98.38	98.57 ∘	98.34	98.53 ∘
page	93.70	95.38 ∘	93.08	94.92 ∘
pendigits	98.52	99.17 ∘	98.49	99.15 ∘
phoneme	76.83	80.07 ∘	77.09	79.94 ∘
pima	77.15	76.78	76.72	76.55
primary	44.66	46.38	47.12	45.88
promoters	91.85	91.49	91.01	91.48
ringnorm	74.90	87.27 ∘	76.40	88.57 ∘
sat	86.74	88.33 ∘	86.76	88.19 ∘
segment	93.03	95.50 ∘	92.90	95.08 ∘
shuttle	96.95	99.53 ∘	96.95	99.36 ∘
sick	95.88	96.50	95.92	96.35
sonar	78.09	82.36	77.28	82.27 ∘
soybean-small	100.00	100.00	100.00	100.00
soybean	92.84	92.85	93.09	93.02
splice	93.95	94.21	93.32	93.61
threenorm	84.50	85.93	85.30	86.70
tic-tac-toe	98.19	97.96	98.14	98.19
twonorm	95.83	95.87	95.93	96.00
vehicle	74.24	76.48	74.35	75.75
vote1	90.59	90.11	90.80	90.39
voting	95.95	95.93	95.95	95.90
vowel-context	82.48	92.21 ∘	80.88	92.15 ∘
vowel-nocontext	69.84	85.43 ∘	68.82	85.10 ∘
waveform	86.34	86.46	86.49	86.42
yeast	56.16	56.21	57.19	56.56
zip	95.52	95.74 ∘	95.38	95.67 ∘
zoo	96.05	96.45	96.44	94.75

A.2. Tasas de acierto para $\mathcal{DN}$ con árboles

Tabla A.6: Tasas de acierto para $\mathcal{DN}$-Ensemble, 1-NN y k-NN.

Dataset	$\mathcal{DN}$-Ensemble	1-NN	k-NN
abalone	22.79	19.97	24.56
anneal	98.61	99.13	99.13
audiology	79.38	78.43	78.43
autos	82.93	74.55	74.27
balance-scale	83.75	86.72	90.19
breast-w	96.64	95.28	96.52
breast-y	73.66	72.85	72.11
bupa	71.83	62.22	62.07
car	93.40	93.05	93.04
credit-a	85.88	81.57	86.13
credit-g	72.30	71.88	73.55
crx	85.54	81.39	86.19
dna	92.66	74.55	85.20
ecoli	83.57	80.66	86.31
glass	71.21	69.95	68.14
heart-c	81.05	76.06	81.14
heart-h	81.45	78.33	81.75
heart-s	93.53	91.01	93.14
heart-statlog	81.19	76.15	80.81
heart-v	74.20	70.70	78.05
hepatitis	81.23	81.40	84.35
horse-colic	85.24	79.11	82.25
hypo	99.28	97.08	97.15
ionosphere	91.06	87.10	89.77
iris	94.80	95.40	95.80
kr-vs-kp	99.44	96.12	96.50
krk	83.50	62.08	77.78
labor	85.03	84.30	90.53
led-24	74.66	56.06	72.34
letter	89.30	96.01	96.01
lrd	83.13	86.24	86.40
lymphography	78.86	81.69	81.65
mushroom	100.00	100.00	100.00
nursery	97.33	98.39	98.39
optdigits	96.48	98.70	98.75
page	97.03	96.08	95.93
pendigits	97.54	99.37	99.34
phoneme	89.50	90.28	90.28
pima	76.21	70.62	73.68
primary	44.96	39.91	46.78
promoters	79.68	79.71	76.87
ringnorm	85.03	61.53	68.93
sat	90.05	90.33	90.84
segment	97.19	97.15	97.15
shuttle	99.97	99.93	99.93
sick	98.75	96.10	96.14
sonar	79.04	86.17	85.44
soybean-small	97.65	100.00	100.00
soybean	92.79	91.20	91.24
splice	94.24	74.43	85.15
threenorm	82.73	74.60	84.03
tic-tac-toe	95.70	98.98	98.98
twonorm	95.07	93.23	95.67
vehicle	73.12	69.59	69.32
vote1	89.75	89.61	89.49
voting	96.57	92.58	92.67
vowel-context	84.24	99.05	99.05
vowel-nocontext	85.93	98.84	98.84
waveform	78.47	73.41	82.37
yeast	60.04	52.61	57.91
zip	89.32	96.89	96.89
zoo	92.61	96.05	94.86

Tabla A.7: Tasas de acierto para las configuraciones de Bagging y Random Forest.

Conjunto	Bagging	$\mathcal{DN}$-Bagging	Random Forest	$\mathcal{DN}$-Random Forest
abalone	23.92	24.05	23.68	24.17
anneal	98.80	98.82	99.69	99.58
audiology	80.88	81.65	79.39	79.89
autos	84.19	84.04	83.99	83.70
balance-scale	82.23	85.99 ○	80.48	83.31 ○
breast-w	96.24	96.81	96.38	96.88
breast-y	73.13	73.48	70.02	71.23
bupa	73.10	72.87	71.77	72.61
car	93.59	94.22 ○	94.44	94.61
credit-a	85.91	86.01	86.04	86.72
credit-g	74.36	74.95	75.62	75.31
crx	86.17	86.25	85.41	86.39
dna	94.89	94.88	94.55	94.30
ecoli	84.76	85.09	84.71	86.16
glass	74.11	74.84	78.72	78.97
heart-c	80.10	82.41	81.39	82.31
heart-h	80.07	80.65	80.59	80.63
heart-s	93.21	93.37	91.75	92.39
heart-statlog	81.19	82.19	82.00	82.78
heart-v	75.75	75.20	75.10	75.35
hepatitis	81.50	81.95	83.33	83.96
horse-colic	85.26	85.37	85.48	84.86
hypo	99.26	99.27	99.01	98.99
ionosphere	92.57	92.91	93.68	93.54
iris	94.60	94.53	94.40	94.80
kr-vs-kp	99.46	99.44	99.26	99.02
krk	84.91	85.55 ○	84.89	85.30 ○
labor	85.23	88.97	86.97	90.57
led-24	74.59	75.03	74.43	74.50
letter	93.72	93.99 ○	96.41	96.56
lrd	86.88	87.14	88.31	88.42
lymphography	79.82	79.97	83.30	84.65
mushroom	100.00	100.00	100.00	100.00
nursery	97.42	97.67 ○	99.05	98.71 •
optdigits	95.75	97.23 ○	98.19	98.23
page	97.38	97.39	97.42	97.47
pendigits	98.23	98.58 ○	99.17	99.19
phoneme	89.56	90.30 ○	91.25	91.23
pima	76.05	76.20	75.58	75.80
primary	44.96	45.55	43.27	43.83
promoters	85.29	85.81	88.98	90.03
ringnorm	88.43	89.40	90.67	92.70
sat	90.77	91.33 ○	91.80	91.84
segment	97.60	97.57	98.11	98.10
shuttle	99.97	99.97	99.99	99.98 •
sick	98.86	98.81	98.45	98.39
sonar	80.30	79.18	84.50	83.72
soybean-small	97.65	97.65	100.00	100.00
soybean	93.03	93.59	93.24	93.56
splice	94.66	94.67	95.27	95.05
threenorm	82.10	84.53	83.97	85.20
tic-tac-toe	94.70	95.77	96.42	95.63
twonorm	90.27	95.40 ○	93.43	94.77
vehicle	74.97	74.87	75.07	74.80
vote1	89.66	90.12	89.54	89.96
voting	96.71	96.30	96.50	96.09
vowel-context	92.23	93.40	97.85	98.19
vowel-nocontext	91.78	92.75	95.94	97.53 ○
waveform	83.25	83.26	84.77	84.96
yeast	60.92	61.75	60.58	61.30
zip	93.17	93.66 ○	96.28	96.14
zoo	93.20	93.30	96.23	96.15

Tabla A.8: Tasas de acierto para las configuraciones de AdaBoost

Conjunto	AdaBoost(W)	$\mathcal{DN}$-AdaBoost(W)	AdaBoost(S)	$\mathcal{DN}$-AdaBoost(S)
abalone	22.59	22.85	23.21	23.27
anneal	99.63	99.66	99.67	99.69
audiology	84.61	85.84	84.56	85.18
autos	86.59	86.17	86.10	86.48
balance-scale	76.36	77.18	75.53	76.08
breast-w	96.61	96.84	96.68	96.87
breast-y	66.43	68.12	67.16	67.98
bupa	70.20	69.85	70.77	69.89
car	96.72	97.26	96.60	97.01
credit-a	85.87	86.01	86.39	86.45
credit-g	73.75	74.36	74.52	74.63
crx	86.01	85.96	85.88	86.42
dna	95.04	95.05	94.95	94.95
ecoli	83.51	84.43	83.99	84.43
glass	78.08	77.75	79.10	79.15
heart-c	80.11	80.96	79.78	80.96
heart-h	79.40	78.68	80.04	78.99
heart-s	90.78	91.58	91.33	91.03
heart-statlog	80.44	81.19	79.44	80.30
heart-v	71.80	72.10	70.40	71.85
hepatitis	84.15	83.96	84.61	83.95
horse-colic	81.76	82.36	82.09	83.12
hypo	98.98	98.99	98.99	99.07
ionosphere	93.85	93.93	94.22	94.08
iris	94.53	94.27	94.13	94.60
kr-vs-kp	99.60	99.62	99.61	99.61
krk	89.47	89.99 ∘	89.39	89.73 ∘
labor	89.10	91.37	87.47	89.83
led-24	69.80	72.58 ∘	71.33	72.89 ∘
letter	97.03	97.04	96.87	96.92
lrd	88.17	87.91	88.10	88.19
lymphography	84.12	84.80	83.72	83.66
mushroom	100.00	99.99	100.00	99.99
nursery	99.75	99.81	99.74	99.79
optdigits	98.40	98.44	98.42	98.38
page	97.07	97.03	97.11	97.11
pendigits	99.38	99.39	99.36	99.40
phoneme	91.18	91.29	91.27	91.37
pima	73.71	74.38	73.71	73.47
primary	41.65	42.84	42.37	42.30
promoters	94.08	93.39	93.78	93.65
ringnorm	94.17	95.13	95.67	95.77
sat	92.13	92.08	92.01	92.07
segment	98.60	98.63	98.57	98.59
shuttle	99.99	99.99	99.99	99.99
sick	99.07	99.03	99.06	99.06
sonar	84.65	85.05	82.41	83.76
soybean-small	97.65	97.30	98.35	99.35
soybean	93.31	93.92	93.31	93.95
splice	94.91	94.92	94.84	94.92
threenorm	84.73	85.17	84.10	85.27
tic-tac-toe	98.92	98.92	98.94	98.98
twonorm	94.07	95.43	94.33	94.80
vehicle	77.41	77.07	77.68	77.26
vote1	89.17	89.72	89.36	89.73
voting	95.31	95.56	95.10	95.44
vowel-context	95.93	96.62	96.26	96.70
vowel-nocontext	95.23	96.45 ∘	95.62	96.34
waveform	84.35	84.48	84.15	84.25
yeast	58.44	59.55	59.36	60.11
zip	96.74	96.71	96.59	96.66
zoo	96.35	96.25	95.06	96.15

Tabla A.9: Tasas de acierto para las configuraciones de MultiBoost.

Conjunto	MultiBoost(W)	$\mathcal{DN}$-MultiBoost(W)	MultiBoost(S)	$\mathcal{DN}$-MultiBoost(S)
abalone	23.57	23.69	24.07	24.05
anneal	99.68	99.63	99.68	99.64
audiology	85.14	85.49	85.06	84.91
autos	86.10	85.90	85.99	85.41
balance-scale	80.60	81.80	80.33	81.41
breast-w	96.60	96.77	96.50	96.60
breast-y	69.68	70.84	69.35	70.57
bupa	72.52	72.42	72.84	72.64
car	95.96	96.47	95.80	96.32
credit-a	86.90	86.70	86.77	86.87
credit-g	75.39	75.66	75.29	75.79
crx	86.84	86.96	86.74	86.70
dna	95.63	95.61	95.52	95.55
ecoli	84.58	85.00	85.15	85.50
glass	77.24	77.38	78.50	77.57
heart-c	81.28	82.50	81.39	81.95
heart-h	79.70	80.34	80.68	79.90
heart-s	90.93	91.83	91.26	91.51
heart-statlog	81.56	81.07	80.78	81.74
heart-v	74.40	74.00	74.40	74.00
hepatitis	83.57	83.45	83.27	83.39
horse-colic	84.67	84.13	84.10	84.61
hypo	99.12	99.13	99.14	99.19
ionosphere	93.73	94.02	93.73	94.13
iris	94.20	94.53	94.53	94.20
kr-vs-kp	99.65	99.65	99.65	99.62
krk	88.49	88.73	88.48	88.66
labor	86.57	90.50	87.57	90.60
led-24	72.60	73.92 ∘	73.14	74.15 ∘
letter	96.67	96.70	96.58	96.62
lrd	87.65	87.44	88.04	88.06
lymphography	83.10	83.73	83.40	84.06
mushroom	100.00	99.99	100.00	99.99
nursery	99.70	99.71	99.68	99.68
optdigits	98.08	98.18	98.10	98.14
page	97.38	97.37	97.39	97.39
pendigits	99.24	99.26	99.25	99.29
phoneme	91.12	91.14	91.14	91.05
pima	75.54	75.17	74.92	75.57
primary	42.27	43.99	43.60	44.07
promoters	94.55	94.67	94.27	93.25
ringnorm	92.67	93.80	94.10	94.80
sat	91.83	91.98	91.72	91.75
segment	98.35	98.36	98.34	98.39
shuttle	99.99	99.99	99.99	99.99
sick	99.06	99.04	99.04	98.99
sonar	83.03	83.79	81.44	81.99
soybean-small	97.65	97.30	98.15	98.80
soybean	93.32	94.27	93.54	94.38
splice	95.56	95.58	95.48	95.43
threenorm	85.00	85.37	83.97	85.17
tic-tac-toe	98.31	98.59	98.31	98.24
twonorm	93.60	95.33	93.67	95.60
vehicle	76.63	76.77	76.48	76.08
vote1	89.84	90.07	89.93	90.23
voting	95.59	96.20	95.56	96.18
vowel-context	95.14	95.94	95.15	95.64
vowel-nocontext	94.41	95.27	94.52	95.17
waveform	84.57	84.82	84.57	84.72
yeast	60.66	61.06	60.97	61.28
zip	96.38	96.32	96.26	96.26
zoo	96.05	96.04	95.25	96.35

Tabla A.10: Tasas de acierto para las configuraciones de Random Subspaces.

Conjunto	Sunspaces(50 %)	$\mathcal{DN}$-Subspaces(50 %)	Subspaces(75 %)	$\mathcal{DN}$-Subspaces(75 %)
abalone	25.53	25.72	24.08	24.59
anneal	98.75	98.79	98.64	98.83
audiology	79.23	79.43	80.84	81.77
autos	85.24	85.66	85.67	86.24
balance-scale	81.42	85.36 ◦	77.58	85.40 ◦
breast-w	96.57	97.01	95.78	96.84 ◦
breast-y	74.10	73.76	74.24	73.27
bupa	67.47	70.24	67.69	72.39 ◦
car	70.02	70.02	92.38	93.00
credit-a	86.20	86.61	86.16	86.41
credit-g	74.70	74.56	73.73	75.01
crx	86.26	86.46	85.99	86.61
dna	95.71	95.77	95.29	95.33
ecoli	85.57	85.63	84.94	84.47
glass	77.08	77.23	73.04	74.99
heart-c	82.11	83.33	78.88	82.28 ◦
heart-h	81.95	82.77	81.04	82.05
heart-s	93.53	93.53	93.53	93.53
heart-statlog	83.85	83.30	81.30	82.89
heart-v	73.95	74.45	73.55	74.15
hepatitis	83.12	83.12	80.79	82.35
horse-colic	84.93	85.07	85.18	85.40
hypo	98.71	98.75	99.21	99.23
ionosphere	93.68	93.48	92.82	92.91
iris	94.60	94.87	94.73	94.80
kr-vs-kp	97.31	97.15	99.21	99.18
krk	42.81	43.46	81.79	83.11 ◦
labor	80.57	88.03	79.83	86.60
led-24	74.37	74.52	74.85	74.92
letter	96.08	96.10	95.88	95.98
lrd	86.09	86.76	85.03	85.52
lymphography	79.82	82.18	78.03	81.72
mushroom	100.00	100.00	100.00	100.00
nursery	91.41	91.40	94.82	94.77
optdigits	98.09	98.16	96.98	97.84 ◦
page	97.41	97.44	97.24	97.28
pendigits	99.02	99.03	98.75	98.97 ◦
phoneme	85.68	87.56 ◦	87.38	89.39 ◦
pima	74.88	75.17	75.34	75.98
primary	45.46	46.02	44.72	45.64
promoters	86.74	89.68	80.73	83.67
ringnorm	90.57	92.30	86.63	89.03
sat	91.65	91.75	90.72	91.44 ◦
segment	97.71	97.82	97.57	97.73
shuttle	99.93	99.96	99.97	99.98
sick	95.76	96.17	98.49	98.49
sonar	82.31	83.08	78.56	80.34
soybean-small	98.95	99.55	97.80	99.35
soybean	94.76	95.18	93.51	94.60 ◦
splice	96.14	96.05	95.08	95.16
threenorm	83.00	85.37	78.07	84.77 ◦
tic-tac-toe	80.96	81.38	88.49	90.72 ◦
twonorm	90.90	95.20 ◦	82.87	94.63 ◦
vehicle	75.09	74.88	74.49	74.32
vote1	89.64	90.19	89.45	90.23
voting	95.38	95.38	96.64	96.57
vowel-context	96.25	96.91	90.81	93.06 ◦
vowel-nocontext	94.84	95.61	87.53	92.26 ◦
waveform	84.45	84.36	82.77	83.21
yeast	59.36	60.31	59.78	61.61 ◦
zip	95.75	95.72	94.29	94.61
zoo	94.08	94.96	92.20	93.20

A.3. Tasas de acierto del análisis de lesiones para $\mathcal{DN}$

Tabla A.11: Tasas de acierto para las configuraciones de Bagging.

Conjunto	Bagging	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	23.92	24.00	24.32	23.83	24.21	24.08	24.05
anneal	98.80	98.81	98.86	98.79	98.82	98.85	98.82
audiology	80.88	81.42	81.78	80.98	81.65	81.61	81.65
autos	84.19	84.49	83.84	84.04	84.18	83.80	84.04
balance-scale	82.23	86.19 ∘	86.35 ∘	84.07 ∘	85.90 ∘	86.54 ∘	85.99 ∘
breast-w	96.24	96.48	96.48	96.65	96.45	96.55	96.81
breast-y	73.13	73.34	73.23	73.37	73.72	73.41	73.48
bupa	73.10	72.50	72.12	72.46	72.64	72.49	72.87
car	93.59	93.66	93.72	93.74	94.21 ∘	93.77	94.22 ∘
credit-a	85.91	86.12	86.36	86.17	85.94	86.32	86.01
credit-g	74.36	74.43	75.12	74.42	74.90	74.96	74.95
crx	86.17	86.28	86.39	86.26	86.23	86.33	86.25
dna	94.89	94.87	94.90	94.85	94.87	94.90	94.88
ecoli	84.76	85.63	84.73	84.91	85.09	85.23	85.09
glass	74.11	74.72	74.62	74.57	74.84	74.53	74.84
heart-c	80.10	82.21	83.03 ∘	81.09	81.78	83.37 ∘	82.41
heart-h	80.07	80.48	80.62	80.14	80.48	80.82	80.65
heart-s	93.21	93.29	93.29	93.37	93.37	93.29	93.37
heart-statlog	81.19	81.78	81.81	81.96	82.00	82.04	82.19
heart-v	75.75	75.70	76.10	74.90	75.45	76.00	75.20
hepatitis	81.50	82.34	82.20	81.70	81.76	82.53	81.95
horse-colic	85.26	85.31	85.32	85.26	85.42	85.21	85.37
hypo	99.26	99.26	99.27	99.26	99.27	99.27	99.27
ionosphere	92.57	92.73	92.45	92.74	92.77	92.60	92.91
iris	94.60	94.40	94.53	94.67	94.47	94.53	94.53
kr-vs-kp	99.46	99.44	99.42	99.44	99.44	99.42	99.44
krk	84.91	84.94	85.16	85.21 ∘	85.57 ∘	85.12	85.55 ∘
labor	85.23	90.63	88.87	87.73	87.27	90.43	88.97
led-24	74.59	74.84	74.94	74.85	74.99	74.96	75.03
letter	93.72	93.78	93.95	93.79	94.03 ∘	93.90	93.99 ∘
lrd	86.88	87.14	87.29	87.11	87.12	87.33	87.14
lymphography	79.82	80.70	80.23	78.88	79.76	80.78	79.97
mushroom	100.00	100.00	100.00	100.00	100.00	100.00	100.00
nursery	97.42	97.41	97.36	97.47	97.64 ∘	97.36	97.67 ∘
optdigits	95.75	97.17 ∘	97.53 ∘	96.24 ∘	97.19 ∘	97.61 ∘	97.23 ∘
page	97.38	97.37	97.39	97.37	97.39	97.39	97.39
pendigits	98.23	98.67 ∘	98.69 ∘	98.50 ∘	98.60 ∘	98.73 ∘	98.58 ∘
phoneme	89.56	89.89	90.23 ∘	89.94	90.23 ∘	90.20 ∘	90.30 ∘
pima	76.05	76.25	76.28	76.25	76.27	76.38	76.20
primary	44.96	45.32	45.11	44.81	44.99	45.11	45.55
promoters	85.29	87.57	87.20	86.17	85.62	88.67	85.81
ringnorm	88.43	89.23	88.27	89.27	88.33	89.07	89.40
sat	90.77	91.17	91.20	91.15	91.08	91.33 ∘	91.33 ∘
segment	97.60	97.59	97.51	97.55	97.58	97.52	97.57
shuttle	99.97	99.97	99.97	99.97	99.97	99.98	99.97
sick	98.86	98.84	98.83	98.85	98.81	98.83	98.81
sonar	80.30	79.81	80.15	79.57	79.70	80.10	79.18
soybean-small	97.65	100.00	97.65	97.65	97.65	100.00	97.65
soybean	93.03	93.88	93.79	93.38	93.59	93.91	93.59
splice	94.66	94.69	94.76	94.65	94.66	94.76	94.67
threenorm	82.10	85.73	85.60	83.63	84.07	86.03 ∘	84.53
tic-tac-toe	94.70	94.99	95.61	95.08	95.86	95.55	95.77
twonorm	90.27	96.67 ∘	96.30 ∘	95.17 ∘	94.53 ∘	96.80 ∘	95.40 ∘
vehicle	74.97	74.94	74.68	74.87	74.65	74.67	74.87
vote1	89.66	90.19	89.82	89.96	89.98	90.58	90.12
voting	96.71	96.48	96.37	96.55	96.32	96.43	96.30
vowel-context	92.23	92.81	93.42	92.78	93.41	93.24	93.40
vowel-nocontext	91.78	92.18	93.26 ∘	92.09	92.78	93.36 ∘	92.75
waveform	83.25	83.35	83.31	83.26	83.28	83.43	83.26
yeast	60.92	61.32	61.54	61.26	61.49	61.67	61.75
zip	93.17	93.28	93.75 ∘	93.27	93.64 ∘	93.77 ∘	93.66 ∘
zoo	93.20	93.11	93.31	93.01	93.50	93.41	93.30
Victorias-Empates-Derrotas		4-58-0	8-54-0	5-57-0	10-52-0	10-52-0	11-51-0

Tabla A.12: Tasas de acierto para las configuraciones de Random Forest.

Conjunto	Random Forest	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	23.68	23.73	23.93	23.85	23.93	24.07	24.17
anneal	99.69	99.58	99.58	99.58	99.55	99.50	99.58
audiology	79.39	80.14	79.81	79.67	80.29	79.67	79.89
autos	83.99	83.99	83.94	84.03	83.98	83.06	83.70
balance-scale	80.48	82.80 ○	84.11 ○	81.85 ○	83.63 ○	84.15 ○	83.31 ○
breast-w	96.38	96.64	96.74	96.51	96.74	96.71	96.88
breast-y	70.02	69.93	71.20	70.17	71.16	70.90	71.23
bupa	71.77	72.00	72.73	72.11	72.43	72.24	72.61
car	94.44	94.55	94.58	94.68	94.80	94.35	94.61
credit-a	86.04	86.06	86.48	85.93	86.22	86.77	86.72
credit-g	75.62	75.17	75.75	75.43	75.47	75.50	75.31
crx	85.41	85.77	86.42	85.74	86.33	86.46	86.39
dna	94.55	94.58	94.45	94.54	94.59	94.36	94.30
ecoli	84.71	85.53	85.77	85.03	85.48	86.28	86.16
glass	78.72	79.02	79.02	78.49	79.25	78.65	78.97
heart-c	81.39	81.20	82.28	81.68	81.75	82.32	82.31
heart-h	80.59	80.41	80.76	80.46	80.84	80.70	80.63
heart-s	91.75	92.22	92.55	92.22	92.88	92.07	92.39
heart-statlog	82.00	81.93	82.37	82.33	82.85	82.07	82.78
heart-v	75.10	74.75	75.85	74.75	75.95	75.55	75.35
hepatitis	83.33	83.85	84.23	84.35	83.88	84.72	83.96
horse-colic	85.48	85.32	85.32	85.51	85.13	85.35	84.86
hypo	99.01	99.00	98.97	99.03	98.96	98.99	98.99
ionosphere	93.68	93.76	93.42	93.76	93.60	93.68	93.54
iris	94.40	94.53	94.67	94.60	94.87	94.60	94.80
kr-vs-kp	99.26	99.18	98.97 •	99.19	99.03	98.99 •	99.02
krk	84.89	87.63 ○	84.70	87.65 ○	84.78	85.34 ○	85.30 ○
labor	86.97	90.47	89.27	88.73	89.30	90.73	90.57
led-24	74.43	74.55	74.60	74.41	74.57	74.48	74.50
letter	96.41	96.57	96.45	96.64 ○	96.51	96.49	96.56
lrd	88.31	88.21	88.52	88.14	88.38	88.33	88.42
lymphography	83.30	83.07	84.55	82.81	84.01	84.39	84.65
mushroom	100.00	100.00	100.00	100.00	100.00	100.00	100.00
nursery	99.05	98.97	98.74 •	99.03	98.80 •	98.57 •	98.71 •
optdigits	98.19	98.20	98.19	98.22	98.27	98.17	98.23
page	97.42	97.46	97.45	97.42	97.48	97.50	97.47
pendigits	99.17	99.22	99.19	99.23	99.17	99.17	99.19
phoneme	91.25	91.20	91.25	91.21	91.26	91.19	91.23
pima	75.58	75.98	76.10	75.82	75.61	76.33	75.80
primary	43.27	43.48	43.27	43.04	43.45	43.69	43.83
promoters	88.98	87.05	89.63	87.20	90.63	89.88	90.03
ringnorm	90.67	90.77	91.97	90.83	92.70	92.43	92.70
sat	91.80	91.86	91.83	91.81	91.73	91.81	91.84
segment	98.11	98.19	98.12	98.26	98.05	98.10	98.10
shuttle	99.99	99.99	99.98	99.99	99.98	99.98 •	99.98 •
sick	98.45	98.42	98.37	98.46	98.43	98.37	98.39
sonar	84.50	84.71	84.19	83.09	84.53	84.06	83.72
soybean-small	100.00	98.75	100.00	98.55	100.00	100.00	100.00
soybean	93.24	93.53	93.72	93.56	93.47	93.60	93.56
splice	95.27	95.05	95.24	95.05	95.09	95.13	95.05
threenorm	83.97	84.63	85.17	83.63	85.30	84.80	85.20
tic-tac-toe	96.42	95.99	95.46	96.23	95.76	95.52	95.63
twonorm	93.43	95.07	95.33	94.67	94.77	96.10	94.77
vehicle	75.07	75.04	75.06	75.13	74.59	74.71	74.80
vote1	89.54	89.47	89.65	89.96	90.05	89.89	89.96
voting	96.50	96.43	96.27	96.34	96.27	96.25	96.09
vowel-context	97.85	98.14	98.23	98.36	98.22	98.10	98.19
vowel-nocontext	95.94	97.32 ○	97.05	97.31 ○	96.64	97.44 ○	97.53 ○
waveform	84.77	84.85	84.93	84.81	84.67	84.87	84.96
yeast	60.58	61.08	61.49	60.70	61.59	61.50	61.30
zip	96.28	96.24	96.26	96.28	96.23	96.25	96.14
zoo	96.23	95.65	95.94	96.15	96.05	96.54	96.15
Victorias-Empates-Derrotas		3-59-0	1-59-2	4-58-0	1-60-1	3-56-3	3-57-2

Tabla A.13: Tasas de acierto para las configuraciones de $\mathcal{DN}$-Ensemble.

Conjunto	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	21.40	22.68	21.65	22.95	22.68	22.79
anneal	98.57	98.64	98.59	98.61	98.64	98.61
audiology	77.71	79.24	77.75	78.94	79.29	79.38
autos	81.96	83.42	81.77	83.22	83.13	82.93
balance-scale	84.12	84.44	79.92	83.62	84.59	83.75
breast-w	96.22	96.05	96.45	95.92	96.27	96.64
breast-y	74.32	73.20	74.28	73.58	73.45	73.66
bupa	68.15	71.45	68.33	71.86	71.57	71.83
car	92.45	92.77	92.57	93.24	92.76	93.40
credit-a	85.57	85.68	85.52	85.77	85.71	85.88
credit-g	71.53	72.24	71.57	72.09	72.49	72.30
crx	85.51	85.61	85.48	85.48	85.59	85.54
dna	92.54	92.73	92.54	92.65	92.73	92.66
ecoli	84.02	83.40	83.75	83.46	84.11	83.57
glass	68.51	71.31	68.33	71.35	71.40	71.21
heart-c	79.90	81.81	78.09	80.20	82.93	81.05
heart-h	80.94	81.00	80.56	81.55	81.54	81.45
heart-s	93.53	93.53	93.53	93.53	93.53	93.53
heart-statlog	79.33	80.70	78.59	80.85	81.30	81.19
heart-v	73.85	74.45	73.55	74.40	74.65	74.20
hepatitis	80.08	81.30	79.49	81.04	81.42	81.23
horse-colic	85.07	85.18	85.16	85.21	85.18	85.24
hypo	99.27	99.28	99.27	99.28	99.28	99.28
ionosphere	90.17	90.95	90.14	91.09	91.03	91.06
iris	94.73	94.80	94.73	94.87	94.80	94.80
kr-vs-kp	99.44	99.44	99.44	99.44	99.44	99.44
krk	82.51	83.36	82.55	83.58	83.30	83.50
labor	87.23	85.30	83.77	83.80	88.43	85.03
led-24	74.02	74.68	74.08	74.67	74.69	74.66
letter	88.43	89.46	88.52	89.38	89.25	89.30
lrd	82.25	83.15	82.32	83.02	83.24	83.13
lymphography	75.71	78.45	75.51	78.66	78.73	78.86
mushroom	100.00	100.00	100.00	100.00	100.00	100.00
nursery	97.18	97.21	97.19	97.32	97.21	97.33
optdigits	93.31	97.28	90.81	96.44	97.45	96.48
page	96.99	97.01	96.99	97.03	97.01	97.03
pendigits	98.02	98.18	97.31	97.58	98.24	97.54
phoneme	87.71	89.35	87.59	89.47	89.43	89.50
pima	75.58	76.44	75.31	76.33	76.52	76.21
primary	43.75	44.91	43.25	45.17	44.99	44.96
promoters	79.85	80.36	79.12	79.31	81.51	79.68
ringnorm	82.97	81.90	82.17	83.40	84.10	85.03
sat	88.35	89.59	88.01	89.49	90.32	90.05
segment	96.81	97.11	96.80	97.19	97.13	97.19
shuttle	99.97	99.97	99.97	99.97	99.97	99.97
sick	98.72	98.72	98.72	98.75	98.72	98.75
sonar	73.66	78.27	73.90	78.57	78.04	79.04
soybean-small	99.75	97.65	97.65	97.65	99.75	97.65
soybean	92.62	92.81	92.28	92.72	93.18	92.79
splice	94.16	94.22	94.17	94.22	94.22	94.24
threenorm	83.23	84.50	75.13	82.43	85.23	82.73
tic-tac-toe	87.86	95.40	87.19	95.66	95.53	95.70
twonorm	96.23	96.53	93.93	94.57	96.27	95.07
vehicle	72.42	73.17	72.57	73.18	73.27	73.12
vote1	89.93	89.20	88.85	89.10	90.19	89.75
voting	96.62	96.57	96.57	96.60	96.62	96.57
vowel-context	80.70	83.67	80.81	84.19	83.54	84.24
vowel-nocontext	81.35	87.68	80.71	86.53	87.44	85.93
waveform	76.45	79.35	75.95	78.32	79.89	78.47
yeast	58.41	60.04	58.33	60.07	60.22	60.04
zip	88.37	89.62	88.33	89.25	89.70	89.32
zoo	92.61	92.61	92.61	92.61	92.61	92.61

Tabla A.14: Tasas de acierto para las configuraciones de Random Subespaces(50 %).

Conjunto	Subspaces(50 %)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	25.53	25.83	25.53	25.59	25.73	25.67	25.72
anneal	98.75	98.83	98.84	98.83	98.84	98.88	98.79
audiology	79.23	79.75	79.88	79.70	79.43	79.79	79.43
autos	85.24	85.51	85.21	85.90	85.65	85.55	85.66
balance-scale	81.42	85.12 ∘	86.27 ∘	83.46	85.33 ∘	86.40 ∘	85.36 ∘
breast-w	96.57	96.87	96.70	96.82	96.83	96.88	97.01
breast-y	74.10	74.13	73.69	73.96	73.83	73.55	73.76
bupa	67.47	68.51	69.29	69.05	69.78	69.75	70.24
car	70.02	70.02	70.02	70.02	70.02	70.02	70.02
credit-a	86.20	86.39	86.38	86.42	86.57	86.38	86.61
credit-g	74.70	74.30	74.37	74.46	74.46	74.52	74.56
crx	86.26	86.57	86.62	86.41	86.43	86.52	86.46
dna	95.71	95.83	95.82	95.87	95.74	95.81	95.77
ecoli	85.57	85.60	85.54	85.63	85.69	85.72	85.63
glass	77.08	76.85	77.17	77.78	77.46	77.64	77.23
heart-c	82.11	82.97	83.23	83.00	83.49	83.47	83.33
heart-h	81.95	82.60	82.53	82.50	82.67	82.70	82.77
heart-s	93.53	93.53	93.53	93.53	93.53	93.53	93.53
heart-statlog	83.85	84.07	83.63	83.70	83.44	83.89	83.30
heart-v	73.95	74.10	74.50	74.30	74.35	74.40	74.45
hepatitis	83.12	83.57	83.63	83.37	83.13	83.55	83.12
horse-colic	84.93	84.88	84.93	84.83	84.94	85.02	85.07
hypo	98.71	98.73	98.75	98.72	98.76	98.75	98.75
ionosphere	93.68	93.42	93.68	93.31	93.51	93.71	93.48
iris	94.60	95.27	94.67	95.00	94.60	94.80	94.87
kr-vs-kp	97.31	97.21	97.20	97.19	97.17	97.19	97.15
krk	42.81	43.59	43.47	43.61	43.44	43.50	43.46
labor	80.57	86.87	86.23	85.73	86.60	88.43	88.03
led-24	74.37	74.58	74.62	74.56	74.53	74.59	74.52
letter	96.08	96.06	96.16	96.06	96.14	96.13	96.10
lrd	86.09	86.52	86.71	86.61	86.97	86.69	86.76
lymphography	79.82	81.36	81.76	81.22	81.77	82.17	82.18
mushroom	100.00	100.00	100.00	100.00	100.00	100.00	100.00
nursery	91.41	91.36	91.38	91.39	91.40	91.37	91.40
optdigits	98.09	98.12	98.18	98.06	98.15	98.19	98.16
page	97.41	97.42	97.41	97.43	97.45	97.40	97.44
pendigits	99.02	99.04	99.07	99.04	99.05	99.05	99.03
phoneme	85.68	86.51 ∘	87.45 ∘	86.57 ∘	87.55 ∘	87.43 ∘	87.56 ∘
pima	74.88	75.42	75.44	75.18	75.31	75.29	75.17
primary	45.46	46.23	46.29	45.79	46.02	46.29	46.02
promoters	86.74	89.71	89.99	89.73	89.75	90.76	89.68
ringnorm	90.57	92.13	92.00	91.53	91.70	92.33	92.30
sat	91.65	91.76	91.75	91.70	91.76	91.80	91.75
segment	97.71	97.71	97.87	97.69	97.79	97.87	97.82
shuttle	99.93	99.95	99.95	99.95	99.95	99.95	99.96
sick	95.76	95.86	96.24	95.91	96.17	96.24	96.17
sonar	82.31	83.71	83.31	82.73	82.84	83.90	83.08
soybean-small	98.95	100.00	99.55	99.35	99.55	100.00	99.55
soybean	94.76	95.07	95.02	95.04	95.07	95.11	95.18
splice	96.14	96.14	96.11	96.11	96.03	96.11	96.05
threenorm	83.00	84.90	85.73	84.07	85.17	85.93	85.37
tic-tac-toe	80.96	81.08	81.24	81.16	81.37	81.14	81.38
twonorm	90.90	95.40 ∘	95.63 ∘	94.33 ∘	94.50 ∘	95.50 ∘	95.20 ∘
vehicle	75.09	74.88	74.95	74.79	74.94	74.86	74.88
vote1	89.64	90.28	90.16	90.10	90.05	90.44	90.19
voting	95.38	95.29	95.43	95.38	95.38	95.15	95.38
vowel-context	96.25	96.74	96.84	96.67	96.92	96.78	96.91
vowel-nocontext	94.84	95.20	95.66	94.93	95.69	95.64	95.61
waveform	84.45	84.46	84.39	84.46	84.44	84.43	84.36
yeast	59.36	60.19	60.23	59.96	60.36	60.47	60.31
zip	95.75	95.71	95.78	95.69	95.70	95.77	95.72
zoo	94.08	94.56	94.96	94.85	94.96	95.16	94.96
Victorias-Empates-Derrotas		3-59-0	3-59-0	2-60-0	3-59-0	3-59-0	3-59-0

Tabla A.15: Tasas de acierto para las configuraciones de Random Subespaces(75 %).

Conjunto	Subspaces(75 %)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	24.08	24.20	24.61	24.24	24.74	24.46	24.59
anneal	98.64	98.68	98.81	98.65	98.84	98.80	98.83
audiology	80.84	81.86	81.77	81.59	81.64	81.73	81.77
autos	85.67	85.71	86.20	85.81	86.53	86.00	86.24
balance-scale	77.58	84.64 ∘	85.44 ∘	82.43 ∘	85.36 ∘	85.53 ∘	85.40 ∘
breast-w	95.78	96.60	96.35	96.64	96.21	96.70	96.84 ∘
breast-y	74.24	74.31	73.44	74.31	73.34	73.23	73.27
bupa	67.69	70.33	72.18 ∘	70.62	72.16 ∘	72.32 ∘	72.39 ∘
car	92.38	92.17	92.65	92.32	92.93	92.59	93.00
credit-a	86.16	86.06	86.33	85.83	86.42	86.33	86.41
credit-g	73.73	74.09	74.77	73.94	74.92	74.85	75.01
crx	85.99	86.20	86.41	86.39	86.64	86.35	86.61
dna	95.29	95.32	95.35	95.34	95.32	95.35	95.33
ecoli	84.94	85.21	84.95	85.12	84.53	85.15	84.47
glass	73.04	74.38	74.95	75.23	75.23	74.85	74.99
heart-c	78.88	81.87	82.53 ∘	80.66	81.72 ∘	83.03 ∘	82.28 ∘
heart-h	81.04	82.03	82.22	81.55	82.15	82.50	82.05
heart-s	93.53	93.53	93.53	93.53	93.53	93.53	93.53
heart-statlog	81.30	82.26	82.52	82.19	83.04	82.70	82.89
heart-v	73.55	73.85	74.20	73.80	74.05	74.35	74.15
hepatitis	80.79	82.00	81.83	82.01	82.21	82.10	82.35
horse-colic	85.18	85.24	85.21	85.18	85.37	85.26	85.40
hypo	99.21	99.23	99.23	99.24	99.23	99.23	99.23
ionosphere	92.82	92.85	93.19	92.85	92.99	93.17	92.91
iris	94.73	94.93	94.47	95.07	94.87	94.60	94.80
kr-vs-kp	99.21	99.20	99.17	99.19	99.19	99.17	99.18
krk	81.79	82.86	82.99 ∘	82.99 ∘	83.12 ∘	83.00 ∘	83.11 ∘
labor	79.83	87.80 ∘	85.53	85.23	85.23	88.70 ∘	86.60
led-24	74.85	74.88	74.93	74.99	74.91	74.96	74.92
letter	95.88	95.84	96.00	95.82	96.04	95.97	95.98
lrd	85.03	85.09	85.52	85.26	85.47	85.46	85.52
lymphography	78.03	79.91	81.38	78.82	81.23	82.07	81.72
mushroom	100.00	100.00	100.00	100.00	100.00	100.00	100.00
nursery	94.82	94.75	94.71	94.75	94.78	94.70	94.77
optdigits	96.98	97.73 ∘	97.92 ∘	97.23	97.83 ∘	97.96 ∘	97.84 ∘
page	97.24	97.25	97.26	97.26	97.27	97.26	97.28
pendigits	98.75	98.92	99.04 ∘	98.86	98.96 ∘	99.03 ∘	98.97 ∘
phoneme	87.38	88.44 ∘	89.24 ∘	88.56 ∘	89.36 ∘	89.28 ∘	89.39 ∘
pima	75.34	75.43	75.96	75.37	76.04	75.94	75.98
primary	44.72	45.65	45.85	45.41	45.76	46.05	45.64
promoters	80.73	84.73	84.15	82.65	83.08	85.10	83.67
ringnorm	86.63	88.43	87.80	88.73	88.10	88.80	89.03
sat	90.72	91.42 ∘	91.49 ∘	91.31 ∘	91.37 ∘	91.56 ∘	91.44 ∘
segment	97.57	97.58	97.64	97.58	97.74	97.68	97.73
shuttle	99.97	99.98	99.98	99.98	99.98	99.98	99.98
sick	98.49	98.48	98.50	98.49	98.49	98.50	98.49
sonar	78.56	79.29	81.25	78.99	79.81	81.54	80.34
soybean-small	97.80	100.00	98.65	99.10	98.45	100.00	99.35
soybean	93.51	94.42	94.42	94.32	94.54	94.67 ∘	94.60 ∘
splice	95.08	95.10	95.10	95.09	95.16	95.11	95.16
threenorm	78.07	84.57 ∘	84.67 ∘	81.70	84.10 ∘	85.60 ∘	84.77 ∘
tic-tac-toe	88.49	89.29	89.52	89.48	90.63	89.49	90.72 ∘
twonorm	82.87	95.47 ∘	96.20 ∘	94.03 ∘	94.63 ∘	96.13 ∘	94.63 ∘
vehicle	74.49	74.54	74.57	74.34	74.42	74.32	74.32
vote1	89.45	90.37	89.61	90.12	89.89	90.55	90.23
voting	96.64	96.71	96.46	96.62	96.48	96.46	96.57
vowel-context	90.81	91.46	93.15 ∘	91.40	93.30 ∘	93.07 ∘	93.06 ∘
vowel-nocontext	87.53	89.31	93.13 ∘	88.73	92.57 ∘	92.76 ∘	92.26 ∘
waveform	82.77	83.16	83.37	82.97	83.25	83.52	83.21
yeast	59.78	61.21	61.25	61.18	61.26	61.39	61.61 ∘
zip	94.29	94.28	94.67 ∘	94.25	94.59	94.68 ∘	94.61
zoo	92.20	92.60	93.09	92.79	93.40	93.09	93.20
Victorias-Empates-Derrotas		7-55-0	13-49-0	5-57-0	12-50-0	15-47-0	16-46-0

Tabla A.16: Tasas de acierto para las configuraciones de AdaBoost(W).

Conjunto	AdaBoost(W)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	22.59	22.48	23.11	22.45	23.06	22.60	22.85
anneal	99.63	99.63	99.63	99.60	99.64	99.60	99.66
audiology	84.61	85.05	85.58	84.96	85.45	86.11	85.84
autos	86.59	86.49	86.20	86.64	86.20	86.29	86.17
balance-scale	76.36	78.30	78.57 ∘	76.72	76.80	78.57 ∘	77.18
breast-w	96.61	96.61	96.71	96.67	96.60	96.67	96.84
breast-y	66.43	67.50	68.18	66.54	68.13	68.57	68.12
bupa	70.20	70.62	70.44	71.45	70.92	70.05	69.85
car	96.72	96.92	96.79	96.89	97.40 ∘	96.78	97.26
credit-a	85.87	85.91	86.09	86.20	86.19	85.99	86.01
credit-g	73.75	74.18	73.88	74.17	74.54	74.09	74.36
crx	86.01	85.86	85.81	86.04	85.80	86.42	85.96
dna	95.04	95.01	94.99	95.03	95.02	95.06	95.05
ecoli	83.51	83.98	84.64	83.62	84.23	84.25	84.43
glass	78.08	78.45	77.79	78.72	77.57	77.84	77.75
heart-c	80.11	80.11	81.09	80.93	81.59	81.65	80.96
heart-h	79.40	79.02	77.93	78.82	78.58	78.04	78.68
heart-s	90.78	91.03	91.41	91.01	91.18	91.26	91.58
heart-statlog	80.44	81.11	81.07	80.19	80.33	81.04	81.19
heart-v	71.80	70.80	71.80	71.95	71.70	71.00	72.10
hepatitis	84.15	84.38	83.62	84.12	83.70	85.18	83.96
horse-colic	81.76	82.61	83.53	82.63	82.61	83.20	82.36
hypo	98.98	98.95	99.05	98.97	98.99	99.04	98.99
ionosphere	93.85	93.62	93.97	94.13	94.02	93.91	93.93
iris	94.53	94.20	94.07	94.13	94.00	94.27	94.27
kr-vs-kp	99.60	99.62	99.61	99.62	99.64	99.61	99.62
krk	89.47	89.60	89.73	89.82	89.94 ∘	89.77	89.99 ∘
labor	89.10	92.07	91.90	90.10	89.27	92.03	91.37
led-24	69.80	71.99 ∘	72.56 ∘	71.72 ∘	72.32 ∘	72.72 ∘	72.58 ∘
letter	97.03	97.00	97.03	96.99	97.03	97.01	97.04
lrd	88.17	88.12	88.42	87.85	88.25	88.53	87.91
lymphography	84.12	83.66	84.33	84.06	84.81	84.60	84.80
mushroom	100.00	99.98	99.99	100.00	100.00	99.98	99.99
nursery	99.75	99.76	99.77	99.76	99.80	99.74	99.81
optdigits	98.40	98.46	98.46	98.41	98.45	98.42	98.44
page	97.07	97.09	97.09	97.11	97.03	97.06	97.03
pendigits	99.38	99.38	99.38	99.36	99.38	99.38	99.39
phoneme	91.18	91.25	91.27	91.30	91.15	91.18	91.29
pima	73.71	73.67	74.06	73.77	74.23	73.94	74.38
primary	41.65	42.54	43.29	41.98	42.87	43.04	42.84
promoters	94.08	94.14	95.28	93.57	94.94	94.28	93.39
ringnorm	94.17	94.47	94.60	95.00	94.67	94.70	95.13
sat	92.13	92.20	92.19	92.21	92.19	92.21	92.08
segment	98.60	98.58	98.53	98.51	98.50	98.59	98.63
shuttle	99.99	99.99	99.99	99.99	99.99	99.99	99.99
sick	99.07	99.05	98.97	99.07	99.02	99.00	99.03
sonar	84.65	85.24	84.97	84.08	85.52	85.63	85.05
soybean-small	97.65	98.15	97.65	97.70	97.25	98.15	97.30
soybean	93.31	94.00	94.01	94.16	94.30	94.19	93.92
splice	94.91	94.88	94.94	94.87	94.92	94.94	94.92
threenorm	84.73	86.33	85.50	84.93	84.87	86.03	85.17
tic-tac-toe	98.92	98.93	98.90	98.97	98.85	98.89	98.92
twonorm	94.07	96.00	95.53	95.80	94.87	95.97	95.43
vehicle	77.41	77.60	77.69	77.74	77.83	77.00	77.07
vote1	89.17	89.22	89.61	89.26	89.75	89.56	89.72
voting	95.31	95.61	95.74	95.72	95.74	95.60	95.56
vowel-context	95.93	96.48	96.54	96.39	96.64	96.47	96.62
vowel-nocontext	95.23	95.81	96.31	95.74	96.22	96.36	96.45 ∘
waveform	84.35	84.41	84.63	84.30	84.59	84.47	84.48
yeast	58.44	59.14	59.45	59.06	59.58	59.54	59.55
zip	96.74	96.73	96.71	96.74	96.74	96.74	96.71
zoo	96.35	95.65	96.25	96.35	96.14	96.15	96.25
Victorias-Empates-Derrotas		1-61-0	2-60-0	1-61-0	3-59-0	2-60-0	3-59-0

Tabla A.17: Tasas de acierto para las configuraciones de AdaBoost(S).

Conjunto	AdaBoost(S)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	23.21	23.12	23.31	23.17	23.50	23.40	23.27
anneal	99.67	99.63	99.64	99.66	99.66	99.64	99.69
audiology	84.56	84.82	85.63	84.75	85.32	85.72	85.18
autos	86.10	86.32	86.29	86.78	86.64	86.48	86.48
balance-scale	75.53	75.71	75.97	75.87	75.75	76.37	76.08
breast-w	96.68	96.70	96.68	96.55	96.78	96.72	96.87
breast-y	67.16	66.96	68.11	67.78	67.60	67.68	67.98
bupa	70.77	70.61	69.85	70.81	70.90	70.47	69.89
car	96.60	96.56	96.55	96.80	97.00	96.67	97.01
credit-a	86.39	86.23	86.39	86.41	85.94	86.39	86.45
credit-g	74.52	74.58	74.94	74.45	74.78	74.62	74.63
crx	85.88	85.87	86.29	85.94	86.64	86.13	86.42
dna	94.95	94.93	95.00	95.08	95.00	95.02	94.95
ecoli	83.99	84.70	84.82	84.55	84.59	85.12	84.43
glass	79.10	79.05	78.08	79.06	78.39	78.31	79.15
heart-c	79.78	80.33	80.30	81.02	80.93	80.97	80.96
heart-h	80.04	79.46	79.16	79.19	79.42	78.62	78.99
heart-s	91.33	91.10	91.65	91.19	91.01	90.44	91.03
heart-statlog	79.44	80.19	80.22	79.89	81.00	79.96	80.30
heart-v	70.40	70.95	71.35	71.35	70.95	72.40	71.85
hepatitis	84.61	84.10	83.69	83.91	83.33	84.53	83.95
horse-colic	82.09	82.80	82.85	82.44	83.18	82.95	83.12
hypo	98.99	98.99	99.04	99.08	99.05	99.05	99.07
ionosphere	94.22	94.22	94.39	94.05	94.33	94.22	94.08
iris	94.13	94.53	94.60	94.27	94.40	94.80	94.60
kr-vs-kp	99.61	99.64	99.63	99.62	99.63	99.60	99.61
krk	89.39	89.43	89.49	89.62	89.73 ∘	89.46	89.73 ∘
labor	87.47	90.93	89.73	89.97	89.23	89.83	89.83
led-24	71.33	72.32 ∘	73.07 ∘	72.45 ∘	72.99 ∘	73.06 ∘	72.89 ∘
letter	96.87	96.85	96.94	96.89	96.96	96.94	96.92
lrd	88.10	88.27	88.32	88.46	88.42	88.61	88.19
lymphography	83.72	84.45	83.93	83.63	84.41	84.51	83.66
mushroom	100.00	100.00	100.00	99.98	100.00	100.00	99.99
nursery	99.74	99.75	99.72	99.77	99.79	99.72	99.79
optdigits	98.42	98.44	98.41	98.41	98.45	98.46	98.38
page	97.11	97.08	97.20	97.11	97.12	97.16	97.11
pendigits	99.36	99.39	99.38	99.38	99.39	99.37	99.40
phoneme	91.27	91.33	91.41	91.30	91.39	91.20	91.37
pima	73.71	73.62	73.76	73.89	73.94	73.79	73.47
primary	42.37	42.21	42.39	41.65	42.25	42.07	42.30
promoters	93.78	93.96	93.11	92.45	92.74	93.31	93.65
ringnorm	95.67	95.37	95.73	95.70	95.40	95.30	95.77
sat	92.01	92.04	92.02	91.98	92.03	92.15	92.07
segment	98.57	98.55	98.54	98.58	98.59	98.52	98.59
shuttle	99.99	99.99	99.99	99.99	99.99	99.99	99.99
sick	99.06	99.05	99.00	99.07	98.99	99.00	99.06
sonar	82.41	82.89	83.58	82.36	83.16	83.12	83.76
soybean-small	98.35	99.55	99.30	99.50	98.35	99.55	99.35
soybean	93.31	94.04	94.04	93.88	94.25	94.23	93.95
splice	94.84	94.93	94.93	94.86	94.92	95.03	94.92
threenorm	84.10	85.17	85.57	84.53	84.93	85.27	85.27
tic-tac-toe	98.94	98.87	98.85	98.81	98.99	98.96	98.98
twonorm	94.33	95.43	95.33	94.97	95.30	95.57	94.80
vehicle	77.68	77.62	76.75	77.60	77.33	77.15	77.26
vote1	89.36	89.22	89.12	89.42	89.31	89.38	89.73
voting	95.10	95.49	95.70	95.44	95.56	95.72	95.44
vowel-context	96.26	96.57	96.64	96.86	96.83	96.74	96.70
vowel-nocontext	95.62	96.15	96.57	95.88	96.29	96.53	96.34
waveform	84.15	84.23	84.33	84.34	84.45	84.31	84.25
yeast	59.36	59.94	59.68	59.40	59.63	59.80	60.11
zip	96.59	96.62	96.64	96.64	96.62	96.62	96.66
zoo	95.06	95.26	95.35	95.25	95.66	95.65	96.15
Victorias-Empates-Derrotas		1-61-0	1-61-0	1-61-0	2-60-0	1-61-0	2-60-0

Tabla A.18: Tasas de acierto para las configuraciones de MultiBoost(W).

Conjunto	MultiBoost(W)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	23.57	23.44	23.79	23.68	23.58	23.79	23.69
anneal	99.68	99.63	99.54	99.65	99.64	99.53	99.63
audiology	85.14	85.62	85.80	85.66	85.62	85.94	85.49
autos	86.10	86.04	86.10	85.70	86.15	85.99	85.90
balance-scale	80.60	82.38 ○	83.08 ○	81.28	81.72	82.77 ○	81.80
breast-w	96.60	96.65	96.57	96.78	96.67	96.90	96.77
breast-y	69.68	69.48	70.21	69.27	70.77	71.31	70.84
bupa	72.52	72.61	72.01	73.28	72.35	71.17	72.42
car	95.96	95.94	95.90	96.13	96.60	96.00	96.47
credit-a	86.90	86.74	86.71	86.67	86.80	86.81	86.70
credit-g	75.39	75.09	75.43	75.21	75.39	75.57	75.66
crx	86.84	86.36	86.71	86.62	86.99	87.33	86.96
dna	95.63	95.61	95.65	95.68	95.62	95.62	95.61
ecoli	84.58	85.20	85.15	84.76	85.39	85.41	85.00
glass	77.24	77.38	76.91	78.22	77.51	77.57	77.38
heart-c	81.28	81.78	82.71	80.93	82.38	82.64	82.50
heart-h	79.70	79.81	80.28	79.80	80.76	79.88	80.34
heart-s	90.93	91.01	91.41	90.93	91.66	91.49	91.83
heart-statlog	81.56	82.00	81.70	81.11	82.04	82.26	81.07
heart-v	74.40	74.25	73.90	74.05	74.55	74.45	74.00
hepatitis	83.57	84.21	84.00	83.71	83.23	83.44	83.45
horse-colic	84.67	84.39	84.59	84.02	84.37	84.50	84.13
hypo	99.12	99.13	99.17	99.11	99.15	99.17	99.13
ionosphere	93.73	93.88	93.85	93.88	94.28	94.25	94.02
iris	94.20	94.13	93.87	94.27	94.00	94.00	94.53
kr-vs-kp	99.65	99.64	99.62	99.64	99.65	99.64	99.65
krk	88.49	88.36	88.41	88.64	88.72	88.45	88.73
labor	86.57	91.77	89.03	89.87	88.70	90.97	90.50
led-24	72.60	73.60 ○	73.98 ○	73.45 ○	74.05 ○	73.99 ○	73.92 ○
letter	96.67	96.64	96.70	96.68	96.72	96.69	96.70
lrd	87.65	87.68	87.67	87.85	87.91	87.87	87.44
lymphography	83.10	83.81	83.99	83.32	84.06	83.80	83.73
mushroom	100.00	99.98	99.99	100.00	100.00	99.98	99.99
nursery	99.70	99.69	99.59	99.70	99.70	99.59 •	99.71
optdigits	98.08	98.20	98.19	98.15	98.19	98.27	98.18
page	97.38	97.35	97.36	97.37	97.38	97.38	97.37
pendigits	99.24	99.28	99.25	99.25	99.29	99.27	99.26
phoneme	91.12	91.09	91.07	91.20	91.18	91.15	91.14
pima	75.54	75.52	75.32	75.18	75.76	75.58	75.17
primary	42.27	43.16	44.14	42.74	43.01	43.40	43.99
promoters	94.55	94.94	94.06	94.27	94.95	94.85	94.67
ringnorm	92.67	93.13	93.37	93.77	93.47	93.60	93.80
sat	91.83	91.92	91.99	91.93	91.94	91.92	91.98
segment	98.35	98.32	98.33	98.34	98.43	98.30	98.36
shuttle	99.99	99.99	99.99	99.99	99.99	99.99	99.99
sick	99.06	99.05	99.02	99.04	99.07	99.06	99.04
sonar	83.03	83.90	83.85	83.84	82.98	84.33	83.79
soybean-small	97.65	98.15	97.65	97.70	97.25	98.15	97.30
soybean	93.32	94.24	94.16	94.10	94.35	94.19	94.27
splice	95.56	95.60	95.54	95.61	95.47	95.55	95.58
threenorm	85.00	85.60	86.27	84.77	85.53	86.33	85.37
tic-tac-toe	98.31	98.26	98.57	98.31	98.43	98.32	98.59
twonorm	93.60	96.13	96.33	94.77	95.50	96.50	95.33
vehicle	76.63	76.19	76.57	76.87	76.54	76.60	76.77
vote1	89.84	89.91	90.25	90.02	90.16	89.93	90.07
voting	95.59	95.90	96.20	95.90	96.20	96.11	96.20
vowel-context	95.14	95.70	95.70	95.46	95.75	95.70	95.94
vowel-nocontext	94.41	95.24	95.75	94.73	95.44	95.68	95.27
waveform	84.57	84.61	84.86	84.68	84.72	84.80	84.82
yeast	60.66	60.60	60.81	60.46	61.23	61.22	61.06
zip	96.38	96.31	96.33	96.27	96.40	96.35	96.32
zoo	96.05	95.85	96.05	96.35	95.75	96.25	96.04
Victorias-Empates-Derrotas		2-60-0	2-60-0	1-61-0	1-61-0	2-59-1	1-61-0

Tabla A.19: Tasas de acierto para las configuraciones de MultiBoost(S).

Conjunto	Subspaces(50 %)	$\mathcal{DN}_P$	$\mathcal{DN}_V$	$\mathcal{DN}_{PA}$	$\mathcal{DN}_{VA}$	$\mathcal{DN}_{VP}$	$\mathcal{DN}$
abalone	24.07	23.98	24.37	23.84	24.06	24.42	24.05
anneal	99.68	99.57	99.62	99.59	99.62	99.61	99.64
audiology	85.06	85.15	85.64	85.28	84.97	85.32	84.91
autos	85.99	85.85	84.92	85.31	84.83	85.81	85.41
balance-scale	80.33	81.23	81.40	80.68	81.33	81.45	81.41
breast-w	96.50	96.73	96.71	96.70	96.62	96.78	96.60
breast-y	69.35	70.64	70.92	70.01	70.67	70.08	70.57
bupa	72.84	72.61	71.56	72.81	72.90	71.89	72.64
car	95.80	95.73	95.64	95.93	96.23	95.78	96.32
credit-a	86.77	86.84	86.88	86.74	86.52	86.88	86.87
credit-g	75.29	75.48	75.87	75.50	75.24	75.32	75.79
crx	86.74	87.03	86.91	86.64	86.78	86.68	86.70
dna	95.52	95.48	95.56	95.54	95.47	95.53	95.55
ecoli	85.15	85.09	85.65	84.94	84.79	85.62	85.50
glass	78.50	79.11	77.47	78.95	78.31	77.51	77.57
heart-c	81.39	81.52	81.84	81.25	81.61	82.54	81.95
heart-h	80.68	80.28	80.79	80.41	80.48	80.99	79.90
heart-s	91.26	91.91	91.25	91.42	91.51	91.67	91.51
heart-statlog	80.78	80.89	81.93	81.26	81.56	81.59	81.74
heart-v	74.40	73.85	74.90	73.65	73.80	75.00	74.00
hepatitis	83.27	83.58	84.08	84.30	83.13	84.08	83.39
horse-colic	84.10	84.42	84.78	83.85	84.64	84.10	84.61
hypo	99.14	99.14	99.17	99.12	99.15	99.17	99.19
ionosphere	93.73	94.10	94.25	93.71	94.13	94.27	94.13
iris	94.53	94.33	94.47	94.40	94.07	94.53	94.20
kr-vs-kp	99.65	99.63	99.62	99.63	99.63	99.60	99.62
krk	88.48	88.27	88.26	88.63	88.73	88.23	88.66
labor	87.57	90.03	90.43	90.43	89.50	91.10	90.60
led-24	73.14	73.85 ○	74.17 ○	73.76 ○	74.18 ○	74.12 ○	74.15 ○
letter	96.58	96.56	96.65	96.58	96.63	96.60	96.62
lrd	88.04	88.17	88.46	88.08	88.33	88.33	88.06
lymphography	83.40	84.53	84.62	84.00	84.21	84.27	84.06
mushroom	100.00	100.00	100.00	99.98	100.00	100.00	99.99
nursery	99.68	99.64	99.53 •	99.67	99.68	99.57	99.68
optdigits	98.10	98.16	98.18	98.12	98.16	98.20	98.14
page	97.39	97.37	97.41	97.42	97.39	97.43	97.39
pendigits	99.25	99.29	99.29	99.26	99.28	99.29	99.29
phoneme	91.14	91.10	91.23	91.17	91.07	91.13	91.05
pima	74.92	75.71	75.40	75.13	75.22	75.76	75.57
primary	43.60	43.24	43.69	43.27	44.40	43.92	44.07
promoters	94.27	92.75	94.68	93.95	94.10	93.81	93.25
ringnorm	94.10	94.53	94.70	95.00	94.43	95.23	94.80
sat	91.72	91.77	91.96	91.82	91.83	91.82	91.75
segment	98.34	98.29	98.42	98.32	98.39	98.39	98.39
shuttle	99.99	99.99	99.99	99.99	99.99	99.99	99.99
sick	99.04	99.04	99.00	99.02	99.01	98.99	98.99
sonar	81.44	82.41	81.99	82.09	82.89	81.93	81.99
soybean-small	98.15	99.30	99.10	99.05	97.85	99.30	98.80
soybean	93.54	94.16	94.30	94.25	94.25	94.32	94.38
splice	95.48	95.53	95.47	95.53	95.49	95.45	95.43
threenorm	83.97	85.47	85.47	84.27	85.33	84.97	85.17
tic-tac-toe	98.31	98.29	98.35	98.34	98.36	98.29	98.24
twonorm	93.67	95.93	96.07	94.97	94.47	95.57	95.60
vehicle	76.48	76.25	75.85	76.65	76.39	76.23	76.08
vote1	89.93	90.48	90.18	89.72	90.37	90.29	90.23
voting	95.56	96.09	96.16	95.79	96.09	96.18	96.18
vowel-context	95.15	95.63	95.90	95.55	95.59	95.74	95.64
vowel-nocontext	94.52	95.04	95.75 ○	94.87	95.38	95.48	95.17
waveform	84.57	84.59	84.58	84.54	84.65	84.52	84.72
yeast	60.97	60.95	61.10	61.12	61.54	61.16	61.28
zip	96.26	96.21	96.28	96.25	96.26	96.26	96.26
zoo	95.25	95.65	95.65	95.35	95.95	95.85	96.35
Victorias-Empates-Derrotas		1-61-0	2-59-1	1-61-0	1-61-0	1-61-0	1-61-0

A.4. Tasas de acierto para $\mathcal{RFW}$

Tabla A.20: Tasas de acierto para $\mathcal{RFW}$ con árboles podados y $p = 1 \ldots 4$.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	24.77	25.33	25.47	25.46
anneal	99.00	99.02	99.01	99.03
audiology	83.68	82.02	80.34	79.24
autos	84.87	83.84	83.27	83.11
balance-scale	78.76	79.29	79.48	79.71
breast-w	96.77	96.72	96.85	96.90
breast-y	73.68	73.72	73.55	73.44
bupa	69.54	70.44	70.79	70.99
car	93.75	93.61	93.28	92.95
credit-a	85.70	86.03	86.32	86.29
credit-g	75.41	75.73	75.24	74.29
crx	85.84	86.19	86.04	86.22
dna	95.18	95.45	95.56	95.61
ecoli	86.13	86.70	86.25	86.35
glass	76.18	77.44	78.32	78.75
heart-c	81.22	82.38	82.64	83.10
heart-h	80.72	81.13	81.50	81.78
heart-s	93.53	93.53	93.53	93.53
heart-statlog	82.78	83.22	83.63	83.89
heart-v	73.80	73.90	73.95	74.55
hepatitis	81.31	82.35	82.93	83.06
horse-colic	84.94	84.94	84.88	84.85
hypo	99.27	99.23	99.20	99.14
ionosphere	93.48	93.45	93.57	93.40
iris	94.93	95.07	95.13	95.13
kr-vs-kp	99.47	99.37	99.21	99.12
krk	88.21	88.82	88.75	88.41
labor	80.90	81.33	81.17	81.50
led-24	75.23	75.23	75.18	75.05
letter	96.46	96.11	95.69	95.27
lrd	87.08	87.57	87.74	87.99
lymphography	81.19	82.14	81.89	82.22
mushroom	100.00	100.00	100.00	100.00
nursery	97.18	96.88	96.75	96.55
optdigits	98.22	98.30	98.19	98.08
page	97.39	97.49	97.51	97.50
pendigits	99.14	99.12	99.08	99.01
phoneme	89.27	89.55	89.64	89.52
pima	74.97	75.30	75.61	75.81
primary	45.90	45.90	46.22	46.43
promoters	87.80	91.22	92.04	91.09
ringnorm	89.30	91.53	92.77	93.33
sat	91.88	91.78	91.75	91.61
segment	98.02	98.11	98.08	98.06
shuttle	99.98	99.98	99.98	99.98
sick	98.85	98.71	98.56	98.53
sonar	81.83	84.53	84.54	85.06
soybean-small	100.00	100.00	100.00	100.00
soybean	95.09	95.31	95.12	94.82
splice	95.19	95.66	95.99	96.07
threenorm	80.70	82.67	83.77	83.83
tic-tac-toe	93.93	93.58	92.44	91.53
twonorm	91.03	93.07	93.87	94.07
vehicle	75.76	75.76	75.73	75.26
vote1	89.91	90.21	90.46	90.58
voting	96.30	95.82	95.59	95.47
vowel-context	95.48	96.82	96.75	96.62
vowel-nocontext	94.85	95.45	95.49	95.31
waveform	84.36	84.76	84.82	84.83
yeast	61.13	62.28	62.17	62.15
zip	96.34	96.61	96.59	96.64
zoo	93.98	95.06	95.65	96.24

Tabla A.21: Tasas de acierto para Bagging y Random Subspaces 50 % y 75 % contra $\mathcal{RFW}$ $p = 1 \ldots 4$ (para árboles podados en ambos casos). Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	Bagging	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$			
Conjunto		1	2	3	4	50 %	1	2	3	4	75 %	1	2	3	4
abalone	23.92		∘	∘	∘	25.53					24.08				
anneal	98.80					98.75					98.64				
audiology	80.88					79.23	∘				80.84				
autos	84.19					85.24					85.67				
balance-scale	82.23	•	•	•	•	81.42					77.58				
breast-w	96.24					96.57					95.78			∘	∘
breast-y	73.13					74.10					74.24				
bupa	73.10	•				67.47					67.69				
car	93.59					70.02	∘	∘	∘	∘	92.38	∘			
credit-a	85.91					86.20					86.16				
credit-g	74.36					74.70					73.73	∘	∘		
crx	86.17					86.26					85.99				
dna	94.89			∘		95.71					95.29				
ecoli	84.76					85.57					84.94				
glass	74.11					77.08					73.04				
heart-c	80.10					82.11					78.88				∘
heart-h	80.07					81.95					81.04				
heart-s	93.21					93.53					93.53				
heart-statlog	81.19					83.85					81.30				
heart-v	75.75					73.95					73.55				
hepatitis	81.50					83.12					80.79				
horse-colic	85.26					84.93					85.18				
hypo	99.26					98.71	∘	∘	∘	∘	99.21				
ionosphere	92.57					93.68					92.82				
iris	94.60					94.60					94.73				
kr-vs-kp	99.46				•	97.31	∘	∘	∘	∘	99.21				
krk	84.91	∘	∘	∘	∘	42.81	∘	∘	∘	∘	81.79	∘	∘	∘	∘
labor	85.23					80.57					79.83				
led-24	74.59	∘				74.37					74.85				
letter	93.72	∘	∘	∘	∘	96.08	∘		•	•	95.88	∘			•
lrd	86.88					86.09				∘	85.03	∘	∘	∘	∘
lymphography	79.82					79.82					78.03				
mushroom	100.00					100.00					100.00				
nursery	97.42	•	•	•	•	91.41	∘	∘	∘	∘	94.82	∘	∘	∘	∘
optdigits	95.75	∘	∘	∘	∘	98.09					96.98	∘	∘	∘	∘
page	97.38					97.41					97.24		∘	∘	
pendigits	98.23	∘	∘	∘	∘	99.02					98.75	∘	∘	∘	∘
phoneme	89.56					85.68	∘	∘	∘	∘	87.38	∘	∘	∘	∘
pima	76.05					74.88					75.34				
primary	44.96					45.46					44.72				
promoters	85.29					86.74					80.73		∘	∘	∘
ringnorm	88.43			∘	∘	90.57					86.63		∘	∘	∘
sat	90.77	∘	∘	∘	∘	91.65					90.72	∘	∘	∘	∘
segment	97.60					97.71					97.57		∘		
shuttle	99.97	∘	∘			99.93	∘	∘	∘	∘	99.97				
sick	98.86					95.76	∘	∘	∘	∘	98.49	∘			
sonar	80.30					82.31					78.56		∘		∘
soybean-small	97.65					98.95					97.80				
soybean	93.03	∘	∘	∘		94.76					93.51	∘	∘	∘	
splice	94.66	∘	∘	∘	∘	96.14	•				95.08		∘	∘	∘
threenorm	82.10					83.00					78.07		∘	∘	∘
tic-tac-toe	94.70			•	•	80.96	∘	∘	∘	∘	88.49	∘	∘	∘	∘
twonorm	90.27			∘	∘	90.90				∘	82.87	∘	∘	∘	∘
vehicle	74.97					75.09					74.49				
vote1	89.66					89.64					89.45				
voting	96.71				•	95.38					96.64				
vowel-context	92.23	∘	∘	∘	∘	96.25					90.81	∘	∘	∘	∘
vowel-nocontext	91.78	∘	∘	∘	∘	94.84					87.53	∘	∘	∘	∘
waveform	83.25	∘	∘	∘	∘	84.45					82.77	∘	∘	∘	∘
yeast	60.92					59.36		∘	∘	∘	59.78		∘	∘	∘
zip	93.17	∘	∘	∘	∘	95.75	∘	∘	∘	∘	94.29	∘	∘	∘	∘
zoo	93.20					94.08					92.20				

Tabla A.22: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$ (para árboles podados en ambos casos). Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	AdaBoost	$\mathcal{RFW}$				AdaBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$			
Conjunto	(W)	1	2	3	4	(S)	1	2	3	4	(W)	1	2	3	4	(S)	1	2	3	4
abalone	22.59	∘	∘	∘	∘	23.21	∘	∘	∘	∘	23.57		∘	∘	∘	24.07		∘		
anneal	99.63					99.67	•				99.68	•				99.68	•			
audiology	84.61				•	84.56				•	85.14			•	•	85.06			•	•
autos	86.59					86.10					86.10					85.99				
balance-scale	76.36	∘	∘	∘	∘	75.53	∘	∘	∘	∘	80.60					80.33				
breast-w	96.61					96.68					96.60					96.50				
breast-y	66.43	∘	∘	∘	∘	67.16	∘	∘	∘	∘	69.68					69.35				
bupa	70.20					70.77					72.52					72.84				
car	96.72	•	•	•	•	96.60	•	•	•	•	95.96	•	•	•	•	95.80	•	•	•	•
credit-a	85.87					86.39					86.90					86.77				
credit-g	73.75					74.52					75.39					75.29				
crx	86.01					85.88					86.84					86.74				
dna	95.04					94.95					95.63					95.52				
ecoli	83.51					83.99					84.58					85.15				
glass	78.08					79.10					77.24					78.50				
heart-c	80.11					79.78					81.28					81.39				
heart-h	79.40					80.04					79.70					80.68				
heart-s	90.78					91.33					90.93					91.26				
heart-statlog	80.44					79.44				∘	81.56					80.78				
heart-v	71.80					70.40					74.40					74.40				
hepatitis	84.15					84.61					83.57					83.27				
horse-colic	81.76					82.09					84.67					84.10				
hypo	98.98	∘	∘			98.99	∘				99.12					99.14				
ionosphere	93.85					94.22					93.73					93.73				
iris	94.53					94.13					94.20					94.53				
kr-vs-kp	99.60			•	•	99.61			•	•	99.65		•	•	•	99.65		•	•	•
krk	89.47	•	•	•	•	89.39	•	•	•	•	88.49					88.48				
labor	89.10					87.47					86.57					87.57				
led-24	69.80	∘	∘	∘	∘	71.33	∘	∘	∘	∘	72.60	∘	∘	∘	∘	73.14	∘	∘	∘	∘
letter	97.03	•	•	•	•	96.87	•	•	•	•	96.67		•	•	•	96.58		•	•	•
lrd	88.17					88.10					87.65					88.04				
lymphography	84.12					83.72					83.10					83.40				
mushroom	100.00					100.00					100.00					100.00				
nursery	99.75	•	•	•	•	99.74	•	•	•	•	99.70	•	•	•	•	99.68	•	•	•	•
optdigits	98.40					98.42					98.08					98.10				
page	97.07		∘	∘	∘	97.11		∘	∘	∘	97.38					97.39				
pendigits	99.38	•	•	•	•	99.36	•	•	•	•	99.24			•	•	99.25			•	•
phoneme	91.18	•	•	•	•	91.27	•	•	•	•	91.12	•	•	•	•	91.14	•	•	•	•
pima	73.71					73.71					75.54					74.92				
primary	41.65	∘	∘	∘	∘	42.37					42.27			∘	∘	43.60				
promoters	94.08					93.78					94.55	•				94.27				
ringnorm	94.17	•				95.67	•	•	•		92.67					94.10	•			
sat	92.13					92.01					91.83					91.72				
segment	98.60	•	•	•	•	98.57	•			•	98.35					98.34				
shuttle	99.99					99.99					99.99					99.99				
sick	99.07		•	•	•	99.06		•	•	•	99.06		•	•	•	99.04		•	•	•
sonar	84.65					82.41					83.03					81.44				
soybean-small	97.65					98.35					97.65					98.15				
soybean	93.31		∘			93.31		∘			93.32	∘	∘	∘		93.54				
splice	94.91			∘	∘	94.84		∘	∘	∘	95.56					95.48				
threenorm	84.73					84.10					85.00	•				83.97				
tic-tac-toe	98.92	•	•	•	•	98.94	•	•	•	•	98.31	•	•	•	•	98.31	•	•	•	•
twonorm	94.07					94.33					93.60					93.67				
vehicle	77.41					77.68				•	76.63					76.48				
vote1	89.17					89.36					89.84					89.93				
voting	95.31					95.10					95.59					95.56				
vowel-context	95.93					96.26					95.14		∘	∘		95.15		∘	∘	
vowel-nocontext	95.23					95.62					94.41					94.52				
waveform	84.35					84.15					84.57					84.57				
yeast	58.44	∘	∘	∘	∘	59.36		∘	∘	∘	60.66					60.97				
zip	96.74	•				96.59					96.38					96.26		∘		
zoo	96.35					95.06					96.05					95.25				

Tabla A.23: Tasas de acierto para $\mathcal{RFW}$ con árboles no podados y $p = 1 \ldots 4$.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	24.76	25.08	25.39	25.42
anneal	99.11	99.22	99.21	99.17
audiology	84.75	83.06	82.09	81.62
autos	85.59	84.76	84.38	83.50
balance-scale	80.22	80.31	80.54	81.23
breast-w	96.45	96.65	96.65	96.85
breast-y	71.00	72.31	72.48	72.70
bupa	69.28	70.73	70.78	71.22
car	94.64	94.94	95.05	95.03
credit-a	84.99	85.97	86.45	86.72
credit-g	74.01	75.90	76.00	75.91
crx	85.13	86.22	86.41	86.43
dna	95.35	95.72	95.69	95.60
ecoli	85.62	86.40	86.07	86.01
glass	76.31	77.25	78.41	79.06
heart-c	81.22	82.18	82.48	82.38
heart-h	78.95	80.31	80.89	81.19
heart-s	90.04	91.10	91.44	92.08
heart-statlog	81.22	81.56	81.93	81.52
heart-v	73.85	75.15	75.90	76.10
hepatitis	81.79	82.87	83.82	83.95
horse-colic	85.43	85.67	85.10	85.13
hypo	99.24	99.15	99.13	99.11
ionosphere	93.42	93.48	93.51	93.42
iris	95.53	95.27	95.27	95.20
kr-vs-kp	99.54	99.46	99.36	99.26
krk	88.69	89.14	88.57	88.02
labor	83.00	84.07	85.27	85.63
led-24	74.74	74.98	74.83	74.58
letter	96.65	96.27	95.78	95.39
lrd	87.20	87.86	87.97	88.10
lymphography	82.61	84.10	84.30	85.11
mushroom	100.00	100.00	100.00	100.00
nursery	99.23	99.39	99.31	99.09
optdigits	98.29	98.35	98.24	98.15
page	97.30	97.42	97.49	97.48
pendigits	99.18	99.16	99.13	99.05
phoneme	89.44	89.66	89.74	89.69
pima	74.93	75.21	75.47	75.76
primary	45.25	45.57	45.54	45.81
promoters	90.75	92.15	92.94	91.24
ringnorm	89.50	91.57	92.57	93.47
sat	92.02	91.95	91.96	91.82
segment	98.12	98.20	98.23	98.16
shuttle	99.99	99.99	99.99	99.98
sick	98.94	98.97	98.93	98.81
sonar	81.74	84.58	84.39	84.82
soybean-small	100.00	100.00	100.00	100.00
soybean	94.61	94.85	94.83	94.54
splice	95.76	96.22	96.10	95.98
threenorm	80.97	82.67	83.47	83.70
tic-tac-toe	96.39	96.92	96.66	95.69
twonorm	91.00	93.23	93.93	93.97
vehicle	76.00	75.74	76.06	75.42
vote1	89.48	90.16	90.51	90.69
voting	96.62	96.57	96.48	96.32
vowel-context	96.11	97.23	97.12	96.82
vowel-nocontext	94.93	95.53	95.51	95.20
waveform	84.36	84.73	84.84	84.84
yeast	60.26	61.40	61.21	60.96
zip	96.46	96.70	96.70	96.70
zoo	94.29	95.65	96.14	96.24

Tabla A.24: Tasas de acierto para Bagging, Random Forests y Random Subspaces 50 % y 75 % contra $\mathcal{RFW}$ $p = 1 \ldots 4$ (para árboles sin podar en ambos casos). Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

Conjunto	Bagging	$\mathcal{RFW}$ 1 2 3 4	Random Forests	$\mathcal{RFW}$ 1 2 3 4	Subspaces 50 %	$\mathcal{RFW}$ 1 2 3 4	Subspaces 75 %	$\mathcal{RFW}$ 1 2 3 4
abalone	23.85	∘ ∘	23.68	∘ ∘	25.31		23.83	∘ ∘
anneal	98.91		99.69		99.05		98.85	
audiology	81.51		79.39	∘	79.74	∘	81.46	∘
autos	85.71		83.99		86.15		86.40	
balance-scale	81.58		80.48		83.21		80.36	
breast-w	96.28		96.38		96.68		95.82	
breast-y	70.40		70.02		73.05		70.56	
bupa	73.13		71.77		68.22		67.98	
car	94.24		94.44		70.29	∘ ∘ ∘ ∘	94.72	
credit-a	85.43		86.04		86.29		85.23	∘
credit-g	72.91	∘ ∘ ∘	75.62		75.52		72.48	∘ ∘ ∘
crx	85.67		85.41		86.03		85.22	
dna	94.92	∘ ∘	94.55	∘ ∘ ∘ ∘	95.74		95.36	
ecoli	85.06		84.71		85.44		85.00	
glass	74.16		78.72		77.09		73.18	∘ ∘
heart-c	80.06		81.39		82.34		79.67	
heart-h	78.82		80.59		82.29		79.57	
heart-s	90.29		91.75		93.29		90.28	
heart-statlog	81.30		82.00		82.74		80.96	
heart-v	75.50		75.10		75.20		74.90	
hepatitis	81.77		83.33		82.71		80.72	
horse-colic	85.40		85.48		84.13		84.51	
hypo	99.26		99.01		98.88	∘ ∘ ∘	99.23	
ionosphere	92.48		93.68		93.73		92.79	
iris	94.53		94.40		94.80		94.80	
kr-vs-kp	99.48		99.26	∘	97.73	∘ ∘ ∘ ∘	99.38	
krk	84.82	∘ ∘ ∘ ∘	84.89	∘ ∘ ∘ ∘	44.30	∘ ∘ ∘ ∘	83.60	∘ ∘ ∘ ∘
labor	85.73		86.97		83.50		81.07	
led-24	73.89	∘ ∘ ∘	74.43		73.46	∘ ∘ ∘ ∘	74.13	∘
letter	93.90	∘ ∘ ∘ ∘	96.41	• •	96.30	∘ • •	96.03	∘ •
lrd	86.95		88.31		86.12	∘ ∘	85.11	∘ ∘ ∘ ∘
lymphography	80.43		83.30		83.08		79.36	∘
mushroom	100.00		100.00		100.00		100.00	
nursery	98.66	∘ ∘ ∘ ∘	99.05	∘ ∘	92.16	∘ ∘ ∘ ∘	96.18	∘ ∘ ∘ ∘
optdigits	95.81	∘ ∘ ∘ ∘	98.19		98.15		97.06	∘ ∘ ∘ ∘
page	97.36		97.42		97.43		97.16	∘ ∘ ∘
pendigits	98.31	∘ ∘ ∘ ∘	99.17		99.07		98.81	∘ ∘ ∘ ∘
phoneme	89.64		91.25	• • • •	86.13	∘ ∘ ∘ ∘	87.57	∘ ∘ ∘ ∘
pima	76.17		75.58		75.18		74.87	
primary	43.92		43.27		46.26		45.58	
promoters	88.00		88.98		90.27		84.41	∘
ringnorm	88.57	∘ ∘	90.67		90.60		86.73	∘ ∘ ∘
sat	90.89	∘ ∘ ∘ ∘	91.80		91.78		90.78	∘ ∘ ∘ ∘
segment	97.70	∘ ∘	98.11		97.73		97.65	∘ ∘
shuttle	99.98	∘	99.99		99.95	∘ ∘ ∘	99.98	
sick	98.98		98.45	∘ ∘ ∘ ∘	96.88	∘ ∘ ∘ ∘	98.78	
sonar	80.40		84.50		82.41		78.47	∘ ∘
soybean-small	97.65		100.00		98.75		97.80	
soybean	92.09	∘ ∘ ∘ ∘	93.24	∘	94.60		93.21	∘ ∘
splice	94.69	∘ ∘ ∘ ∘	95.27	∘ ∘	96.16		95.24	∘ ∘
threenorm	82.27		83.97		83.00		78.03	∘ ∘ ∘
tic-tac-toe	96.29		96.42		84.82	∘ ∘ ∘ ∘	91.64	∘ ∘ ∘ ∘
twonorm	90.23	∘ ∘	93.43		91.20		82.90	∘ ∘ ∘ ∘
vehicle	75.03		75.07		75.28		74.60	
vote1	89.01		89.54		90.51		89.38	
voting	96.73		96.50		95.77		96.73	
vowel-context	93.67	∘ ∘ ∘ ∘	97.85	•	96.43		91.79	∘ ∘ ∘ ∘
vowel-nocontext	91.90	∘ ∘ ∘ ∘	95.94		94.86		87.65	∘ ∘ ∘ ∘
waveform	83.27	∘ ∘ ∘ ∘	84.77		84.57		82.80	∘ ∘ ∘ ∘
yeast	60.53		60.58		58.00	∘ ∘ ∘	59.04	∘ ∘
zip	93.25	∘ ∘ ∘ ∘	96.28	∘ ∘ ∘	95.86	∘ ∘ ∘ ∘	94.38	∘ ∘ ∘ ∘
zoo	93.60		96.23		94.17		92.60	

Tabla A.25: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$ (para árboles sin podar en ambos casos). Las marcas $\circ$ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas $\bullet$ indican una derrota.

Conjunto	AdaBoost (W)	$\mathcal{RFW}$ 1 2 3 4	AdaBoost (S)	$\mathcal{RFW}$ 1 2 3 4	MultiBoost (W)	$\mathcal{RFW}$ 1 2 3 4	MultiBoost (S)	$\mathcal{RFW}$ 1 2 3 4
abalone	22.54	○ ○ ○ ○	22.96	○ ○ ○ ○	23.23	○ ○ ○ ○	24.26	
anneal	99.59		99.68		99.57		99.57	
audiology	84.66		85.05		83.94		85.19	
autos	85.95		86.45		86.19		86.10	
balance-scale	74.57	○ ○ ○ ○	74.87	○ ○ ○ ○	78.81	○	79.48	
breast-w	96.70		96.60		96.60		96.63	
breast-y	67.53	○	67.47	○ ○ ○	69.00		69.73	
bupa	70.03		69.91		72.35		72.29	
car	96.38	• • • •	95.93	• •	95.78	•	95.59	•
credit-a	85.90		86.17		86.54		86.64	
credit-g	74.59		74.21		75.17		75.61	
crx	85.97		86.59		86.35		86.65	
dna	94.92	○ ○	94.98	○	95.56		95.52	
ecoli	83.59		84.85		84.58		85.39	
glass	78.59		78.82		77.40		78.58	
heart-c	80.30		79.97		80.60		80.80	
heart-h	79.46		79.15		79.62		79.97	
heart-s	90.70		91.44		90.92		91.67	
heart-statlog	79.96		80.33		81.11		79.89	
heart-v	70.85	○	71.00		73.10		73.25	
hepatitis	83.79		83.58		83.45		83.38	
horse-colic	81.62	○ ○	81.47	○ ○	84.21		83.83	
hypo	98.98		98.98		99.10		99.15	
ionosphere	94.10		94.02		93.82		93.88	
iris	94.33		94.47		94.40		94.33	
kr-vs-kp	99.62	• •	99.60	• •	99.60	• •	99.59	• •
krk	88.67		89.05	•	88.14	○ ○	88.15	○ ○
labor	86.80		85.70		87.67		87.90	
led-24	70.05	○ ○ ○ ○	71.22	○ ○ ○ ○	72.11	○ ○ ○ ○	72.96	○ ○ ○ ○
letter	97.03	• • • •	96.87	• • • •	96.66	• • •	96.63	• • •
lrd	87.82		88.23		87.82		88.31	
lymphography	84.11		83.78		83.32		83.04	
mushroom	100.00		100.00		100.00		100.00	
nursery	99.72	• • • •	99.70	• • • •	99.60	• • • •	99.65	• • • •
optdigits	98.42		98.34		98.09		98.09	
page	97.04	○ ○ ○	97.10	○ ○ ○	97.34		97.40	
pendigits	99.36	• • • •	99.39	• • • •	99.23	•	99.28	•
phoneme	91.11	• • • •	91.37	• • • •	91.09	• • • •	91.14	• • • •
pima	73.58		74.07		75.41		75.84	
primary	42.04		41.21	○ ○	42.63		42.60	
promoters	89.59		93.02		91.27		92.58	
ringnorm	94.03	•	95.33	• •	92.57		94.20	•
sat	92.12		92.06		91.92		91.75	
segment	98.50		98.50		98.35		98.33	
shuttle	99.99		99.99		99.99		99.99	
sick	99.00		99.00		99.01		99.02	
sonar	84.69		82.79		83.70		81.69	
soybean-small	97.65		98.15		97.65		98.15	
soybean	93.04	○ ○ ○	92.71	○ ○ ○	93.18	○ ○	93.37	
splice	94.55	○ ○ ○ ○	94.72	○ ○ ○ ○	95.42	○	95.34	○ ○
threenorm	84.50		84.83		84.03		84.13	
tic-tac-toe	98.77	• • • •	98.76	• • • •	98.30	• • • •	98.40	• • • •
twonorm	94.17		94.30		93.73		93.57	
vehicle	77.83		77.14		76.98		76.56	
vote1	88.92		89.17		89.24		89.82	
voting	95.08		95.35		95.56		95.45	
vowel-context	97.14		96.56		96.53		96.06	
vowel-nocontext	95.61		95.91		94.73		94.67	
waveform	84.43		84.32		84.64		84.44	
yeast	58.58	○ ○ ○	59.32		60.62		60.73	
zip	96.73		96.59		96.38	○ ○	96.19	○ ○ ○
zoo	97.05		95.06		96.35		95.25	

Tabla A.26: Tasas de acierto para $\mathcal{RFW}$ con árboles podados y $p = 1 \ldots 4$, para el caso de un error artificial del 10 % en el conjunto de datos.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	22.67	23.1	23.44	23.39
anneal	89.18	89.25	89.25	89.32
audiology	76.45	74.51	73.04	72.74
autos	74.78	73.17	72.48	72.11
balance-scale	73.17	73.63	73.83	74.18
breast-w	87.37	87.73	87.93	87.91
breast-y	64.58	64.76	65.04	65.32
bupa	60.97	61.23	61.52	61.52
car	83.62	83.51	82.95	82.48
credit-a	77.51	77.97	78.23	78.39
credit-g	69.07	69.31	69.05	68.96
crx	79.32	79.86	79.97	80.23
dna	85.62	85.85	85.84	85.76
ecoli	73.76	74.47	74.5	74.86
glass	68.7	70.75	71.12	71.77
heart-c	71.08	72.45	73.15	73.38
heart-h	73.11	73.42	73.62	73.90
heart-s	82.94	83.34	83.67	83.75
heart-statlog	76.52	76.85	76.56	77.22
heart-v	67.8	68.3	68.45	68.50
hepatitis	75.81	76.14	77.12	76.82
horse-colic	78.98	78.85	79.53	79.50
hypo	89.06	89.08	89.08	89.02
ionosphere	83.04	83.01	83.36	83.16
iris	85.07	85.07	85.53	85.67
kr-vs-kp	89.29	89.25	88.97	88.73
krk	78.37	78.49	78.13	77.70
labor	82.6	83.27	82.73	81.30
led-24	67.94	68.14	68.04	68.15
letter	86.35	85.93	85.41	84.96
lrs	79.33	79.51	79.51	79.42
lymph	71.96	73.12	74	73.93
mushroom	89.99	89.99	89.99	89.99
nursery	87.04	86.73	86.19	85.67
optdigits	88.33	88.35	88.21	88.09
page-blocks	87.68	87.79	87.79	87.75
pendigits	89.1	89.09	88.98	88.84
phoneme	79.8	80.07	80.12	80.11
pima-diabetes	69.01	68.81	68.98	68.71
primary-tumor	41.27	41.6	41.47	41.62
promoters	77.75	79.91	81.06	82.38
ringnorm	80.4	82.3	82.8	82.77
sat	82.78	82.61	82.49	82.29
segment	87.69	87.82	87.89	87.82
shuttle	89.97	89.98	89.98	89.98
sick	88.74	88.77	88.8	88.74
sonar	69.67	70.72	70.81	71.83
soybean-small	88.1	88.95	88.95	88.95
soybean	85.61	85.53	84.93	84.64
splice	86.03	86.38	86.62	86.50
threenorm	69	71.67	71.93	71.97
tic-tac-toe	80.26	80.51	80.29	80.18
twonorm	80.83	82.7	83.8	84.17
vehicle	67.58	67.68	67.97	67.99
vote1	82.99	83.22	83.17	83.17
voting	87.29	87.31	87.17	87.04
vowel-context	84.33	85.26	84.88	84.29
vowel-nocontext	83.04	84.07	83.7	83.04
waveform-5000	76.55	76.81	76.92	76.86
yeast	55.77	56.2	56.39	55.78
zip	86.6	86.81	86.87	86.83
zoo	83.55	84.65	84.84	84.94

Tabla A.27: Tasas de acierto para Bagging y las dos versiones de Random Subspaces contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles podados y conjuntos de entrenamiento con error artificial del 10 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	Bagging	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$			
Conjunto		1	2	3	4	50 %	1	2	3	4	75 %	1	2	3	4
abalone	21.79		∘	∘	∘	22.96					21.13	∘	∘	∘	∘
anneal	89.02					89.20					89.18				
audiology	74.81					72.52	∘				74.68				
autos	72.61					74.33					73.46				
balance-scale	75.12	•				74.51					71.73				
breast-w	87.01					87.24					86.18	∘	∘	∘	∘
breast-y	65.94					64.41					64.72				
bupa	64.92					59.99					60.33				
car	83.22					63.95	∘	∘	∘	∘	82.75				
credit-a	77.74					77.41					77.48				
credit-g	69.09					68.87					68.53				
crx	79.67					79.54					79.59				
dna	85.17					85.69					85.43				
ecoli	74.05					74.02					73.82				
glass	69.06					68.76					66.93				
heart-c	71.16					72.18					69.92				
heart-h	72.66					74.71					72.60				
heart-s	82.54					84.15					83.10				
heart-statlog	75.15					76.85					75.41				
heart-v	70.25					68.70					67.95				
hepatitis	76.02					75.96					75.25				
horse-colic	79.20					79.30					79.12				
hypo	89.11					88.89					89.06				
ionosphere	82.33					83.33					81.68				
iris	85.27					85.93					85.00				
kr-vs-kp	89.12					87.78	∘	∘	∘	∘	89.26				•
krk	75.20	∘	∘	∘	∘	38.99	∘	∘	∘	∘	72.57	∘	∘	∘	∘
labor	84.37					81.90					81.07				
led-24	67.15	∘	∘	∘	∘	67.37					67.65				
letter	83.82	∘	∘	∘	∘	86.02	∘		•	•	85.85	∘		•	•
lrs	79.00					79.06					77.25	∘	∘	∘	∘
lymph	69.33					71.10					67.25	∘	∘	∘	∘
mushroom	89.99					89.99					89.99				
nursery	87.42	•	•	•	•	82.42	∘	∘	∘	∘	85.12	∘	∘	∘	∘
optdigits	86.11	∘	∘	∘	∘	88.12					87.09	∘	∘	∘	∘
page-blocks	87.71					87.67					87.50				
pendigits	88.51	∘	∘	∘	∘	88.99					88.78	∘	∘		
phoneme	80.38					76.97	∘	∘	∘	∘	78.21	∘	∘	∘	∘
pima-diabetes	68.54					68.07					68.87				
primary-tumor	39.56					40.35					40.80				
promoters	77.15					76.40					72.06			∘	∘
ringnorm	82.67					81.50					77.67			∘	∘
sat	82.06	∘	∘			82.56					81.54	∘	∘	∘	∘
segment	87.31					87.50					86.91	∘	∘	∘	∘
shuttle	89.96		∘	∘	∘	89.93		∘	∘	∘	89.96		∘		∘
sick	88.83					86.16	∘	∘	∘	∘	88.73				
sonar	70.20					70.64					66.00				
soybean-small	85.10					87.70					85.60				
soybean	83.45	∘	∘			84.23		∘			83.43	∘	∘		
splice	85.29	∘	∘	∘	∘	86.53					85.56		∘	∘	∘
threenorm	71.70					68.90					66.83				
tic-tac-toe	81.59					74.39	∘	∘	∘	∘	77.78	∘	∘	∘	∘
twonorm	81.90					81.77					77.33		∘	∘	∘
vehicle	66.85					68.26					66.91				
vote1	83.21					82.55					82.81				
voting	87.03					86.57					86.92				
vowel-context	82.11	∘	∘	∘	∘	84.62					80.52	∘	∘	∘	∘
vowel-nocontext	81.22	∘	∘	∘		83.36					76.95	∘	∘	∘	∘
waveform-5000	75.94		∘	∘	∘	76.72					75.14	∘	∘	∘	∘
yeast	55.87					53.42	∘	∘	∘	∘	54.20		∘	∘	
zip	84.00	∘	∘	∘	∘	85.94	∘	∘	∘	∘	84.43	∘	∘	∘	∘
zoo	80.79					82.45					80.79				

Tabla A.28: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles podados y conjuntos de entrenamiento con error artificial del 10 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	AdaBoost	$\mathcal{RFW}$				AdaBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$			
Conjunto	(W)	1	2	3	4	(S)	1	2	3	4	(W)	1	2	3	4	(S)	1	2	3	4
abalone	19.33	∘	∘	∘	∘	20.80	∘	∘	∘	∘	20.66	∘	∘	∘	∘	21.64		∘	∘	∘
anneal	82.38	∘	∘	∘	∘	82.42	∘	∘	∘	∘	85.10	∘	∘	∘	∘	85.09	∘	∘	∘	∘
audiology	73.31					74.00					75.28					75.25				
autos	70.75					71.35					71.62					72.08				
balance-scale	65.80	∘	∘	∘	∘	65.83	∘	∘	∘	∘	71.15			∘	∘	71.35				∘
breast-w	84.04	∘	∘	∘	∘	84.21	∘	∘	∘	∘	86.78					86.64			∘	∘
breast-y	57.55	∘	∘	∘	∘	58.27	∘	∘	∘	∘	58.44	∘	∘	∘	∘	58.39	∘	∘	∘	∘
bupa	62.71					64.22					65.06					65.98				
car	81.34	∘	∘	∘		80.87	∘	∘	∘		83.07					82.78				
credit-a	73.03	∘	∘	∘	∘	73.38	∘	∘	∘	∘	75.46			∘	∘	76.23				
credit-g	67.48					67.41					68.86					68.89				
crx	75.70	∘	∘	∘	∘	76.51		∘	∘	∘	78.09					78.61				
dna	81.29	∘	∘	∘	∘	81.23	∘	∘	∘	∘	84.88	∘	∘	∘	∘	84.83	∘	∘	∘	∘
ecoli	71.43					71.99					73.57					73.69				
glass	67.84					68.88					69.19					70.37				
heart-c	69.97					69.25					71.70					70.87				
heart-h	73.24					72.03					74.71					74.97				
heart-s	80.66					81.67					82.31					82.03				
heart-statlog	71.11	∘	∘	∘	∘	71.11	∘	∘	∘	∘	72.52				∘	73.37				
heart-v	64.45					65.35					68.20					68.20				
hepatitis	75.39					75.93					77.79					78.00				
horse-colic	74.39	∘	∘	∘	∘	74.79	∘		∘	∘	78.87					78.82				
hypo	85.29	∘	∘	∘	∘	85.27	∘	∘	∘	∘	88.02	∘	∘	∘	∘	87.89	∘	∘	∘	∘
ionosphere	81.82					81.50					82.67					82.36				
iris	81.47			∘	∘	81.53					84.07					85.00				
kr-vs-kp	82.28	∘	∘	∘	∘	82.33	∘	∘	∘	∘	86.32	∘	∘	∘	∘	86.22	∘	∘	∘	∘
krk	71.43	∘	∘	∘	∘	71.06	∘	∘	∘	∘	74.86	∘	∘	∘	∘	74.44	∘	∘	∘	∘
labor	80.97					79.57					82.57					82.67				
led-24	61.69	∘	∘	∘	∘	63.34	∘	∘	∘	∘	64.80	∘	∘	∘	∘	65.67	∘	∘	∘	∘
letter	80.99	∘	∘	∘	∘	80.94	∘	∘	∘	∘	84.56	∘	∘	∘	∘	84.57	∘	∘	∘	
lrs	79.25					79.36					79.53					79.29				
lymph	70.48					73.40					72.23					73.88				
mushroom	83.28	∘	∘	∘	∘	83.20	∘	∘	∘	∘	87.76	∘	∘	∘	∘	87.77	∘	∘	∘	∘
nursery	81.63	∘	∘	∘	∘	81.27	∘	∘	∘	∘	85.97	∘	∘			85.97	∘	∘		
optdigits	88.21					88.07					88.04					87.96		∘		
page-blocks	85.35	∘	∘	∘	∘	85.33	∘	∘	∘	∘	87.21	∘	∘	∘	∘	87.10	∘	∘	∘	∘
pendigits	88.94					88.84	∘	∘			89.05					88.95				
phoneme	77.72	∘	∘	∘	∘	78.16	∘	∘	∘	∘	80.31					80.46				
pima-diabetes	66.38					65.74			∘		67.24					66.85				
primary-tumor	37.70					34.21	∘	∘	∘	∘	37.40					35.19	∘	∘	∘	∘
promoters	75.08					76.48					79.22					79.04				
ringnorm	83.17					83.43					83.17					84.13	•			
sat	82.60					82.56					82.67					82.50				
segment	85.07	∘	∘	∘	∘	85.47	∘	∘	∘	∘	86.44	∘	∘	∘	∘	86.30	∘	∘	∘	∘
shuttle	85.72	∘	∘	∘	∘	84.86	∘	∘	∘	∘	88.72	∘	∘	∘	∘	88.32	∘	∘	∘	∘
sick	85.50	∘	∘	∘	∘	85.85	∘	∘	∘	∘	88.12	∘	∘	∘	∘	88.12	∘	∘	∘	∘
sonar	72.15					69.66					71.25					69.85				
soybean-small	77.40		∘	∘	∘	76.35	∘	∘	∘	∘	79.70					81.10				
soybean	79.31	∘	∘	∘	∘	78.41	∘	∘	∘	∘	81.93	∘	∘	∘	∘	81.60	∘	∘	∘	∘
splice	82.26	∘	∘	∘	∘	82.40	∘	∘	∘	∘	85.02	∘	∘	∘	∘	84.98	∘	∘	∘	∘
threenorm	71.93					72.90					73.17					73.83				
tic-tac-toe	82.97	•		•	•	82.91				•	84.04	•	•	•	•	84.44	•	•	•	•
twonorm	82.73					83.17					83.13					83.57				
vehicle	68.02					68.15					67.77					68.02				
vote1	78.80	∘	∘	∘	∘	78.86	∘	∘	∘	∘	81.13					80.90				
voting	83.80	∘	∘	∘	∘	83.22	∘	∘	∘	∘	85.54	∘	∘			85.52				
vowel-context	81.21	∘	∘	∘	∘	82.26	∘	∘	∘	∘	82.29	∘	∘	∘	∘	82.80		∘	∘	
vowel-nocontext	81.56		∘	∘		81.70		∘	∘		82.12		∘			82.59		∘		
waveform-5000	76.15					75.95					76.82					76.30				
yeast	52.22	∘	∘	∘	∘	53.02	∘	∘	∘	∘	54.49					55.00				
zip	86.51			∘		86.16	∘	∘	∘	∘	86.27	∘	∘	∘	∘	85.97	∘	∘	∘	∘
zoo	81.96					82.66					82.66					83.16				

Tabla A.29: Tasas de acierto para $\mathcal{RFW}$ con árboles no podados y $p = 1 \ldots 4$, para el caso de un error artificial del 10 % en el conjunto de datos.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	22.46	22.85	23.02	23.03
anneal	88.7	89.18	89.31	89.37
audiology	75.79	74.33	73.8	73.27
autos	73.96	72.83	71.86	71.38
balance-scale	72.61	73.19	73.54	73.86
breast-w	86.81	87.14	87.38	87.57
breast-y	58.95	58.95	59.55	60.04
bupa	60.6	60.69	61.27	60.93
car	83.55	83.94	83.63	83.34
credit-a	74.14	75.06	75.49	76.09
credit-g	67.97	69.14	69.44	69.44
crx	77.36	78.13	78.39	78.52
dna	85.25	85.42	85.15	85.10
ecoli	73.67	74.5	74.67	74.80
glass	68.27	70.28	70.74	71.35
heart-c	70.57	70.96	70.9	71.70
heart-h	71.36	72.53	72.66	73.27
heart-s	80.81	82.87	82.19	81.87
heart-statlog	75.07	75.63	75.63	75.59
heart-v	68.45	68.75	69.15	69.65
hepatitis	75.48	76.47	77.25	77.50
horse-colic	78.92	79.38	79.95	79.95
hypo	88.39	88.58	88.69	88.77
ionosphere	82.96	82.99	83.38	83.18
iris	83.87	84.2	84.67	84.87
kr-vs-kp	88.17	88.57	88.61	88.36
krk	78.45	77.44	76.25	75.53
labor	84.3	85.17	85.03	83.23
led-24	67.39	67.5	67.42	67.17
letter	86.42	85.96	85.4	84.97
lrs	79.38	79.47	79.55	79.55
lymph	73.15	75.87	76.17	75.77
mushroom	88.92	89.62	89.8	89.89
nursery	88.4	88.71	88.45	88.21
optdigits	88.44	88.39	88.31	88.12
page-blocks	87.55	87.75	87.77	87.79
pendigits	89.14	89.17	89.08	88.96
phoneme	79.89	80.26	80.21	80.18
pima-diabetes	69.15	68.68	68.67	68.58
primary-tumor	39.55	39.85	40.03	39.94
promoters	76.92	78.11	79.2	79.37
ringnorm	80.47	82.2	82.7	82.83
sat	82.93	82.8	82.72	82.58
segment	86.9	87.29	87.38	87.29
shuttle	89.77	89.8	89.83	89.85
sick	88.1	88.4	88.53	88.63
sonar	69.48	70.91	70.72	71.54
soybean-small	84.15	88.15	88.95	88.95
soybean	84.73	84.79	84.63	84.42
splice	85.58	85.66	85.61	85.21
threenorm	69.2	71.67	71.93	72.03
tic-tac-toe	83.06	84.34	83.96	83.73
twonorm	81	82.7	84	84.30
vehicle	67.51	67.74	68.01	67.95
vote1	82.14	82.2	82.27	82.30
voting	85.7	86.19	86.23	86.44
vowel-context	84.43	85.12	84.81	84.28
vowel-nocontext	83.14	84.2	83.85	82.77
waveform-5000	76.53	76.8	76.99	76.86
yeast	54.95	55.41	54.95	54.14
zip	86.81	86.95	86.99	87.00
zoo	83.95	84.95	85.24	85.54

Tabla A.30: Tasas de acierto para Bagging, Random Forests y Random Subspaces 50 % y 75 % contra $\mathcal{RFW}$ $p = 1 \ldots 4$ para árboles sin podar y conjuntos de entrenamiento con error artificial del 10 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	Bagging	$\mathcal{RFW}$				Random	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$			
Conjunto		1	2	3	4	Forests	1	2	3	4	50 %	1	2	3	4	75 %	1	2	3	4
abalone	21.70			∘	∘	21.50			∘	∘	22.70					20.92	∘	∘	∘	∘
anneal	88.29		∘	∘	∘	86.38	∘	∘	∘	∘	89.38					89.05				
audiology	73.76					68.88	∘	∘	∘	∘	72.42					72.83				
autos	72.27					69.17	∘				73.07					71.26				
balance-scale	74.02					71.36					75.57	•				73.17				
breast-w	86.71					85.56		∘	∘	∘	87.44					86.11			∘	∘
breast-y	57.93					56.07					61.89					58.88				
bupa	65.16	•				64.95					60.51					59.83				
car	82.15	∘	∘			83.61					64.31	∘	∘	∘	∘	82.68				
credit-a	75.67					75.41					76.57					75.10				
credit-g	68.46					69.61					69.61					66.76			∘	∘
crx	77.96					77.48					79.64					77.64				
dna	84.90					83.54	∘	∘	∘	∘	85.32					84.96				
ecoli	73.58					73.51					73.51					73.63				
glass	68.88					71.10					68.56					66.08				
heart-c	70.33					70.11					72.88					70.08				
heart-h	72.35					74.66					75.49	•				71.81				
heart-s	82.28					82.95					82.93					80.97				
heart-statlog	73.89					74.19					76.56					73.70				
heart-v	69.80					69.40					71.25					70.30				
hepatitis	76.85					77.43					76.93					75.20				
horse-colic	78.84					80.15					79.33					78.03				
hypo	88.67					88.42					88.93	•				88.83	•			
ionosphere	82.33					83.64					83.19					81.65				
iris	85.00					81.60					86.00					84.27				
kr-vs-kp	88.11		∘			86.99	∘	∘	∘	∘	87.55		∘	∘	∘	88.63				
krk	74.99	∘	∘	∘		72.42	∘	∘	∘	∘	40.32	∘	∘	∘	∘	73.70	∘	∘	∘	∘
labor	85.83					83.23					82.90					81.30				
led-24	66.42	∘	∘	∘		66.92					66.14	∘	∘	∘		66.50	∘	∘	∘	
letter	83.91	∘	∘	∘	∘	84.20	∘	∘	∘	∘	86.16	∘		•	•	85.96	∘		•	•
lrs	79.04					79.68					79.04					76.88	∘	∘	∘	∘
lymph	69.81		∘	∘	∘	73.70					72.64					66.31	∘	∘	∘	∘
mushroom	88.95		∘	∘	∘	89.68	•			∘	89.99	•	•	•	•	89.91	•	•	•	
nursery	86.70	∘	∘	∘	∘	87.40	∘	∘	∘	∘	83.21	∘	∘	∘	∘	86.67	∘	∘	∘	∘
optdigits	86.14	∘	∘	∘	∘	88.27					88.22					87.14	∘	∘	∘	∘
page-blocks	87.62					86.96	∘	∘	∘	∘	87.73					87.29		∘	∘	∘
pendigits	88.60	∘	∘	∘	∘	89.07					89.11					88.84	∘	∘	∘	
phoneme	80.44					80.63					77.17	∘	∘	∘	∘	78.32	∘	∘	∘	∘
pima-diabetes	68.31					67.48					67.90					69.04				
primary-tumor	39.09					37.94					40.41					39.97				
promoters	74.56					79.25					72.71					65.22	∘	∘	∘	∘
ringnorm	82.77					83.53					81.60					77.70			∘	∘
sat	82.08	∘	∘	∘		82.65					82.69					81.61	∘	∘	∘	∘
segment	86.94					86.29		∘	∘	∘	87.12					86.35		∘	∘	∘
shuttle	89.87	•	•			86.61	∘	∘	∘	∘	89.95	•	•	•	•	89.88	•	•		
sick	88.59	•				88.54					87.22	∘	∘	∘	∘	88.67	•			
sonar	70.06					73.84					70.30					66.10				
soybean-small	81.80					88.30					83.20					79.10			∘	∘
soybean	82.01	∘	∘	∘	∘	80.86	∘	∘	∘	∘	84.39					82.65	∘	∘	∘	
splice	84.27	∘	∘	∘		84.02	∘	∘	∘	∘	85.70					83.97	∘	∘	∘	∘
threenorm	71.73					73.33					69.03					67.00				
tic-tac-toe	82.32					83.34					78.30	∘	∘	∘	∘	80.67	∘	∘	∘	∘
twonorm	82.00					83.70					81.87					77.57		∘	∘	∘
vehicle	66.62					67.18					68.24					66.23				
vote1	81.93					80.48					82.32					82.09				
voting	85.86					85.59					86.62					86.16				
vowel-context	82.81	∘	∘	∘		82.54	∘	∘	∘		84.69					80.78	∘	∘	∘	∘
vowel-nocontext	81.35	∘	∘	∘		83.28					83.14					76.94	∘	∘	∘	∘
waveform-5000	75.96			∘		76.82					76.69					75.14	∘	∘	∘	∘
yeast	55.60					54.61					51.58	∘	∘	∘	∘	52.86	∘	∘	∘	
zip	84.05	∘	∘	∘	∘	86.53		∘	∘	∘	86.18	∘	∘	∘	∘	84.58	∘	∘	∘	∘
zoo	80.79					81.75					82.95					81.18				

Tabla A.31: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles sin podar y conjuntos de entrenamiento con error artificial del 10 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	AdaBoost	$\mathcal{RFW}$				AdaBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$			
Conjunto	(W)	1	2	3	4	(S)	1	2	3	4	(W)	1	2	3	4	(S)	1	2	3	4
abalone	19.51	∘	∘	∘	∘	20.53	∘	∘	∘	∘	20.57	∘	∘	∘	∘	21.59			∘	
anneal	81.59	∘	∘	∘	∘	81.85	∘	∘	∘	∘	83.43	∘	∘	∘	∘	83.75	∘	∘	∘	∘
audiology	70.92	∘				73.16					73.47					74.54				
autos	70.38					71.99					71.16					71.90				
balance-scale	65.63	∘	∘	∘	∘	66.18	∘	∘	∘	∘	70.58		∘	∘	∘	71.07			∘	∘
breast-w	83.45	∘	∘	∘	∘	83.72	∘	∘	∘	∘	86.21				∘	85.89			∘	∘
breast-y	56.47					56.55					57.04					57.35				
bupa	62.98					64.07					65.41					65.27				
car	80.87	∘	∘	∘	∘	81.16	∘	∘	∘	∘	81.44	∘	∘	∘	∘	81.95	∘	∘	∘	
credit-a	74.09					73.59					75.51					75.67				
credit-g	68.23					67.56					69.11					69.22				
crx	76.26					77.16					77.77					78.04				
dna	81.14	∘	∘	∘	∘	81.42	∘	∘	∘	∘	84.74					84.73				
ecoli	71.04		∘	∘	∘	71.37				∘	72.65					73.60				
glass	66.86					67.95					68.55					70.04				
heart-c	69.45					69.24					70.40					71.03				
heart-h	71.56					72.68					73.51					74.40				
heart-s	81.84					80.83					80.84					82.86				
heart-statlog	71.11			∘	∘	71.00			∘		72.81					72.78				
heart-v	64.65					64.80					67.50					67.50				
hepatitis	76.21					76.48					76.41					78.14				
horse-colic	74.47		∘	∘	∘	74.74		∘	∘	∘	77.51					77.70				
hypo	85.36	∘	∘	∘	∘	85.46	∘	∘	∘	∘	87.78	∘	∘	∘	∘	87.72	∘	∘	∘	∘
ionosphere	81.94					81.99					82.59					82.33				
iris	81.73					81.80					83.87					84.27				
kr-vs-kp	82.52	∘	∘	∘	∘	82.39	∘	∘	∘	∘	85.33	∘	∘	∘	∘	85.66	∘	∘	∘	∘
krk	70.44	∘	∘	∘	∘	70.44	∘	∘	∘	∘	73.14	∘	∘	∘	∘	73.61	∘	∘	∘	∘
labor	80.00					80.70					82.87					83.10				
led-24	60.68	∘	∘	∘	∘	62.97	∘	∘	∘	∘	63.97	∘	∘	∘	∘	65.19	∘	∘	∘	∘
letter	80.04	∘	∘	∘	∘	80.15	∘	∘	∘	∘	83.70	∘	∘	∘	∘	84.10	∘	∘	∘	∘
lrs	79.32					79.44					79.40					79.25				
lymph	70.33					71.45					71.76					72.46				
mushroom	83.51	∘	∘	∘	∘	82.96	∘	∘	∘	∘	86.21	∘	∘	∘	∘	86.52	∘	∘	∘	∘
nursery	81.43	∘	∘	∘	∘	81.27	∘	∘	∘	∘	83.70	∘	∘	∘	∘	83.81	∘	∘	∘	∘
optdigits	88.08					88.05					87.90	∘	∘	∘		87.93	∘	∘		
page-blocks	85.00	∘	∘	∘	∘	85.02	∘	∘	∘	∘	87.01	∘	∘	∘	∘	86.97	∘	∘	∘	∘
pendigits	88.87	∘	∘	∘		88.74	∘	∘	∘		88.95	∘	∘			88.92	∘	∘		
phoneme	77.50	∘	∘	∘	∘	78.46	∘	∘	∘	∘	80.28					80.55				
pima-diabetes	65.47	∘	∘			65.99	∘			∘	67.25					67.08				
primary-tumor	35.69					34.60	∘	∘	∘	∘	36.49					36.99				
promoters	75.88					78.11					74.51					78.85				
ringnorm	82.30					83.37					84.37					84.37				
sat	82.60					82.47					82.67					82.57				
segment	84.87	∘	∘	∘	∘	85.32	∘	∘	∘	∘	86.29		∘	∘	∘	86.23		∘	∘	∘
shuttle	85.56	∘	∘	∘	∘	84.57	∘	∘	∘	∘	87.92	∘	∘	∘	∘	87.62	∘	∘	∘	∘
sick	85.53	∘	∘	∘	∘	85.73	∘	∘	∘	∘	87.91			∘	∘	88.04				∘
sonar	72.27					70.81					70.72					70.57				
soybean-small	76.15		∘	∘	∘	77.20		∘	∘	∘	77.70			∘	∘	81.15				
soybean	78.03	∘	∘	∘	∘	77.34	∘	∘	∘	∘	79.81	∘	∘	∘	∘	80.35	∘	∘	∘	∘
splice	80.56	∘	∘	∘	∘	80.92	∘	∘	∘	∘	84.61	∘	∘	∘		84.66	∘	∘	∘	
threenorm	71.37					72.23					73.60					73.53				
tic-tac-toe	82.28					82.95					83.48					83.73				
twonorm	82.90					82.90					83.07					83.20				
vehicle	68.29					68.05					67.88					68.20				
vote1	79.03	∘	∘	∘	∘	79.17					79.90					80.27				
voting	83.41					83.60		∘	∘	∘	85.34					85.48				
vowel-context	81.67	∘	∘	∘	∘	82.12	∘	∘	∘	∘	82.45	∘	∘	∘	∘	83.02		∘	∘	
vowel-nocontext	81.16	∘	∘	∘		82.17		∘			81.97		∘	∘		82.26		∘		
waveform-5000	76.30					76.09					76.78					76.32				
yeast	51.93	∘	∘	∘		52.53	∘	∘	∘		54.20					54.43				
zip	86.38	∘	∘	∘	∘	86.18	∘	∘	∘	∘	86.24	∘	∘	∘	∘	85.80	∘	∘	∘	∘
zoo	81.85					81.68					82.06					82.97				

Tabla A.32: Tasas de acierto para $\mathcal{RFW}$ con árboles podados y $p = 1 \ldots 4$, para el caso de un error artificial del 20 % en el conjunto de datos.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	20.38	20.80	20.65	20.80
anneal	78.95	79.01	79.00	79.07
audiology	64.42	63.37	62.79	61.18
autos	61.17	60.77	60.96	60.28
balance-scale	66.38	66.82	66.76	66.81
breast-w	77.67	77.74	77.92	78.05
breast-y	61.15	61.59	62.20	62.17
bupa	53.21	53.58	54.07	54.03
car	74.04	73.99	73.77	73.32
credit-a	70.84	70.83	70.65	70.59
credit-g	62.77	63.56	63.51	63.33
crx	72.04	72.33	71.96	72.03
dna	75.88	76.11	75.93	75.53
ecoli	66.92	67.07	66.95	67.04
glass	61.66	62.74	62.94	62.80
heart-c	63.38	64.25	64.38	64.93
heart-h	68.54	68.57	68.47	68.51
heart-s	73.35	74.65	75.37	76.18
heart-statlog	61.63	63.44	64.78	64.89
heart-v	60.20	61.50	61.65	62.10
hepatitis	70.23	70.61	71.13	72.16
horse-colic	72.61	72.28	72.33	72.25
hypo	78.76	78.82	78.92	78.92
ionosphere	70.28	71.07	71.41	72.07
iris	74.33	74.87	75.33	75.73
kr-vs-kp	79.09	78.89	78.72	78.53
krk	68.21	68.18	67.72	67.32
labor	65.57	65.87	65.40	65.07
led-24	60.78	60.89	60.76	60.61
letter	76.16	75.64	75.11	74.79
lrs	71.47	71.77	71.56	71.70
lymph	61.17	62.79	63.81	63.42
mushroom	79.98	79.99	79.99	79.99
nursery	77.03	76.52	76.05	75.72
optdigits	78.39	78.36	78.20	78.05
page-blocks	77.82	77.92	77.93	77.92
pendigits	79.07	79.04	78.93	78.79
phoneme	69.91	70.29	70.32	70.26
pima-diabetes	61.25	61.62	61.55	61.33
primary-tumor	34.16	34.63	34.90	34.97
promoters	69.13	69.72	70.15	70.12
ringnorm	68.97	70.10	70.30	70.40
sat	73.23	73.16	73.03	72.89
segment	77.57	77.61	77.72	77.60
shuttle	79.96	79.97	79.97	79.97
sick	78.91	78.91	78.94	78.95
sonar	64.90	66.10	66.93	66.97
soybean-small	77.90	79.90	80.50	80.50
soybean	75.49	75.44	75.01	74.65
splice	76.53	76.84	76.81	76.56
threenorm	61.60	62.47	63.80	63.67
tic-tac-toe	70.50	71.34	71.51	71.33
twonorm	71.37	72.20	72.87	73.23
vehicle	59.98	60.43	60.50	60.63
vote1	74.19	74.19	74.12	74.29
voting	77.61	77.56	77.45	77.22
vowel-context	71.13	71.32	71.21	70.93
vowel-nocontext	70.75	70.68	70.25	69.47
waveform-5000	68.41	68.61	68.55	68.44
yeast	48.69	49.08	48.96	48.52
zip	76.82	77.01	77.06	77.05
zoo	74.57	74.67	74.76	74.47

Tabla A.33: Tasas de acierto para Bagging y las dos versiones de Random Subspaces contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles podados y conjuntos de entrenamiento con error artificial del 20 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	Bagging	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$			
Conjunto		1	2	3	4	50 %	1	2	3	4	75 %	1	2	3	4
abalone	19.05	∘	∘	∘		20.51					18.72	∘	∘	∘	∘
anneal	78.53					78.99					78.77				
audiology	63.07					60.81	∘				62.06				
autos	62.88					61.42					62.39				
balance-scale	67.03					64.74					64.07				
breast-w	77.35					77.64					76.91				
breast-y	58.50					61.81					60.37				
bupa	60.39	•	•	•	•	55.62					53.16				
car	73.41					57.99	∘	∘	∘	∘	73.24				
credit-a	70.38					70.49					70.72				
credit-g	62.36					63.31					62.91				
crx	71.58					71.94					71.61				
dna	75.17	∘	∘			75.70					75.49				
ecoli	66.56					66.17					66.51				
glass	61.02					62.21					59.25				
heart-c	63.85					64.81					62.97				
heart-h	68.67					69.80					68.67				
heart-s	74.97					76.64					74.38				
heart-statlog	61.67					64.85					62.11				
heart-v	62.00					63.45					62.15				
hepatitis	71.10					72.11					70.50				
horse-colic	72.77					71.90					72.66				
hypo	78.64					78.91					78.93				
ionosphere	70.88					71.42					68.94				
iris	75.20					76.67					74.33				
kr-vs-kp	78.94					77.93	∘	∘	∘		78.81				
krk	64.80	∘	∘	∘	∘	35.14	∘	∘	∘	∘	62.90	∘	∘	∘	∘
labor	71.20					65.63					63.47				
led-24	59.30	∘	∘	∘	∘	59.86	∘	∘	∘	∘	60.48				
letter	74.28	∘	∘	∘	∘	75.95		•	•	•	75.74	∘		•	•
lrs	70.71					70.54					68.32	∘	∘	∘	∘
lymph	62.90					61.81					60.66				
mushroom	79.93					80.00					79.99				
nursery	77.39	•	•	•	•	73.62	∘	∘	∘	∘	75.58	∘	∘	∘	
optdigits	76.33	∘	∘	∘	∘	78.20					77.07	∘	∘	∘	∘
page-blocks	77.53		∘	∘	∘	77.73					77.49	∘	∘	∘	∘
pendigits	78.50	∘	∘	∘	∘	78.99				•	78.65	∘	∘	∘	
phoneme	70.70	•				67.84	∘	∘	∘	∘	68.69	∘	∘	∘	∘
pima-diabetes	62.05					62.46					61.83				
primary-tumor	32.39					36.08					34.49				
promoters	70.35					70.15					67.16				
ringnorm	71.30					70.03					67.20				
sat	72.21	∘	∘	∘	∘	73.10					71.91	∘	∘	∘	∘
segment	76.62	∘	∘	∘	∘	77.26					76.68	∘	∘	∘	∘
shuttle	79.95					79.92	∘	∘	∘	∘	79.95				
sick	78.35	∘	∘	∘	∘	77.03	∘	∘	∘	∘	78.88				
sonar	66.75					66.84					64.60				
soybean-small	73.95					78.80					74.50				
soybean	72.74	∘	∘	∘		73.80	∘	∘			73.00	∘	∘	∘	
splice	75.63	∘	∘	∘		76.99					76.10				
threenorm	64.47					62.10					61.63				
tic-tac-toe	71.23					67.68	∘	∘	∘	∘	68.23	∘	∘	∘	∘
twonorm	71.97					71.63					69.57				
vehicle	59.24					59.94					58.11				
vote1	73.87					73.39					73.50				
voting	77.40					76.67					77.24				
vowel-context	70.45					70.48					66.92	∘	∘	∘	∘
vowel-nocontext	69.19					70.46					64.99	∘	∘	∘	∘
waveform-5000	67.79					68.42					67.04	∘	∘	∘	∘
yeast	48.23					46.59		∘	∘		47.00		∘		
zip	74.41	∘	∘	∘	∘	76.28	∘	∘	∘	∘	74.60	∘	∘	∘	∘
zoo	74.05					73.28					72.07				

Tabla A.34: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles podados y conjuntos de entrenamiento con error artificial del 20 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	AdaBoost	$\mathcal{RFW}$				AdaBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$			
Conjunto	(W)	1	2	3	4	(S)	1	2	3	4	(W)	1	2	3	4	(S)	1	2	3	4
abalone	16.71	∘	∘	∘	∘	17.88	∘	∘	∘	∘	18.43	∘	∘	∘	∘	19.00	∘	∘	∘	∘
anneal	66.79	∘	∘	∘	∘	67.18	∘	∘	∘	∘	73.83	∘	∘	∘	∘	72.45	∘	∘	∘	∘
audiology	59.44	∘				60.75					61.95					63.19				
autos	54.71	∘				55.40					57.37					57.40				
balance-scale	57.03	∘	∘	∘	∘	57.15	∘	∘	∘	∘	63.35	∘	∘	∘	∘	63.06	∘	∘	∘	∘
breast-w	74.43	∘	∘	∘	∘	72.24	∘	∘	∘	∘	75.49	∘	∘	∘	∘	75.12	∘	∘	∘	∘
breast-y	52.60	∘	∘	∘	∘	53.06	∘	∘	∘	∘	53.40	∘	∘	∘	∘	53.24	∘	∘	∘	∘
bupa	54.02					57.45					55.49					60.07	•	•		
car	67.19	∘	∘	∘	∘	67.92	∘	∘	∘	∘	72.01	∘	∘	∘		71.93	∘	∘	∘	
credit-a	64.13	∘	∘	∘	∘	65.49	∘	∘	∘	∘	66.80	∘	∘	∘	∘	67.72				
credit-g	58.97	∘	∘	∘	∘	59.20	∘	∘	∘	∘	61.06					60.96				
crx	64.22	∘	∘	∘	∘	65.29	∘	∘	∘	∘	66.97	∘	∘	∘	∘	67.91	∘	∘	∘	∘
dna	69.36	∘	∘	∘	∘	69.35	∘	∘	∘	∘	75.01	∘	∘			74.99	∘	∘		
ecoli	62.45	∘	∘	∘	∘	63.11	∘	∘	∘	∘	65.31					65.49				
glass	58.42					59.66					60.94					61.29				
heart-c	62.80					61.56					62.54					63.82				
heart-h	63.71	∘	∘	∘	∘	61.77	∘	∘	∘	∘	66.35					65.91				
heart-s	70.85					71.06					74.49					74.87				
heart-statlog	58.96				∘	61.44					62.33					63.30				
heart-v	59.60					57.05					60.20					59.10				
hepatitis	63.42		∘	∘	∘	63.21			∘	∘	66.48					68.51				
horse-colic	65.44	∘	∘	∘	∘	65.28	∘	∘	∘	∘	70.74					71.38				
hypo	75.59	∘	∘	∘	∘	71.63	∘	∘	∘	∘	76.81	∘	∘	∘	∘	76.39	∘	∘	∘	∘
ionosphere	66.57			∘	∘	66.92				∘	69.88					69.42				
iris	73.13					72.13					73.80					74.00				
kr-vs-kp	68.43	∘	∘	∘	∘	68.47	∘	∘	∘	∘	72.73	∘	∘	∘	∘	72.76	∘	∘	∘	∘
krk	57.83	∘	∘	∘	∘	57.67	∘	∘	∘	∘	63.28	∘	∘	∘	∘	62.51	∘	∘	∘	∘
labor	62.03					62.93					66.13					68.73				
led-24	52.26	∘	∘	∘	∘	54.66	∘	∘	∘	∘	56.59	∘	∘	∘	∘	57.48	∘	∘	∘	∘
letter	69.10	∘	∘	∘	∘	68.91	∘	∘	∘	∘	73.77	∘	∘	∘	∘	73.64	∘	∘	∘	∘
lrs	70.24					70.40					71.26					70.92				
lymph	55.72		∘			56.04			∘		59.52					60.30				
mushroom	69.19	∘	∘	∘	∘	69.00	∘	∘	∘	∘	74.91	∘	∘	∘	∘	74.94	∘	∘	∘	∘
nursery	67.29	∘	∘	∘	∘	66.62	∘	∘	∘	∘	74.63	∘	∘	∘	∘	73.93	∘	∘	∘	∘
optdigits	77.71	∘	∘	∘		77.54	∘	∘	∘	∘	77.85	∘	∘			77.74	∘	∘	∘	
page-blocks	73.06	∘	∘	∘	∘	72.99	∘	∘	∘	∘	76.47	∘	∘	∘	∘	76.40	∘	∘	∘	∘
pendigits	78.25	∘	∘	∘	∘	78.20	∘	∘	∘	∘	78.77	∘	∘			78.69	∘	∘	∘	
phoneme	68.23	∘	∘	∘	∘	67.84	∘	∘	∘	∘	68.57	∘	∘	∘	∘	70.45				
pima-diabetes	60.08					57.53		∘		∘	60.14					59.65				
primary-tumor	31.01		∘	∘	∘	26.78	∘	∘	∘	∘	31.01		∘	∘	∘	26.84	∘	∘	∘	∘
promoters	59.03	∘	∘	∘	∘	57.60	∘	∘	∘	∘	62.03					63.51				
ringnorm	67.40					67.87					69.53					70.70				
sat	72.65	∘				72.47	∘	∘	∘		72.95					72.84				
segment	72.77	∘	∘	∘	∘	73.00	∘	∘	∘	∘	74.54	∘	∘	∘	∘	74.54	∘	∘	∘	∘
shuttle	71.99	∘	∘	∘	∘	71.36	∘	∘	∘	∘	77.97	∘	∘	∘	∘	77.12	∘	∘	∘	∘
sick	71.73	∘	∘	∘	∘	71.47	∘	∘	∘	∘	75.57	∘	∘	∘	∘	76.15	∘	∘	∘	∘
sonar	64.02					65.05					66.89					65.16				
soybean-small	60.60	∘	∘	∘	∘	62.70	∘	∘	∘	∘	69.00		∘	∘	∘	69.05		∘	∘	∘
soybean	67.50	∘	∘	∘	∘	66.44	∘	∘	∘	∘	71.29	∘	∘	∘	∘	71.01	∘	∘	∘	∘
splice	70.24	∘	∘	∘	∘	70.51	∘	∘	∘	∘	75.06	∘	∘	∘	∘	74.87	∘	∘	∘	∘
threenorm	61.60					62.53					62.77					63.67				
tic-tac-toe	68.75					68.09			∘	∘	71.27					71.01				
twonorm	70.57					71.27					73.20					71.97				
vehicle	59.74					59.50					59.90					60.63				
vote1	67.47	∘	∘	∘	∘	67.45	∘	∘	∘	∘	70.83					70.76				
voting	72.16	∘	∘	∘	∘	71.79	∘	∘	∘	∘	74.99	∘	∘			75.36				
vowel-context	62.07	∘	∘	∘	∘	64.04	∘	∘	∘	∘	66.04	∘	∘	∘	∘	68.34	∘	∘	∘	∘
vowel-nocontext	66.34	∘	∘	∘	∘	67.12	∘	∘	∘		68.29	∘	∘	∘		68.72				
waveform-5000	67.91					67.33		∘	∘	∘	68.47					68.04				
yeast	43.96	∘	∘	∘	∘	45.11	∘	∘	∘	∘	47.36					47.65				
zip	76.27	∘	∘	∘	∘	75.80	∘	∘	∘	∘	76.23	∘	∘	∘	∘	75.81	∘	∘	∘	∘
zoo	71.93					69.31					71.11					70.62				

Tabla A.35: Tasas de acierto para $\mathcal{RFW}$ con árboles no podados y $p = 1 \ldots 4$, para el caso de un error artificial del 20 % en el conjunto de datos.

Conjunto	$\mathcal{RFW}1$	$\mathcal{RFW}2$	$\mathcal{RFW}3$	$\mathcal{RFW}4$
abalone	20.16	20.64	20.44	20.59
anneal	77.01	77.77	78.22	78.31
audiology	62.39	61.27	60.91	60.43
autos	61.27	60.53	60.57	59.83
balance-scale	64.01	64.79	64.47	64.84
breast-w	76.27	76.75	76.99	77.29
breast-y	55.20	54.94	54.18	54.36
bupa	53.66	53.72	53.77	54.03
car	73.07	73.34	72.93	72.53
credit-a	67.06	68.10	68.16	68.12
credit-g	60.47	61.91	62.13	62.62
crx	68.17	68.65	68.70	68.86
dna	75.56	75.63	75.33	74.86
ecoli	65.49	65.94	66.41	66.03
glass	60.95	62.32	62.47	62.09
heart-c	62.65	63.38	63.68	63.61
heart-h	64.59	65.67	65.78	66.73
heart-s	73.15	74.08	75.15	75.17
heart-statlog	58.33	61.30	61.63	62.44
heart-v	60.35	60.80	61.65	61.90
hepatitis	68.03	68.61	69.40	70.44
horse-colic	71.60	71.60	71.65	71.78
hypo	77.54	77.82	78.14	78.35
ionosphere	69.82	70.79	71.13	71.59
iris	73.47	73.67	74.07	74.27
kr-vs-kp	75.17	75.84	75.91	75.83
krk	67.33	65.48	64.18	63.60
labor	63.33	64.20	64.87	65.47
led-24	60.08	60.20	59.72	59.20
letter	75.92	75.42	74.82	74.44
lrs	71.30	71.58	71.49	71.70
lymph	61.48	63.59	63.73	65.17
mushroom	76.58	77.89	78.50	78.81
nursery	77.06	77.74	77.62	77.50
optdigits	78.36	78.38	78.20	78.08
page-blocks	77.21	77.52	77.64	77.68
pendigits	79.05	79.12	79.01	78.88
phoneme	70.04	70.38	70.44	70.38
pima-diabetes	61.13	61.40	61.33	61.26
primary-tumor	33.54	33.98	34.10	34.07
promoters	66.62	67.59	66.72	66.38
ringnorm	68.93	70.13	70.17	70.37
sat	73.22	73.20	73.08	73.07
segment	75.12	75.55	75.55	75.46
shuttle	79.00	79.20	79.28	79.40
sick	76.71	77.17	77.48	77.91
sonar	64.66	66.05	66.97	66.93
soybean-small	76.00	79.25	80.10	80.30
soybean	73.24	73.75	73.95	73.73
splice	75.32	75.46	75.12	74.47
threenorm	61.57	62.27	63.73	63.70
tic-tac-toe	71.47	71.93	72.28	71.98
twonorm	71.33	72.17	72.60	73.13
vehicle	59.80	60.31	60.05	60.08
vote1	71.65	72.63	73.52	73.64
voting	76.02	76.64	76.87	76.71
vowel-context	70.64	70.82	70.51	70.43
vowel-nocontext	70.08	70.15	69.82	68.99
waveform-5000	68.39	68.54	68.51	68.45
yeast	47.25	47.70	47.33	46.85
zip	76.86	77.08	77.17	77.16
zoo	74.77	75.05	74.85	74.85

Tabla A.36: Tasas de acierto para Bagging, Random Forests y Random Subspaces 50 % y 75 % contra $\mathcal{RFW}$ $p = 1 \ldots 4$ para árboles sin podar y conjuntos de entrenamiento con error artificial del 20 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	Bagging	$\mathcal{RFW}$				Random	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$				Subspaces	$\mathcal{RFW}$			
Conjunto		1	2	3	4	Forests	1	2	3	4	50 %	1	2	3	4	75 %	1	2	3	4
abalone	18.83	∘	∘	∘	∘	18.64	∘	∘	∘	∘	20.13					18.46	∘	∘	∘	∘
anneal	76.45		∘	∘	∘	71.82	∘	∘	∘	∘	79.06	•	•	•		77.55				
audiology	60.63					57.26	∘	∘	∘		60.05					59.79				
autos	61.60					56.05					61.32					61.02				
balance-scale	65.54					61.44		∘	∘	∘	66.55					65.22				
breast-w	75.99					74.03	∘	∘	∘	∘	77.69					76.82				
breast-y	54.71					53.72					56.98					55.26				
bupa	60.39	•	•	•	•	59.33					55.39					53.05				
car	69.90	∘	∘	∘	∘	72.84					58.46	∘	∘	∘	∘	71.99				
credit-a	67.99					67.91					69.86					67.25				
credit-g	61.16					61.66					61.89					59.48				∘
crx	68.65					67.54					70.10					68.91				
dna	75.00					73.37	∘	∘	∘	∘	75.28					75.17				
ecoli	66.17					64.57					64.66					64.33				
glass	60.93					62.59					61.27					58.79				
heart-c	62.98					64.35					65.07					62.82				
heart-h	68.26	•				66.94					68.06					66.19				
heart-s	74.31					76.19					76.69					73.65				
heart-statlog	60.63					64.85	•				64.78	•				60.04				
heart-v	60.90					60.25					64.55					61.75				
hepatitis	69.21					71.23					70.16					69.08				
horse-colic	71.49					72.99					71.11					70.22				
hypo	77.95					77.33			∘	∘	78.87	•	•	•	•	78.38	•			
ionosphere	70.74					70.70					71.28					68.54				
iris	74.33					73.00					76.87					73.80				
kr-vs-kp	75.34					73.84	∘	∘	∘	∘	76.81	•			•	76.48	•			
krk	63.95	∘	∘			59.67	∘	∘	∘	∘	36.20	∘	∘	∘	∘	63.16	∘	∘		
labor	68.90					67.83					64.70					63.40				
led-24	58.23	∘	∘	∘	∘	59.00	∘	∘			58.07	∘	∘	∘	∘	58.79	∘	∘	∘	
letter	73.91	∘	∘	∘	∘	72.01	∘	∘	∘	∘	75.94		•	•	•	75.47	∘		•	•
lrs	70.71					71.36					70.56					68.02	∘	∘	∘	∘
lymph	60.53					60.41					62.77					58.57				
mushroom	76.24		∘	∘	∘	78.09	•			∘	79.95	•	•	•	•	79.50	•	•	•	•
nursery	72.59	∘	∘	∘	∘	74.93	∘	∘	∘	∘	74.32	∘	∘	∘	∘	76.91		∘	∘	∘
optdigits	76.26	∘	∘	∘	∘	78.36					78.27					76.87	∘	∘	∘	∘
page-blocks	77.17			∘	∘	75.97	∘	∘	∘	∘	77.68	•				76.81	∘	∘	∘	∘
pendigits	78.45	∘	∘	∘	∘	78.91		∘			79.03					78.48	∘	∘	∘	∘
phoneme	70.78					69.83					68.01	∘	∘	∘	∘	68.71	∘	∘	∘	∘
pima-diabetes	61.88					60.92					62.28					61.44				
primary-tumor	31.86					31.56					35.52					33.60				
promoters	69.37					62.83					67.15					61.35				
ringnorm	71.37					72.23					69.87					67.30				
sat	72.17	∘	∘	∘	∘	73.12					73.06					71.67	∘	∘	∘	∘
segment	75.36					74.67		∘	∘	∘	75.65					74.43		∘	∘	
shuttle	79.16	•			∘	72.83	∘	∘	∘	∘	79.90	•	•	•	•	79.47	•	•	•	
sick	77.38					77.96	•	•			77.60					78.23	•	•	•	
sonar	66.51					69.32					66.79					64.55				
soybean-small	71.30					75.30					75.95					71.25				
soybean	69.74	∘	∘	∘	∘	69.33	∘	∘	∘	∘	73.53					70.82	∘	∘	∘	∘
splice	74.07	∘	∘			72.75	∘	∘	∘	∘	75.40				•	72.64	∘	∘	∘	∘
threenorm	64.43					64.53					61.67					61.70				
tic-tac-toe	70.10					71.09					70.46					69.36	∘	∘	∘	∘
twonorm	72.00					74.33					71.80					69.77				
vehicle	59.27					59.66					59.78					57.60		∘		
vote1	71.43					70.39				∘	74.07					72.46				
voting	75.86					74.50			∘		77.13					75.93				
vowel-context	70.18					66.41	∘	∘	∘	∘	70.13					66.25	∘	∘	∘	∘
vowel-nocontext	69.04					68.41					70.11					64.31	∘	∘	∘	∘
waveform-5000	67.80					68.76					68.38					67.02	∘	∘	∘	∘
yeast	47.65					47.20					45.09		∘	∘		45.15	∘	∘		
zip	74.38	∘	∘	∘	∘	76.70			∘	∘	76.32	∘	∘	∘	∘	74.39	∘	∘	∘	∘
zoo	74.15					65.16	∘	∘	∘	∘	73.38					72.07				

Tabla A.37: Tasas de acierto para las dos versiones de AdaBoost y MultiBoost contra $\mathcal{RFW}$ $p = 1 \ldots 4$, para árboles sin podar y conjuntos de entrenamiento con error artificial del 20 % en ambos casos. Las marcas ∘ indican una victoria significativa de la versión de $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

	AdaBoost	$\mathcal{RFW}$				AdaBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$				MultiBoost	$\mathcal{RFW}$			
Conjunto	(W)	1	2	3	4	(S)	1	2	3	4	(W)	1	2	3	4	(S)	1	2	3	4
abalone	16.24	∘	∘	∘	∘	17.61	∘	∘	∘	∘	17.85	∘	∘	∘	∘	18.84		∘	∘	∘
anneal	65.55	∘	∘	∘	∘	66.00	∘	∘	∘	∘	69.69	∘	∘	∘	∘	69.88	∘	∘	∘	∘
audiology	57.65	∘				59.48					58.98					60.89				
autos	54.03	∘	∘	∘		55.64	∘				57.35					56.86				
balance-scale	56.35	∘	∘	∘	∘	57.19	∘	∘	∘	∘	61.62		∘	∘	∘	61.96		∘		∘
breast-w	72.40	∘	∘	∘	∘	71.90	∘	∘	∘	∘	74.76			∘	∘	74.55		∘	∘	∘
breast-y	51.24					53.72					52.85					53.92				
bupa	54.82					58.43					56.55					59.85				
car	66.75	∘	∘	∘	∘	67.83	∘	∘	∘	∘	68.75	∘	∘	∘	∘	69.95	∘	∘	∘	∘
credit-a	65.16					66.06					67.04					67.35				
credit-g	59.61				∘	59.66				∘	60.10				∘	60.50				
crx	65.77			∘	∘	65.35		∘	∘	∘	67.10					67.72				
dna	68.51	∘	∘	∘	∘	69.05	∘	∘	∘	∘	74.98					75.02				
ecoli	62.31		∘	∘	∘	62.84		∘	∘	∘	64.57					65.43				
glass	59.10					59.89					60.36					61.71				
heart-c	61.54					61.64					62.70					62.94				
heart-h	59.01		∘	∘	∘	60.70				∘	64.48					64.79				
heart-s	72.02					70.98					74.90					74.19				
heart-statlog	59.44					60.81					61.19					63.30				
heart-v	57.50					56.40					58.30					58.00				
hepatitis	65.60					62.85				∘	67.22					67.69				
horse-colic	65.08	∘	∘	∘	∘	65.24	∘	∘	∘	∘	69.12					70.05				
hypo	74.26	∘	∘	∘	∘	71.61	∘	∘	∘	∘	75.99	∘	∘	∘	∘	75.83	∘	∘	∘	∘
ionosphere	66.97					67.45					69.59					69.97				
iris	72.87					71.87					73.47					73.60				
kr-vs-kp	68.53	∘	∘	∘	∘	68.68	∘	∘	∘	∘	71.73	∘	∘	∘	∘	72.36	∘	∘	∘	∘
krk	55.11	∘	∘	∘	∘	56.62	∘	∘	∘	∘	60.82	∘	∘	∘	∘	61.47	∘	∘	∘	∘
labor	64.97					65.13					67.20					68.30				
led-24	51.06	∘	∘	∘	∘	54.54	∘	∘	∘	∘	54.76	∘	∘	∘	∘	57.04	∘	∘	∘	∘
letter	66.64	∘	∘	∘	∘	67.77	∘	∘	∘	∘	71.89	∘	∘	∘	∘	72.72	∘	∘	∘	∘
lrs	70.56					70.34					70.68					70.79				
lymph	55.80		∘	∘	∘	57.12				∘	58.44					59.90				
mushroom	69.41	∘	∘	∘	∘	68.97	∘	∘	∘	∘	73.04	∘	∘	∘	∘	73.34	∘	∘	∘	∘
nursery	66.10	∘	∘	∘	∘	65.86	∘	∘	∘	∘	69.90	∘	∘	∘	∘	69.85	∘	∘	∘	∘
optdigits	77.59	∘	∘	∘	∘	77.59	∘	∘	∘	∘	77.75	∘	∘	∘		77.70	∘	∘	∘	
page-blocks	72.51	∘	∘	∘	∘	72.55	∘	∘	∘	∘	76.05	∘	∘	∘	∘	76.26	∘	∘	∘	∘
pendigits	78.10	∘	∘	∘	∘	78.13	∘	∘	∘	∘	78.59	∘	∘	∘	∘	78.65	∘	∘	∘	
phoneme	68.20	∘	∘	∘	∘	67.94	∘	∘	∘	∘	68.61	∘	∘	∘	∘	70.31				
pima-diabetes	60.37					57.66					59.73					59.47				
primary-tumor	28.53	∘	∘	∘	∘	25.34	∘	∘	∘	∘	28.35	∘	∘	∘	∘	25.66	∘	∘	∘	∘
promoters	60.53					58.97					60.67					62.65				
ringnorm	67.33					69.13					68.90					70.43				
sat	72.72					72.44	∘	∘	∘	∘	72.93					72.77				
segment	71.83	∘	∘	∘	∘	72.57	∘	∘	∘	∘	74.23	∘	∘	∘	∘	74.36		∘	∘	∘
shuttle	71.35	∘	∘	∘	∘	70.90	∘	∘	∘	∘	75.81	∘	∘	∘	∘	75.77	∘	∘	∘	∘
sick	71.55	∘	∘	∘	∘	71.95	∘	∘	∘	∘	75.40	∘	∘	∘	∘	76.02		∘	∘	∘
sonar	67.20					64.95					65.35					66.21				
soybean-small	58.65	∘	∘	∘	∘	60.05	∘	∘	∘	∘	64.95	∘	∘	∘	∘	67.60			∘	∘
soybean	63.90	∘	∘	∘	∘	64.66	∘	∘	∘	∘	67.70	∘	∘	∘	∘	68.60	∘	∘	∘	∘
splice	67.51	∘	∘	∘	∘	68.37	∘	∘	∘	∘	74.30	∘	∘			74.22	∘	∘		
threenorm	60.83					62.33					62.47					64.13				
tic-tac-toe	69.01		∘	∘	∘	68.56	∘	∘	∘	∘	69.99			∘		70.15				
twonorm	71.17					70.90					72.43					71.93				
vehicle	59.45					59.71					59.82					60.52				
vote1	67.33	∘	∘	∘	∘	67.64	∘	∘	∘	∘	69.25		∘	∘	∘	69.41			∘	∘
voting	71.42	∘	∘	∘	∘	71.59	∘	∘	∘	∘	74.18		∘	∘	∘	74.76				
vowel-context	62.08	∘	∘	∘	∘	64.00	∘	∘	∘	∘	65.69	∘	∘	∘	∘	67.75	∘	∘	∘	∘
vowel-nocontext	66.24	∘	∘	∘	∘	67.09	∘	∘	∘		68.01		∘			68.96				
waveform-5000	67.78					67.36		∘	∘	∘	68.54					68.00				
yeast	43.67	∘	∘	∘	∘	45.15		∘			46.85					47.74				
zip	76.30	∘	∘	∘	∘	75.86	∘	∘	∘	∘	76.14	∘	∘	∘	∘	75.75	∘	∘	∘	∘
zoo	69.75					67.05		∘	∘	∘	70.14					69.70				

Tabla A.38: Tasas de acierto para $\mathcal{RFW}$-Bagging y $\mathcal{RFW}$-Random Subspaces 50 % y 75 % contra sus versiones sin $\mathcal{RFW}$, tanto para árboles podados *(P)* como sin podar *(U)*). Las marcas ∘ indican una victoria significativa de la versión $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

Conjunto	$\mathcal{RFW}$-Bagging *(P)*		$\mathcal{RFW}$-Bagging *(U)*		$\mathcal{RFW}$-Subspaces 50 % *(P)*		$\mathcal{RFW}$-Subspaces 50 % *(U)*		$\mathcal{RFW}$-Subspaces 75 % *(P)*		$\mathcal{RFW}$-Subspaces 75 % *(U)*	
abalone	25.51	∘	25.48	∘	25.86		25.70		25.46		25.13	
anneal	98.99		99.15		98.88		98.98		99.01		99.20	
audiology	84.44	∘	84.04		78.64		81.25		82.59		82.94	
autos	83.26		83.75		83.93		84.42		84.14		85.45	
balance-scale	83.23		82.52		83.22		84.22		79.31		80.73	
breast-w	96.81		96.74		96.88		96.78		96.81		96.68	
breast-y	73.61		72.03		73.54		73.47		73.86		72.29	
bupa	73.11		72.79		67.90		68.54		69.69		70.07	
car	94.05		94.80		70.02		70.17		92.18		94.10	
credit-a	86.81		86.45		86.54		86.74		86.49		86.41	
credit-g	76.07		75.62	∘	73.75		75.19		75.28		75.49	∘
crx	86.86		86.62		86.43		86.48		86.91		86.28	
dna	95.33		95.44		95.70		95.77		95.53		95.67	
ecoli	86.37		86.49		86.37		85.80		86.70		86.25	
glass	78.60	∘	78.31		78.41		78.60		77.84		77.50	
heart-c	81.69		81.33		83.27		83.13		82.11		82.20	
heart-h	80.75		79.87		82.36		82.73		81.85		81.23	
heart-s	93.53		91.35		93.53		93.37		93.53		91.92	
heart-statlog	83.11		82.48		83.70		82.48		83.26		82.33	
heart-v	75.35		75.45		74.20		75.70		73.80		75.55	
hepatitis	82.41		82.85		83.78		84.51		82.28		83.43	
horse-colic	85.07		85.67		84.74		84.37		85.04		85.48	
hypo	99.23		99.21		98.65		98.82		99.18		99.22	
ionosphere	93.39		93.42		93.42		93.54		93.73		93.65	
iris	94.93		95.07		94.33		94.47		95.00		95.07	
kr-vs-kp	99.41		99.48		97.16		97.55		99.18		99.31	
krk	88.26	∘	88.38	∘	43.57		45.09		83.41	∘	84.57	
labor	84.33		86.43		81.10		84.60		81.30		84.13	
led-24	75.09		74.88	∘	74.70		73.55		75.08		74.72	
letter	96.24	∘	96.35	∘	95.23	•	95.43	•	96.27	∘	96.45	∘
lrd	87.72		87.74		87.80		87.86		87.46	∘	87.55	∘
lymphography	82.25		82.46		83.07		85.16		81.45		83.68	
mushroom	100.00		100.00		100.00		100.00		100.00		100.00	
nursery	97.36		99.08	∘	91.44		92.29		94.49		96.21	
optdigits	98.14	∘	98.20	∘	98.08		98.14		98.24	∘	98.30	∘
page	97.49		97.46		97.40		97.45		97.48	∘	97.38	
pendigits	99.11	∘	99.14	∘	98.93		98.99		99.11	∘	99.16	∘
phoneme	90.02		90.09		86.48	∘	86.73		88.75	∘	88.93	∘
pima	76.03		76.03		75.29		75.47		75.64		75.62	
primary	45.57		44.63		45.91		46.61		46.20		45.87	
promoters	91.19		92.41		91.92		93.09		90.95	∘	92.45	∘
ringnorm	91.83	∘	92.17	∘	93.43		93.57		91.40	∘	91.60	∘
sat	91.72	∘	91.81	∘	91.64		91.88		91.82	∘	92.05	∘
segment	97.92		97.99		97.90		97.91		98.07		98.12	
shuttle	99.98		99.99	∘	99.95		99.96		99.98		99.99	
sick	98.80		99.00		95.58		96.74		98.53		98.71	
sonar	82.85		82.75		84.42		84.42		84.19	∘	84.09	∘
soybean-small	99.75		99.75		100.00		100.00		100.00		100.00	
soybean	95.05	∘	94.32	∘	94.88		94.64		95.14	∘	95.18	∘
splice	95.33	∘	95.71	∘	96.11		95.99		95.68		96.09	∘
threenorm	84.23		84.33		84.13		84.10		83.03	∘	83.27	∘
tic-tac-toe	94.56		96.85		81.25		84.34		89.31		94.41	∘
twonorm	93.53		93.57		93.93		94.03		92.60	∘	92.40	∘
vehicle	75.42		75.66		74.93		75.06		75.46		75.79	
vote1	90.48		90.12		90.30		90.92		90.35		90.51	
voting	96.07		96.50		95.31		95.50		96.09		96.69	
vowel-context	96.33	∘	96.94	∘	96.74		97.00		96.43	∘	96.87	∘
vowel-nocontext	95.09	∘	95.18	∘	95.10		95.10		95.63	∘	95.65	∘
waveform	84.77	∘	84.79	∘	84.77		84.77		84.57	∘	84.61	∘
yeast	62.04		61.78		60.38		58.84		61.84	∘	60.96	
zip	96.14	∘	96.20	∘	96.57	∘	96.65	∘	96.47	∘	96.56	∘
zoo	95.37		95.67		96.34		96.34		95.45		96.05	

Tabla A.39: Tasas de acierto para $\mathcal{RFW}$-AdaBoost y $\mathcal{RFW}$-MultiBoost contra sus versiones sin $\mathcal{RFW}$, tanto para las versiones con repesado (W) como para las de remuestro (S), y tanto para árboles podados *(P)* como sin podar *(U)*. Las marcas ∘ indican una victoria significativa de la versión $\mathcal{RFW}$ sobre el método correspondiente, mientras que las marcas • indican una derrota.

Conjunto	$\mathcal{RFW}$-AdaBoost-W *(P)*		$\mathcal{RFW}$-AdaBoost-W *(U)*		$\mathcal{RFW}$-AdaBoost-S *(P)*		$\mathcal{RFW}$-AdaBoost-S *(U)*		$\mathcal{RFW}$-MultiBoost-W *(P)*		$\mathcal{RFW}$-MultiBoost-W *(U)*		$\mathcal{RFW}$-MultiBoost-S *(P)*		$\mathcal{RFW}$-MultiBoost-S *(U)*	
abalone	24.28	∘	24.03	∘	24.39		24.03		25.20	∘	24.92	∘	25.05		25.25	
anneal	99.64		99.70		99.68		99.70		99.69		99.66		99.52		99.66	
audiology	85.19		84.66		84.61		85.02		85.15		84.04		84.94		84.27	
autos	86.29		85.26		84.87		85.59		84.68		84.56		85.71		84.90	
balance-scale	74.59	•	74.35		74.37		74.78		80.06		79.02		79.90		79.71	
breast-w	96.72		96.82		96.75		96.60		96.75		96.77		96.75		96.77	
breast-y	67.24		66.79		68.18		67.82		69.90		69.85		70.63		69.32	
bupa	68.82		70.41		70.69		71.41		72.50		72.73		72.13		71.76	
car	96.89		96.64		96.83		96.37		96.03		95.78		95.80		95.70	
credit-a	86.16		86.14		85.94		85.93		86.86		86.78		86.83		86.70	
credit-g	74.06		74.51		73.92		74.33		75.51		75.23		75.64		75.79	
crx	86.03		86.28		85.94		86.52		86.61		86.65		87.06		86.97	
dna	95.19		94.95		94.77		94.72		95.91		95.73		95.80		95.69	
ecoli	83.89		83.98		83.78		84.07		85.50		84.82		85.01		84.88	
glass	79.20		79.05		78.12		78.16		78.73		79.06		77.55		77.96	
heart-c	79.97		79.91		80.00		79.84		81.42		81.68		81.16		81.03	
heart-h	79.05		79.22		78.93		79.97		80.12		80.11		80.55		80.01	
heart-s	91.01		91.10		91.27		91.60		91.43		91.42		91.58		91.83	
heart-statlog	80.15		81.26		79.85		79.85		81.26		81.67		81.00		81.33	
heart-v	71.10		70.65		71.75		70.25		74.00		72.85		74.40		73.65	
hepatitis	83.97		83.89		83.76		84.23		83.64		83.58		83.90		83.69	
horse-colic	82.39		81.43		82.34		81.52		84.69		84.29		84.23		84.50	
hypo	98.92		98.92		98.93		98.90		99.08		99.02		99.09		99.07	
ionosphere	93.68		93.68		94.28		93.94		93.96		93.91		93.91		93.99	
iris	93.87		94.07		94.40		94.33		94.40		94.60		94.33		94.53	
kr-vs-kp	99.65		99.59		99.63		99.62		99.64		99.60		99.64		99.61	
krk	91.94	∘	91.20	∘	91.62	∘	90.85	∘	90.20	∘	89.70	∘	90.03	∘	89.39	∘
labor	87.13		88.23		88.50		88.33		88.53		89.03		89.27		88.50	
led-24	73.15	∘	72.83	∘	72.91	∘	73.14	∘	74.45	∘	73.90	∘	74.20	∘	73.99	∘
letter	96.93		96.97		96.90		96.89		96.70		96.73		96.65		96.63	
lrd	88.52		88.21		88.42		88.50		88.46		88.17		88.27		88.25	
lymphography	85.07		84.53		83.93		84.08		85.16		84.40		84.38		84.03	
mushroom	100.00		100.00		99.99		100.00		100.00		100.00		99.99		100.00	
nursery	99.85		99.78		99.84		99.76		99.74		99.60		99.73		99.67	
optdigits	98.51		98.54		98.51		98.54		98.40	∘	98.42		98.38		98.34	
page	97.10		97.09		97.05		97.04		97.40		97.43		97.40		97.38	
pendigits	99.37		99.34		99.35		99.33		99.23		99.26		99.26		99.24	
phoneme	90.93		90.93		91.03		90.88		90.84		90.86		90.83		90.75	
pima	73.91		74.07		73.56		74.09		75.45		75.13		75.30		75.58	
primary	40.33		40.77		39.20		40.03		40.51		43.90		40.59		43.04	
promoters	93.06		91.14		92.34		92.36		93.82		90.24		93.26		91.88	
ringnorm	95.63		95.40		95.63		96.10		95.10		94.67		95.73		95.57	
sat	92.10		92.04		91.98		92.05		91.97		91.81		91.82		91.92	
segment	98.48		98.42		98.41		98.37		98.33		98.29		98.27		98.29	
shuttle	99.99		99.99		99.99		99.99		99.99		99.99		99.99		99.99	
sick	98.89		98.86		98.96		98.94		98.94		98.86		98.92		98.92	
sonar	84.36		85.02		82.74		83.42		83.80		83.47		82.93		82.98	
soybean-small	99.10		98.65		98.95		98.45		99.10		98.65		98.70		98.25	
soybean	94.39		93.69		93.78		93.22		94.46		94.30		94.42		94.50	
splice	95.45		94.57		94.87		94.49		96.03		95.90		95.94		95.76	
threenorm	84.17		84.63		83.33		84.00		84.80		84.77		84.03		83.93	
tic-tac-toe	98.88		98.58		98.85		98.58		98.08		97.76		98.12		97.80	
twonorm	95.20		94.60		93.67		94.47		94.17		94.13		94.50		94.53	
vehicle	76.98		77.03		76.58		76.81		76.98		76.48		76.16		76.50	
vote1	89.44		89.40		89.33		89.37		90.25		89.77		90.16		89.68	
voting	94.80		95.47		95.12		95.14		95.79		95.72		95.52		95.84	
vowel-context	97.49	∘	97.74		96.89		97.20		96.77	∘	97.51		96.05		96.61	
vowel-nocontext	96.36		96.54		96.22		96.39		95.75		95.74		95.57		95.58	
waveform	84.60		84.68		84.37		84.17		84.89		84.87		84.57		84.55	
yeast	59.36		59.51		59.18		58.81		61.19		60.83		60.94		60.12	
zip	96.70		96.74		96.66		96.61		96.56		96.61		96.48		96.47	
zoo	96.15		95.95		95.75		95.84		96.05		95.95		95.95		96.15	

Bibliografía

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Jaume Amores, Nicu Sebe, and Petia Radeva. Boosting the distance estimation: Application to the -nearest neighbor classifier. *Pattern Recognition Letters*, 27(3):201–209, 2006.

[3] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W.P. Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.

[5] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[6] Eta S. Berner, editor. *Clinical decision support systems: Theory and Practice*. Health Informatic Series. Springer, 2nd edition, 2007.

[7] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[8] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Dept., Univ. of California, Berkeley, 1996.

[9] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Mach. Learn.*, 36(1-2):85–103, 1999.

[10] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[11] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[12] Leo Breiman. Using iterated bagging to debias regressions. *Machine Learning*, 45(3):261–277, December 2001.

[13] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.

[14] Carla E. Brodley. Recursive automatic bias selection for classifier construction. *Machine Learning*, 20(1-2):63–94, 1995.

[15] Robert K. Bryll, Ricardo Gutierrez-Osuna, and Francis K. H. Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, 2003.

[16] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[17] Bruno Caprile, Stefano Merler, Cesare Furlanello, and Giuseppe Jurman. Exact bagging with k-nearest neighbour classifiers. In *Multiple Classifier Systems*, pages 72–81, 2004.

[18] Philip Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 90–98. Morgan Kaufmann, 1995.

[19] Philip K. Chan and Salvatore J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 39–44. AAAI Press, 1995.

[20] Nitesh V. Chawla, Kevin W. Bowyer Lawrence O. Hall, and W. Philip Kegelmeyer. Learning ensembles from bites, a scalable and accurate approach. *Journal of Machine Learning Research*, 5:421–451, 2004.

[21] Stefan W. Christensen, Ian Sinclair, and Philippa A. S. Reed. Designing committees of models through deliverate weighting of data points. *Journal of Machine Learning Research*, 4(1):39–66, 2004.

[22] Andreas Christmann, Ingo Steinwart, and Mia Hubert. Robust learning from bites for data mining. *Computational Statistics & Data Analysis*, 52(1):347–361, 2007.

[23] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960.

[24] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge University Press, 1 edition, March 2000.

[25] Pádraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *In 11th European Conference on Machine Learning*, pages 109–116. Springer, 2000.

[26] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[27] D. G. T. Denison, N. M. Adams, C. C. Holmes, and D. J. Hand. Bayesian partition modelling. *Comput. Stat. Data Anal.*, 38(4):475–485, 2002.

[28] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, 2000.

[29] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40(2):139–157, 2000.

[30] Carlotta Domeniconi and Bojun Yan. Nearest neighbor ensemble. In *ICPR (1)*, pages 228–231, 2004.

[31] Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[32] Harris Drucker. Improving regressors using boosting techniques. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 107–115, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[33] W. Duch, K. Grudzinski, and G. Stawski. Symbolic features in neural networks. In *Proc. 5th Conference on Neural Networks and Soft Computing*, pages 180–185, Zakopane, 2000.

[34] Wlodzislaw Duch and Geerd H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87(3):341–371, 1995.

[35] Günther Eibl and Robert Schapire. Multiclass boosting for weak classifiers. In *Journal of Machine Learning Research*, pages 6–189, 2005.

[36] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *In Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, 1999.

[37] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

[38] Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995.

[39] Yoav Freund. An adaptive version of the boost by majority algorithm. In *In Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 102–113, 2000.

[40] Yoav Freund. A more robust boosting algorithm, 2009. En http://arxiv.org/abs/0905.2138v1, última comprobación: 1 de Julio de 2010.

[41] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[42] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[43] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[44] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[45] Jerome H. Friedman and Usama Fayyad. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[46] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Twayne Publishers, Boston, 1990.

[47] J. Gama and P. Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.

[48] Nicolás García-Pedrajas, César García-Osorio, and Colin Fyfe. Nonlinear "boosting" projections for ensemble construction. *Journal of Machine Learning Research*, 8:1–33, 2007.

[49] K. Grabczewski and N. Jankowski. Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003*, volume 2714, pages 359–366. Springer, 2003.

[50] César Guerra-Salcedo and Darrell Whitley. Genetic approach to feature selection for ensemble creation. In *In Proc. of Genetic and Evolutionary Computation Conference*, pages 236–243. Morgan Kaufmann, 1999.

[51] L. Yu H. Liu and H. Motoda. *Data Mining Handbook*, chapter Feature Extraction, Selection and Construction, pages 409–422. Lawrence Erlbaum Associates, 2003.

[52] David J. Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

[53] Ralf Herbrich. *Learning Kernel Classifiers: Theory and Algorithms (Adaptive Computation and Machine Learning)*. The MIT Press, December 2001.

[54] José Hernández Orallo, Mª José Ramírez Quintana, and Cèsar Ferri Ramírez. *Introducción a la Minería de Datos*. Pearson Prentice Hall, 2004.

[55] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[56] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing, A Guide to Theory, Algorithm, and System Development*. Prentice Hall, 2001.

[57] Michael I. Jordan and Rober A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[58] Chandrika Kamath and Erick Cantú-Paz. Creating ensembles of decision trees through sampling. In *Proceedings, 33rd symposium on the interface of computing science and statistics, Costa Mesa, CA*, 2001.

[59] Chandrika Kamath, Erick Cantú-Paz, and David Littau. Approximate splitting for ensembles of trees using histograms. In Robert L. Grossman, Jiawei Han, Vipin Kumar, Heikki Mannila, and Rajeev Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002*, 2002.

[60] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

[61] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207, 1996.

[62] R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Thirteenth International Conference on Machine Learning*, pages 275–283. Morgan Kaufmann, 1996.

[63] Ron Kohavi and Ross Quinlan. Decision tree discovery. In *in Handbook of Data Mining and Knowledge Discovery*, pages 267–276. University Press, 1999.

[64] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1995.

[65] Ludmila Kuncheva and Christopher J. Whitaker. Using diversity with three variants of boosting: Aggressive, conservative, and inverse. In *Multiple Classifier Systems 2002*, pages 81–90, 2002.

[66] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, July 2004.

[67] Ludmila I. Kuncheva. On the optimality of naïve bayes with dependent binary features. *Pattern Recognition Letters*, 27(7):830–837, 2006.

[68] Ludmila I. Kuncheva and Juan J. Rodríguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):500–508, 2007.

[69] Ludmila I. Kuncheva and Juan J. Rodríguez. An experimental study on rotation forest ensembles. In *7th International Workshop on Multiple Classifier Systems, MCS 2007*, volume 4472 of *LNCS*, pages 459–468. Springer, 2007.

[70] Seong-Whan Lee. *Advances in Handwriting Recognition*, volume 34 of *Series in Machine Perception and Artificial Intelligence*. World Scientific Publishing Co. Pte. Ltd., 1999.

[71] Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4), 2004.

[72] Yuansong Liao and John E. Moody. Constructing heterogeneous committees using input feature grouping: Application to economic forecasting. In *Advances in Neural Information Processing Systems 12, NIPS*, pages 921–927, 1999.

[73] Alan Wee-Chung Liew, Hong Yan, and Mengsu Yang. Pattern recognition techniques for the emerging field of bioinformatics: A review. *Pattern Recognition*, 38(11):2055–2073, 2005.

[74] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proc. 14th International Conference on Machine Learning*, pages 211–218. Morgan Kaufmann, 1997.

[75] Gonzalo Martínez-muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38:1483–1494, 2005.

[76] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Cascading for nominal data. In *7th International Workshop on Multiple Classifier Systems, MCS 2007*, volume 4472 of *LNCS*, pages 231–240. Springer, 2007.

[77] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Cascading with vdm and binary decision trees for nominal data. In Oleg Okun and Giorgio Valentini, editors, *Workshop on Supervised and Unsupervised*

Ensemble Methods and Their Applications, SUEMA'2007, pages 28–42, 2007.

[78] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Cascading with vdm and binary decision trees for nominal data. In Oleg Okun and Giorgio Valentini, editors, *Supervised and Unsupervised Ensemble Methods and their Applications*, volume 126 of *Studies in Computational Intelligence*, pages 165–178. Springer, 2008.

[79] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Disturbing neighbors diversity for decision forests. In Oleg Okun and Giorgio Valentini, editors, *Workshop on Supervised and Unsupervised Ensemble Methods and their Applications, SUEMA 2008*, pages 67–71, 2008.

[80] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Disturbing neighbors diversity for decision forests. In Oleg Okun and Giorgio Valentini, editors, *Applications of Supervised and Unsupervised Ensemble Methods*, volume 245 of *Studies in Computational Intelligence*. Springer, 2009.

[81] Jesús Maudes, Juan J. Rodríguez, and César García-Osorio. Disturbing neighbors ensembles for linear svm. In Jon Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems, 8th International Workshop, MCS 2009*, volume 5519 of *Lecture Notes in Computer Science*, pages 191–200. Springer, 2009.

[82] Jesús Maudes, Juan J. Rodríguez, César García-Osorio, and Carlos Pardo. Random projections for svm ensembles. In *23rd Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA-AIE 2010*, pages 87–95. Springer, 2010.

[83] Prem Melville and Raymond J. Mooney. Constructing diverse classifier ensembles using artificial training examples. In *In Proceedings of the 18th International Joint Conference on Artifical Intelligence*, pages 505–510, 2003.

[84] Stefano Merler, Bruno Caprile, and Cesare Furlanello. Parallelizing adaboost by weights dynamics. *Computational Statistics & Data Analysis*, 51(5):2487–2498, 2007.

[85] C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(239–281), 2003.

[86] Y. Y. Nguwi and A. Z. Kouzani. A study on automatic recognition of road signs. In *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, pages 1–6, 2006.

[87] Steven J. Nowlan and Geoffrey E. Hinton. Evaluation of adaptive mixtures of competing experts. In *Advances in Neural Information Processing Systems 3, NIPS*, pages 774–780. Morgan Kaufmann, 1990.

[88] David Opitz and Richard Maclin. Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[89] Nikunj C. Oza and Kagan Tumer. Input decimation ensembles: Decorrelation through dimensionality reduction. In *LNCS*, pages 238–247. Springer, 2001.

[90] T. V. Pham and A. W. M. Smeulders. Quadratic boosting. *Pattern Recognition*, 41(1):331–341, 2008.

[91] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. MIT Press, 1998.

[92] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[93] J. Ross Quinlan. Simplifying decision trees. *Int. J. Hum.-Comput. Stud.*, 51(2):497–510, 1999.

[94] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI-01 workshop on Empirical Methods in AI*, 2001.

[95] Juan J. Rodríguez, César García-Osorio, and Jesús Maudes. Forests of nested dichotomies. *Pattern Recognition Letters*, 31(2):125–132, 2010.

[96] Juan J. Rodríguez and Ludmila I. Kuncheva. Naïve bayes ensembles with a random oracle. In *7th International Workshop on Multiple Classifier Systems, MCS 2007*, volume 4472 of *LNCS*, pages 450–458. Springer, 2007.

[97] Juan J. Rodríguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, Oct 2006.

[98] Juan J. Rodríguez and Jesús Maudes. Ensembles of grafted trees. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI-06, 17th European Conference on Artificial Intelligence*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 803–804. IOS Press, 2006.

[99] Juan J. Rodríguez and Jesús Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29(8):1049–1059, jun 2008.

[100] Juan J. Rodríguez, Jesús Maudes, and Carlos J. Alonso. Rotation-based ensembles of RBF networks. In Michel Verleysen, editor, *ESANN'2006, 14th European Symposium on Artificial Neural Networks*, pages 605–610, Belgium, 2006. d-side.

[101] Juan J. Rodríguez, Jesús Maudes, Carlos Pardo, and César García-Osorio. Disturbing neighbors ensembles for regression. In *XIII Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA - TTIA 2009*, pages 369–378. Asociación Española para la Inteligencia Artificial, 2009.

[102] Lior Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific Publishing Co. Pte. Ltd., 2010.

[103] Lior Rokach and Oded Maimon. *Data mining with decision trees: theroy and applications*, volume 69. World Scientific Publishing Co. Pte. Ltd., 2008.

[104] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.

[105] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[106] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[107] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[108] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[109] A.K. Seewald. How to make stacking better and faster while also talking care of unknown weakness. In *Machine Learning Proceeedings of the 19th International Conference ICLM 2002*, pages 554–561, 2002.

[110] A.K. Seewald. *Towards understanding Stacking*. PhD thesis, Vienna University of Technology, 2003.

[111] A.K. Seewald and J. Fürnkranz. An evaluation of grading classifiers. In *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001*, pages 115–124. Springer, 2001.

[112] Jeffrey S. Simmonoff. *Analyzing Categorical Data*. Springer Texts in Statistics. Springer, 2003.

[113] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1229, 1986.

[114] Yijun Sun, Sinisa Todorovic, and Jian Li. Reducing the overfitting of adaboost by controlling its data distribution skewness. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(7):1093–1116, 2006.

[115] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, November 1999.

[116] Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, Vol.40(No.2):980–991, 2000.

[117] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005. `http://www.cs.waikato.ac.nz/ml/weka`.

[118] D. Wolpert. Stacked generalization. *Neural networks*, 5:241–260, 1992.

[119] Xindong Wu and Vipin Kumar. *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC, 2009.

[120] Tzay Y. Young, editor. *Handbook of pattern recognition and image processing (vol. 2): computer vision*. Academic Press, Inc., Orlando, FL, USA, 1994.

[121] Chun-Xia Zhang and Jiang-She Zhang. A local boosting algorithm for solving classification problems. *Comput. Stat. Data Anal.*, 52(4):1928–1941, 2008.

[122] Chun-Xia Zhang and Jiang-She Zhang. Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recognition Letters*, 29(10):1524–1536, July 2008.

[123] Zhi-Hua Zhou and Yuan Jiang. Nec4.5: Neural ensemble based c4.5. *IEEE Trans. Knowl. Data Eng.*, 16(6):770–773, 2004.