UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR

PROGRAMA DE DOCTORADO
«INVESTIGACIÓN EN INGENIERÍA»

Estudio de métodos de construcción de ensembles de clasificadores y aplicaciones

José Francisco Diez Pastor

BURGOS, 15 de Junio de 2015

La investigación realizada en esta tesis doctoral ha sido parcialmente subvencionada por el Ministerio de Economía y Competitividad, proyecto TIN-2011-24046.

Agradecimientos

Esta memoria de tesis está dedicada a todos aquellos que la han hecho posible.

Un agradecimiento muy especial a mis dos tutores César y Juanjo, sin cuya ayuda, consejos y visión no habría podido llevar a cabo esta tesis. Agradezco especialmente el interés en las revisiones de los contenidos de esta tesis y los artículos que la componen, porque sin estas revisiones mi inglés no lo entendería ni yo.

Otro agradecimiento muy importante es para Lucy Kuncheva, con quien tuve el placer de colaborar en dos ocasiones, en la preciosa ciudad de Bangor. No solo es una investigadora brillante, es una bellísima persona que trata a sus doctorandos e investigadores visitantes con mucho cariño. De mi primera estancia en Bangor también quiero recordar a Ramón Mollineda y su familia, que compartieron su sentido del humor y su tiempo con nosotros. De mi segunda estancia quiero recordar a toda la pandilla de españoles que nos juntamos, que hicieron mi estancia mucho más feliz.

De mis compañeros de universidad quiero dar las gracias a Jesús Maudes por las ideas tan buenas que tiene y sus chistes, a Carlos Pardo porque tiene soluciones para todo y conocimientos ancestrales que solo conoce él, a Lolo y Andrés que con discusiones de matrimonio amenizan el despacho, a Carlos López por su tiempo dedicado en resolver cualquier duda y porque es un placer colaborar con él, a Raúl Marticorena por todo lo aprendido en la dirección de proyectos, a Julián Luengo y Ángel Arroyo con quienes compartí mi primer año de docencia, a Álvar Arnaiz que me ha ayudado a soltar estrés con sus visitas al despacho. Y a todos los demás con los que todavía no he tenido la ocasión de colaborar por los cafés. También recordar a otros que ahora no están en el área, pero me ayudaron en su momento como José Miguel Robledo y Pedro Santos.

Quiero agradecer también a los alumnos que he tenido hasta ahora, por ayudarme a darme cuenta que lo que quiero es dedicarme a la universidad.

Lógicamente tengo que agradecer y reconocer el esfuerzo de todos aquellos que trabajan para hacer posible la investigación de otros: a los desarrolladores de Weka, de ImageJ, de Keel, los mantenedores de repositorios como el de la UCI y todos aquellos que colaboran en herramientas y bibliotecas Open Source.

Por último, quiero agradecer los ánimos a mi familia y amigos. A mi

padre, mi madre, mi hermana, mi sobrina, mis primos y mis tios. A mis amigos de Burgos por las cañas de los viernes y los sábados y a mis amigos de Ciruelos de Cervera que después de varios años llamándome “Doctor”, por fin podrán hacerlo con motivo. Y también a mis compañeros y amigos de la Escuela de Idiomas y el Drink in English, porque si pensáis que mi inglés es malo, deberíais haberlo visto antes.

Fin del comunicado.

Acknowledgements

I would like to thank Lucy Kuncheva for her collaboration. She is not only a brilliant researcher, she is a beautiful person who treats her PhD students and visiting researchers with love.

Acknowledge the work of the reviewers, their interesting comments have improved enormously the papers of the thesis.

Thanks to Weka, Keel, ImageJ, LaTeX and Ubuntu developers. Thanks to the UCI Repository's maintainers.

UNIVERSIDAD DE BURGOS

Estudio de métodos de construcción de ensembles de clasificadores y aplicaciones

La tesis «Estudio de métodos de construcción de ensembles de clasificadores y aplicaciones», que presenta D. José Francisco Díez Pastor para optar al título de doctor, ha sido realizada dentro del programa de «Investigación en Ingeniería», en el Área de Lenguajes y Sistemas Informáticos perteneciente al Departamento de Ingeniería Civil de la Universidad de Burgos bajo la dirección de los doctores D. César Ignacio García Osorio y Juan José Rodríguez Diez.

Los directores autorizan la presentación del presente documento como memoria para optar al grado de Doctor por la Universidad de Burgos.

Vº. Bº. del Director:	Vº. Bº. del Director:	El doctorando:
Dr. D. César Ignacio García Osorio	Dr. D. Juan José Rodríguez Díez	D. José Francisco Diez Pastor

Burgos, Tuesday 2nd June, 2015

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR

Programa de doctorado «Investigación en Ingeniería»
Tesis

TÍTULO:
Estudio de métodos de construcción de ensembles de clasificadores y aplicaciones

AUTOR: José Francisco Diez Pastor

DIRECTORES: Dr. D. César Ignacio García Osorio y Dr. D. Juan José Rodríguez Diez

RESUMEN:
La inteligencia artificial es el área de conocimiento que se dedica a la creación de sistemas informáticos con un comportamiento inteligente. Dentro de este área se puede considerar que el aprendizaje computacional estudia la creación de sistemas que aprenden por sí mismos. En el aprendizaje supervisado se le proporcionan al sistema tanto las entradas como la salida esperada, cuando la salida es de tipo categórico, se trata de un clasificador y cuando la salida es numérica, se trata de un regresor. En ocasiones en ciertos problemas ocurre que el número de ejemplos de un tipo es mucho mayor que el número de ejemplos de otro tipo, cuando esto ocurre se habla de conjuntos desequilibrados. La combinación de varios clasificadores o regresores es lo que se denomina ensemble, y a menudo ofrece mejores resultados que cualquiera de los miembros que lo forman.

Esta tesis, se centra en el desarrollo de nuevos algoritmos de construcción de ensembles, sobre todo haciendo hincapié a las técnicas de incremento de la diversidad en ensembles homogéneos (cuando todos los miembros están construidos usando la misma técnica).

En la primera parte de la tesis, se presenta un breve estudio de los

métodos más representativos de las distintas técnicas de construcción de ensembles, aprendizaje en conjuntos desequilibrados y breves nociones sobre validación experimental. En la segunda, aparecen todas las publicaciones que han sido realizadas en el contexto de esta tesis. Esta segunda parte puede ser dividida en tres bloques.

En un primer bloque, se explora la utilización de la fase constructiva de la metaheurística GRASP como una manera de inyectar aleatoriedad en algoritmos de construcción de árboles. Inyectar aleatoriedad en el propio algoritmo del clasificador base es una de las técnicas usadas para incrementar la diversidad de un ensemble. Esta técnica, que ha sido llamada "GRASP Forest" ha sido utilizada con éxito en árboles de clasificación y árboles de regresión.

La diversidad es clave en los ensembles, pero se quiere incrementar la diversidad sin afectar gravemente a la precisión de los clasificadores base. Profundizando en la idea anterior, se ha desarrollado un segundo método "GAR-Forest" *GRASP with annealed randomness*.

En este método se parte de la idea intuitiva de que los nodos que más influencia tienen en la correcta clasificación de las instancias son los nodos inferiores y hojas, mientras que los nodos que más afectan la estructura global del árbol (y por lo tanto la diversidad) son la raíz y los nodos superiores. Partiendo de esa idea se ha diseñado un método que utiliza la metaheurística GRASP, para controlar el nivel de aleatoriedad en cada uno de los niveles en el árbol. Creando árboles en donde la raíz es completamente aleatoria y el nivel de aleatoriedad que disminuye según se construye el árbol.

En un segundo bloque se aborda el problema de los ensembles para conjuntos desequilibrados. Existen varias estrategias para lidiar con el problema del desequilibrio, una de ellas es utilizar distintos métodos de preprocesado como *Undersampling* o *SMOTE*, los parámetros óptimos de estos métodos son dependientes del problema y a menudo son difíciles de encontrar. En este bloque se presenta un método llamado "Random Balance", basado en la idea de variar aleatoriamente las proporciones entre las clases, confiando en esta heurística, se elimina la necesidad de ajustar parámetros, a la vez que se aumenta la diversidad del ensemble.

Como se ha mencionado previamente, cuando se aborda el problema del desequilibrio, las técnicas más comúnmente usadas son aquellas que afectan la proporción entre las clases (re-pesado, sobremuestreo y submuestreo,

etc.). En otro trabajo de este bloque se estudia el efecto de distintas técnicas (*Random Oracles*, *Random Feature Weights*, *Rotation Forest* y *Disturbing Neighbours*), originalmente destinadas a aumentar la diversidad en ensembles no desequilibrados. Se realizan varios análisis sobre el impacto del tamaño del ensemble en el rendimiento del ensemble, se estudia en qué ocasiones estas técnicas mejoran a los ensembles del estado del arte para desequilibrados y qué combinación de ensemble y técnica de diversidad es la que ofrece mejores resultados para distintas medidas. También se realiza un estudio que trata de predecir en qué ocasiones es más apropiado utilizar técnicas de incremento de la diversidad basándose en distintas meta-características propias del conjunto de datos.

Finalmente, se aplican algunas de estas técnicas a la solución de varias aplicaciones reales. Se han aplicado ensembles para la predicción de la calidad superficial en procesos de mecanizado y para el desarrollo de un sistema de detección de defectos en piezas metálicas mediante imágenes de radiografía.

PALABRAS CLAVE:
Minería de datos, ensembles, diversidad, GRASP, Random Balance, Boosting projections, boosting, conjuntos de datos desequilibrados, análisis de radiografía

ABSTRACT:
Artificial intelligence is the knowledge area devoted to the creation of computer systems with intelligent behavior. Within this area, Machine Learning can be defined as the area that studies the creation of systems that learn by themselves. In supervised learning, the system receives both the inputs and the expected output, when the output is categorical, it is named classifier and when the output is numeric, it is named regressor. Sometimes it happens that the number of examples belonging to a class is much greater than the number of examples belonging to other, when this happens in a certain problem, it is called unbalanced problem. The combination of multiple classifiers or regressors is called ensemble, and often provides better results than any of the members which comprise it.

This thesis focuses on the development of new ensemble building algorithms, especially emphasizing the techniques of increasing diversity in homogeneous ensembles (those ensembles whose members are all built using the same technique).

In the first part of the thesis, a brief survey of the most representative ensemble building techniques is presented, some theory about imbalanced learning is shown and brief notions on experimental validation are described. In the second part, all publications that have been produced in the context of this thesis are included. This second part can be divided into three blocks.

In the first block, the use of the construction phase of GRASP metaheuristic as a way to inject randomness into tree construction algorithms is explored. Injecting randomness directly into the base classifier algorithm is one of the most commonly used techniques to increase the diversity of an ensemble. This technique, which has been called “GRASP Forest” has been successfully applied to classification and regression trees.

Diversity is essential in the ensembles, but one wants to increase the diversity without seriously affecting the accuracy of the base classifiers. Expanding on the above idea, it has been developed a second method “GAR-Forest” (GRASP with annealed randomness). This method is based on one intuitive idea: nodes that have the most influence on the correct classification of the instances are the lower nodes and leaves, while nodes that affect the overall structure of the tree (and hence diversity) are the root and the upper nodes. Starting from this idea, it has been developed a method that uses GRASP metaheuristic to control the level of randomness in each levels of the tree. Creating trees where the root is totally random and the

level of randomness decreases as the tree is constructed.

In a second block of publications the problem of using ensembles for imbalanced learning is tackled. There are several strategies for dealing with the problem of imbalance, one is to use preprocessing methods such as Undersampling or SMOTE, the optimum parameters of these methods are problem dependent and often difficult to find. In this block it is presented a method called "Random Balance", which is based on the idea of randomly vary the proportions between classes, by doing that, the need to adjust parameters is eliminated, while the diversity of the ensemble is increased.

As previously mentioned, the problem of imbalance is frequently addressed with techniques that affect the ratio between classes (reweighing, oversampling and undersampling, etc.). In another paper belonging to this block, it is studied the effect on different techniques (Random Oracles, Random Feature Weights, Rotation Forest and Disturbing Neighbours), originally designed to increase diversity in balanced ensembles on imbalanced problems. In this paper , several analyzes have been carried out: on the impact of how the size of an ensemble affects its performance, on the suitability of diversity enhancing techniques in imbalanced classification, to contrast which combination between ensemble learning and diversity scheme has the best overall results for different metrics of performance, to extract rules based on data complexity metrics to find when it is more suitable to combine ensemble methods with diversity techniques.

Finally, some of the ensemble techniques studied in the previous two blocks are applied to the solution of several real-world applications. Ensembles have been applied to the prediction of surface quality in machining processes and for developing a defect detection system in metal pieces using X-ray images.

KEYWORDS:
Data mining, ensembles, diversity, GRASP, Random Balance, Boosting projections, boosting, imbalanced learning, radiography analysis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to ensemble based systems

This chapter will review the basics of Data Mining, classification and ensemble based systems.

In recent years, technological advances have led to an exponential increase in the amount of data available for processing. Unfortunately, this unmanageable data overflow has no increased our knowledge, we are being inundated with data, but we are unable to extract all the knowledge that it conveys.

Data mining emerges in this context of necessity of knowledge extraction in a world where data increases exponentially. [23].

The major tasks in Data Mining are:

- Prediction
  - Clasification.
  - Regression.
- Association rule learning.
- Clustering.

This thesis focuses on the study and proposal of new methods of classification and regression, using ensembles, a concept explained later.

Classification is a process through which a group of items (or instances) is categorized as belonging to certain subsets, called classes. The classification is used in many areas, for example in medical diagnosis, a possible use of classification could be the diagnosis of whether a patient has a particular disease or not, using data obtained through specific analyses. In the analysis

of banking data, using data from different sources, a classification algorithm could forecast if a client can deal with a mortgage or will fall into default. Ultimately, classification can be understood as the process that takes as input a set of data, that can be of any nature, defining a particular instance of an object or situation and the process output is the class to which the object or situation belongs. The class can be a binary class (grant mortgage / deny mortgage) or it can have multiple values (flu / pneumonia / cold / ...). If, instead of discrete, the value to predict is continuous, the data mining task is a regression problem. Ensembles are combinations of classifiers or regressors, that use different techniques to obtain better predictive performance than could be obtained from any of the constituent learning algorithms [59]. The intuitive idea behind these new algorithms is the same as that in real life when one is faced with a difficult decision, a second or third opinion is sought. When important decisions need to be taken, it is usual to consult third parties about their opinion, in the case of medical diagnosis, it is usual that doctors consult with other specialists to see whether they have the same opinion or disagree. Speaking in a formal way, an ensemble of size T has a set of T hypothesis, $\mathscr{H} = \{h_1, \ldots, h_T\}$ where each one of them is the output of a predictor. The hypothesis of the ensemble is the combination of the hypothesis of each of its constituent base predictors (classifiers or regressors)[1]. This is a new philosophy when it comes to seek to improve the accuracy in pattern recognition. Instead of looking for the best set of features and the best classification method, now the idea is to find the best set of predictors and the best combination method [37]. Ensembles based algorithms have proven to be the best technique to solve complicated problems of classification and regression [19, 59, 10]. Although the ensembles have been studied mainly applied to classification problems, their origin is linked to the decomposition of generalization error in regression.

If the output of an ensemble of regressors is defined as a weighted average of its members

$$\bar{f}(x) = \sum_{k=1}^{T} w_k f_k(x) \tag{1.1}$$

where $f_k(x)$ represents the k-th regressor of the ensemble and the weigths

[1]The combination may be realized in many ways: voting, weighted voting, using a referee, using a meta-classifier having as inputs the outputs of the first level classifiers, etc.

w_k sum 1. The generalized error of the ensemble

$$\varepsilon(x) = (y(x)) - \bar{f}(x))^2 \tag{1.2}$$

can be decomposed as

$$\varepsilon(x) = \bar{e}(x) - \bar{a}(x) \tag{1.3}$$

Where the first term $\bar{e}(x)$, is the average error of the base regressors of the ensemble

$$\bar{e}(x) = \sum_{k=1}^{T} w_k (y(x) - f_k(x))^2 \tag{1.4}$$

And the second $\bar{a}(x)$, is the diversity of the ensemble.

$$\bar{a}(x) = \sum_{k=1}^{T} w_k (f(x) - \bar{f}(x))^2 \tag{1.5}$$

Therefore, according to the equation 1.3, to reduce the error of an ensemble we can proceed in two ways, reducing the average error of the members of the ensemble or increasing their diversity. In general, both for classification and regression, the total error of an ensemble is the sum of the error of the individual classifiers/regressors (also called bias) and the error variance.

1.2 Reasons for using ensembles based systems

In [19], Dietterich indicates the three main reasons why a set of classifiers can be better than a single classifier. According to Dietterich these reasons are: statistical, computational and representational.

More recently, in [59] Robi Polikar completes and extends this list of reasons with four reason more that ar listed below.

1.2.1 Statistical reasons

A learning algorithm can be seen as the search for the best hypothesis h, within the space $\mathscr{H}$ of hypotheses, that best fits the data. The statistical problem arises when the training set is very small compared to the set of hypotheses. This means that the learning algorithm can find several hypothesis h, that are different, but achieve the same accuracy on the training dataset.

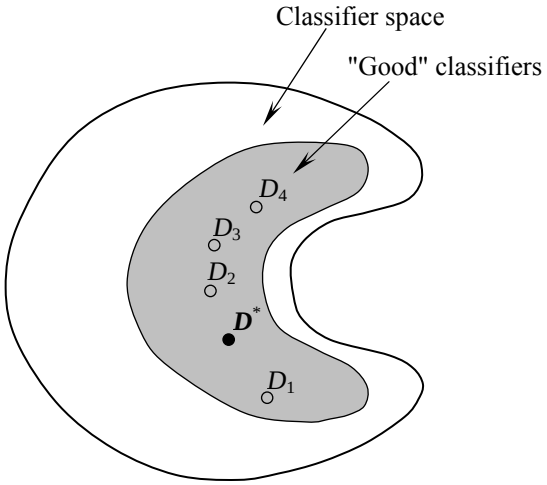


Figure 1.1: The statistical reason. D^* is the best classifier for one particular problem. The outer curve represents the set of hypotheses, the inner curve represents the set of hypotheses with the best accuracy on the training set. It can be seen that the average of the best hypothesis is more successful than any of them individually. Figure extracted from [45].

1.2.2 Computational reasons

Many Machine Learning algorithms operate in a way that can make them get stuck in local optima. For example, neural networks use gradient descent to minimize the error on the training set, and decision trees use a greedy algorithm to build the tree. In these cases, it is computationally very hard to find the best hypothesis. To find the optimal neural network or the optimal decision tree is a NP problem [40, 6].

An ensemble constructed from classifiers that have started their training at different points provides a better approximation than any of the individual classifiers.

1.2.3 Representational reasons

In many learning methods, the function $\mathscr{F}$ that correctly classifies the data can not be represented by any h of the hypotheses set $\mathscr{H}$ of the learning method. However, forming a new hypothesis based on the combination of the set of hypotheses $\mathscr{H}$ it is possible to expand the space of functions and it is possible that a set of classifiers approximates the function much better than any of the individual classifiers separately.

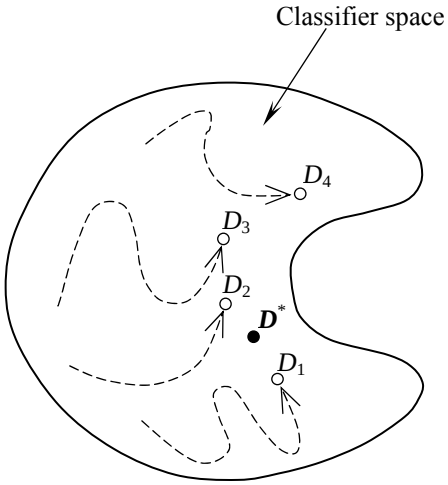


Figure 1.2: The computational reason. D^* is the best classifier for one particular problem. The outer curve represents the set of hypotheses, the dotted lines show hypothetical paths during training of the classifiers, it is observed that many of them get stuck in local minima during training. Figure extracted from [45].

1.2.4 Big Data

In certain applications the amount of data to be analyzed is too large to be handled by a single classifier. Training a classifier with a huge amount of data is often a bad idea. Partitioning the data into subsets and training different classifiers with different portions of data, and then combining the outputs of these classifiers is a much more efficient solution. In [29] García-Pedrajas presents a review of data mining algorithms that have been adapted to handle large volumes of data using ensembles and partitioning.

1.2.5 Too few data

This is the opposite case. In order to make a data mining algorithm adequately learn the data distribution, the data used for its training must be sufficiently representative. In the absence of an appropriate data, re-sampling techniques can be employed to produce overlapping random subsets of the available data that can be used to train different classifiers, whose predictions will be combined to get the final prediction. This approach has proven to be very effective.

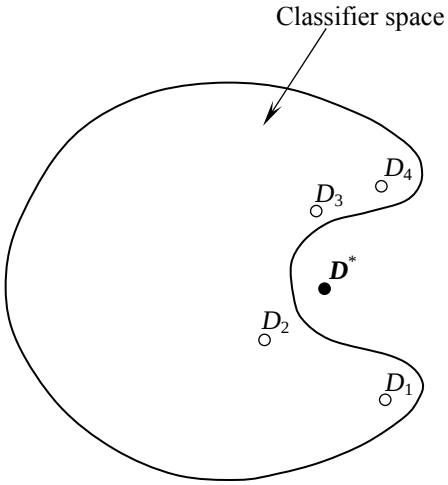


Figure 1.3: The representational reason. D^* is the best classifier for one particular problem, it is located outside the set of hypotheses that can be used by a certain classifier, but it can be obtained by combining classifiers. Figure extracted from [45].

1.2.6 Divide and conquer

This is a version of the representational reason presented by Dietterich in [19]. Regardless of the amount of available data, certain problems are too difficult to solve with a single classifier. The decision boundary may be too complex or outside the range of functions that can be implemented by the chosen classifier, for example a linear classifier can not separate groups that are not linearly separable. However, the combination of several simple classifiers can provide the flexibility needed to represent complex boundaries.

1.2.7 Data fusion

If we have several datasets obtained from multiple sources with features of different nature (heterogeneous features), it is possible that a single classifier cannot be used to learn the information contained in the combination of sources. For example, in Medicine, the evidences about the cause of an illness may come from several diagnostic test, for example a patient scan, a blood test, each of these generate data with different number of characteristics and of different types. Due to the different nature of the input sources, a single classifier may have problems to combine them, making the learning process impossible. On the other hand, in an ensemble, each classifier can be trained on different data and then the results combined (note that, in this

case, is not the data, but the predictions which are combined).

1.3 Ensembles: Taxonomy and most common techniques

This section will briefly present the different ensemble learning algorithms, depending on the techniques used to create the base classifiers or according to the methods of combining the outputs of classifiers. All ensemble learning algorithms are based on the use of two strategies. The first one is designed to build a set of classifiers as different as possible. The second one is designed to combine the outputs of the individual classifiers to make the final output of the ensemble as accurate as possible.

1.3.1 Methods of generating diverse classifiers

As several authors claim [45, 63], diversity is a fundamental requirement to achieve good performance ensembles. A formal explanation of why this is so, for the case of ensembles of regresors, was shown in formulas 1.2 to 1.5. Many methods have been used to enforce diversity between the classifiers forming an ensemble, Dietterich and Kuncheva [19, 44] identified four basic approaches to increase diversity in ensembles:

1. Using different combination schemes.
2. Using different base classifiers.
3. Using different feature subsets
4. Using different data sub-sets.

This thesis will follow a taxonomy based on Rokach [63] that tries to cover previous works. The different ensemble construction algorithms are classified, according to the strategies used to enforce diversity, in the following categories:

1. Heterogeneous. Those ensembles that use more than one learning algorithms to construct its members. For example, using an classifier belonging to each of the most popular super-families (combining neural networks with decision trees, SVMs, nearest neighbors, ...).
2. Homogeneous. When all the base classifiers of the ensemble are generated using the same learning algorithm.

(a) Manipulating the training set vertically (that is, using different instances for each base classifier, or modifying the way they are used).

(b) Manipulating the training set horizontally (that is, using different attributes for each base classifier, or modifying the way they are used).

(c) Manipulating the class representation (Using a differant class representation for each base classifier.).

(d) Manipulating the behavior of the learning algorithm (for example, using different values for the parameters of the algorithm).

(e) Hybrid (several of the above strategies are used at the same time).

1.3.1.1 Manipulating the training set vertically

The strategy of manipulating the training set, either vertically or horizontally (Secc. 1.3.1.2) is very useful when the classifiers that compose the ensemble are *unstable*, that is to say, small variations in the training set cause large variations in the trained classifier. An example of unstable classifier are decision trees.

In these methods the diversity is achieved by training each classifier with a different variation of the dataset. This variation is obtained by creating, deleting, or resampling instances from the original data set.

Popular ensemble construction methods such as bagging [7] and boosting [64] belong to this category.

1.3.1.1.1 Resampling and reweighing based methods

A common strategy to achieve diversity is to train each base classifier with a different set of data, however, on many occasions the amount of training data is limited and to split this small number of instances in even smaller different subsets for each classifier, can lead to biased classifiers.

One possible solution to this problem are resampling methods. These methods train each base classifiers with different samples S_t taken from the original dataset S.

The resample extraction may be completely random and with replacement as Bagging. Or it may be pseudo-random giving more probabilities of being in sampling t to those instances more erroneously classified by classifiers 1 to $t-1$, as Boosting.

Some of these methods implement the resampling by reweighing.

Bagging Bagging [7] is the best known ensemble method based on resampling. This method tries to increase the diversity between base classifiers by training each one of them with different samples of the dataset, and also it outperforms the precision of any base classifiers by combining the outputs of all of them. The training set for each classifier is the same size as the original dataset, but obtained from it by resampling with replacement. To classify a new instance each classifier returns its prediction and these predictions are combined using the average of probabilities for each class.

Require: I (an inducer), T (the number of iterations), S (the training set), μ (the subsample size).
Ensure: $M_t; t = 1, \ldots, T$
1: $t \leftarrow 1$
2: **repeat**
3: $S_t \leftarrow$ Sample μ instances from S with replacement.
4: Build classifier M_t using I on S_t
5: $t++$
6: **until** $t > T$

Figure 1.4: Bagging pseudocode. Extracted from [63]

Wagging Wagging [4] is a variant of Bagging. In this case each classifier is not trained with a resampling of the dataset, but is trained using the original dataset where each instance has associated a weight which value has been randomly assigned. Bagging can be considered a particular case of Wagging where the weights follow a Poisson distribution.

AdaBoost Boosting family methods have their origin in 1990, Schapire [64] demonstrated that it is possible to obtain a strong classifier from the combination of weak classifiers[2].

Of all ensemble methods, Adaboost [24] is certainly the one that has received more attention. AdaBoost is the original version, which has been revised successively resulting in new variants: AdaBoost.M1, which is able to deal with multiclass datasets or AdaBoost.R [25] which is able to deal with regression datasets. AdaBoost, unlike Bagging, is a dependent method, this means that a classifier i needs information obtained from the classifier $i-1$ and other previous classifiers. The principal idea of this algorithm is to focus on the most difficult to classify instances, initially,

[2] A weak classifier is a classifier good enough to get a better performance than a totally random classifier

all instances have associated the same weight, but in each iteration, the weight of incorrectly classified instances is increased, while the weight of correctly classified instances is decremented. As a result of this change in the weights, each classifier is forced to pay more attention to instances that have offered more difficulties to previous classifiers. In addition, a weight β_t, based on their performance, is assigned to each classifier, classifiers with better performance are associated with a higher weight. Finally, when classifying a new instance, the weight associated with each classifier is used following formula.

$$H(x) = \arg\max_{y \in \text{dom}(y)} \left(\sum_{t:M_t(x)=y} \log \frac{1}{\beta_t} \right) \tag{1.6}$$

Where β_t is calculated according to the following formula.

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t} \tag{1.7}$$

Being ε_t the resubstitution error [3].

Algorithm AdaBoost.M1
Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.
Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

Figure 1.5: AdaBoost.M1 pseudocode. Extracted from [24]

AdaBoost tends to reduce both the bias and variance terms of error. Though AdaBoost is a very widely used method, it sometimes can not achieve the expected results. According to [60], AdaBoost tends to overfit, AdaBoost tries to build classifiers that improve outcomes for instances

[3] The error rate on the training data

misclassified by previous classifiers, over a large number of iterations, this can cause that the classifiers added to the ensemble to overly focus on the noisy instances and begins to degrade its performance with other instances.

MultiBoost MultiBoost [78] combines AdaBoost with Wagging [4]. AdaBoost and Wagging are techniques with very different properties. AdaBoost tends to minimize both the bias and variance and Wagging only minimizes the variance. MultiBoost assumes that if the underlying mechanisms of these two algorithms are effective, their combination can produce even better method. Multiboost can be seen as Wagging using AdaBoost as base classifier. It is recommended to set the size of subcommittees to $\sqrt{N}$, thus, Multiboost size $N = 100$ is equivalent to Wagging size $N = 10$ whose base classifier is AdaBoost size $N = 10$.

1.3.1.1.2 Methods based on creating new instances

There are two cases in which the methods that provide diversity by subsampling and re-weighting, just as is done in bagging and boosting, do not work properly: *a*) when the dataset is small, *b*) the number of instances of one of the classes is limited. In these cases the best strategy is to add artificial instances to the training set. A method for increasing diversity in small size datasets is DECORATE [54]. In the DECORATE method the first base classifier of the ensemble is constructed using the original dataset, the following classifiers are trained using the original set and artificial data created by randomly picking data points from an approximation of the training data distribution. To increase the diversity, the labels of the artificial data are chosen using probabilities that are inversely proportional to the predictions of the previous classifier. To avoid increasing the error of the ensemble, the base classifiers are checked and discarded when they increase the ensemble total error.

When one class has much less examples than the others (imbalanced datasets see Sec. 1.3.3), specific techniques are used to create artificial instances, such as SMOTE (see sec. 1.3.4.3), this instance creation technique can be combined with resampling methods above mentioned to create ensembles methods that are specially designed to deal with unbalanced sets, such SMOTEBagging and SMOTEBoost (Secc 1.3.4.4).

1.3.1.1.3 Partitioning the space of hypotheses

One can define the partitioning of datasets as the division of the dataset of instances into several smaller datasets. There are several reasons for this:

1. To overcome the limitations of some classifiers to handle large data sets.
2. To Scale and parallelize classifiers.
3. Just to increase the diversity of the ensemble.
4. Balancing each of the partitions.
5. Trying to increase the precision of ensemble

To overcome the limitations of some classifiers to handle large data sets This limitation, described in [59] as one of the reasons to work with ensembles can be overcome by splitting the training set into several partitions. By doing so, ensembles that use this technique are not only able to reduce processor time and memory size needs of each classifier, but also increase the diversity since each classifier is trained with a different data set.

Scale and parallelize classifiers. Partition-based methods are highly parallelizable since the process of building the ensemble can be run across multiple machines, each machine can build one or several base classifiers using datasets of smaller size. Experiments show this approach is fast, accurate, and scalable. [14]

Increasing the diversity of the ensemble. Sometimes a random partitioning can be used as well to increase diversity, this is the strategy followed in Random Oracle [46]. The oracle divides the data set into two parts without regard to the class of the instances or the internal structure of the data set. This division can be carried out in two different ways: split the instances to one side or another of a random hyperplane (Random Linear Oracle), or in and out of a random hypersphere (Random Spherical Oracle). In the training stage, a different classifier is built for each of the two datasets. In the prediction stage, firstly, it is determined which partition the instance belongs to, and finally it is returned only the prediction of the classifier associated with that partition.

Balance each of the partitions. In [55] it is explored the idea of turning a dataset where one class (*negative class*) is much larger than the other (*positive class*) in several sets of balanced data. The dataset S, consisting of S_N and S_P, where S_N is M times larger than S_P, is divided into M balanced sets. The division of negative instances in groups can be done by clustering or just randomly.

Trying to increase the accuracy of the base classifiers. Sometimes it may be convenient to divide the set of hypotheses, so that each member of the ensemble explore different parts of the search space. Doing this has two effects, on the one hand, the problem is simplified, resulting in minor errors individual for each of base classifiers, on the other hand, there is an increase in diversity

In [57] it is described the method Mixture of Experts. In this method the instances in the training set are divided into several subspace and each expert (classifier) assigned to a different subspace. The subspaces of this method have "soft borders" since overlapping is allowed between neighboring subspaces.

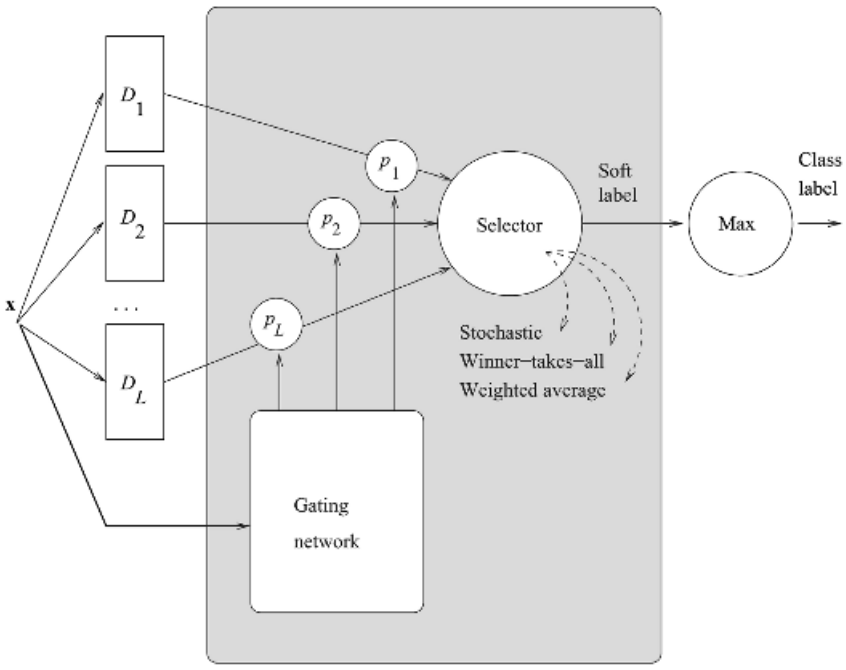


Figure 1.6: How the algorithm mixture of experts works. Extracted from [45]

1.3.1.2 Manipulating the training set horizontally

The other major family of techniques to increase the diversity by manipulating the training set is the family encompassing methods that transforms attributes. This time the number of instances does not change (instances are not created, or deleted, nor resampled), what changes is the way in which each instance is represented.

1.3.1.2.1 Adding new attributes

One way to introduce diversity in an ensemble is to add new attributes whose values are different in each of the ensemble members. *Disturbing Neighbours* [51] uses N randomly selected instances, the disturbing neighbours, to train a k-NN classifier, then it creates N binary attributes for each instance (with value 1 if the corresponding disturbing neighbour is the closest to the instance, 0 otherwise) and an additional attribute whose value is the class predicted by the k-NN classifier, hence, the feature space is expanded with N+1 new attributes.

1.3.1.2.2 Assign weights to the attributes

It is possible to introduce diversity by giving a different importance/weight to every attribute for each member of the ensemble. Random Subspaces, presented by Ho in [38], is a special case of this approach. A set of binary random weights is used, where a weight of value 0 means that the attribute is not included in the subset for the respective ensemble member. The result is that different classifiers are constructed using different subsets of the attributes. Ho proved that the diversity gain compensates for the loss of precision in each individual classifier. This technique has the additional advantage of accelerating training and facilitating the creation of classifiers for large data sets.

The Random Forest ensemble [8] is a bagging ensemble with random trees. A random tree differs from the standard tree only in the number of attributes that are considered during its training. In random trees, a random subset of attributes is considered for the splitting of each node. This can be seen as a variation of Random Subspaces, but instead of using the same subset for the whole tree, the set is different in each node evaluated in the construction of the tree.

Proposed more recently, the Random Feature Weights ensemble [53] associates a vector of weights with each tree of the ensemble. This vector is used to modify the way in which the merit function of the attributes is calculated. For a training set D and a weight vector $\mathbf{w}$, the new merit function for attribute a_i is defined as $f_{\mathbf{w}}(a_i, D) = w_i f(a_i, D)$, where $f(a_i, D)$ denotes the original merit function of a_i for D. Thus, the method introduces a bias which favours the selection and use of attributes with higher associated weight. The vector of weights is randomly created for each tree in the ensemble. These weights are real-valued (not just 0 and 1, as in Random Subspaces and Random Forest) so it is possible to draw a parallel with Wagging, but using features instead of instances.

The last two examples, can also be included in the family of methods which manipulate the learning algorithm (Secc 1.3.1.4)

1.3.1.2.3 Projections

These are frequently used to reduce the dimensionality of the data. In the context of ensemble learning, projecting can be construed as a diversifying heuristic. A well known example is Rotation Forest [62], an ensemble method for decision trees which uses principal component analysis (PCA) to project different groups of attributes for each base classifier. A similar approach, but using supervised projections, is Boosting Projections [28, 31], this method follows the philosophy of boosting, focusing on difficult instances, to do that a supervised projection is obtained using only the misclassified instances at each iteration. The next classifier is trained using all available examples, in the space given by the supervised projection. Random Projections have been used to provide an extra diversity and embed the original set into a space of lower dimension [65]. Random Subspaces can be also seen as a special case of Random Projections.

1.3.1.3 Manipulating the class representation.

The last category of techniques that manipulate the data set comprises those methods that manipulate the class representation. Although in this case the manipulation of data is not performed to increase diversity, but to overcome a limitation of the base classifiers.

Many learning algorithms are designed to solve binary classification problems. By changing the representation of the class, it is possible to get a multiclass ensemble consisting of binary classifiers.

The strategy is to use a different encoding of the class for each base classifier of the ensemble. To do that, a matrix M is created that is formed of k rows and t columns, where k is the number of classes, and t is the number of base classifiers in the ensemble. There are basically two alternatives to fill this matrix, and the resulting number of classifiers depends on the alternative used.

- 1-1 decomposition: With this scheme $k(k-1)/2$ binary classifiers, each binary classifier solves a problem consisting of distinguishing between a pair of classes i and j.
- 1-All decomposition: This time k binary classifiers are requiered (as many classifiers as classes), each trained to distinguish class i from the rest.

After this, the final output of the ensemble is the class in which most of the base classifiers agree.

1.3.1.3.1 Error Correcting Output Codes

Error Correcting Output Codes (ECOC) originally emerged in information theory to correct the bit inversions produced during transmission through a noisy communication lines. In data mining, these codes have been used to transform binary classifiers into multiclass classifiers [20]. The idea behind ECOC is that every word differs from the most similar word to it at least in the Hamming distance. The Hamming distance [35] is used for data transmission, if two possible words formed by binary characters differ by a distance d of bits, then p errors in the transmission can be corrected at least, being $d \geq 2p+1$. In ECOC based ensembles, a class is represented by a binary word of length T, and two classes differ at least in the Hamming distance. In this method a matrix M is also created formed of k rows and t columns, where k is the number of classes, and t is the number of base classifiers in the ensemble. In the most common strategy t is $2^{k-1}-1$. So with four classes seven classifiers are needed. four words of size seven are generated so that, the Hamming distance between each of the words is four (that is, $\lfloor T/2 \rfloor + 1$).

When a new instance is classified, that instance is predicted using the T classifiers, each of which generates a vector of size T, the vector is compared with the one which describe each class and class whose distance is lower is returned as prediction.

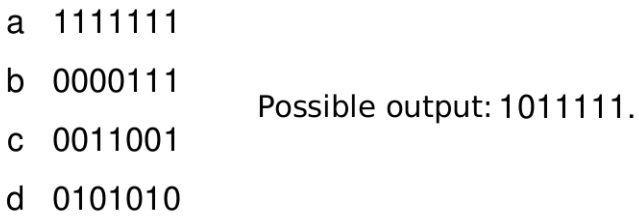


Figure 1.7: Example of how ECOC works

1.3.1.4 Manipulating the behavior of the learning algorithm.

1.3.1.4.1 Manipulate the parameters that regulate the functioning of the classifier.

Some learning algorithms such as neural networks or SVM are very sensitive to the fine tunning of its parameters. This means that the same classification algorithm trained with the same training set and evaluated using the same test set can obtain very different results depending on the values of the parameters that configure the method. A simple way to create diverse ensemble of classifiers is to use different parameter values in each of the members of the ensemble. For example in [72] this strategy is studied in ensembles of SVMs, this paper conduct an analysis of the relationships between bias, variance, kernel type and its parameters.

1.3.1.4.2 Injecting randomness inside the learning algorithm.

The strategy of injecting randomness has generally been applied to classification trees. Classification trees are one of the preferred algorithms to be used as a base classifier in ensembles, because they are unstable, that means that small changes in the training set or in the construction method will produce very different models, this instability makes it easy to incorporate diversity within the tree construction algorithm.

In [19], a simple technique randomizes the selection of one of the twenty best splits in each node. Random Forest [8] increases diversity when it combines training set sampling with random selection of subsets of attributes at each tree node; thus, the splits are only considered in each node within the selected subset of attributes. Extremely randomized trees [33] take k random attributes into account at each node, as in Random Forest, but the splitting point in each attribute is also randomly chosen. The Random Feature Weights method [52] makes use of all attributes, each with a differing probability of being used as a split. These probabilities are subject to the weight that is assigned to each attribute. These weights

are randomly generated and are different between trees, so diversity is the ensemble is ensured.

This thesis presents two new method for creating diverse decision trees called GRASP Forest and GAR-Forest. GRASP Forest method have been used in clasification trees (see Chapter 2) and regresion trees (see Chapter 3). The Grasp Forest method was expanded and improved resulting in GAR-Forest (GAR means GRASP with Annealed Randomness).

1.3.1.5 Hybrid methods

It is quite common that ensemble methods simultaneously use more than one of the above techniques. A clear example is Random Forest, which combines resampling with injecting randomness in the learning algorithm. Or RotBoost [83] which combines projections with weigthed resampling.

Hybrid methods are particularly common when working with unbalanced sets because imbalance learning ensembles usually combine balancing techniques (artificial instance creation, undersampling, ...) with bias reduction techniques (RUSBoost [67] and SMOTEBoost [16]) or variance reduction techniques (SMOTEBagging [75] and UnderBagging).

1.3.2 Methods of combining multiple classifiers

The generation of classifiers that are accurate and diverse is only the first part of building and ensemble. The second, but not less important, is the method of obtaining the ensemble outputs by combining the outputs of the base classifiers. In [59] the methods of combining multiple classifiers are classified into two categories according to how are the combination rules:

- Methods without learned combination rules.
- Methods in which the combination rules are learned.

1.3.2.1 Methods without learned combination rules.

In these methods it is assumed that only the predictions of the base classifiers are available to perform the final prediction.

Some ways to combine the outputs of the base classifiers are:

- Majority voting. The predicted class is the one with the most votes among base classifiers
- Average of probabilities. Predictions for each class are the average of the predictions of each of the base classifiers.
- Product of probabilities.
- Weighted average. A weight is assigned to each base classifier of the ensemble that determines how much the classifier contributes to the average. This weight may be proportional to its accuracy.

1.3.2.2 Methods in which the combination rules are learned.

The methods that learn the combination rules are often called meta-classifiers. These methods are particularly useful when the base classifiers do not have the same success in classifying instances (failed instances for each classifier are very different between different classifiers). This happens when the base classifiers are generated using different methods; or when generated with the same method but with very different parameters.

1.3.2.2.1 Stacking

When decision boundaries of the base classifiers are very different, the problem of combining the results arises. Stacked generalization [81] solves this problem by building a classifier that takes as inputs the outputs of the base classifiers and learn to map the outputs of the base classifiers with the correct output. In other words, to combine the predictions of the base classifiers, there is not voting, instead a "meta" classifier is used.

The base classifiers are trained with the training set and the meta classifier is trained with the predictions of the base classifiers. The predictions of the classifiers are obtained from a different partition set than the one used for training. This is achieved by dividing the training set into several partitions (see figure 1.8).

1.3.2.2.2 Grading

The Grading method [66] is very similar to Stacking. This time, the meta-classifier, rather than being trained with the set of predictions returned by the base classifiers, it is trained with a dataset consisting of as many rows

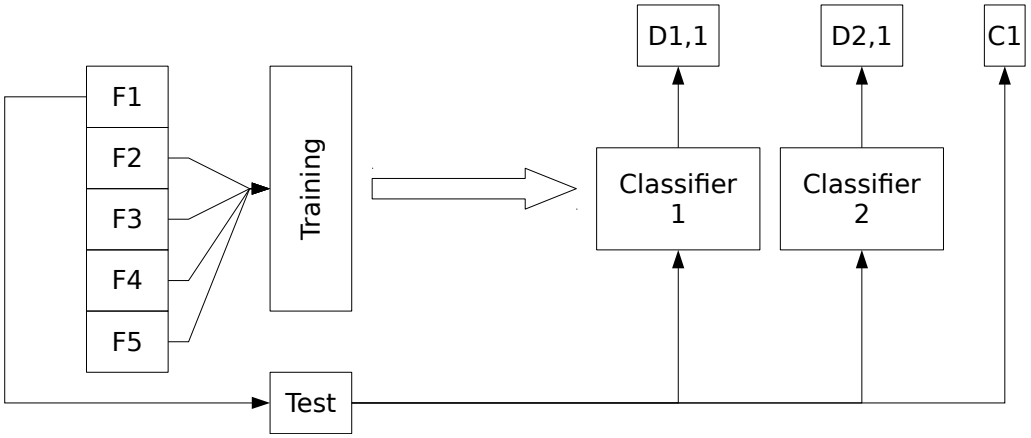


Figure 1.8: The training set is divided into several partitions. Iteratively, one partition is chosen to obtain the predictions and the rest to train the base classifiers. (ie. with the classifiers trained using partitions from F2 to F5, the predictions of partition F1 are obtained.). By repeating this process several times the training set of the meta classifier is built.

as instances and as many columns as base classifiers. In each pair row_i-column_j, the cell contains a 1 (+) if the instance_i was successful classified by the classifier_j or 0 (−) otherwise (see figure 1.9).

Attributes	Class
$x_{11}\ \ldots\ x_{1n_a}$	t
$x_{21}\ \ldots\ x_{2n_a}$	f
$\ldots\ \ldots\ \ldots$	$\ldots$
$x_{n_e1}\ \ldots\ x_{n_en_a}$	t

(a) training set

C_1	C_2	$\ldots$	C_{n_c}
t	t	$\ldots$	f
f	t	$\ldots$	t
$\ldots$	$\ldots$	$\ldots$	$\ldots$
f	f	$\ldots$	t

(b) predictions

C_1	C_2	$\ldots$	C_{n_c}
$+$	$+$	$\ldots$	$-$
$+$	$-$	$\ldots$	$-$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$-$	$-$	$\ldots$	$+$

(c) "graded" predictions

C_1	C_2	$\ldots$	C_{n_c}	Cl
t	t	$\ldots$	f	t
f	t	$\ldots$	t	f
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
f	f	$\ldots$	t	t

(d) t. s. for stacking

Attributes	Cl
$x_{11}\ \ldots\ x_{1n_a}$	$+$
$x_{21}\ \ldots\ x_{2n_a}$	$+$
$\ldots\ \ldots\ \ldots$	$\ldots$
$x_{n_e1}\ \ldots\ x_{n_en_a}$	$-$

$\ldots$

Attributes	Cl
$x_{11}\ \ldots\ x_{1n_a}$	$-$
$x_{21}\ \ldots\ x_{2n_a}$	$-$
$\ldots\ \ldots\ \ldots$	$\ldots$
$x_{n_e1}\ \ldots\ x_{n_en_a}$	$+$

(e) n_c training sets for grading

Figure 1.9: Grading table and comparison with Stacking (figure from [66])

1.3.3 Introduction to imbalanced learning

In classification, the imbalance problem occurs when some classes contains many more cases than others[13].

Many real-life problems can be described as imbalanced. The areas in which this problem has been studied more deeply are:

- Bioinformatics: translation initiation site (TIS) recognition in DNA

sequences [30], gene recognition [3], mining cancer gene expression [82].

- Monitoring of industrial processes: non-destructive testing in weld flaws detection through visual inspection [47, 21], tools breakage detection [12].
- Finance: predicting credit card customer churn [1], fraud detection [58, 18], risk predictions in credit data [27].
- Security: spam detection [32, 69].
- Software engineering: software defect detection [76, 50, 48].

Classification of imbalanced data is hard and standard learning algorithms usually perform badly on them because standard classifiers are driven by accuracy, hence the minority class may simply be ignored [74], moreover standard classification methods operate under the assumption that the data sample is an accurate representation of the population of interest, which is not always the case with imbalance problems and finally standard classification can not handle different errors costs coming from different classes.

According to [49] there are at least six problems related to imbalance datasets that contribute to make the problem harder:

1. overlapping [70].
2. lack of density and information [77].
3. noisy examples [9].
4. small disjuncts [79].
5. the significance of borderline instances to discriminate between positive and negative classes [56].
6. differences in the data distributions between training and test stages [61].

Because the difference between the class sizes, accuracy is not an appropriate measure to assess performance in unbalanced datasets. Commonly used measures of performance for imbalanced data are the Area Under the ROC (Receiver Operation Characteristic) curve [22], the F-Measure [73] and the Geometric Mean [43] (see section 1.4.1).

1.3.4 Classification methods for imbalanced problems

In recent years, numerous techniques have been developed to deal with the problem of class-imbalance datasets. This section is a sort summary of the subset of methods tested in the articles of the thesis that address the problem of imbalance, longer summaries can be found in these articles (see chapters 5 and 6).

The methods of this summary are organized using the following classification. We can consider to approaches:

Data level approaches: the balanced is achieved through the modification of the data distribution, this methods are: random undersampling, oversampling, SMOTE.

Ensemble learning: methods designed to deal directly with imbalanced datasets, such as, modifications of bagging: SMOTEBagging; modifications of AdaBoost: SMOTEBoost, RUSBoost; methods based on partitioning, and multiobjective methods.

1.3.4.1 Random Undersampling.

This technique will randomly drop some of the examples of the majority class. When it comes to sampling without replacement, an example of the minority can appear only once in the sub-sampling; with replacement, the same example can appear multiple times.

1.3.4.2 Oversampling

Oversampling [2] consists on adding exact copies of some minority class examples. With this technique overfitting is more common than in the prior technique.

1.3.4.3 SMOTE

Synthetic Minority Over-sampling Technique [15] creates new examples using the following procedure: a member of the minority class is selected and its k nearest neighbours (from the minority class) are identified. One of them is randomly selected. Then, the new example added to the set is a random point in the line segment defined by the member and its neighbour. This method tries to avoid overfitting using a random procedure to create the new samples, but this can introduce noise or nonsensical samples.

1.3.4.4 SMOTEBagging

SMOTEBagging [75], combines Bagging with different amounts of SMOTE and Oversampling in each iteration. The resulting dataset is completely balanced and consists of three parts: *i*) a sample with replacement of the majority class, keeping the original size; *ii*) oversampling of the minority class; and *iii*) SMOTE of the minority class. The amounts of oversampling and SMOTE are not equal in all ensemble members: the oversampling percentage varies in each iteration (ranging from 10% in the first iteration to 100% in the last.) The rest of the positive instances are generated by the SMOTE algorithm.

1.3.4.5 SMOTEBoost and RUSBoost

SMOTEBoost [16] and RUSBoost [67] are both modifications of AdaBoost.M2 [26], in each iteration, besides the instance reweighting done according to the algorithm Adaboost.M2, SMOTE or Random undersampling are applied to the training set of the base classifier. Ensembles based on boosting tend to perform better than bagging based ensembles, however their base classifiers are more sensitive to noise (they focus on previously failed instances that can be noise), this disadvantage is attenuated in SMOTEBoost and RUSBoost because they introduce a high degree of randomness by creating or deleting instances.

1.3.4.6 Partitioning

Although the most popular methods are modifications or variations of bagging or boosting, there are methods that do not do resampling, oversampling or undersampling and instead of that, they make partitions.

Partitioning [55] is similar to undersampling based ensembles, it breaks the majority class into several disjoint partitions and constructs several models each using one partition from the majority class and the entire minority class.

1.3.4.6.1 Multiobjective Methods

Most of the above methods, at the same time that increase accuracy in minority class, they decrease overall accuracy compared to traditional learning algorithms. Some approaches combine both types of classifiers,

one trained with the original skewed data and other trained according to one of the previous approaches in an attempt to cope with the imbalance.

Reliability Based Classifier [68] trains two classifiers and then chooses between the output of the classifier trained on the original skewed distribution and the output of the classifier trained according to a learning method addressing the curse of imbalanced data. This decision is guided by a parameter whose value maximizes, on a validation set, the accuracy and a measure designed to evaluate the performance of a classifier in imbalanced classifiers, such as the geometric mean.

The preprocessing methods used for dealing with the problem of imbalance such as Undersampling or SMOTE have an issue: the optimum value of the parameters of these methods are problem dependent and are often difficult to find.

Chapter 5 of this thesis presents a method called "Random Balance" that overcomes this problem and increases the diversity of the ensemble by randomly changing the class proportions.

Most literature about imbalanced learning addresses the problem of imbalance using reweighing, oversampling, undersampling and other techniques that affect the ratio between classes. Chapter 6 of this thesis presents a study of the effect on diversity techniques (Random Oracles, Random Feature Weights, Rotation Forest and Disturbing Neighbours) in imbalanced learning.

1.4 Experimental methodology

This section will present the basic fundamentals of the experimental methodology: what measures are evaluated 1.4.1, how to measure 1.4.2, and what are the statistical test used to compare different models 1.4.3.

1.4.1 Performance measures

Basically when evaluating a classifier what is measured is the rate of success/error o accuracy.

$$\text{accuracy} = \frac{\text{number of instances correctly labeled}}{\text{number of instances}} \tag{1.8}$$

But it is not always appropriate to use the accuracy to compare methods. Sometimes the cost of making a mistake is not the same, but depends on the class, for example, financial fraud, errors in medical diagnosis (the cost of making an unnecessary test is less than the cost to discharge a patient with a dangerous disease). On other occasions due to the data distribution, the accuracy is not useful, for example, let us imagine an industrial process that produces a defective product per 1000 correct products, a defect prediction system that always predicts that the product is correct would have a success rate of 99.9%, it would be very accurate, but not useful.

In examples of this nature, binary and unbalanced datasets, it is more appropriate to evaluate and measure the performance of classifiers using ROC analysis [22], the F measure [73] and the Geometric Mean [43].

To understand these measures, it is necessary to explain some basic concepts: given a test dataset for a 2-class imbalanced problem, containing P examples of the positive (minority) class and N examples of the negative (majority) class. There are four posible outcomes: that the instance is positive and it is predicted as positive, *true positive*, that the instance is negative and it is predicted as positive, *false positive*, that the instance is negative and it is predicted as negative, *true negative*, or is positive and it is predicted as negative, *false negative*.

One way to visualize these outcomes is using a tabular representation called confusion matrix, shown in Table 6.3.

Table 1.1: Confusion matrix in binary problems

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

The True Positive Rate (TPR), also named Sensitivity or Recall in some fields, is defined as TP/P, and False Positive Rate (FPR) is defined as FP/N. The precision is defined as $TP/(TP+FP)$.

Using these previous measures it is posible to define the F-Measure as

$$FMeasure = 2 \times \frac{precision \times recall}{precision + recall}$$

The Geometric Mean is defined as

$$GMean = \sqrt{TP/P \times TN/N}$$

.

The ROC Curve is obtained from the probabilities assigned to the instances by the classifier, each probability threshold gives a TPR and FPR that defines a point in the curve. The Area Under the ROC curve (AUC) is a way to represent the performance of a binary classifier using a scalar.

The ROC curve is a graphical representation of the performance of classifiers. In this representation:

- a classifier that always predicts negative, will be represented by a single point in (0,0).
- a classifier that always predicts positive, will be represented by a single point in (1,1).
- a perfect classifier will be represented by a point of coordinates (0,1).
- a Random classifier will be represented by a diagonal line.

A classifier is better than others if its corresponding point is on the top left corner. The best possible area under the curve is 1 and the worst is 0.5

1.4.2 Training and test datasets

The proper evaluation of a model requires the use of a dataset totally independent of the dataset used for training. The reason of this is that if we use the same dataset used for training, or with instances in common, the estimation of the performance of the model would be too optimistic. We are not seeking models with just a good behaviour on the training set, we want more than that, we want the model to be able to generalize, that is, a model with good behaviour on instances that have not been used in training. That is why first a dataset is used for training the model. The larger this dataset, the better the model. And then, a different dataset is used for evaluating the model. The larger this dataset is, the more accurate is the estimation of the model performance. So in practice, the available data is used to satisfy these two objectives, in a way that the data used for training is not later used for evaluation.

The *k-fold cross-validation* is a process that avoids overlapping between training and test datasets, the procedure is as follows:

1. The data is divided into k subsets, folds, of the same size.
2. Successively each fold is used as test and the rest is used for training.

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

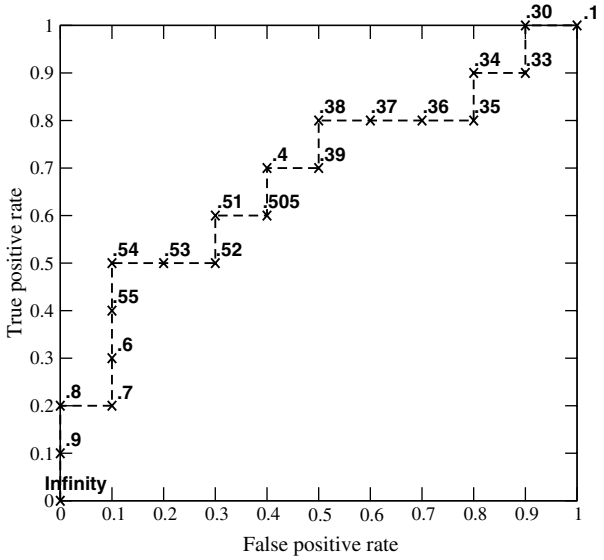


Figure 1.10: ROC Curve. To create a ROC curve is necessary to order the instances based on the probability returned by the classifier, considering that values close to 1 means that the instance is predicted as positive and vice versa. Starting with the instances with highest probabilities, every time the instance is correctly predicted a segment is drawn upward, and every time the prediction fails, it is drawn to the left. [22]

Often, the folds are obtained in such a way that the proportion of classes follows the proportion in the full dataset, that is called *stratified cross-validation*.

In the experiments presented in this thesis, the configuration used is 5×2, the data set is halved in two folds. One fold is used for training and the other for testing, and then the roles of the folds are reversed. This process is repeated five times, with different folds each time.. The results are the averages of these ten experiments. Cross validation was stratified: the class proportions was approximately preserved for each fold.

Table 1.2: Class sizes for certain datasets in the Keel repository [4]

Name	Size Minority	Size Majority
keel_shuttle-c2-vs-c4	6	123
keel_ecoli-0-1-3-7_vs_2-6	7	274
keel_glass5	9	205
keel_glass-0-6_vs_5	9	99
keel_glass-0-4_vs_5	9	83
keel_glass-0-1-6_vs_5	9	175

As the datasets we are dealing with are imbalanced, the use of 5-fold stratified cross validation (another configuration commonly used in the literature) will make the measures calculated in each fold too unreliable when there are very few instances in the minority class (some of them with only 6 instances in the minority class, (see Table 1.2), with 5-fold stratified cross validation a minority class in a fold will have only one or two instances, the value of the performance measure could be minimum or maximum depending on a single instance). By using 2-fold stratified cross validation we alleviate this, since in each fold half of the instances of the minority class are present instead of only a fifth.

1.4.3 Method comparison

There are several types of methods for comparing classifiers, these methods can be arranged depending on the number of models to be evaluated and the number of datasets used for the comparison. The comparison of classifiers is a huge area of study and this section will only briefly describe those statistical tests used in the experiments of this thesis.

- two models.
  - Student's t-test.
  - Sign test.
- More than two models, several sets.
  - Friedman test.
  - Iman Daveport test.
  - One versus the rest. Bounferroni-Dunn.
  - One versus the rest. Hochberg.

1.4.3.1 Student's t-test

To compare how accurate are two models on a certain dataset, a evaluation using cross-validation may be sufficient. However, to demonstrate convincingly that a model A works significally better than a model B in a particular dataset we must use the Student's t-Test.

The Student's t-Test indicates whether the means of two samples are significantly different or else the differences can be merely due to chance.

When for every observation of a group there is associated one observation in the other group, is called the paired t-test. This is what happens when estimates are obtained by cross-validation.

In this test, the null hypothesis is that the difference between averages is zero. When the the mean of the differences is not consistent with an average equal to zero the hypothesis is rejected.

1.4.3.2 Comparison of 2 models on several datasets

According to [17], when comparing various classifiers on different datasets, new problems arise, that did not existed when using only one dataset. Although it is possible to perform a paired t test with the results of each dataset, the results for each dataset are not comparable with each other. There are specific tests to evaluate this type of experiments.

The signs test requires simply to count for how many data sets a model wins the other. If the two models are equivalent, each of them should win in half of the datasets. A classifier is significantly better than the other if it, has at least the number of wins (ties count as half a win) that appears on Table 1.3. For example, for 20 data sets and considering a confidence $\alpha = 0.10$, one method is significally better than the other, if it wins at least in 14 datasets.

datasets	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
α =0.05	5	6	7	7	8	9	9	10	10	11	12	12	13	13	14	15	15	16	17	18	18
α =0.10	5	6	6	7	7	8	9	9	10	10	11	12	12	13	13	14	14	15	16	16	17

Table 1.3: Number of wins that has to reach a model to be significantly superior than the other

1.4.3.3 Comparison of various models over multiple datasets

Average ranks is a method that computes a rank for each model, taking into account the scores on all datasets and the scores of other methods. The procedure is as follows:

1. Scores are obtained for each data set and model.
2. Using the scores, models are ordered from best to worst.
3. Each model recieves a number, its rank, according to its position. The best receives a 1, the second receives a rank of 2, and so on.
4. When there is a tie, the ranks are shared out. For example, if the top three models for a given dataset tied, each one of them would receive rank $(1+2+3)/3 = 2$ for this dataset.
5. For each classifier the average rank for all datasets is calculated.
6. Classifiers are sorted using that average rank.
7. From these rankings one can perform various statistical tests.

Table 1.4 shows an example.

	Scores (AUC)				Ranks			
	C4.5	C4.5+m	C4.5+cf	C4.5+m+cf	C4.5	C4.5+m	C4.5+cf	C4.5+m+cf
adult	0.763	0.768	0.771	0.798	4.0	3.0	2.0	1.0
breast cancer	0.599	0.591	0.590	0.569	1.0	2.0	3.0	4.0
cancer wisconsin	0.954	0.971	0.968	0.967	4.0	1.0	2.0	3.0
cmc	0.628	0.661	0.654	0.657	4.0	1.0	3.0	2.0
ionosphere	0.882	0.888	0.886	0.898	4.0	2.0	3.0	1.0
iris	0.936	0.931	0.916	0.931	1.0	2.5	4.0	2.5
liver disorders	0.661	0.668	0.609	0.685	3.0	2.0	4.0	1.0
lung cancer	0.583	0.583	0.563	0.625	2.5	2.5	4.0	1.0
lymphography	0.775	0.838	0.866	0.875	4.0	3.0	2.0	1.0
mushroom	1.000	1.000	1.000	1.000	2.5	2.5	2.5	2.5
primary tumor	0.940	0.962	0.965	0.962	4.0	2.5	1.0	2.5
rheum	0.619	0.666	0.614	0.669	3.0	2.0	4.0	1.0
voting	0.972	0.981	0.975	0.975	4.0	1.0	2.0	3.0
wine	0.957	0.978	0.946	0.970	3.0	1.0	4.0	2.0
				Average Rank:	3.143	2.000	2.893	1.964

Table 1.4: Average Ranks example. The best classifier according to average ranks is C4.5+m+cf. Example extracted from [17]

1.4.3.3.1 Friedman's Test and Iman Davenport's Test

The null hypothesis is that if the models are equivalent, their average ranks should be equivalent. Friedman statistic is as follows:

$$\chi_F^2 = \frac{12N}{k(k+1))}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{1.9}$$

Where N is the number of datasets, k is the number of models, R_j the average ranks of model j.

According to Iman and Davenport [41], this test is too conservative and they establish an alternative statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{1.10}$$

If the null hypothesis is rejected, that is, if the models are not equivalent,It is possible to use a "post-hoc" test to detect differences between models.

1.4.3.3.2 Bonferroni-Dunn's Test and Hochberg's Test

These methods are used to compare one model against an other, usually the best model according to the average rank of all the models.

A model is significally bether than other if the difference between their average ranks are at least:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} \tag{1.11}$$

Where q_α is the confidence level, for example, 90% and 95%.

There are many other methods that are more powerful than Bonferroni-Dunn, this means that can detect more significant differences between methods. One of this methods is the Hochberg procedure [39]. The Hochberg procedure is more powerful, but require the hypotheses to be independent, that is the compared methods must be different, not same methods with different parameters.

1.5 Applications

Part of the work done in this thesis is the application of Data Mining methods to solve real problems. The selected problems were of industrial processes of manufacturing and defect inspection.

The manufacturing problems can be solved by dividing them into different parts each of them likely to be posed as a problem that can be solved with a Data Mining technique: for example, to determine the values of the process' parameters that will yield the desired product quality or maximize manufacturing system performance using the available resources [5].

The inspection of defects is a very important task to ensure the quality of industrial processes. Human inspection is not as consistent as an automated process since it is a process that requires high concentration, besides it is affected by the occurrence of fatigue, different levels of skill, experience or ways of working of each operator. From a Machine Learning point of view a defect detection system is a binary classification problem being the classes 'defect', or 'not defect'.

Throughout the development of this thesis we have studied these two processes. It has been studied the use of ensemble learning technique called "Boosting projections" to improve the surface roughness prediction in high-torque milling operations (chapter 7). And it has been applied imbalance learning ensembles to the novel problem of detecting defects in magnesium alloys castings. The latter problem has been addressed in two ways, the first of which was by classifying pixels in the image as belonging to a defect or not, using the technique of sliding window (chapter 8); and the second by classifying regions of interest (ROIs) in defects or no-defects (chapter 9).

1.6 Motivations and objectives

After describing briefly some theoretical concepts in the previous sections, the problems that have motivated the development of this thesis will be outlined.

- As seen in Section 1.3.1, the diversity among base classifiers is essential to get accurate ensembles. But diversity has to be introduced without losing much accuracy in the base classifiers. Many heuristics have been used to increase the diversity. In this thesis GRASP heuristic is used as the inspiration for a new method to create diverse ensembles of trees with a controlled degree of randomness. This is addressed in Chapters 2 and 3.

- Delving deep into the previous idea, it is going to be studied the possibility of building decision trees that are accurate and have a great

source of randomness at the same time. The motivation is an intuitive idea: since the prediction of the tree is calculated at leaves, nodes closer to the leaves are the more influential on the final prediction of the tree, while nodes that affect the overall structure of the tree (and hence diversity) are the root and the upper nodes. So it is going to investigated the idea of building trees that are random at the root and deterministic in the leaves, with a level of randomness that decreases as the tree is built. This is addressed in Chapter 4.

- When Data Mining is applied to real world dataset, sometimes the problem of imbalance arises, the size of a class is much larger than the other. There are several strategies for dealing with imbalanced datasets, one is to reduce the size of the majority class and another to increase the size of the minority class. It happens that it is difficult to find out which is the optimal strategy for a particular dataset. This motivates an interesting line of research: to study whether it is possible or not to rely on randomness and repetition to address this problem. The goal is to eliminate the need to choose the proper technique and adjust their optimal parameters. The results and conclusions of this research line are given in Chapter 5.

- Continuing with unbalanced learning, most of the techniques used in ensembles for imbalanced learning are approaches based on modify the dataset "vertically", by bootstrapping, reweighed, oversampling or undersampling; not only to deal with the imbalance but to introduce diversity. There has not been so far a study on the influence of the techniques that modify the dataset "horizontally" affecting attributes rather than instances. Nor about the combination of both approaches. This study is conducted in Chapter 6.

- One of the practical areas where it is possible to apply ensembles of classifiers is in the prediction in industrial processes. Often these problems are commonly addressed using neural networks, genetic algorithms and fuzzy logic, there are not many studies using ensembles. It will be studied the performance of different ensemble algorithms in a problem of predicting the final quality of an industrial process (Chapter 7).

- Another practical area is defect inspection. Thanks to Grupo Antolin

Irausa, SA, we have access to a repository of X-ray images of magnesium alloys pieces. Magnesium is a very light material but may present internal porosity. Since the portions of the piece containing pores are much smaller than those that are free of these defects, this is a problem of unbalanced classification and may be interesting to try to solve this problem using imbalanced learning ensembles. This is studied in Chapters 8 and 9.

1.7 Discussion of results

The research conducted during the course of this thesis has led to the publication of several articles in conferences and journals. This section provides a summary of the results of each. The list of articles published as research results is:

1. GRASP Forest: A New Ensemble Method for Trees.

   Authors José-Francisco Díez-Pastor, César García-Osorio, Juan José Rodríguez, Andrés Bustillo

   Type Conference

   Published in 10th International Workshop on. Multiple Classifier Systems (MCS 2011), pages 66-75.

   Year 2011

2. GRASP Forest for regression: GRASP Metaheuristic Applied to the Construction of Ensembles of Regression Trees.

   Authors José-Francisco Díez-Pastor, César García-Osorio, Juan José Rodríguez.

   Type Conference

   Published in XIV Conferencia de la Asociación Española para la Inteligencia Artificial. (CAEPIA 2011).

   Year 2011

3. Tree ensemble construction using a GRASP-based heuristic and annealed randomness.

   Authors José-Francisco Díez-Pastor, César García-Osorio, Juan J. Rodríguez

Type Journal

Published in Information Fusion 20: 189-202

Year 2014

4. Random Balance: Ensembles of Variable Priors Classifiers for Imbalanced Data

   Authors Jose F Diez-Pastor; Juan J. Rodriguez; César I Garcia-Osorio; Ludmila I Kuncheva

   Type Journal

   Published in Knowledge-Based Systems. (In press)

   Year 2015

5. Diversity techniques improve the performance of the best imbalance learning ensembles

   Authors Jose F Diez-Pastor; Juan J. Rodriguez; César I Garcia-Osorio; Ludmila I Kuncheva

   Type Journal

   Published in Information Sciences

   Year Review (Minor changes)

6. Boosting Projections to improve surface roughness prediction in high-torque milling operations

   Authors José-Francisco Díez-Pastor, Andrés Bustillo, Guillem Quintana, César García-Osorio

   Type Journal

   Published in Soft Computing. 16(8): 1427-1437

   Year 2012

7. Imbalanced Learning Ensembles for Defect Detection in X-Ray Images.

   Authors José-Francisco Díez-Pastor, César García-Osorio, Víctor Barbero-García, Alan Blanco-Álamo

   Type Conference

Published in The 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems. (IEA/AIE 2013), pages 654-663.

Year 2013

8. Segmentación de defectos en piezas de fundido usando umbrales adaptativos y ensembles (Segmentation of defects in castings using adaptive thresholds and ensembles).

   Authors José-Francisco Díez-Pastor, Álvar Arnaiz-González, César García-Osorio, Juan J. Rodriguez

   Type Conference

   Published in XVII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2014), pages 345-349

   Year 2014

1.7.1 GRASP Forest: A New Ensemble Method for Trees

This conference paper proposes a new technique for constructing ensembles of decision trees called GRASP Forest. This technique seeks to increase the diversity of the ensemble while controlling the performance of base classifiers. It is based on the generative step of GRASP and is similar to Random Forest. While Random Forest choose the best attribute from a randomly selected subset of attributes, GRASP Forest creates a subset of selected good attributes candidates and then the attribute is randomly chosen. This heuristic is also used to select the split value for each numeric attribute.

The highlights of the paper are:

- The GRASP metaheuristic and its application to tree construction is described. The resulting method is based on J48 (Java version of C4.5).
- The level of randomness is controlled using a parameter α. A parameter that varies from 0 (totally random) to 1 (totally deterministic).
- An experimental study is performed comparing our method with Bagging, AdaBoost.M1, Multiboost, Random Subspaces and Random Forest over 62 datasets from the UCI repository.

  - Average ranks were used for comparing multiple methods.
  - The proposed method outperforms the rest of ensemble methods.
  - Binary trees outperforms non-binary trees for almost all of the methods and variants tested.
  - The best performance of the proposed method is obtained when α is 0.2 or 0.3.

1.7.2 GRASP Forest for regression: GRASP metaheuristic applied to the construction of ensembles of regression trees

In this conference paper the idea of the previous paper is adapted to regression tasks.

The highlights of the paper are:

- GRASP metaheuristic is applied to REPTree regression tree algorithm.
- A experimental study is performed comparing our method with Bagging, Iterated Bagging AdaBoost.R2, Random Subspaces and Random Forest over 61 regression datasets from the UCI repository and from Luis Torgo Collection.
  - The proposed method achieves the best average rank.
  - The best performance of the proposed method is obtained when α is 0.4 or 0.5.

1.7.3 Tree ensemble construction using a GRASP-based heuristic and annealed randomness

This Journal Paper extends the idea of GRASP Forest and presents a new method for constructing ensembles of decision trees. The new method is called GAR-Forest (GRASP with Annealed Randomness). This method introduces randomness during the process of constructing the tree: maximum randomness at the root and minimum randomness at the leaves. The paper presents the intuitive idea that the nodes that have the greatest influence on the correct classification of the instances are the lower nodes and leaves, while the root and the top nodes influence the structure of the tree but not the correct classification of examples.

The key aspects of the paper are:

- The attribute and split point that shape the nodes of the trees are randomly selected from a list of good candidates.
- The minimum value of the items belonging to the list of candidates is calculated from the maximum, minimum and a parameter α as in the previous paper.
- The size of this list varies: from a list containing all possible items to a list containing only the best element.
- The size of the list associated with a given node depends on the percentage of instances that reach that node. A parameter τ controls how rapidly α is increased with the reduction of instances reaching the node.
- An experimental study is performed comparing GRASP Forest and GAR Forest with Bagging, AdaBoost.M1, Multiboost, Random Subspaces and Random Forest over 62 datasets from the UCI repository.
  - Average ranks were used for comparing mutiple methods.
  - The best ranks correspond to GAR-Forest with different values of τ.
  - We analyze the results, grouping the methods into families, because of the high number of methods compared and the fact that many of them differ only in the value of a parameter. The proposed method get the best position again.
  - New experiments were performed using datasets with artificial noise added. The results are even more convincing in the presence of noise, demonstrating the robustness of the method.
  - Kappa-error relative movement diagrams were used to visualize the behavior of the methods. Diagrams show that both GRASP-Forest and GAR-Forest build trees that are more accurate than those built by Random Forest.

1.7.4 Random Balance: Ensembles of Variable Priors Classifiers for Imbalanced Data

In this paper a new approach for building ensembles for two-class imbalanced datasets is proposed. The paper presents a new preprocessing

technique called Random Balance. This technique, based on a simple randomisation heuristic, can be used within an ensemble to increase the diversity and deal with imbalance. In the paper, it is also described a new ensemble method for imbalanced learning which combines Random Balance with AdaBoost.M2 and is called RB-Boost (Random Balance Boost).
The highlights of the paper are:

- The presented technique is based on a simple randomisation heuristic. The dataset obtained after applying the preprocessing technique will have random class proportions. The classes are either reduced, using Random Undersampling or augmented with artificial examples using SMOTE.
- The paper presents a study about the instance inclusion probability, showing the chances that an instance appears in the resulting datasets depending on the original size of classes. This study shows that the problem of discarding important instances of the majority class is ameliorated respecting other methods that use undersampling.
- The paper explains the intuition behind the method in a graphical way. When the classifier is represented as a point in the True Positive Rate - False Positive Rate space, it is shown how base classifiers trained using datasets with large variability in the ratio between classes are represented throughout the entire space.
- The paper includes a section on the state of the art in imbalanced learning.
- The paper includes an exhaustive experimental study. Ensembles using the proposed preprocessing technique are compared with SMOTEBoost, RUSBoost, SMOTEBagging and a long list of methods more over 86 datasets from KEEL repository and the HDDT collection.
  - The ensembles using the proposed method obtains the top positions in the average ranks according to AUC, F-Measure and Geometric Mean.
  - Several fusion rules are evaluated concluding that combination that performs better is the simple average of probabilities.
  - Regarding to base classifiers, decision trees perform better than nearest neighbours and SVM.

- This simple idea bypasses the need to tune the sensitive parameter used in most common preprocessing methods for imbalanced classification and simultaneously outperforms state of the art methods.

1.7.5 Diversity techniques improve the performance of the best imbalance learning ensembles

Ensembles of classifiers have gained popularity in dealing with imbalanced learning problems. When dealing with imbalanced data, commonly the techniques they used are those that affect the proportion between classes, for example re-weighting, oversampling and undersampling. In this paper it is studied the effect on imbalanced problems of other techniques originally intended to increase the ensemble diversity. The highlights of the paper are:

- The paper includes a brief introduction to the state of the art in imbalanced learning and classifier ensembles.
- The paper includes a novel taxonomy of the different diversifying techniques based on data manipulation.
- The paper includes a vast experimental study: in which seventeen of the state of the art methods for imbalance learning: RAMOBoost, Random Balance Boost, RUSBoost, SMOTEBoost and many more are tested by themselves and in combination with four different diversifying techniques: Random Oracles, Random Feature Weigths, Disturbing Neighbors and Rotation Forest. Experiments were conducted using 104 datasets from KEEL repository and the HDDT collection. The key parts of this experimental study are:
  - Average ranks are used for multiple method comparisons. The ranks were computed over AUC, F-Measure and Geometric Mean, comparing each ensemble method with its diversity enhanced variants.
  - In the average rank computed using the AUC and according to Hochberg's test, all ensembles are significaly worse than at least one of its counterpart enhanced with diversity techniques.
  - In the case of using the F-measure to compute the average ranks, the improvement is not statistically significant for all cases, but the improvement exists for the best methods. The best non-enhanced methods are Random Balance Boost, Bagging + Random Balance

and RUSBoost, and these methods are improved significantly when combined with Rotation Forest strategy.

- Something similar happens when examining the rank calculated with the G-mean. Bagging + Random Undersampling, Ensemble of Random Balance and RUSBoost are improved significantly when combined with Random Oracles and Disturbing Neighbors.

- The overall winner is Rotation Forest according to de AUC. RAMOBoost combined with Rotation Forest obtain the top position in the rank according to the F-Measure and Bagging + Random Undersampling combined with Random Oracles is the method with better rank according to the Geometric Mean.

- Another interesting finding is that the best combinations according to the F-measure use oversampling strategies while the best combinations according to the G-mean use undersampling. The method that gets the top position in the rank according to the AUC do not uses any balancing strategy.

- The paper presents a preliminary study trying to predict when to apply diversity techniques. This study is performed from meta-features obtained using a data complexity library. We use HotSpot to identify for which meta-features exists a high probability that a certain diversity technique improves the ensemble.

- Another experimental setup was conducted to test the suitability of Disturbing Neigbours, for dealing with datasets that have presence of noisy and borderline examples. It was found that methods which have been combined with Disturbing Neigbours perform better than their non-enhanced counterpart.

- It has been found that diversity-enhancing techniques can improve the performance of the best imbalance learning ensembles. This is a curious finding because all diversity-enhancing techniques that we applied are "imbalance-blind". This is doubly interesting because most research in imbalanced focuses on methods that address the imbalance of classes and neglects the use for techniques of increasing diversity.

1.7.6 Boosting Projections to improve surface roughness prediction in high-torque milling operations

The process consisting on removing material from a piece of raw material and cutting it into a desired shape is called "machining". Milling is a subtype of machining using rotatory cutters to remove the material. Surface roughness is a quality measure defined as the the vertical deviations of a real surface from its ideal form. Surface roughness of a milled workpiece will depend on many parameters. This paper describes how to use Boosting Projections to predict surface roughness in milling operations. Although the roughness is a continuous variable, this problem has become a classification problem by discretizing roughness in the levels suggested by the ISO Standard 4288:1996. The main parts of the article are:

- There is a section explaining the data set generation.
- There is a section that briefly reviews the ensemble learning techniques used to predict this model.
- Novel techniques for this type of problems have been used, such as Boosting Projections and Ordinal Classification.
- Although neural networks are commonly used for this type of problems, the best results are obtained with ensembles of trees, particularly Boosting Projections.

1.7.7 Imbalanced Learning Ensembles for Defect Detection in X-ray Images

This paper describes the process of detection of defects in high complexity metallic pieces through the analysis of X-ray images. There are several highlights in this paper:

- Metallic pieces used in this paper can present great amount of porosity, due to the magnesium inherent porosity. To our knowledge this is the first time that magnesium pieces are analyzed.
- The automatic system to be constructed should work with the same images that are currently used by human operators. This real application has many challenges: several different pieces, different views, variability introduced by the inspection process, such as the small

variations that occur in the position of the piece, when this is placed in the X-ray device.

- Due to the high complexity of images a system based on sliding window has been used. In this way a problem of image processing is transformed into a binary classification problem, and since the number of defective regions is much smaller than the non-defective, it is an unbalanced learning problem.
- The window is systematically moved along the image. For each window a set of features are extracted: basic statistics, Haralick and local binary patterns. From all these attribute selection is performed.
- CFS and SVM Attribute eval obtains the set of attributes that performs better in the results. Bigger window sizes leads to better performance.
- Experiments conclude that in this problem, Bagging achieved the best results, no improvements are obtained by combining Bagging with preprocessing techniques as SMOTE or Undersampling.

1.7.8 Segmentación de defectos en piezas de fundido usando umbrales adaptativos y ensembles (Segmentation of defects in castings using adaptive thresholds and ensembles)

In this paper we continue working with images and the problem described in the previous paper. This time we try to provide a solution that works almost in real time obtaining good results. The key aspects in this paper are:

- This paper presents a method that combines two approaches: the simplicity and speed common in image processing based methods, and the robustness to the variability provided by data mining based methods.
- The proposed method consists of two steps, in the first, all candidate regions be a defect are detected using local thresholds. In the second, feature extraction and classification of these candidate regions into defects or not-defect is performed.
- Four different local threshold algorithms were tested: Bersen, Mean, Niblack and Sauvola.

- Several features were extracted: basic stats, Haralick and geometric measures.

- Rotation Forest was used as a classifier. The performance of the method was evaluated comparing the predicted defect image with the "Ground Truth" using the geometric mean.

1.8 Conclusions

This thesis addres standard classification problems, imbalanced classification problems and applications using ensembles. Some of the contributions made in each of these fields are listed bellow.

1.8.1 Standard classification

It has been proposed the use of GRASP, a metaheuristic strategy widely used for optimization problems, for the construction of decision trees, in order to inject randomness, and thus to increase the diversity of ensembles. The results obtained were favourable t when compared with state of the art ensembles.

In the first method that follows to this idea, GRASP Forest, the level of randomness was the same throughout the process of building the tree.

The method is sophisticated and improved resulting in GAR-Forest. GAR-Forest. The purpose of GAR-Forest is to increase ensemble diversity, without harming the accuracy of the individual classifiers that constitute it. It is introduced another idea: lower nodes and leaves have the greatest influence on the correct classification of the instances so it is possible to introduce higher levels of randomness at the root and the top nodes to influence the structure of the tree and introduce lower levels of randomness in lower nodes and leaves to keep the tree still accurate. The combination of both ideas have proven to be competitive in experimental studies

The improvements made by the proposed methods over other ensemble methods are even more evident in presence of noisy datasets.

1.8.2 Regression

The use of GRASP metaheuristic as a way of injecting diversity has been also applied in ensembles of regression trees. Once again the results are

very competitive and outperform Random Forest, AdaBoost.R2 and other ensembles of regression trees.

1.8.3 Imbalance classification

Random Balance, a new preprocessing technique specially designed to be used in combination with ensemble methods, has been proposed and evaluated showing very good results. This technique is based on the idea of creating preprocessed output datasets in which the ratio between classes varies randomly. With this idea and taking inspiration from other methods, such as SMOTEBoost and RUSBoost, this technique has been combined with boosting creating a new ensemble method: RB-Boost. This intuitive heuristic avoids the need to tune the proportion parameter, common to most methods for imbalanced classification. Despite their simplicity, method using Random Balance clearly have exceeded the performance of other state-of-the-art ensembles.

Generally, the problem of unbalanced data sets, is addressed by manipulating the ratio between classes in order to reduce it. This has been done in Random Balance, but this time the ratio is random.

Diversity techniques have not been systematically studied for their effect on imbalanced problems. In the development of this thesis, a paper has been prepared that presents an exhaustive experimental study that combines techniques especially designed to work with imbalanced data with ensemble diversifying techniques that do not favourably impact the imbalance ratio of the dataset. An exhaustive experimental setup with five different analysis is carried out:

1. To analyze the impact of ensemble size (with and without diversity added) in the quality of the results. It was found that in general, bigger ensemble sizes leads to better results. There is one exception: RUSBoost using the G-mean as performance measure, for this method the best sizes are 20 and 30. This behaviour happens even when combining the method with diversity techniques.

2. To check whether the diversity enhancing techniques are well suited for ensembles in imbalanced classification. It was found that for the AUC all methods enhanced with diversity techniques perform significantly better than the original method. When taking into account the F-Measure and G-Mean, improvement does not happen for all the

methods, but it is clearly visible for the best methods according to these measures.

3. To find which combination between ensemble learning and diversity scheme has the best overall results for different metrics of performance. Rotation Forest and ensemble methods that combines resampling and oversampling with Rotation Forest monopolize the best positions in the average rank computed using the AUC. A similar conclusion can be extracted from the average rank computed using the F-Measure. The best combinations use oversampling strategies trying to obtain more balance and Rotation Forest, but this time Rotation Forest (alone without balancing strategies) does not achieve a good position. For the G-Mean, there is no clear trend. Although it seems that ensembles that uses Undersampling (RUSBoost, Bagging+RUS) or Random Balance obtains the best results when they are enhanced with Random Linear Oracles and Disturbing Neighbors.

4. To extract rules based on data complexity metrics to find when is more suitable to combine ensemble methods with diversity techniques. We prove that is possible to establish a relationship between the meta-features and the fact that the diversity technique improves the ensemble or not. From the generated set of rules, it can be extracted some interesting conclusions, for example, when the overlap of the per-class bounding boxes is high or the Directional-vector maximum Fisher's discriminant ratio is low [5], it would be a good idea to apply some diversity techniques.

5. To analyze the behaviour of diversity techniques for noisy and borderline examples in imbalanced datasets. It was found that methods which have been combined with Disturbing Neigbours occupy the top positions of the ranking for the three measures.

We found that enhancing diversity pays off. All diversity-enhancing techniques that were applied are "imbalance-blind", but surprisingly diversity-enhanced ensembles ranked better than their original counterpart. Another interesting finding was the results obtained for one measure can not be extrapolated to others, and one method that is the best according to a measure, not necessarily is the best according to others.

[5] which indicates the classes are hardly separables when projected into the maximum separability vector

1.8.4 Applications

Surface roughness prediction is a problem commonly addressed using neural networks or decision trees. There were not many papers on this subject that used ensembles. In this thesis Boosting Projection has been used for surface roughness prediction in a vertical milling machine obtaining better results than classical ensemble methods.

Besides the predicted output of industrial processes, defect detection is the other major area of application Machine Learning methods in an industrial context. Defect inspection ensure the safety and reliability of industrial processes and is key to achieving and maintaining competitiveness. Considering the techniques based on non-destructive test, those based on X-ray analysis are the most used. Radiography has proven to be one of the most effective techniques to identify defects in the process of injection-molded material. Two different approaches to the problem of detection of defects in radiographic images of magnesium parts have been developed, the aim of both is to speed up the process of defect detection and the elimination of the ambiguity and subjectivity of the human inspection.

1.9 Future lines

During the making of this thesis, some ideas have emerged that have not been yet fully developed, but that might be interesting for future work and, eventually, for further papers..

1.9.1 Multiclass classification

The success applying the GRASP metaheuristic as a technique to inject randomness into the tree construction process of J48 (C4.5) lead us to several possible lines of future research. For example, to study how this heuristic can be adapted to other tree construction algorithms, such as Functional Trees, Model Trees or Hellinger Distance Decision Tree (HDDT). Other algorithms such as rule construction, are also likely to be modified benefit from this heuristics in order to increase diversity.

1.9.2 Imbalance classification

Imbalanced datasets present some problems that have a strong influence on imbalanced classification [49, 36]. Some of these problems are: overlap-

ping [70], noisy examples [9], small disjuncts [79] or borderline examples [56]. One promising future line is to use the ideas of Random Balance to combine preprocessing techniques that have been designed to address the aforementioned problems (for example, the resampling strategy CBO [42] has been used successfully with small disjuncts; cleaning techniques such as ENN [80] or CNN [34] have been used with noisy datasets; and variants of SMOTE, such as, Safe-Level-SMOTE [11] or SPIDER [71] with borderline examples.) It can be very difficult to determine when a dataset suffers from these problems, unless the dataset is artificially generated, and much more difficult to adjust the value of the parameters of each technique for each dataset. We plan to use the various preprocessing techniques randomly at the begining, then progressively betting on those that offer better results using an adaptive mechanism.

In the paper with the systematic study of the diversity techniques applied to imbalanced learning (Chapter 6) we perform a preliminary study that tries to predict when the use of diversity-enhancing techniques could be more beneficial. One interesting future line would be to continue with this study using meta-features specially designed for unbalanced or meta-features that measures any of the intrinsic problems aforementioned. The objective of this study could be to determine whether it is possible to find the balancing technique and diversity strategy best suited to a certain dataset problem based on this meta-features.

Another interesting finding in this paper was that the best methods according to a measure are not necessarily the best according to others. The AUC and the F-Measure presents similar patterns, regarding the order of the methods in the ranks calculated with these measures, but with the Geometric Mean the ranks are quite different compared to the other two measures. It is therefore interesting to study ways of combining classifiers to try to get good results in both measures. We are exploring the idea of stacking, using classifiers that maximize a measure on the first level of the stack and the other on at the second level.

1.9.3 Applications

This thesis has addressed the problem of detecting and locating defects in metallic pieces using X-ray images. This is the first step of a full inspection system, the next steps are: extract features from the defect, classifying the type of defect in terms of its features (pore, bubble, impurity ...)

and because some types of defects are more severe than others provide a summary of each type of defect and a final assessment of the metallic piece's quality . We plan to address each of the following stages of the inspection process using ensemble learning.

Apart from this we are starting to use classifiers in another problem consisting also in analyzing X-ray images, but for a different application. This time it is about images of dental radiographs. There are two types of dental radiographs: intraoral and extraoral. Extraoral images are hard to analyze because teeth occlude with each other frequently. We have interest in applying ensemble learning to solve problems like: teeth detection and segmentation, dental age assessment based on teeth shape, prediction of a person's diet based on analyzing the erosion of the teeth and other interesting problems in the fields of archaeology and forensic science.

Chapter 2

GRASP Forest: A New Ensemble Method for Trees

Authors José-Francisco Díez-Pastor, Cesar García-Osorio, Juan José Rodríguez, Andrés Bustillo

Type Conference

Published in 10th International Workshop on. Multiple Classifier Systems (MCS 2011), pages 66-75.

Year 2011

Abstract

This paper proposes a method for constructing ensembles of decision trees: GRASP Forest. This method uses the metaheuristic GRASP, usually used in optimization problems, to increase the diversity of the ensemble. While Random Forest increases the diversity by randomly choosing a subset of attributes in each tree node, GRASP Forest takes into account all the attributes, the source of randomness in the method is given by the GRASP metaheuristic. Instead of choosing the best attribute from a randomly selected subset of attributes, as Random Forest does, the attribute is randomly chosen from a subset of selected good attributes candidates. Besides the selection of attributes, GRASP is used to select the split value for each numeric attribute. The method is compared to Bagging, Random Forest, Random Subspaces, AdaBoost and MutliBoost, being the results very competitive for the proposed method.

Index terms— Classifier ensembles, Bagging, Random Subspaces, Boosting, Random Forest, decision trees, GRASP

2.1 Introduction

Classifier ensembles [16] are combinations of several classifiers which are called *base classifiers* or *member classifiers*. Ensembles often give better results than individual classifiers. The kind of ensemble most often used is the homogeneous ensemble, in which all the base classifiers are built using the same method. In these ensembles, the diversity is commonly forced by training each base classifier with a variant of the training data set: Bagging [1] uses different random samples of the training set, Random Subspace [14] uses different subsets of attributes, AdaBoost [11] and Multiboost [21] adaptively change the distribution of the training set based on the performance of the previous classifiers, this way, the instances more difficult for the previous classifiers have a higher probability of being in the next training sample. Other methods, like Random Forest [2], increase the randomness by combining the sampling of the training set with the random selection of subsets of attributes in each node of the tree. This way, in each node, the splits only consider the selected subset of attributes. In [6], they use a very simple technique to randomize the election of the split among the twenty best split in each node. Recently, the method called Random Feature Weights [17] proposes to use all the attributes, but with different probabilities of being considered in the splits that depend on a weight associated to the attribute. To assure the diversity, the weights are randomly generated for each tree in the ensemble.

Decision trees are frequently used as base classifiers because they are efficient and unstable, that is, small changes in the training set or in the construction method will produce very different classifiers.

The algorithms for building decision trees are top-down methods. In the root of the tree they use all the instances to find which attribute is the best to split the instances in two subsets assigned to two new nodes[1], children of the root node. This process is recursively repeated in each new node till a stop criteria is verified. The best attribute is determined in each node by evaluating a merit function. Some common split criteria are: Information Gain and Gain Ratio [20], or Gini Index [3]. In this paper the merit function used is Gain Ratio.

The meta heuristic GRASP (Greedy Randomize Adaptive Search Procedure) [8, 9], a widely used strategy in optimization problems, has been

[1] If the attribute a_i is nominal, they create a new branch for each possible value of a_i.

recently used in [19] to modify the way the attribute is selected in the process of building a binary decision tree. The controlled randomness introduced by GRASP is able to build less complex trees without affecting the accuracy.

This work takes as starting point the idea used in [19], extends it using GRASP also in the selection of the split value for each attribute, and combining these trees in the construction of an ensemble, the GRASP Forest. Using GRASP in the selection of the split values gives an extra level of randomness that helps in increasing the diversity in the ensemble. This increased diversity compensates the loss in accuracy of the individual trees, improving in overall the ensemble accuracy.

The rest of the paper is organised as follows. Next section describes the proposed method. Section 3 describes the experiments and the results. Finally, Section 4 gives the conclusions and presents some future lines of research.

2.2 Method

Usually, to built a decision tree we have a training dataset D, several attributes $a_1, a_2, \ldots, a_n$ and a merit function $f(a_i, D)$ that gives a value to the i-th attribute. One of the most used merit function is *Information Gain* defined as

$$Gain(D,a) = Entropy(D) - \sum_{v} \frac{|D_v|}{|D|} Entropy(D_v) \tag{2.1}$$

where D is the data set, a is the candidate attribute, v indicates the values of the attribute and D_v is the subset of the data set D formed by the examples, where $a = v$. The entropy is defined as

$$Entropy(D) = \sum_{i=1}^{c} -p_i \log_2(p_i) \tag{2.2}$$

where c is the number of classes and p_i the probability of class i.

The GRASP method [8, 9] is a iterative process, each iteration has two steps:

1. Build an initial solution using a method that is greedy, random and adaptive.
2. Local search from the built solution trying to improve it.

Table 2.1: Backpack problem (weight limit of the backpack: 10 weight units).

Element	Weight	Value	Ratio Ratio Value/Weight
1	10	11	1.10
2	6	9	1.50
3	4	1	0.25

With each iteration the best found solution is updated and the process ends when a stop condition is reached.

As the building method is greedy, random and adaptive, every time a new element is added to the solution, instead of choosing the best possible element, one is randomly chosen from a short list of good candidates called the *Restricted Candidate List* (*RCL*). This list is created with those items whose values are close enough to the value of the best item. This closeness is defined by a percentage α of the best value. The idea behind GRASP is that the best solution in each step does not always lead the process to the global optimal solution of the problem. A good example is the backpack problem, for example, given 3 objects with weights, values and ratios (value/weight) shown in table 4.1 and a backpack capacity 10, is necessary to select a subset of them that fits in the backpack and that maximizes the value. A greedy method would take in each step the element with best possible value-weight ratio: first element 2, then element 3, with total value of 10; however, the best solution would have been to choose the element 1 that improves by one the previous solution.

The content of the RCL is defined as:

$$RCL = \{i : Value_i/Weight_i \geq \alpha Ratio_{max} + (1-\alpha)Ratio_{min}\}$$

If $\alpha = 1$ the list would have only one element, the element chosen by the greedy procedure; if $\alpha = 0$ the list would have all possible elements and the selection would be totally random.

In the work described in this paper, from GRASP we only use the construction of the solution by a greedy, random and adaptive procedure. Greedy, as the construction of trees is greedy by nature, random due to the random selection of attributes and split points, and adaptive because depending on the maximum and minimum gain ratio of the attributes the number of these considered for selection is different for each node. The aim is to increase the randomness in the process of building the base classifiers.

The method GRASP Forest (see Algorithm 4) works by using Algorithm 7 to create L decision trees that are added to the ensemble.

Algorithm 1: GRASP Forest

Input: Dataset D_T, set of attributes Attributes, size of ensemble L, value α between 0 and 1 to control the level of randomness
Output: Ensemble of decision trees
for $l = 1$ *to* L **do**
  GTree ← `TrainDecisionTree` (D_T,Attributes, α)
  Add GTree to the ensemble;
end

The way the trees are built is similar to the traditional algorithms. In each node, the merit function is evaluated for the m attributes. With these values a list of candidates is created from which the attribute to be used in the node is chosen. Note that with $\alpha = 1$ this algorithm would choose the same attribute as a traditional method, with $\alpha = 0$ the selection of the attribute is totally random.

Algorithm 2: TrainDecisionTree (for numeric attributes)

Input: Dataset D_T, set of attributes Attributes, value between 0 and 1 to control the level of randomness α
Output: Tree
if Attributes *is empty or number of examples* $<$ *minimum allowed per branch* **then**
  Node.label = most common value label in examples
  return Node;
else
  for $j = 1$ *to* m **do**
    model [j]← `GraspSplit` (D_T,Attributes,j,α)
  end
  maxGain ← Max(model.gain); minGain ← Min(model.gain)
  List ← $\{j = 1,2,\ldots,m|$ model [j].gain $\geq \alpha$maxGain $+ (1-\alpha)$minGain$\}$
  Randomly choose $j_g \in$ List; Att = j_g, splitPoint = model [j_g].splitPoint
  $D_l \leftarrow \{x \in D_T | x_{ij_g} \leq splitPoint\}$; $D_r \leftarrow \{x \in D_T | x_{ij_g} > splitPoint\}$;
  Node.son[0] = `TrainDecisionTree` (D_l,Attributes, α);
  Node.son[1] = `TrainDecisionTree` (D_r,Attributes, α);
end

Algorithm 8 shows how the idea of GRASP is also used in the process of choosing the splitting point for numeric attributes. The normal way of selecting the splitting point would be to find the point that maximizes the merit function value (in this work Gain Ratio). However, in *GraspSplit*, we again create a list of good candidates, with all points with value higher than

a minimal value determined by α, and one of them is randomly chosen and returned together with its merit function value.

Thus, the randomness in each node is both in selecting the attribute and in the choice of the splitting point within that attribute. With $\alpha = 1$, the generated tree is the same as with a traditional algorithm[2]. With $\alpha = 0$ the generated tree will be completely random as it happens in *Extremely Randomized Trees* [12].

Algorithm 3: GraspSplit

Input: Dataset D_T, set of attributes Attributes, attribute index j, value α between 0 and 1 to control the level of randomness
Output: model
`/* Compute values for all possible split point and their index */`
infoGain $\leftarrow$ List of all possible split info gains
infoGainIndex $\leftarrow$ List of all possible split indexes
maxGain $\leftarrow$ Max(infoGain); minGain $\leftarrow$ Min(infoGain)
List $\leftarrow \{j = 1,2,\ldots,m|\text{infoGain}[j] \geq \alpha\text{maxGain} + (1-\alpha)\text{minGain}\}$
Randomly choose $j_g \in$ List
model.gain = infoGain $[j_g]$, model.splitPoint = infoGainIndex $[j_g]$
return model

2.3 Results

The proposed method was implemented in the Weka library [13] by modifying J48, the Weka implementation of C4.5 [20] in conjunction with Random Committee[3]. The rest of decision trees and other ensembles are from this library. The size of the ensembles was set to 50. Since, when using trees in ensembles, we are interested in increasing diversity, we validate our method with low values of α, from 0.1 to 0.5. We compare our method with the following ensembles, whose settings are the default parameters of Weka unless otherwise indicated:

1. Bagging [1].
2. Boosting: AdaBoost.M1 [11] and Multiboost [21]. In both version the variants with resampling and reweighting were used (in the tables

[2]Except in the case of multiple attributes with the same value of the merit function, the traditional algorithm will always choose the same, according to the method of calculating the maximum, the version using GRASP will choose randomly between them, and similarly in the case of the splitting points.

[3]A Random Committee is an ensemble of randomizable base classifiers. Each base classifier is built using the same data but a different seed for the generation of randomness. The final predicted probabilities are simply the average of the probabilities generated by the individual base classifiers.

Table 2.2: Summary of the data sets used in the experiments.
#N: Numeric features, #D: Discrete features,#E: Examples, #C: Classes.

Dataset	#N	#D	#E	#C
abalone	7	1	4177	28
anneal	6	32	898	6
audiology	0	69	226	24
autos	15	10	205	6
balance-scale	4	0	625	3
breast-w	9	0	699	2
breast-y	0	9	286	2
bupa	6	0	345	2
car	0	6	1728	4
credit-a	6	9	690	2
credit-g	7	13	1000	2
crx	6	9	690	2
dna	0	180	3186	3
ecoli	7	0	336	8
glass	9	0	214	6
heart-c	6	7	303	2
heart-h	6	7	294	2
heart-s	5	8	123	2
heart-statlog	13	0	270	2
heart-v	5	8	200	2
hepatitis	6	13	155	2
horse-colic	7	15	368	2
hypo	7	18	3163	2
ionosphere	34	0	351	2
iris	4	0	150	3
krk	6	0	28056	18
kr-vs-kp	0	36	3196	2
labor	8	8	57	2
led-24	0	24	5000	10
letter	16	0	20000	26
lrd	93	0	531	10

Dataset	#N	#D	#E	#C
lymphography	3	15	148	4
mushroom	0	22	8124	2
nursery	0	8	12960	5
optdigits	64	0	5620	10
page	10	0	5473	5
pendigits	16	0	10992	10
phoneme	5	0	5404	2
pima	8	0	768	2
primary	0	17	339	22
promoters	0	57	106	2
ringnorm	20	0	300	2
sat	36	0	6435	6
segment	19	0	2310	7
shuttle	9	0	58000	7
sick	7	22	3772	2
sonar	60	0	208	2
soybean	0	35	683	19
soybean-small	0	35	47	4
splice	0	60	3190	3
threenorm	20	0	300	2
tic-tac-toe	0	9	958	2
twonorm	20	0	300	2
vehicle	18	0	846	4
vote1	0	15	435	2
voting	0	16	435	2
vowel-context	10	2	990	11
vowel-nocontext	10	0	990	11
waveform	40	0	5000	3
yeast	8	0	1484	10
zip	256	0	9298	10
zoo	1	15	101	7

represented with S and W). For Multiboost the approximate number of subcommittees was 10.

3. Random Subspaces [14]: with two different configurations, with 50% and 75% of the original set of attributes.

4. Random Forest [2]: three different configurations, random subsets of attributes of size 1, 2 and base 2 logarithm of the number of attributes in the original set.

Table 2.3: Wins, ties and losses; the comparison of the ensembles with binary and with non-binary trees is shown (U: unpruned trees, P: pruned trees).

Method	All			Significative (0.05)		
	V	T	L	V	T	L
Bagging (P)	17	35	10	3	59	0
Bagging (U)	15	39	8	3	59	0
AdaBoost-W (P)	16	34	12	2	60	0
AdaBoost-W (U)	15	36	11	2	60	0
AdaBoost-S (P)	18	35	9	2	60	0
AdaBoost-S (U)	15	34	13	2	60	0
MultiBoost-W (P)	13	36	13	3	59	0
MultiBoost-W (U)	14	37	11	2	60	0
MultiBoost-S (P)	12	37	13	2	60	0
MultiBoost-S (U)	16	34	12	2	60	0
Random Subspaces 50% (P)	18	36	8	0	62	0
Random Subspaces 50% (U)	17	35	10	0	62	0
Random Subspaces 75% (P)	20	35	7	3	59	0
Random Subspaces 75% (U)	17	36	9	2	60	0
Random Forest $K = \log numAtt$	42	2	18	3	59	0
Random Forest $K = 1$	24	4	34	0	62	0
Random Forest $K = 2$	34	1	27	1	60	1

For all ensembles, both pruned and not pruned trees were used as base classifiers, except for Random Forest, because pruning is not recommended in this case [2]. Binary trees were used for two reasons, first, this was the kind of trees used in [19], second, in the case of nominal attributes, the use of GRASP metaheuristic for selecting the splitting point is only possible for binary trees (in a non-binary tree, a node that uses a nominal attribute has as many children as nominal values, and it is not necessary to create the list of splitting points since only one splitting point is possible). For Random Forest, that does not work with binary nodes, we used the preprocessing described in [3], where nominal attributes with k values are transformed into k binary attributes. It could have been possible to optimize the value of α of our method by using a validation data subset or internal cross-validation, but this would not have been fair for the other methods, besides we were interested in analysing the global effect of α.

Before comparing the new method, to check that the good results of GRASP Forest over the other ensembles are not due to the use of binary trees degrading their performance, we compared the other ensembles using binary and non binary trees. In Table 2.3 the wins, ties and losses are

Table 2.4: Ensemble methods sorted by average rank (U: unpruned trees, P: pruned trees).

	Method	Ranking
1	GRASP Forest $\alpha = 0.2$ (U)	10.0645161290323
2	GRASP Forest $\alpha = 0.3$ (U)	10.4838709677419
3	RandomForest $K = \log numAtt$	11.1129032258065
4	GRASP Forest $\alpha = 0.4$ (P)	11.1935483870968
5	GRASP Forest $\alpha = 0.3$ (P)	11.3306451612903
6	GRASP Forest $\alpha = 0.4$ (U)	11.4274193548387
7	GRASP Forest $\alpha = 0.1$ (U)	11.5887096774194
8	MultiBoost-S (P)	11.9435483870968
9	MultiBoost-S (U)	12.3467741935484
10	GRASP Forest $\alpha = 0.5$ (U)	12.7661290322581
11	GRASP Forest $\alpha = 0.2$ (P)	12.8548387096774
12	GRASP Forest $\alpha = 0.5$ (P)	13.2177419354839
13	AdaBoost-S (P)	13.3709677419355
14	MultiBoostAB-W (U)	13.6290322580645
15	RandomForest $S = 2$	13.6370967741935
16	MultiBoostAB-W (P)	13.6451612903226
17	GRASP Forest $\alpha = 0.1$ (P)	13.6532258064516
18	AdaBoost-S (P)	13.6774193548387
19	AdaBoost-W (P)	14.3145161290323
20	AdaBoost-W (U)	15.1209677419355
21	Random-Subspaces-50% (U)	15.4435483870968
22	RandomForest $K = 1$	16.5806451612903
23	Random-Subspaces-50% (P)	17.1854838709677
24	Bagging (P)	17.7903225806452
25	Bagging (U)	18.0483870967742
26	Random-Subspaces-50% (U)	20.7096774193548
27	Random-Subspaces-75% (P)	20.8629032258064

shown both total and with a 0.05 significance level for the corrected t paired Student test [18] using 5×2 cross validation [7] and the same 62 datasets used in the next experiments. We can see that the performance is slightly better when the base classifiers are binary trees. Anyway, the results with binary trees are clearly not worse, so the use of binary trees in the comparison of these ensembles with GRASP forest is fair.

Finally, we did the comparison with our method. The experiments were performed using 5×2 cross validation, over 62 data sets from the UCI repository [10] (see table 4.2). Table 4.3 shows the results as an average ranking [5][4]. Various configurations for the parameter α obtained

[4]For each dataset, the methods are sorted according to their performance. The best method has rank 1, the second rank 2, and so on. If several methods have the same result, they are assigned an average value.

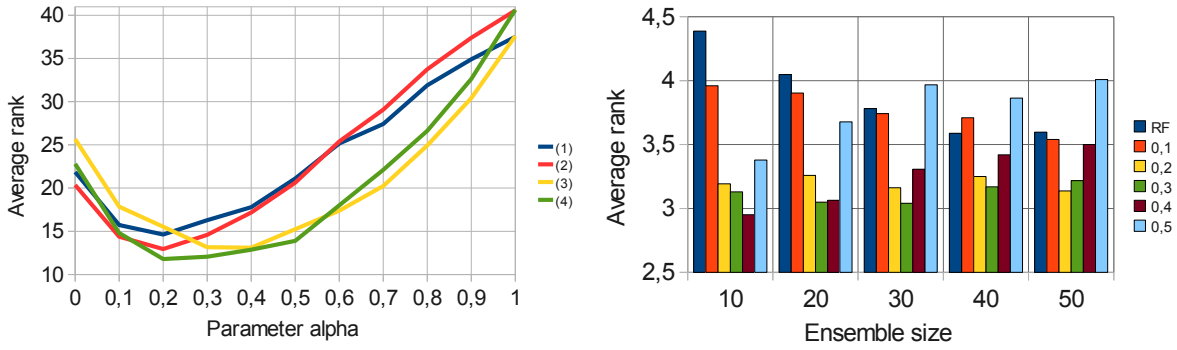


Figure 2.1: *Left*: Average rankings in function of α and four different GRASP Forest configurations: (1) and (2) using GRASP only to choose the attribute, (3) and (4) using GRASP both for attribute and splitting point selection; (1) and (3) using pruned trees as base classifiers, (2) and (4) using not pruned trees as base classifiers (average rankings calculated from all 44 configurations). *Right*: Average rankings for different ensemble sizes and six different methods, Random Forest with $K = \log numAtts$ and GRASP Forest with five different α values and unprunned trees (GRASP used both for attribute and splitting point selection).

favourable results compared to most traditional ensembles.

Figure 2.1 shows average rankings for different configurations of GRASP forest: using GRASP for both the attribute and splitting point selection, using GRASP only for attribute selection, using pruned trees as base classifiers, using not pruned trees as base classifiers, and using 11 different values of α between 0 and 1. The ensemble size was 50. On the left, the average rankings are calculated from all ensemble configurations, 44 in total (2 variants of GRASP $\times$ 2 different base tree classifiers $\times$ 11 values of α). On the right, the average rankings are calculated for each size of the ensemble and for six different methods.

It is possible to appreciate how, in general, the ensembles that use GRASP in the two steps of the tree construction, both attribute and split point selection, get better results both with prune and not pruned trees. The global optimum for α is around 0.2, the point from which the increase in diversity does not compensate the lost in accuracy of the individual trees in the ensemble. As well, for ensembles with few base classifiers, the best results are obtained with trees with low randomness ($\alpha = 0.5$), but as the size of the ensemble increases, the trees with the best rankings are those with higher level of randomness ($\alpha = 0.2$ and $\alpha = 0.3$).

For each method, its average rank is calculated as the mean across all the datasets.

2.4 Conclusion and future lines

GRASP is a metaheuristic strategy widely used for optimization problems, recently it has been used for the construction of trees less complex than the ones built with deterministic algorithms. In this paper, we propose an evolution of the use of this metaheuristic in the construction of trees that is used to increase the diversity of ensembles. The results are favourable compared with traditional ensembles.

There are several future research lines. This paper presents a method that injects randomness into two steps in the construction of the tree: in the choice of the splitting attribute and in the selection of the splitting point within that attribute. In this work, the parameter that determines the randomness is the same for both steps, a line of future work will be to study how the performance of the method could be affected if independent values are used in these steps. Another aspect to consider is the performance of this new method in combination with traditional ensembles. Rather than considering these methods as competitors against which to measure the GRASP Forest, consider them as allies that the GRASP Forest could improve.

We have shown that when the average ranking is calculated as a function of α, there is a kind of global optimum. However, this does not mean that a value of α could optimally work for all data sets. In [19] is stated that intermediate values of α improve the accuracy of the trees in simple databases and high values improve the accuracy in complex databases. Another line of future work could be to carry out an exhaustive study of what is the effect of the value of α in terms of different meta-features [4] or complexity measures [15] for several datasets. This way, this knowledge could be used in the algorithm to adaptively choose the value of α taking into account the characteristics of the dataset.

Last but not least, we would like to work on adapting GRASP Forest for regression problems.

References

[1] L. Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[2] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125.

[3] L. Breiman et al. *Classification and Regression Trees*. 1st ed. Chapman & Hall/CRC, 1984. ISBN: 0412048418.

[4] C. Castiello, G. Castellano, and A. Fanelli. "Meta-data: Characterization of input features for meta-learning". In: *Modeling Decisions for Artificial Intelligence* (2005), pp. 457–468.

[5] J. Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine Learning Research* 7 (2006), p. 30.

[6] T. Dietterich. "Ensemble methods in machine learning". In: *Multiple classifier systems* (2000), pp. 1–15.

[7] T.G. Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[8] T.A. Feo and M.G.C. Resende. "A probabilistic heuristic for a computationally difficult set covering problem". In: *Operations Research Letters* 8 (1989), pp. 67–71.

[9] T.A. Feo and M.G.C. Resende. "Greedy randomized adaptive search procedures". In: *Journal of Global Optimization* 6.2 (1995), pp. 109–133. ISSN: 0925-5001.

[10] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. 2010.

[11] Y. Freund and R.E. Schapire. "Experiments with a new boosting algorithm". In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer. 1996, pp. 148–156.

[12] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely randomized trees". In: *Machine Learning* 63.1 (2006), pp. 3–42. ISSN: 0885-6125.

[13] Mark Hall et al. "The WEKA data mining software: an update". In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[14] T.K. Ho. "The random subspace method for constructing decision forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[15] T.K. Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.3 (2002), pp. 289–300. ISSN: 0162-8828.

[16] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[17] J. Maudes et al. "Random Feature Weights for Decision Tree Ensemble Construction". In: *Information Fusion* In Press, Accepted Manuscript (2010). ISSN: 1566-2535. DOI: DOI:10.1016/j.inffus.2010.11.004.

[18] C. Nadeau and Y. Bengio. "Inference for the Generalization Error". In: *Machine Learning* 52 (3 2003), pp. 239–281. ISSN: 0885-6125.

[19] J. Pacheco et al. "Uso del metaheurístico GRASP en la construcción de árboles de clasificación". In: *Rect@* 11 (2010), pp. 139–154. ISSN: 1575605X.

[20] R. J. Quinlan. *C4.5: Programs for Machine Learning*. 1st ed. Morgan Kaufmann, 1993. ISBN: 1558602380.

[21] G.I. Webb. "Multiboosting: A technique for combining boosting and wagging". In: *Machine learning* 40.2 (2000), pp. 159–196.

Chapter 3

GRASP Forest for regression: GRASP metaheuristic applied to the construction of ensembles of regression trees

Authors José-Francisco Díez-Pastor, Cesar García-Osorio, Juan José Rodríguez.

Type Conference

Published in XIV Conferencia de la Asociación Española para la Inteligencia Artificial. (CAEPIA 2011).

Year 2011

Abstract

This paper proposes the application of the GRASP metaheuristic, commonly used in optimization problems, as a method for increasing diversity in ensembles of regression trees. Most ensemble algorithms try to increase diversity by modifying the training set in some way, e. g. Random Forest does it and also introduces randomness in the process of building the tree by selecting among different random subsets of attributes for each node.

In our method, the source of diversity is due to GRASP; attributes and splitting points within these are selected at random from a list of candidates generated as proposed by GRASP. The method has been compared to Bagging, AdaBoost.R2, Random Subspaces, Iterated Bagging and Random Forest, obtaining very competitive results for the proposed method.

Index terms— Regression ensembles, Bagging, Random Subspaces, Boosting, Random Forest, regression trees, GRASP

3.1 Introduction

Ensembles [19] are combinations of several models which are called *base models*. Ensembles often give better results than individual models. Although they have been studied mainly for classification, there are also ensemble methods for regression.

To make sense of combining multiple models they have to be different. The underlying logic is that if they all give the same output, the whole ensemble could be replaced by a single model. Therefore a key aspect in the construction of ensembles is to get diverse base models.

The kind of ensemble most often used is the homogeneous ensemble, in which all the base models are built using the same method. In these ensembles, the diversity is commonly forced by training each base model with a variant of the training data set.

In Bagging [1], each base model is trained using a random sample, with replacement, of the training data. Some examples of the training will appear several times in the sample, while others will not appear. This can be used both for regression and classification.

In Random Subspaces [18], all the training data is used to train all the base models, but the models are trained in different random subspaces. Again, if the models are for classification, this builds an ensemble for classification, similarly, if one use regressors as base models, an ensemble for regression is obtained.

In AdaBoost.R2 [8], that is based on AdaBoost [14], each training example has a weight. Initially, all the examples have the same weight. The construction of the ensemble members must take into account the examples weights. After an ensemble member is constructed, the examples weights are adjusted. The idea is to give more weight to the examples with greater errors in the previous iterations. Hence, in the construction of the next member, these examples will be more important. The ensemble members also have weights that depend on their error. The prediction of the ensemble is a weighted median of the predicted value of the base models.

Iterated Bagging [3] combines several Bagging ensembles. The first Bagging ensemble is constructed as usual. Based on the predictions of the previous Bagging ensemble, the values of the predicted variable are altered. The next Bagging ensemble is trained with these altered values. These values are the residuals: the difference between the real and the predicted

values. Nevertheless, these predictions are not obtained using all the members in the Bagging ensemble. The error of the predictions for a training example would be too optimistic, the majority of the ensemble models have been trained with that example. These predictions are calculated using the out-of-bag estimation: the prediction for an example is obtained using only those ensemble members that were not trained with that example. The prediction of an Iterated Bagging ensemble is the sum of the predictions of its Bagging ensembles.

Other methods, like Random Forest [2], increase the randomness by combining the sampling of the training set with the random selection of subsets of attributes in each node of the tree. This way, in each node, the splits only consider the selected subset of attributes. It can be used for regression or for classification just changing the type of base models.

The increase of diversity in ensembles of trees has been widely studied. In [6], they use a very simple technique consisting on randomizing the election of the split among the twenty best split in each node. Recently, the method called Random Feature Weights [20] proposes to use all the attributes, but with different probabilities of being considered in the splits that depend on a weight associated to the attribute. To assure the diversity, the weights are randomly generated for each tree in the ensemble.

The rest of the paper is organised as follows. Next section describes the proposed method. Section 7.6 describes the experiments and the results. Finally, Section 7.7 gives the conclusions and presents some future lines of research.

3.2 The GRASP metaheuristic applied to the construction of regression trees

Decision and regression trees are frequently used as base models because they are efficient and unstable, that is, small changes in the training set or in the construction method will produce very different models.

The algorithms for building regression trees are top-down methods. In the root of the tree they use all the instances to find which attribute is the best to split the instances in sub-divisions assigned to new nodes, children of the root node. This process is recursively repeated in each new node till a stop criteria is verified. The best attribute is determined in each node by evaluating a merit function. Some common split criteria are: Impurity [22],

Table 3.1: Backpack problem (weight limit of the backpack: 10 weight units).

Element	Weight	Value	Ratio Value/Weight
1	10	11	1.10
2	6	9	1.50
3	4	1	0.25

or Gini Index [4].

The regression tree used for this work is REPtree [23], a fast tree learner that builds a regression tree using variance, and prunes it using Reduced Error Pruning (REP) [9]. This type of tree is implemented in the Weka machine learning library [17].

The meta heuristic GRASP (Greedy Randomize Adaptive Search Procedure) [10, 11], a widely used strategy in optimization problems, is an iterative process, each iteration has two steps:

1. Build an initial solution using a method that is greedy, random and adaptive.

2. Improve the initial solution by means of a local search.

With each iteration the best found solution is updated and the process ends when a stop condition is reached.

As the building method is greedy, random and adaptive, every time a new element is added to the solution, instead of choosing the best possible element, one is randomly chosen from a short list of good candidates called the *Restricted Candidate List* (*RCL*). This list is created with those items whose values are close enough to the value of the best item. This closeness is defined by a percentage α of the best value. The content of the RCL is defined as:

$$RCL = \{i : Score_i \geq \alpha Score_{max} + (1-\alpha) Score_{min}\} \tag{3.1}$$

If $\alpha = 1$ the list would have only one element, the one with best score that will be chosen by the greedy procedure; if $\alpha = 0$ the list would have all possible elements and the selection would be totally random.

The idea behind GRASP is that the best solution in each step does not always lead the process to the global optimal solution of the problem. A good example is the backpack problem. For instance, given 3 objects of different values and weights, as shown in Table 4.1), and a backpack

capacity 10, it is necessary to select a subset of them that fits in the backpack and that maximizes the value. A greedy method would take in each step the element with best possible value/weight ratio: first element 2 and then element 3, with total value of 10; however, the best solution would have been to choose the element 1 that improves by one the previous solution.

GRASP has firstly been used to modify the way attributes are selected in the process of building a binary classification tree in [21]. The controlled randomness introduced by GRASP is able to build less complex trees without affecting the accuracy. In that work the trees were constructed using the strategy proposed by GRASP; at each node, the attribute to split the node is chosen from a *Restricted Candidate List* with the attributes with highest information gain, generating a set of classification trees (something equivalent to the first step of GRASP). After that the best tree of the set is selected (this is equivalent to the second step) and that single tree is the one used for classification.

Recently in [7] it has been proposed a new method of constructing ensembles of classification trees using GRASP. The starting point is, as in [21], the idea of using a Restricted Candidate List to choose the attributes; to increase even more the diversity, a second list is also used for selecting the splitting points within the attributes. When all the trees has been generated, instead of doing a search to pick the best one, all the trees are kept and used as base models of an ensemble. This paper describes the adaptation of the method proposed in [7] to solving regression problems.

3.2.1 Method

Usually, to build a regression tree we have a training dataset D, several attributes $a_1, a_2, \ldots, a_m$ and a merit function $f(a_i, D)$ that gives a value to the i-th attribute. Variance Gain, Impurity and Gini Index are commonly used as merit function in regression trees.

For this work, the REPtree implementation in Weka library has been modified. The merit function for this tree is the Variance Gain. In case of nominal attributes, the Variance Gain is calculated as:

$$\mathrm{VGain}(D,a) = \mathrm{Var}(\mathrm{LBL}(D)) - \sum_{v} \mathrm{Var}(\mathrm{LBL}(D_{a=v})) \qquad (3.2)$$

where D is the data set, $\mathrm{LBL}(D)$ is the set of all the values to predict, a is the candidate attribute, v represents an attribute value, and $D_{a=v}$ is a subset

of D with only those instances for which the attribute a has value equal to v, $\{\mathbf{x}_i : x_{i,a} = v\}$. For continuous attributes, the Variance Gain is calculated as:

$$\text{VGain}(D,a) = \text{Var}(\text{LBL}(D)) - \min_x(\text{Var}(\text{LBL}(D_{a>x}) + \text{Var}(\text{LBL}(D_{a\leq x})) \tag{3.3}$$

For all values x of attribute a in the training set, the *split variance*, the sum of the variances on both sides of the value is calculated, the minimum of all these values is subtracted from the total variance, the result is the variance gain for the attribute.

The method GRASP Forest for Regression (see Algorithm 4) works by using Algorithm 7 to create L regression trees that are added to the ensemble.

Algorithm 4: GRASP Forest for Regression

Input: Training dataset D_T, set of attributes Attributes, size of ensemble L, value α to control the level of randomness (between 0 and 1)
Output: Ensemble of regression trees
for $l = 1$ **to** L **do**
    GTree ← `TrainRegressionTree` (D_T, Attributes, α); *// Algorithm 7*

    Add GTree to the ensemble;
end

The way the trees are built is similar to the original REPtree algorithm. In each node, the merit function is evaluated for all the attributes. With these values, a restricted candidate list (see Equation 3.1) is created from which the attribute to be used in the node is chosen. Note that with $\alpha = 1$ this algorithm would choose the same attribute as a traditional method, with $\alpha = 0$ the selection of the attribute is totally random.

Algorithm 8 shows how the idea of GRASP is also used in the process of selecting the splitting point for numeric attributes. The normal way of doing this would have been to find the point that maximizes the merit function value (in this work Variance Gain). This requires minimizing the split variance (see second term of Equation 3.3). However, in *GraspSplit*, we again create a list of good candidates, with all splitting points with split variance value lower than a maximal value determined by α, and one of them is randomly chosen.

Thus, the randomness in each node is both in choosing the attribute and in the selection of the splitting point within that attribute. If the value of α

Algorithm 5: TrainRegressionTree (for numeric attributes)

Input: Training dataset D_T, set of m attributes Attributes, value α to control the level of randomness (between 0 and 1)
Output: Regression tree
if Attributes *is empty* **or** *number of examples* $<$ *minimum allowed per branch* **then**
  Node.label = average of label values in the examples;
  return Node;
else
  for $j = 1$ **to** m **do**
    model $[j] \leftarrow$ `GraspSplit` (D_T, Attributes, j, α); *// see Algorithm 8*
  end
  maxVG $\leftarrow$ Max(model.VarGain); minVG $\leftarrow$ Min(model.VarGain);
  List $\leftarrow \{j = 1,2,\ldots,m|$ model $[j]$.VarGain $\geq \alpha$maxVG $+ (1-\alpha)$minVG$\}$;
  Randomly choose $j_g \in$ List; Node.att = j_g;
  Node.splittingPoint = model $[j_g]$.splittingPoint;
  $D_l \leftarrow \{\mathbf{x} \in D_T | x_{i,j_g} \leq$ Node.splittingPoint$\}$;
  $D_r \leftarrow \{\mathbf{x} \in D_T | x_{i,j_g} >$ Node.splittingPoint$\}$;
  Node.son [0] = `TrainRegressionTree` (D_l, Attributes, α);
  Node.son [1] = `TrainRegressionTree` (D_r, Attributes, α);
end

is 1, the generated tree is the same as with a traditional algorithm. If the value of α is 0, the generated tree will be completely random.

3.3 Results

The proposed method was implemented in Weka library [17] by modifying REPtree, and used in conjunction with Random Committee, an ensemble of randomizable base regressors. Each base regressor is built using the same data but a different seed for the generation of randomness. The final predictions are simply the average of the values generated by the individual base regressor. The rest of regression trees and other ensembles are from this library. The size of ensembles was set to 100.

We validate our method with $\alpha = 0.5$ comparing with the following ensembles (whose settings are the default parameters of Weka unless otherwise indicated):

1. Bagging [1].
2. Boosting (specifically, AdaBoost.R2 [8], its variant for regression). This method can be used with different loss functions. Three are

Algorithm 6: GraspSplit

Input: Dataset D_T, set of attributes Attributes, attribute index j, value α between 0 and 1 to control the level of randomness
Output: model
Variance $\leftarrow$ variance of all output values;
/* *Compute split variance values for all possible splitting points* */
splittingPointVars $\leftarrow$ List of all possible split variance values;
splittingPointValues $\leftarrow$ List of all possible splitting points;
maxSplitVar $\leftarrow$ Max(splittingPointVars);
minSplitVar $\leftarrow$ Min(splittingPointVars);
List $\leftarrow \{j = 1,\ldots,m|\text{splittingPointVars}[j] \leq \alpha\text{minSplitVar} + (1-\alpha)\text{maxSplitVar}\}$;

Randomly choose $j_g \in$ List;
model.VarGain = Variance $-$ splittingPointVars $[j_g]$;
model.splittingPoint = splittingPointValues $[j_g]$;
return model;

proposed in [8] and used in this work: linear, square and exponential. The suffixes "-Li", "-Sq" and "-Ex" are used to denote the function used. Moreover, methods based on AdaBoost can be used in two ways [13]. In the reweighting version, the base model is trained with all the training data, it must take into account the weight distribution. In the resampling version, the base model is trained with a sample from the training data. This sample is constructed taken into account the weights. These versions are denoted with "-W" and "-S".

3. Random Subspaces [18] with two different configurations, with 50% and 75% of the original set of attributes.

4. Random Forest [2], random subsets of attributes of size logarithm in base 2 of the number of attributes in the original set.

5. Iterated Bagging [3]: two configurations were considered: 10×10 (Bagging is iterated 10 times, the ensemble size of each Bagging is 10), and 5×20 (Bagging is iterated 5 times, the ensemble size of each Bagging is 20). In both cases, the maximum ensemble size is 100.

For all ensembles, both pruned and not pruned trees were used as base regressors.

The experiments were performed using 5×2 cross validation, over 61 data sets (see Table 4.2) available in the arff format used by Weka[1], most of

[1] `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`

Table 3.2: Summary of the data sets used in the experiments.
#N: Numeric features, #D: Discrete features,#E: Examples, #C: Classes.

Dataset	#N	#D	#E
2d-planes	10	0	40768
abalone	7	1	4177
ailerons	40	0	13750
auto93	16	6	93
auto-horse	17	8	205
auto-mpg	4	3	398
auto-price	15	0	159
bank-32nh	32	0	8192
bank-8FM	8	0	8192
baskball	4	0	96
bodyfat	14	0	256
bolts	7	0	40
breast-tumor	1	8	286
cal-housing	8	0	20640
cholesterol	6	7	303
cleveland	6	7	303
cloud	4	2	108
cpu-act	21	0	8192
cpu	6	1	209
cpu-small	12	0	8192
delta-ailerons	5	0	7129
delta elevators	6	0	9517
detroit	13	0	13
diabetes-numeric	2	0	43
echo-months	6	3	130
elevators	18	6	16599
elusage	1	1	55
fishcatch	5	2	158
friedman	10	0	40768
fruitfly	2	2	125
gascons	4	0	27

Dataset	#N	#D	#E
house-16H	16	0	22784
house-8L	8	0	22784
housing	12	1	506
hungarian	6	7	294
kin8nm	8	0	8192
longley	6	0	16
lowbwt	2	7	189
machine-cpu	6	0	209
mbagrade	1	1	61
meta	19	2	528
mv	7	3	40768
pbc	10	8	418
pharynx	1	10	195
pole	48	0	15000
pollution	15	0	60
puma32H	32	0	8192
puma8NH	8	0	8192
pw-linear	10	0	200
pyrimidines	27	0	74
quake	3	0	2178
schlvote	4	1	38
sensory	0	11	576
servo	0	4	167
sleep	7	0	62
stock	9	0	950
strike	5	1	625
triazines	60	0	186
veteran	3	4	137
vineyard	3	0	52
wisconsin	32	0	194

them from the UCI repository [12] and from Luis Torgo Collection[2].

Table 4.3 shows the results as an average ranking [5]. For each dataset, the methods are sorted according to their Root relative squared error. The best method is given rank 1, the second rank 2, and so on. If several methods have the same result, they are assigned an average value. For each method, its average rank is calculated as the mean across all the datasets.

Grasp Forest for Regression using $\alpha = 0.5$ obtained favourable results

[2] `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html`

Table 3.3: Ensemble methods sorted by average rank (U: unpruned, P: pruned trees).

	Method	Ranking
1	GRASP Forest $\alpha = 0.5$ (U)	7.82
2	RandomForest $K = \log_2 N$ (U)	9.00
3	AdaBoostR2-S-Ex (P)	9.74
4	Iterated Bagging 5×20 (P)	9.97
5	Bagging (U)	11.00
6	AdaBoostR2-S-Sq (P)	11.39
7	AdaBoostR2-S-Li (P)	11.62
8	Iterated Bagging 5×20 (U)	12.26
9	Iterated Bagging 10×10 (P)	12.66
10	Bagging (P)	12.74
11	GRASP Forest $\alpha = 0.5$ (P)	12.74
12	AdaBoostR2-S-Sq (U)	12.82
13	RandomForest $K = \log_2 N$ (P)	13.00
14	AdaBoostR2-S-Li (U)	13.62
15	AdaBoostR2-W-Sq (U)	13.77
16	AdaBoostR2-S-Ex (U)	14.39
17	AdaBoostR2-W-Ex (P)	14.75
18	AdaBoostR2-W-Li (P)	15.38
19	Random-Subspaces-50% (U)	15.54
20	AdaBoostR2-W-Sq (P)	15.79
21	Iterated Bagging 10×10 (U)	15.84
22	Random-Subspaces-75% (P)	15.97
23	AdaBoostR2-W-Li (U)	16.84
24	Random-Subspaces-75% (U)	17.33
25	AdaBoostR2-W-Ex (U)	17.36
26	Random-Subspaces-50% (P)	17.77

compared to most traditional ensembles.

Fig. 3.1 shows the evolution of the average ranks for different configurations of GRASP forest, using pruned and unpruned trees as base regressors, and using 11 different values of α between 0 and 1 for each. The ensemble size was 100. The average rankings are calculated from all ensemble configurations, 22 in total (2 different base tree regressors $\times$ 11 values of α).

Another experiment carried out was the comparison between Random Forest and GRASP Forest, the ones that performed better according to the average ranking (Table 4.3). As both are methods that depend on the choice of a parameter (the size K of the subset of attributes to evaluate at each node in Random Forest, and the value of α which controls the

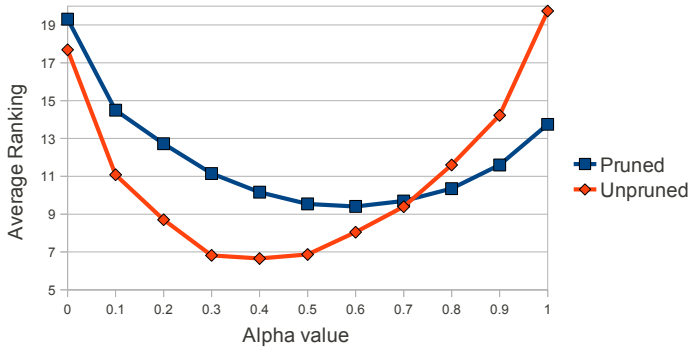


Figure 3.1: Average rankings in function of α and two different base trees: pruned and unpruned. Unpruned are better than pruned in most of $alpha$ values tested, values between 0.3 and 0.5 reach the best average ranking

randomness in GRASP Forest), a grid search was done to find the optimal values. In the case of Random Forest, the best value among $K = 0$, $K = 1$ and $K = \log_2 numAtt$ was used; and in the case of GRASP Forest, the experiments were carried out with the value of α among all the values between 0.1 and 0.9 at intervals of 0.2.

The Wilcoxon matched-pairs signed ranks was used, according to this test GRASP Forest is better than Random Forest with a level of significance $p \leq 0.06284$.

3.4 Conclusion and future lines

The GRASP metaheuristic, widely used in optimization problems, had been previously applied to the construction of trees and ensembles of decision trees. In this paper, we apply this metaheuristic as a way of injecting diversity into the construction method of regression trees, the results are very competitive and outperform traditional ensembles of regression trees.

There are several possible lines of future research, in our opinion, the most interesting of them would be finding a modification of this method that were independent of the choice of a parameter. Others could be the use of the same techniques used to improve the performance of GRASP, such as Path Relinking [16] or use other heuristics such as Tabu Search [15] pursuing the same goal of increasing diversity in the process construction of constructing ensembles of trees.

References

[1] L. Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[2] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125.

[3] L. Breiman. "Using iterated bagging to debias regressions". In: *Machine Learning* 45.3 (2001), pp. 261–277. ISSN: 0885-6125.

[4] L. Breiman et al. *Classification and Regression Trees*. 1st ed. Chapman & Hall/CRC, 1984. ISBN: 0412048418.

[5] J. Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine Learning Research* 7 (2006), p. 30.

[6] T. Dietterich. "Ensemble methods in machine learning". In: *Multiple classifier systems* (2000), pp. 1–15.

[7] J. F. Diez-Pastor et al. "GRASP Forest: A New Ensemble Method for Trees". In: *10th International Workshop, MCS 2011, Lecture Notes in Computer Science, Vol. 6713*. 2011, pp. 66–75. ISBN: 978-3-642-21557-5.

[8] H. Drucker. "Improving regressors using boosting techniques". In: *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 107–115. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.314.

[9] T. Elomaa and M. Kääriäinen. "An analysis of reduced error pruning". In: *Journal of Artificial Intelligence Research* 15.1 (2001), pp. 163–187. ISSN: 1076-9757.

[10] T.A. Feo and M.G.C. Resende. "A probabilistic heuristic for a computationally difficult set covering problem". In: *Operations Research Letters* 8 (1989), pp. 67–71.

[11] T.A. Feo and M.G.C. Resende. "Greedy randomized adaptive search procedures". In: *Journal of Global Optimization* 6.2 (1995), pp. 109–133. ISSN: 0925-5001.

[12] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. 2010.

[13] Y. Freund and R.E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997). cited By (since 1996) 2237, pp. 119–139. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0031211090&partnerID=40&md5=6aa263ad916f3130742d61a6bf8337c3.

[14] Y. Freund and R.E. Schapire. "Experiments with a new boosting algorithm". In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer. 1996, pp. 148–156.

[15] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1998. ISBN: 9780792381877. URL: http://books.google.com/books?id=mFYt0C5cqtAC.

[16] F. Glover, M. Laguna, and R. Martí. "Fundamentals of scatter search and path relinking". In: *Control and Cybernetics* 39.3 (2000), pp. 653–684.

[17] Mark Hall et al. “The WEKA data mining software: an update”. In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[18] T.K. Ho. “The random subspace method for constructing decision forests”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[19] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[20] J. Maudes et al. “Random Feature Weights for Decision Tree Ensemble Construction”. In: *Information Fusion* In Press, Accepted Manuscript (2010). ISSN: 1566-2535. DOI: DOI:10.1016/j.inffus.2010.11.004.

[21] J. Pacheco et al. “Uso del metaheurístico GRASP en la construcción de árboles de clasificación”. In: *Rect@* 11 (2010), pp. 139–154. ISSN: 1575605X.

[22] J.R. Quinlan. “Learning with continuous classes”. In: *5th Australian joint conference on artificial intelligence*. Citeseer. 1992, pp. 343–348.

[23] I.H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann series in data management systems. Morgan Kaufman, 2005. ISBN: 9780120884070.

Chapter 4

Tree ensemble construction using a GRASP-based heuristic and annealed randomness

Authors José-Francisco Díez-Pastor, Cesar García-Osorio, Juan J. Rodríguez

Type Journal

Published in Information Fusion 20: 189-202

Year 2014

Abstract

Two new methods for tree ensemble construction are presented: G-Forest and GAR-Forest. In a similar way to Random Forest, the tree construction process entails a degree of randomness.

The same strategy used in the GRASP metaheuristic for generating random and adaptive solutions is used at each node of the trees. The source of diversity of the ensemble is the randomness of the solution generation method of GRASP. A further key feature of the tree construction method for GAR-Forest is a decreasing level of randomness during the process of constructing the tree: maximum randomness at the root and minimum randomness at the leaves. The method is therefore named "GAR", GRASP with Annealed Randomness.

The results conclusively demonstrate that G-Forest and GAR-Forest outperform Bagging, AdaBoost, MultiBoost, Random Forest and Random

subspaces. The results are even more convincing in the presence of noise, demonstrating the robustness of the method.

The relationship between base classifier accuracy and their diversity is analysed by application of kappa-error diagrams and a variant of these called Kappa-error relative movement diagrams.

Index terms— GRASP metahuristic, Decision Trees, Classifier ensembles, Boosting, Bagging, Random Forest, Random Subspaces, Kappa-error Relative Movement Diagrams

4.1 Introduction

In this paper, we introduce a new ensemble method. Ensemble methods combine several classifiers, referred to as member classifiers or base classifiers, the combination usually achieve better performance than any of the single classifiers [36].

They are intended to generate many classifiers and to combine their outputs, in such a way that the combination of classifiers outperforms the accuracy of any individual classifier when used alone. This requires non-coincidence of the mistakes of individual classifiers. Hence, the classifiers in the ensemble should be diverse [51].

Various methods can ensure high diversity levels in the ensemble classifiers; Kuncheva [36] has described four basic approaches to this problem:

1. Use different methods of combining classifiers;
2. Use different models of classifiers;
3. Use different feature subsets;
4. Use different subsets of the training data.

The most commonly used type of ensemble, the homogeneous ensemble, involves the same method of base classifier construction. Either of the last two approaches may be used to force diversity in these ensembles. In Random Subspaces [28] each base classifier is trained with a different subset of the attributes, while in Bagging [2] they are trained with a different random sample of the original training set. AdaBoost [17] and Multiboost [57] modify the training set distribution, by taking the results of the previous classifiers into account. Instances with which previous classifiers have had difficulty are weighted in accordance with that level of difficulty, to build the training data for the next classifier in the ensemble. This weight can increase the probability of their use in the next sample. Alternatively, instead of sampling, the weight can be directly applied to the construction of the next classifier, to focus on the classification of unsuccessfully classified instances.

Decision trees perform a partition of the input space into regions, and independently classify each one. This partitioning approach is a recursive method, each step of which divides the space, using the best attribute according to a splitting criterion. Base classifiers can often be decision

trees, as their efficiency and instability help the classification process: slight modifications to the training dataset or the construction method will generate dramatically different classifiers.

Certain methods may be specifically designed for tree ensemble construction. They use a procedure that produces different trees, even though it is the same for all trees in the ensemble, due to the randomness introduced in the way in which the attributes and split points are chosen at each node. In [10], a simple technique randomizes the selection of one of the twenty best splits in each node. Random Forest [3] increases diversity when it combines training set sampling with random selection of subsets of attributes at each tree node; thus, the splits are only considered in each node within the selected subset of attributes. The Random Feature Weights method [44] makes use of all attributes, each with a differing probability of being used as a split. These probabilities are subject to the weight that is assigned to each attribute. These weights are randomly generated and are different between trees, so diversity is the ensemble is ensured.

The GRASP metaheuristic [14, 15] is an iterative procedure, each iteration consists of two parts:

1. Construction of the starting solution.

2. Attempt to find improvements on the built solution through a local search, updating the solution when an improvement is achieved.

. This iterative process continues until a stopping condition is satisfied. In the first part the solution is generated incrementally using a randomized and adaptable method. This method consists of adding elements to the solution. These elements are randomly extracted from the *RCL* (*Restricted Candidate List*), the list of elements "close enough" to the best element. This closeness is defined in terms of a parameter α. On the contrary, a deterministic algorithm takes the best decision at each step.

GRASP and similar metaheuristics try to overcome the limitations of deterministic algorithms (based on adding what is apparently the best element to the partial solution); that is, that they are often unable to ensure that the solution is a global optimum, because of their limited exploration of the search space.

The knapsack problem shown in Table 4.1 is a clear example, given a knapsack capacity and a specific set of attributes for 4 objects: object weight, object price and the the ratio between these two values. The objective is

Table 4.1: Knapsack problem

Element	**W**eight	**V**alue	$Ratio = V/W$
1	20	20	1.0
2	12	18	1.5
3	10	14	1.4
4	10	8	0.8

Weight limit of the knapsack: 20 weight units

to find the subset that maximizes the sum of prices and does not reach the weight limit. One greedy deterministic strategy would be to select the object with best ratio every time, the selected object would be element 2, no more space is left in the knapsack for any other object, and the total value would be 18. If the selection criterion is the value, the selected object would be element 1, which would fill the bag and give a total value of 20; nevertheless, a superior solution would involve a choice of elements 3 and 4 that improves the preceding solution by two. This optimal solution can not be found using deterministic greedy algorithms (such as those usually used by decision trees algorithms), whereas the solution is accessible for randomized algorithms, such as GRASP.

The items in the *RCL* are those that meet the following criteria:

$$RCL = \{i : Value_i \geq \alpha \cdot MaxValue + (1-\alpha) \cdot MinValue\} \tag{4.1}$$

If $\alpha = 1$, only the best item could be selected, the same that would have been selected by the deterministic greedy strategy. If $\alpha = 0$, all possible elements will be in the list and the selection will be random.

G-Forest, proposed in [12] and close to the Random Forest method, is based on the GRASP heuristic (solely as regards the greedy, random and adaptive solutions). Unlike Random Forest, rather than selecting the best attribute from among a random subset of attributes, one attribute is selected at random from a list of candidate attributes. A similar procedure selects the split point. The length of the candidate attribute list depends on the quality of those attributes (attribute value for splitting the dataset into subsets, each with instances whose class values are as homogeneous as possible) and on a parameter α. The same approach as in the GRASP metaheuristic that generates random and adaptive solutions is used to form this list. A list with $\alpha = 1$ only includes the best attribute (or multiple attributes if they all share the maximum value) and a list with $\alpha = 0$ would include all possible attributes.

Usually, increased diversity leads to a loss of accuracy in the individual classifiers[1]. A novel method for tree ensemble construction is presented, which increases diversity trying not to affect accuracy. To do this, the following ideas are taken as starting point:

- The choice made at the root of a tree is the most influential in its final structure. Small changes in the root of a tree can produce totally different trees.
- The correct classification of instances is mainly influenced by the lower nodes and leaves[2].

A method of tree construction is proposed in which both attribute and split point selection are absolutely random at the tree root (that is, random and with no bias), and then the randomness decreases as the tree building process advances (that is, the selection of attributes is still random, but with a bias in favour of the best attributes and split points), to the point that, at the leaves, the choice of attributes and split points becomes highly deterministic.

In this paper we extend the work of [12] by:

- Presenting an improvement, GAR-Forest, which dynamically varies the parameter α of G-Forest.
- Analysing its tolerance to class noise.
- Studying how the method influences ensemble diversity.
- Examining the results after performing parameter optimization.

In the same way as in [12], random selections of attributes and split points are taken from a list of reliable candidates, the size of which is controlled by a parameter α. The significant novelty is the fact that the parameter value varies from $\alpha = 0$ at the root up to $\alpha = 1$ at the leaves, growing in a manner that is inversely proportional to the number of instances that reach each node.

The layout of this paper is as follows: Section 4.2 discusses the theoretical background of trees used as ensemble members and GRASP metaheuristics. Section 4.3 explains the experimental setting and shows the results. In

[1] Nevertheless, ensemble accuracy improves, as greater diversity offsets loss of base classifier accuracy.
[2] Considering that trees are constructed starting at the root and growing downward to the leaves.

section 4.4, kappa-error diagrams and kappa-error relative movement diagrams are used to explain the effect of the method in the ensemble accuracy and diversity. The influence of one particular parameter of the proposed method is discussed in section 4.5. Finally, in section 7.7, the conclusions and future research lines are presented.

4.2 Decision trees and their use as ensembles members

The procedures used to build decision trees are top-down algorithms. All instances at tree root are used to identify the best attribute with which to split the dataset into two or more subsets linked to new nodes that are the offspring of the root node. As a process, a recursive repetition of each new node continues until the stop criterion is verified. A merit function determines the best selection of the attribute for each node. Commonly used split criteria are: Gini Index [4], Information Gain and Gain Ratio [53]. Gain ratio is used in this work as the merit function.

The construction of a decision tree requires a training dataset T, with attributes $a_1, \ldots, a_n$, and a splitting criterion $f(T, a_i)$, which computes a value to the i-th attribute. A widely used splitting criterion is *Gain Ratio* defined in [52] as:

$$Gain(T, a_i) = \mathscr{S}(T) - \sum_{v \in V(a_i)} \frac{|T_v|}{|T|} \mathscr{S}(T_v) \tag{4.2}$$

where, T is the training dataset, a_i is the attribute to evaluate, $V(a_i)$ is the set of all different values for attribute a_i, and T_v is the subset of the data set T formed by the examples, where $a_i = v$, and $\mathscr{S}$, the entropy, for c class labels, is defined as

$$\mathscr{S}(T) = \sum_{i=1}^{c} -p_i \log_2(p_i) \tag{4.3}$$

where p_i represents the probability of class label i.

When dealing with numeric attributes, the splits are binary, with two subsets, one with all the instances whose attribute value is lower or equal than the chosen split point, and the other with all the instances with attribute values greater than the chosen split point. Besides, numeric attributes, unlike nominal attributes, have many possible split points. To select the best one, the instances are ordered according to one particular attribute, then the merit function is evaluated for each split point for which the corresponding

instance has a different class from the next one. Typical algorithms for tree construction select the attribute with the highest value of the Gain Ratio for each node (or any other chosen merit function), assuming that this is the best attribute with which to split the data set.

The construction of an ensemble involves two conflicting objectives: to build base classifiers that are as diverse as possible and to preserve their accuracy. A commonly used technique to introduce diversity in tree ensembles is to discard certain information when calculating the splitting point:

1. Ignore attributes: these attributes may vary at each node, as in Random Forest [3], or they may be the same for the whole tree, but vary from one tree to another in the ensemble, as in Random Subspaces [28].

2. Ignore both attributes and splitting points: for example, extremely randomized trees [23] take k random attributes into account at each node, as in Random Forest, but the splitting point in each attribute is also randomly chosen.

Another technique to introduce further diversity is to introduce a bias in the choice of attributes and split points:

1. Random Feature Weights [44] tries to increase the diversity of the ensemble by adding a different set of random weights to each tree. This set has one weight per attribute. Unlike traditional algorithms, that only take into account the Gain Ratio or similar measures, Random Feature Weights jointly uses the Gain Ratio and the random weight. A high weight contribute positively to the choice of the attributes, a low weight reduce the probability of the attribute to be chosen.

2. Randomization [10] uses a very straightforward technique, at each decision point, it computes the best 20 splits and chooses one at random.

3. Random Linear Oracle [37] can be seen as a tree with a random split at the root and in an arbitrary direction, not necessarily parallel to the feature axes, but in a direction defined by any random linear combination of attributes; the rest of the nodes of the tree are constructed in the traditional way. This approach encourages extra diversity in the ensemble while allowing for high individual ensemble-member accuracy.

4.2.1 The use of the GRASP metaheuristic as a means of increasing diversity in the tree construction

The use of heuristics and meta-heuristics in statistics and data mining is a relevant approach with a promising future [58]. Many problems in data mining can be considered as a combinatorial optimization problem. The operations research community has contributed significantly to the data mining field. A survey of heuristic and metaheuristic methods that solve data mining problems can be found in [47].

Metaheuristics are usually chosen over other optimization methods, when it is necessary to find a good solution to a complex problem with many local optima.

The family of the metaheuristics includes such methods as simulated annealing [34], genetic algorithms [26], tabu search [25, 24], variable neighbourhood search (VNS) [45] and GRASP.

Some data mining problems to which this type of metaheuristics has been applied are attribute selection [59, 33, 8], learning the structure of a Bayesian network [39], clustering [1, 50], and decision trees [18, 41, 55].

GRASP (Greedy Randomized Adaptive Search Procedure) [14, 15] is a metaheuristic strategy initially proposed to solve the set covering problem, which is known to be NP-complete. Widely used as a strategy in optimization problems, this is a constructive strategy, based on gradually adding elements to a solution until it is complete and meets a stop criterion. The idea behind GRASP, and what differentiates this method from other constructive strategies, is that the best step at each point of the process does not necessary lead to the best solution of the problem.

Recently, GRASP has been applied to alter the selection method of attributes in the construction of binary decision trees [49, 48]. The idea is to randomly choose attributes from a Restricted Candidate List that lists attributes with sufficiently close values to the best one:

$$RCL = \{i : \mathsf{Value}_i \geq \alpha \cdot \mathsf{bestValue} + (1-\alpha) \cdot \mathsf{worstValue}\} \tag{4.4}$$

Note, that $\alpha = 1$ gives a RCL that only contains the attribute for which the merit functions gives the best value. With $\alpha = 0$ all attributes may be selected, regardless of whether their values are good or bad. GRASP controlled randomness managed to build less complex trees without reducing accuracy, while the optimum results were obtained with high values of α

(0.9 or 0.95).

The methods in this study use the GRASP construction phase to construct ensembles of classification trees. One further combines the ideas of [12] with annealed randomness. This process of annealing the randomness is explained below.

GAR-Trees (GRASP with Annealed Randomness trees) are constructed by slight modifications to the traditional tree construction algorithm. Attribute evaluation at each node involves a specific merit function, in this case the Gain Ratio, with a value that represents the efficiency of the attribute at dividing the dataset into subsets, each having class values of their instances that are fairly homogeneous. The attribute to be used in the node is selected from a candidate list, constructed from those same values. Note that for $\alpha = 1$, the same attribute as a traditional method would be chosen and for $\alpha = 0$, the attribute will be fully random.

Our interest consists in increasing diversity without overly reducing accuracy, so α will take an initial value of 0 at the root of the tree and it will be increased until it reaches a maximum value, close to 1, in the lower nodes. Equation 4.5 shows the function that controls how randomness is reduced.

$$\alpha = \left(1 - \frac{\mathsf{numInstancesNode}}{\mathsf{numInstances}}\right)^{\tau} \tag{4.5}$$

In this equation, α is determined by the fraction of instances that reach the node and a parameter τ which controls how rapidly α is increased with the reduction of instances reaching the node. Figure 4.1 shows the evolution of α in terms of the percentage of instances reaching a node and the exponent τ.

A parallel may be drawn with simulated annealing [34], in which the heuristic search can accept a solution that is worse than the best solution found at that time. The probability of this solution will depend on the degree to which the new solution is worse than the best one and a further parameter, which is a synthetic temperature inspired by the annealing process in metallurgy. As this parameter value gradually descends, so too does the randomness of the method (the lower the value, the lower the randomness). In the proposed method, randomness decreases as the number of instances that reach each node descends. The randomness of the decisions may be said to fall at each node, as the tree construction process progresses.

A linear relationship exists with an exponent of $\tau = 1$ between the

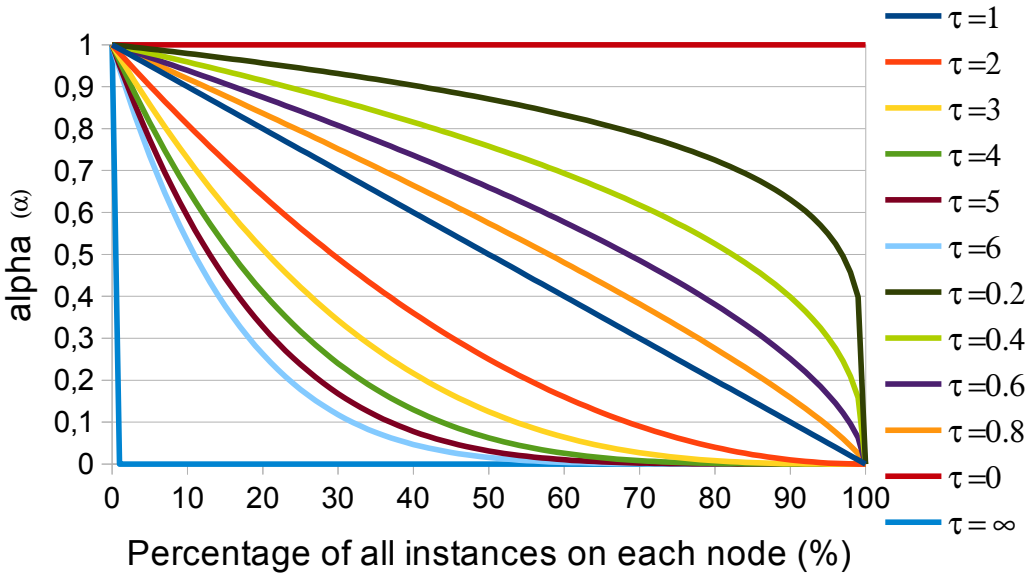


Figure 4.1: Evolution of α in terms of the percentage of instances reaching a node and the exponent τ.

fraction of instances that reach the nodes and the degree of randomness when selecting the attribute and the splitting point. With exponents greater than 1, the level of randomness at the upper tree nodes, where the number of instances is a large fraction of all instances, decreases more slowly, unlike in the lower nodes, where they increase more rapidly. The reverse takes place, with exponents of less than 1; a rapid decrease in randomness at higher nodes and more slowly at lower nodes. Two extreme cases exist, with an exponent of $\tau = 0$, the value of α is equal to 1 throughout the whole tree building process. When $\alpha = 1$, the attributes and split points selected are those with higher values of the merit function, as it happens in the traditional algorithm. With exponent $\tau = \infty$, the value of α always remains at 0. With $\alpha = 0$, each attribute and split point have the same probability of being chosen, so the tree construction process is totally random.

Algorithm 7 shows the GAR-Tree construction procedure, it resembles the traditional algorithms. When there is a sufficient number of instances, the following sequence of actions is executed:

1. The α value for each node is calculated using Equation 4.5 (a key difference between GAR-Trees and G-Trees is that in the later α is a parameter that is constant throughout the tree construction);

2. The importance of each attribute is calculated using the merit function.

3. The set of elements forming the candidate list is calculated using the

Algorithm 7: TrainDecisionTree (for numeric attributes)

Input: Training dataset T, set of m attributes Attributes, total number of instances totalNum, a value that controls the level of annealed randomness τ
Output: Tree
if Attributes *is empty or number of examples* $<$ *minimum allowed per branch* **then**
  Node.label $\leftarrow$ most common value label in examples
  return Node;
else
  numInstancesNode $\leftarrow$ size(T)
  $\alpha \leftarrow (1 - (\text{numInstancesNode}/\text{totalNum}))^{\tau}$
  for $j = 1$ *to* m **do**
    model[j]$\leftarrow$ `GraspSplit` (T, Attributes, j, α)
  end
  maxGain $\leftarrow$ Max(model.gain); minGain $\leftarrow$ Min(model.gain)
  List $\leftarrow \{j = 1,2,\ldots,m |$ model[j].gain $\geq \alpha$maxGain $+ (1-\alpha)$minGain$\}$
  Randomly choose $j_g \in$ List; Att = j_g, splitPoint = model[j_g].splitPoint
  $T_l \leftarrow \{\mathbf{x}_i \in T | \mathbf{x}_{i,j_g} \leq splitPoint\}$
  $T_r \leftarrow \{\mathbf{x}_i \in T | \mathbf{x}_{i,j_g} > splitPoint\}$
  Node.child[0] $\leftarrow$ `TrainDecisionTree` (T_l, Attributes, totalNum, exponent)
  Node.child[1] $\leftarrow$ `TrainDecisionTree` (T_r, Attributes, totalNum, exponent)
end

maximum and minimum merit values, and the α value.

4. The model, which contains the attribute and split point values, is randomly chosen from the candidate list and used to split the training set.

As shown in Algorithm 8, a similar strategy is used for the selection of splitting points. A list of good candidates is generated once again, one of them is randomly selected and its gain and split value are returned to the main procedure. This contrasts with the traditional algorithm that always returns the best split value. Randomness therefore exists in each node, when the attribute is selected and when the splitting point within that attribute is selected.

4.3 Experimental setup and results

In this section, ensembles of classification trees that use the GRASP strategy are compared to ensembles of regular classification trees. In the experiments, both kinds of GRASP trees are used. The term G-Trees refers to trees without randomness annealing, as presented in [12], and the term GAR-Trees refers to the improved version presented in this paper, that

Algorithm 8: GraspSplit

Input: Training dataset T, set of m attributes Attributes, attribute index j, value between 0 and 1 to control the level of randomness α
Output: model
$\mathbf{X} \leftarrow T$ ordered according to attribute j
`/* Compute values for all possible split points and their index */`
infoGain$\leftarrow$ List of all possible split info gains
infoGainIndex$\leftarrow$ List of all possible split indexes
maxGain $\leftarrow$ Max(infoGain)
minGain $\leftarrow$ Min(infoGain)
List $\leftarrow \{k = 1,2,\ldots,size(\text{infoGain}) | \text{infoGain}[k] \geq \alpha\text{maxGain} + (1-\alpha)\text{minGain}\}$
Randomly choose $k_g \in$ List
model.gain $\leftarrow$ infoGain$[k_g]$
splitIndex $\leftarrow$ infoGainIndex$[k_g]$
model.splitPoint $\leftarrow (\mathbf{x}_{\text{splitIndex},j} + \mathbf{x}_{\text{splitIndex}+1,j})/2$
return model

uses annealed randomness. The scoring function is the comparisons was accuracy.

From the implementation point of view, both GRASP-based trees are modifications of J48, the Java version of C4.5 [53] provided by Weka [27].

In the experiments, these trees were used as base classifiers of Bagging. From this library are also the other decision trees and ensembles. The ensembles size was set to 100.

The G-Trees were tested with six values for the parameter α, between 0.05 and 0.3 (at intervals of 0.05). Interest in increasing the diversity of ensemble methods based on these trees is aided by low alpha values, which assure a high degree of randomness.

GAR-Trees were tested with integer exponents from $\tau = 1$ to $\tau = 6$.

We compare both methods with the ensembles listed below. Unless specified, no changes to the default parameters were made:

1. Boosting: Multiboost [57] and AdaBoost.M1 [17]. In the two versions, the variants with reweighting and resampling were used (represented as W and S in the table). Multiboost had 10 subcommittees.

2. Bagging [2].

3. Random Subspaces [28]: two separate configurations were considered: 50% and 75% of the feature size.

4. Random Forest [3]: four separate configurations, random attribute

subsets with sizes of 1, 2, square root and log base 2 of the feature space size.

The pruned and non-pruned versions were applied to all ensembles of J48 trees. Only the unpruned version of Random Forest was used, as pruning is not recommended [3]. Pruning was also by-passed for ensembles of G-Trees and GAR-Trees, as pruning in randomized trees usually does not give good results. Moreover, unpruned trees yielded the best results in [12].

Four were the reasons for using binary trees:

1. This type of tree was applied in [49], a preliminary work that used GRASP to modify the attribute selection process in the construction of a binary decision tree.

2. When a non-binary tree handles a nominal attribute, a node is created with as many children as different values has that nominal attribute. Having only one possible decision to divide the dataset, it is not possible to create a candidate list, so the GRASP strategy is not applicable.

3. In [12], with binary trees as base classifiers, the performance of all the ensembles improved slightly.

4. Most attribute scores (except a few, like Relief) [54] overestimate multivalued attributes, which can be avoided by inducing binary trees.

As the implementation of Random Forest in Weka does not have the option of using binary nodes, the pre-processing method discussed in [4] was applied, in which nominal k value attributes are converted into k binary attributes.

A 5×2 cross validation [11] was performed using 62 UCI data sets [16], their main characteristics are shown in Table 4.2. Table 4.3 contains the results as an average ranking [9]. The configurations under consideration are sorted from best to worst in average ranks, for each data set, each configuration receiving a numeric rank: the best method is ranked 1, the second best, 2.... If several configurations yield the same result, they receive an average rank. The average rank for each configuration is obtained by averaging across all the data sets. According to Demsar [9] a reasonable fair comparison between algorithms is provided by average ranks.

Table 4.3 shows the average ranking for all tested configurations over the 62 dataset listed in Table 4.2. The first 5 positions correspond to 5

Table 4.2: Datasets used in the experiments (the number of **N**umeric and **D**iscrete features, **E**xamples and **C**lasses are indicated with: #N, #D, #E and #C).

Dataset	#N	#D	#E	#C
abalone	7	1	4177	28
anneal	6	32	898	6
audiology	0	69	226	24
autos	15	10	205	6
balance-scale	4	0	625	3
breast-w	9	0	699	2
breast-y	0	9	286	2
bupa	6	0	345	2
car	0	6	1728	4
credit-a	6	9	690	2
credit-g	7	13	1000	2
crx	6	9	690	2
dna	0	180	3186	3
ecoli	7	0	336	8
glass	9	0	214	6
heart-c	6	7	303	2
heart-h	6	7	294	2
heart-s	5	8	123	2
heart-statlog	13	0	270	2
heart-v	5	8	200	2
hepatitis	6	13	155	2
horse-colic	7	15	368	2
hypo	7	18	3163	2
ionosphere	34	0	351	2
iris	4	0	150	3
krk	6	0	28056	18
kr-vs-kp	0	36	3196	2
labor	8	8	57	2
led-24	0	24	5000	10
letter	16	0	20000	26
lrd	93	0	531	10

Dataset	#N	#D	#E	#C
lymphography	3	15	148	4
mushroom	0	22	8124	2
nursery	0	8	12960	5
optdigits	64	0	5620	10
page	10	0	5473	5
pendigits	16	0	10992	10
phoneme	5	0	5404	2
pima	8	0	768	2
primary	0	17	339	22
promoters	0	57	106	2
ringnorm	20	0	300	2
sat	36	0	6435	6
segment	19	0	2310	7
shuttle	9	0	58000	7
sick	7	22	3772	2
sonar	60	0	208	2
soybean	0	35	683	19
soybean-small	0	35	47	4
splice	0	60	3190	3
threenorm	20	0	300	2
tic-tac-toe	0	9	958	2
twonorm	20	0	300	2
vehicle	18	0	846	4
vote1	0	15	435	2
voting	0	16	435	2
vowel-context	10	2	990	11
vowel-nocontext	10	0	990	11
waveform	40	0	5000	3
yeast	8	0	1484	10
zip	256	0	9298	10
zoo	1	15	101	7

of the 6 tested configurations of GAR-Forest. The first thing to check for any significant differences between the ranks of the compared methods. The Iman-Davenport [32] test, an enhanced version of the Friedman test, was used to resolve this issue. The Iman-Davenport test gives a p-value of 1.5569e-66, meaning that it rejects the hypothesis of the equivalence of the algorithms under comparison. Having verified the statistical difference of the methods, to detect pairwise differences between the best method and the remaining methods, the Hochberg test [30] was used as a post-hoc test, which was found to be more powerful and less conservative than the Bonferroni-Dunn test [13, 21], which makes it more suitable for comparisons with a large number of methods. Using the Hochberg test, the methods that are significantly different from the best method in Table 4.3 at a confidence level of 0.10 appear below the horizontal dashed line. Those that are significantly different at a confidence level of 0.05 appear below the horizontal unbroken line. According to the Hochberg test, the best method in Table 4.3 is equivalent to those methods above the horizontal dashed line with significance level of 0.1, and equivalent to those above the horizontal unbroken line with significance level of 0.05, and it is significantly superior to those below the aforementioned lines.

The high number of methods compared and the fact that many of them differ only in the value of a parameter (so some correlation may exist between them) makes it advisable to analyze the results, grouping the methods into families. Those methods that use the same base algorithm have been specifically grouped together, thus it is possible to make a hierarchical analysis similar to that used in [19]. Table 4.4 shows the rankings for each family and the corresponding Iman-Davenport p-values. The lines indicate the methods that, according to the Hochberg test, are significantly different from the method with best ranking.

Table 4.5 compares the best methods for each family. GAR-Forest is the best according to the ranking. The order in which the best methods appear in Table 5 can give a general idea of their relative performance, although these results should be treated with caution, because the selection of the best methods has been done using the same set of results used to form the ranking.

Table 4.3: Average ranks for the different algorithms (**U**npruned and **P**runed trees).

Rank	Method
10.718	GAR-Forest $\tau = 4$
10.798	GAR-Forest $\tau = 3$
11.177	GAR-Forest $\tau = 5$
11.444	GAR-Forest $\tau = 2$
11.484	GAR-Forest $\tau = 6$
12.468	G-Forest $\alpha = 0.30$
12.613	Random Forest $k = \sqrt{N}$
12.919	G-Forest $\alpha = 0.20$
13.000	G-Forest $\alpha = 0.25$
13.379	GAR-Tree $\tau = 1$
14.073	G-Forest $\alpha = 0.15$
14.089	Random Forest $k = \log_2 N$
14.315	G-Forest $\alpha = 0.10$
14.637	Multiboost -S (U)
15.242	Multiboost -S (P)
15.452	AdaBoost -S (P)
16.000	Random Forest $k = 3$
16.242	Multiboost -W (U)
16.274	G-Forest $\alpha = 0.05$
16.315	Adaboost -S (U)
16.419	Random Forest $k = 2$
16.718	Multiboost -W (P)
17.419	AdaBoost -W (P)
18.218	AdaBoost -W (U)
18.879	Random Forest $k = 1$
19.315	Random Subspaces 50% (U)
20.468	Random Subspaces 50% (P)
22.589	Bagging (P)
22.766	Bagging (U)
25.226	Random Subspaces 75% (P)
25.347	Random Subspaces 75% (U)

Table 4.4: Rankings by algorithm families (the p-values for Iman-Davenport are shown at the bottom of each ranking).

Rank	Method
1.419	Bagging (U)
1.581	Bagging (P)
	p-value=2.066e-1
2.040	AdaBoost -S (P)
2.508	AdaBoost -W (P)
2.540	Adaboost -S (U)
2.911	AdaBoost -W (U)
	p-value=2.203e-3
2.234	Multiboost -S (U)
2.419	Multiboost -S (P)
2.613	Multiboost -W (U)
2.734	Multiboost -W (P)
	p-value=1.452e-1
2.379	Random Forest $k = \sqrt{N}$
2.750	Random Forest $k = logN$
3.113	Random Forest $k = 3$
3.129	Random Forest $k = 2$
3.629	Random Forest $k = 1$
	p-value=1.680e-4
1.758	Random Subspaces 50% (U)
2.234	Random Subspaces 50% (P)
2.903	Random Subspaces 75% (U)
3.105	Random Subspaces 75% (P)
	p-value=2.189e-10
2.960	G-Forest $\alpha = 0.30$
3.194	G-Forest $\alpha = 0.25$
3.274	G-Forest $\alpha = 0.20$
3.556	G-Forest $\alpha = 0.15$
3.685	G-Forest $\alpha = 0.10$
4.331	G-Forest $\alpha = 0.05$
	p-value=7.745e-4
3.169	GAR-Forest $\tau = 5$
3.202	GAR-Forest $\tau = 4$
3.363	GAR-Forest $\tau = 3$
3.500	GAR-Forest $\tau = 6$
3.613	GAR-Forest $\tau = 2$
4.153	GAR-Forest $\tau = 1$
	p-value=3.882e-2

Table 4.5: Comparison of the best method in each algorithm family.

Rank	Method
2.927	GAR-Forest $\tau = 5$
3.379	Random Forest $k = \sqrt{N}$
3.387	G-Forest $\alpha = 0.30$
3.839	Multiboost -S (U)
3.960	Adaboost -S (P)
4.823	Random Subspace 50% (U)
5.685	Bagging (U)
	p-value=3.998e-15

4.3.1 Noise Robustness

Noisy or corrupted data may downgrade the performance of a classification method, it is worth assessing the robustness of a method against noise.

One advantage of ensembles over other classifiers is their potential robustness to noise and other imperfections in the data. Real-world problems do have noise, due to errors in data measurement and in labelling. So, it is relevant to study the behaviour of any learning algorithm in the presence of noise.

In order to test this robustness, artificial datasets were composed by shuffling a percentage of the class values of the original datasets, in the same way as described in [10]. For these artificial noisy data sets, the methods were also evaluated using 5×2-fold stratified cross validation.

Table 4.6 shows the methods sorted according to their average ranks for the data sets with 10% class noise and Table 4.8 with 20% class noise. As in Table 4.3, a horizontal dashed line separates those methods that are significantly different from the best one at a confidence level of 0.10, and a horizontal unbroken line separates the methods that are significantly different from the best one at a confidence level of 0.05.

In Table 4.6, the first five positions correspond to the GAR-Forest method. Also, the only rankings that are not significantly different from the best belong to those methods that use the GRASP strategy for constructing the trees, that is, G-Trees and GAR-Trees. In Table 4.8, the first position corresponds to one of the GAR-Forest configurations, but the next correspond to ensembles that use G-Trees. The two methods introduced in this paper have a ranking that is equivalent to the ranking of the best methods. The table also shows the poor performance of Adaboost due to noise, which occupies the final positions in the ranking.

Table 4.6: Average rank for the different algorithms (**U**npruned and **P**runed trees). Dataset with 10% class noise.

Rank	Method
8.008	GAR-Forest $\tau = 3$
8.734	GAR-Forest $\tau = 5$
8.968	GAR-Forest $\tau = 4$
9.226	GAR-Forest $\tau = 2$
9.234	GAR-Forest $\tau = 6$
9.685	G-Forest $\alpha = 0.25$
10.153	G-Forest $\alpha = 0.30$
10.468	GAR-Forest $\tau = 1$
10.548	G-Forest $\alpha = 0.15$
10.597	G-Forest $\alpha = 0.20$
11.831	G-Forest $\alpha = 0.10$
12.581	G-Forest $\alpha = 0.05$
14.040	Random Forest $k = \sqrt{N}$
14.460	Random Forest $k = \log_2 N$
15.266	Random Subspaces 50% (U)
15.419	Random Forest $k = 3$
15.573	Random Subspaces 50% (P)
16.403	Random Forest $k = 2$
16.573	Bagging(P)
18.879	Random Forest $k = 1$
19.758	Bagging (U)
20.024	Random Subspaces 75%(P)
20.218	Multiboost -S (U)
20.363	Multiboost -S (P)
21.371	Multiboost -W (P)
22.798	Random Subspaces 75% (U)
23.411	Multiboost -W (U)
24.145	AdaBoost -S(P)
24.823	Adaboost -S (U)
26.169	AdaBoost -W (U)
26.274	AdaBoost -W (P)

4.3.2 Optimized version

In previous experiments, Random Forest was tested with various fixed values of k, G-Forest with various fixed values of α and GAR-Forest with several fixed values for the exponent τ. In this section, a comparison has been made between these methods using a five folds internal cross validation for parameter setting.

Table 4.13 shows the average ranking of the 62 data sets shown in Table 4.2, at different noise levels (the ranking was given by the ordering of the methods according to their accuracy). The Iman-Davenport test gives p-values of 4.956e-2, 1.786e-5 and 3.082e-8, which means that it rejects the hypothesis of the equivalence of the algorithms under comparison. Taking Random Forest as the reference method, a post-hoc Hochberg test was also performed. The methods that were significantly different from Random

Table 4.7: Ranking by algorithm families with dataset with 10% of class noise (the p-values for Iman-Davenport are shown at the bottom of each ranking).

Rank	Method
3.024	G-Forest $\alpha = 0.25$
3.177	G-Forest $\alpha = 0.30$
3.371	G-Forest $\alpha = 0.20$
3.452	G-Forest $\alpha = 0.15$
3.895	G-Forest $\alpha = 0.10$
4.081	G-Forest $\alpha = 0.05$
p-value=3.743e-3	

3.065	GAR-Forest $\tau = 3$
3.234	GAR-Forest $\tau = 4$
3.363	GAR-Forest $\tau = 6$
3.379	GAR-Forest $\tau = 5$
3.653	GAR-Forest $\tau = 2$
4.306	GAR-Forest $\tau = 1$
p-value=9.834e-3	

Forest, according to the Hochberg test, at a confidence level of 0.05, appear above the horizontal line. In the case of data sets without the added noise, although the G-Forest is not significantly better than random forest, its improvement, GAR-Forest, increases the difference and makes it significant. With noisy data sets, both methods are significantly better than Random Forest. The actual scores are shown in tables 5.7, 4.11 and 4.12, where the wins, draws and losses are also shown at the bottom of the table, the second line shows the statistical significant results, according to the Student's t-test for $\alpha = 0.05$ (the corrected version suggested in [46], the default in Weka). The significant victories and losses are also marked in the tables with ✓and * respectively. Table 5.7 shows the results for the original data sets. Tables 4.11 and 4.12 show the results for the datasets with 10% and 20% class noise.

4.4 Kappa-error diagrams

According to the decomposition of generalization error [35, 22], the increased accuracy of ensembles with respect to individual classifiers is due to an increase in diversity or a reduction in the error of the individual classifiers in the ensemble. Visualization techniques such us kappa-error diagrams are useful to understand the behaviour of ensembles in terms of diversity and error [42].

In this visualization technique, each point represent a couple of classifiers

Table 4.8: Average rank for the different algorithms (**U**npruned and **P**runed trees). Dataset with 20% class noise.

Rank	Method
8.847	GAR-Forest $\tau = 4$
9.032	G-Forest $\alpha = 0.25$
9.065	G-Forest $\alpha = 0.3$
9.597	G-Forest $\alpha = 0.1$
9.726	G-Forest $\alpha = 0.2$
9.823	GAR-Forest $\tau = 6$
10.081	G-Forest $\alpha = 0.15$
10.339	GAR-Forest $\tau = 5$
10.702	GAR-Forest $\tau = 3$
10.823	GAR-Forest $\tau = 2$
11.065	G-Forest $\alpha = 0.05$
11.677	Random Subspaces 50% (P)
11.798	GAR-Forest $\tau = 1$
13.403	Random Subspaces 50% (U)
14.016	Bagging (P)
14.782	Random Forest $k = \sqrt{N}$
15.218	Random Forest $k = \log_2 N$
15.669	Random Forest $k = 3$
16.000	Random Forest $k = 2$
17.363	Random Subspaces 75% (P)
17.798	Bagging (U)
18.379	Random Forest $k = 1$
21.306	Random Subspaces 75% (U)
21.742	Multiboost -S (P)
22.710	Multiboost -S (U)
22.944	Multiboost -W (P)
23.702	Multiboost -W (U)
26.032	AdaBoost -S (P)
27.145	AdaBoost -W(P)
27.355	Adaboost -S (U)
27.863	AdaBoost -W(U)

belonging to the ensemble, so, with N classifiers in the ensemble, the number of points will be $N \cdot (N-1)/2$. For every combination of two classifiers there is a point, its coordinates are the kappa diversity measure and the mean error of the classifiers. An appropriate measure to evaluate the diversity is kappa, which is defined as:

$$\kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \tag{4.6}$$

where Θ_1, for L classes and m test samples, is defined as a measure of the agreement between the two classifiers

$$\Theta_1 = \frac{\sum_{i=1}^{L} C_{i,i}}{m} \tag{4.7}$$

where, C is a contingency matrix, each $C_{i,j}$ is the cardinality of the set of instances classified as class i by one classifier and as class j by the other.

Table 4.9: Ranking by algorithm families with dataset with 20% of class noise (the p-values for Iman-Davenport are shown at the bottom of each ranking).

Rank	Method
3.258	G-Forest $\alpha = 0.10$
3.323	G-Forest $\alpha = 0.25$
3.484	G-Forest $\alpha = 0.30$
3.508	G-Forest $\alpha = 0.15$
3.589	G-Forest $\alpha = 0.20$
3.839	G-Forest $\alpha = 0.05$
p-value=5.856e-1	

2.782	GAR-Forest $\tau = 4$
3.202	GAR-Forest $\tau = 6$
3.266	GAR-Forest $\tau = 5$
3.645	GAR-Forest $\tau = 3$
3.710	GAR-Forest $\tau = 2$
4.395	GAR-Forest $\tau = 1$
p-value=3.725e-5	

And Θ_2, an estimate of the probability that two classifiers agree by chance, is defined as:

$$\Theta_2 = \sum_{i=1}^{L} \left(\sum_{j=1}^{L} \frac{C_{i,j}}{m} . \sum_{j=1}^{L} \frac{C_{j,i}}{m} \right) \tag{4.8}$$

The values of κ are in the interval -1 to 1. If the pair of classifiers predict the same class for all test samples, $\kappa = 1$. If the classification for both classifiers were totally random, κ would be 0. If the agreement in the classification were less than the expected by chance, $\kappa < 0$.

It would be advisable that all the points were at the bottom left of these diagrams, that would mean that the classifiers in the ensemble are both accurate and diverse. Unfortunately, accuracy and diversity are in conflict, as the classifiers can not be very accurate and very diverse at the same time.

Figure 4.2 shows the kappa-error diagrams for *Car* data set, and the four methods: Random Forests with $k = \sqrt{N}$, bagging of G-Trees with $\alpha = 0.3$, and bagging of GAR-Trees with $\tau = 4$ and $\tau = 5$. Figure 4.3 shows the kappa-error diagrams for the same four methods, but applied to the Krk dataset.

With regard to Random Forest, in general, the new proposed method not only reduces the average error of the classifiers in the ensemble, but also the diversity.

These diagrams only show the information for one data set. A much more useful representation technique for simultaneous visualization of information for several data sets, is the *kappa-error relative movement*

Table 4.10: Scores of the selected methods for datasets without added noise (results where G-Forest and GAR-Forest are significantly better/worse are marked with ✓/*).

Dataset	Random Forest	G-Forest	GAR-Forest
abalone	24.42	25.02	24.81
anneal	99.15	98.84	98.64
audiology	74.78	76.19	76.55
autos	76.40	73.08	75.03
balance-scale	84.32	86.72	85.41
breast-w	96.91	96.80	96.62
breast-y	71.05	71.54	70.91
bupa	68.53	66.03	66.44
car	92.64	96.23✓	96.28✓
credit-a	86.03	86.52	86.35
credit-g	75.10	74.24	74.54
crx	86.06	87.10	86.90
dna	95.10	94.88	94.97
ecoli	84.76	85.89	85.95
glass	76.17	72.43	74.58
heart-c	80.99	83.24	82.25
heart-h	81.70	80.48	80.20
heart-s	92.85	93.17	92.69
heart-statlog	83.26	83.11	82.07
heart-v	75.40	76.60	76.90
hepatitis	83.48	83.10	83.87
horse-colic	84.29	85.05	85.38
hypo	98.77	98.80	98.98
ionosphere	92.71	92.36	92.99
iris	94.67	95.20	94.93
kr-vs-kp	98.77	99.34✓	99.29✓
krk	78.61	77.16*	79.88✓
labor	87.76	88.79	87.75
led-24	74.17	74.71	74.30
letter	95.07	95.36	95.50
lrd	87.19	87.27	87.83
lymphography	84.73	82.84	82.57
mushroom	100.00	100.00	100.00
nursery	97.45	99.15✓	99.35✓
optdigits	97.85	97.81	98.16
page	97.06	97.19	97.26
pendigits	98.97	98.97	99.07
phoneme	89.03	86.79	88.69
pima	75.76	75.86	76.35
primary	42.72	43.54	42.37
promoters	87.36	84.34	85.66
ringnorm	95.80	93.47	91.33
sat	90.89	90.41	90.99
segment	97.31	97.48	97.72
shuttle	99.98	99.97	99.98
sick	98.19	98.29	98.46
sonar	79.62	77.88	80.67
soybean-small	100.00	100.00	99.15
soybean	91.74	92.27	92.56
splice	95.66	96.00	95.38
threenorm	83.07	80.40	81.07
tic-tac-toe	91.11	94.95	96.56✓
twonorm	94.80	95.27	95.33
vehicle	74.11	75.30	75.37
vote1	90.80	91.63	91.31
voting	96.09	95.45	96.00
vowel-context	91.92	92.40	92.30
vowel-nocontext	92.00	91.96	92.22
waveform	84.89	85.81✓	85.38
yeast	59.62	60.66	60.84
zip	95.89	95.30	96.00
zoo	93.67	93.47	93.49
	Win-tie-loses	34-2-26	38-1-23
Statistically significant win-tie-loses		4-57-1	5-57-1

Table 4.11: Scores of the selected methods for datasets with 10% added noise (results where G-Forest and GAR-Forest are significantly better/worse are marked with ✓/*).

Dataset	Random Forest	G-Forest	GAR-Forest
abalone	21.97	22.79	21.92
anneal	85.68	88.80✓	88.80✓
audiology	67.17	67.70	70.00
autos	62.54	63.22	64.00
balance-scale	72.86	74.53	73.82
breast-w	86.32	87.58	87.32
breast-y	61.05	64.13	63.57
bupa	62.26	60.52	60.75
car	81.18	85.43✓	85.80✓
credit-a	76.20	77.71	77.54
credit-g	68.58	68.42	68.94
crx	78.20	78.90	78.81
dna	84.48	84.98	85.43
ecoli	72.32	74.64	74.70
glass	68.60	67.38	66.92
heart-c	73.27	72.87	72.54
heart-h	75.17	75.58	75.10
heart-s	83.09	83.58	83.91
heart-statlog	75.11	75.48	74.96
heart-v	69.60	68.30	68.50
hepatitis	78.07	76.52	76.27
horse-colic	78.26	79.46	79.08
hypo	88.48	88.81	88.73
ionosphere	83.59	82.90	82.91
iris	83.07	86.80	86.93
kr-vs-kp	86.27	88.52✓	88.25✓
krk	66.75	67.20	70.00✓
labor	78.90	75.36	75.39
led-24	66.76	67.48	66.91
letter	83.17	85.12✓	85.36✓
lrd	78.87	78.23	78.83
lymphography	72.84	73.11	70.81
mushroom	89.71	89.86	89.56
nursery	86.52	88.62✓	88.98✓
optdigits	88.09	87.82	88.21
page	86.79	87.52	87.63
pendigits	88.78	88.92	89.03
phoneme	79.11	77.13	79.60
pima	68.49	68.98	69.19
primary	37.52	37.82	37.64
promoters	73.40	73.58	76.79
ringnorm	85.60	82.67	82.53
sat	82.07	81.68	82.15
segment	85.90	86.92	87.10
shuttle	86.85	89.87✓	89.86✓
sick	88.41	88.06	88.08
sonar	70.19	71.44	69.90
soybean-small	87.70	89.42	88.55
soybean	80.44	82.20	83.43
splice	84.73	85.79	86.03✓
threenorm	73.47	71.93	70.40
tic-tac-toe	80.67	82.63	83.40✓
twonorm	84.73	86.00	85.33
vehicle	67.21	68.23	68.09
vote1	81.19	82.16	82.30
voting	85.38	86.71	87.13
vowel-context	77.01	80.73	80.36
vowel-nocontext	78.26	80.83	80.75
waveform	77.03	77.81	77.43
yeast	54.23	54.95	54.49
zip	86.26	86.01	86.48
zoo	80.59	83.77	82.38
	Win-tie-loses	46-0-16	47-0-17
Statistically significant win-tie-loses		6-56-0	9-53-0

Table 4.12: Scores of the selected methods for datasets with 20% added noise (results where G-Forest and GAR-Forest are significantly better/worse are marked with ✓/*).

Dataset	Random Forest	G-Forest	GAR-Forest
abalone	19.06	20.30	19.32
anneal	72.00	76.82✓	76.77✓
audiology	53.98	58.41	59.56
autos	52.09	54.14	55.89
balance-scale	63.14	65.06	64.42
breast-w	74.77	77.40	76.40
breast-y	55.87	56.64	55.52
bupa	55.76	55.24	56.12
car	69.65	73.83✓	74.35✓
credit-a	68.09	68.96	68.52
credit-g	62.40	61.50	61.08
crx	69.45	69.97	69.68
dna	73.74	74.93	75.50✓
ecoli	65.60	67.32	66.85
glass	60.84	60.56	61.96
heart-c	64.22	64.89	63.56
heart-h	65.92	68.57	67.14
heart-s	75.29	75.44	73.97
heart-statlog	64.30	64.22	62.30
heart-v	60.90	62.40	62.60
hepatitis	70.46	68.01	67.88
horse-colic	70.60	71.41	71.09
hypo	77.21	78.08	77.42
ionosphere	71.00	71.68	70.20
iris	72.53	75.20	74.00
kr-vs-kp	73.62	75.94✓	75.58✓
krk	54.46	57.12✓	59.96✓
labor	68.45	67.38	64.58
led-24	58.76	60.04✓	59.17*
letter	71.31	74.85✓	75.00✓
lrd	70.17	70.51	70.70
lymphography	63.38	64.19	64.86
mushroom	78.20	79.00✓	77.63
nursery	75.91	78.17✓	78.45✓
optdigits	77.98	77.99	78.21
page	75.74	77.58✓	77.32✓
pendigits	78.61	78.95✓	78.99
phoneme	68.63	68.33	70.35
pima	59.06	62.71	61.30
primary	28.91	31.39	31.27
promoters	56.23	56.42	60.00
ringnorm	72.93	69.93	69.07
sat	72.78	72.57	72.77
segment	74.65	76.27✓	76.11
shuttle	73.43	79.33✓	79.31✓
sick	77.79	77.22	76.96
sonar	66.92	66.15	67.60
soybean-small	73.70	78.33	78.77
soybean	67.79	70.89	71.22
splice	74.02	75.94	75.68✓
threenorm	63.67	63.00	61.60
tic-tac-toe	68.98	70.17	70.23
twonorm	74.53	74.20	75.13
vehicle	58.53	60.02	59.60
vote1	68.23	70.94	71.13
voting	73.01	75.40	74.44
vowel-context	60.91	66.12✓	64.75
vowel-nocontext	63.86	67.17✓	67.47✓
waveform	69.11	69.64	69.16
yeast	46.24	47.44	46.66
zip	76.37	76.10	76.57
zoo	63.78	69.33	68.73
	Win-tie-loses	48-0-14	49-0-13
Statistically significant win-tie-loses		14-48-0	11-51-0

Table 4.13: Ranks of the three methods for different levels of noise (the p-values at the foot of the tables are for the Iman-Davenport test, the final columns show the p-values for the Hochberg test when the Random Forest is taken as the reference method, methods above the line are significantly different from Random Forest, at a confidence level of 0.05).

Rank	Method	p-Hochberg
1.758	GAR-Forest	3.50e-2
2.056	G-Forest	4.72e-1
2.185	Random Forest	

p-value=4.956e-2

(a) No noise

Rank	Method	p-Hochberg
1.766	GAR-Forest	1.0e-4
1.766	G-Forest	1.0e-4
2.468	Random Forest	

p-value=1.786e-5

(b) 10% noise

Rank	Method	p-Hochberg
1.629	G-Forest	0
1.806	GAR-Forest	2.0e-5
2.565	Random Forest	

p-value=3.082e-8

(c) 20% noise

diagram [43]. This diagram is obtained by summarizing each kappa-error diagram for a data set as a single point, which is calculated as the centroid of the points in the cloud (the centroids are represented by an asterisk in figures 4.2 and 4.3). The aim of these diagrams is to summarize as a single point the results of two methods applied to the same dataset. The y coordinate is the increase in the error of the second method with respect to the first, and the x coordinate is the increase in the kappa value of the second method respect to the first.

In Figure 4.4, these kappa-error relative movement diagrams are used to

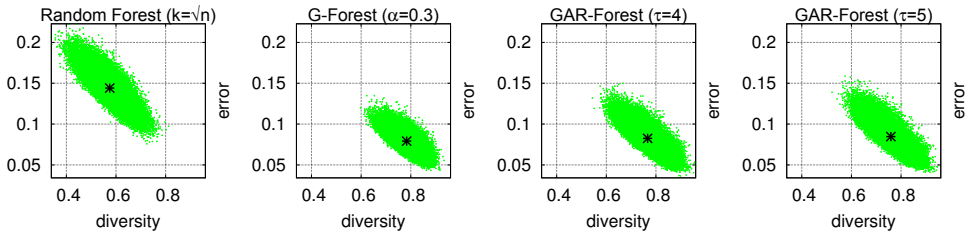


Figure 4.2: Kappa-error diagrams for the car data set.

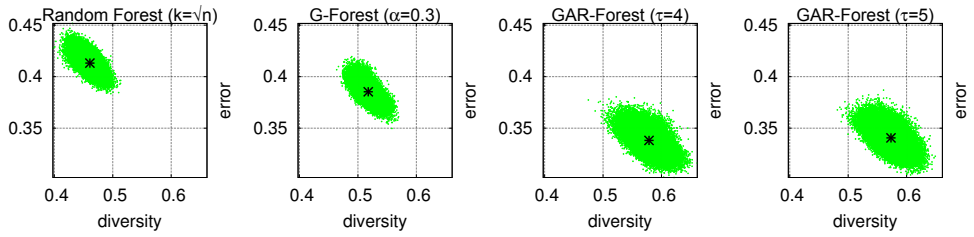


Figure 4.3: Kappa-error diagrams for the krk data set.

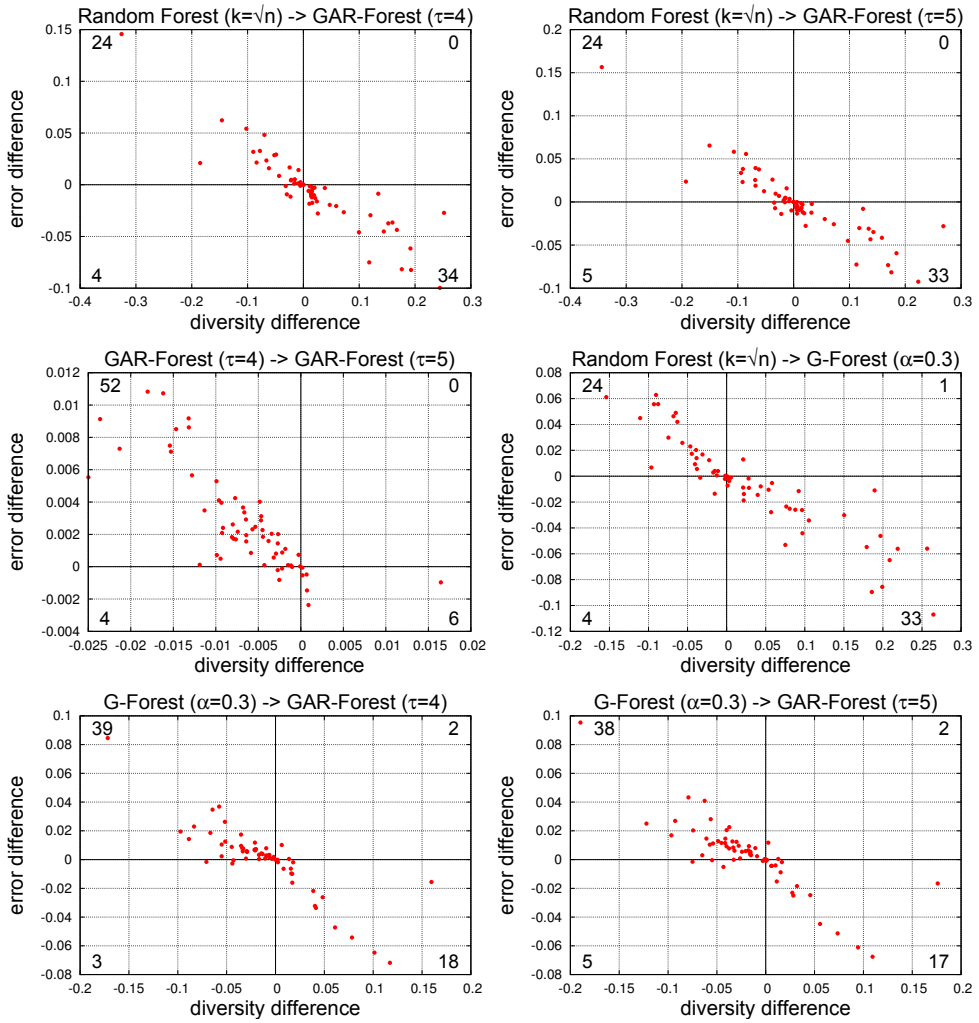


Figure 4.4: Kappa-error relative movement diagrams (the number of points in each quadrant appears at the corresponding corner). Points in the bottom-left quadrant represent datasets for which the second method is better than the first in both error (lower average error in the base classifiers in the ensemble) and diversity (lower kappa), points in the bottom-right quadrant indicate that the second algorithm gives better error but worse diversity, and so on.

show the comparison of several pairs of methods. In the top row, Random Forest with $k = \sqrt{N}$ is compared against GAR-Forest with $\tau = 4$ (left) and $\tau = 5$ (right). In both diagrams, the majority of points are in the lower right quadrant, which means that, for both values of τ, the new method gives more accurate classifiers than Random Forest, but at expense of diversity. In general, both diagrams are quite similar, and it does not appear that the magnitude of the improvement is related to the value of the exponent. However, in the majority of data sets, it is clear that the method with exponent $\tau = 5$ yields ensembles with higher error and diversity than the method with exponent $\tau = 4$, when the two exponent values (middle left) of GAR-Forest are compared. Although, as shown before, this increased diversity was unable to compensate the increased error, and GAR-Forest with exponent $\tau = 4$ yielded, in general, better results than with exponent $\tau = 5$ (see tables 4.3, 4.6 and 4.8).

A comparison between Random Forest with $k = \sqrt{N}$ and Bagging of G-Trees with $\alpha = 0.3$ yields results that are similar to those obtained when comparing with GAR-Trees.

Finally, in the comparison between bagging of G-Trees, G-Forest, and bagging of the two tested configurations of GAR-Trees, GAR-Forest (two lower diagrams in Figure 4.4), it can be seen that an increase in diversity at the expense of a decrease in accuracy occurs for most of the data sets (39 datasets in the case of GAR-Forest with $\tau = 4$, 38 datasets when the exponent is $\tau = 3$). For a few datasets, the improved diversity hardly affects the error (only 3 for $\tau = 4$ and 5 for $\tau = 5$). The error and diversity of GAR-Forest is worse than G-Forest in only 2 datasets. For the rest of dataset, diversity worsens, but the average error of the individual base classifiers in the ensemble improves.

4.5 Influence of the parameter

This section shows how the exponent τ, which is the parameter of GAR-Forest that controls the speed with which randomness decreases, affects the accuracy of the ensemble.

Figure 4.5 shows, for each data set, the evolution of the error as a function of the exponent τ. The value for τ ranges from 0 to 8 at intervals of 0.2. The results shown for the different exponents were obtained from a 5×2 fold cross-validation of GAR-Forest with 100 classification trees. In most of the

data sets, the worst exponent value was 0. But the best value was dependent on the dataset and no clear trend was observed. However, from the average error and average rank (bottom row of Figure 4.5) the best values for τ seems to be between 3 and 5. We do not know yet why the best results are in this range. This issue, which we hope to address in the future, deserves further investigation.

4.6 Conclusion and future lines

This article has examined two new methods for constructing decision tree ensembles: G-Forest and GAR-Forest. Both use the same GRASP meta-heuristic strategy for generating random and adaptive solutions at each node. GAR-Forest introduces another key idea: to make the randomness dependent on the fraction of instances that reach each node, so the randomness is lower when this fraction is smaller. The combination of these ideas offers competitive results. Improvements over ensemble methods based on traditional trees are more obvious when datasets with added noisy are used.

The aim of GAR-Forest is to increase ensemble diversity, but tries not to harm the accuracy of the individual classifiers that constitute it; these two criteria guide the way the decision tree is built. The attribute and split point that will determine the children of a node are randomly chosen from a set of good candidates, in order to increase diversity. The elements in this set are required to have a value of the merit function within certain limits, in order not to have an excessive effect on precision. Additionally, in order to encourage diversity, randomness is higher at the root, where it has more impact on the final structure of the tree. This randomness is lower in the leaves and nodes that are far away from the root to reduce loss of accuracy, so that decisions taken in these nodes can be more informed. The speed of transition from random decisions to more informed ones is controlled by a parameter τ and the traditional trees and the extremely random trees can be considered specific cases for the extreme values of this parameter.

The strategy of injecting randomness into the tree construction process of J48 (C4.5) gives rise to several possible lines of future research. One line would be to study how this heuristic can be adapted to other tree construction algorithms, such as Functional Trees [20, 38], Model Trees [56], or decision tree algorithm for imbalanced data sets, such as Hellinger Distance Decision Tree (HDDT) [6] and Class Confidence Proportion Decision Tree (CCPDT)

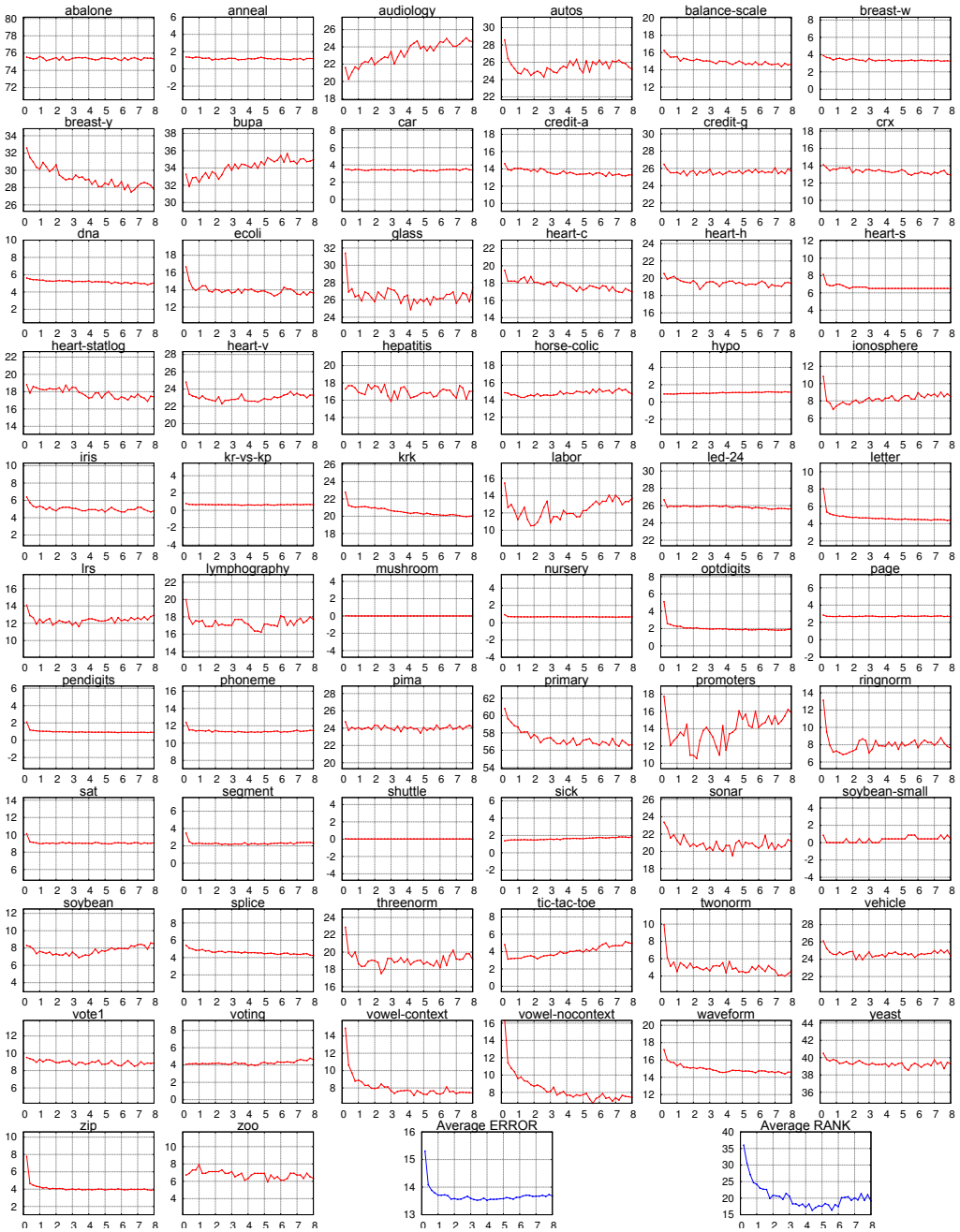


Figure 4.5: Influence of the exponent in the error (x-axis is the τ parameter and y-axis the error).

[40].

It could also be appealing to apply these heuristics to similar data mining algorithms, such as the classic construction of rules like RIPPER [7], or the most recent one such as FURIA [31].

Although, no clear pattern has become evident, in terms of the precision of the exponent τ value, while it evolved, as discussed in Section 4.5, a further research line would be to study the relationship between the optimum value of the exponent τ and different meta-features [5] or measures of complexity [29] for several datasets.

References

[1] N. Belacel, P. Hansen, and N. Mladenovic. "Fuzzy J-means: a new heuristic for fuzzy clustering". In: *Pattern Recognition* 35.10 (2002), pp. 2193–2200.

[2] L. Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[3] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125.

[4] L. Breiman et al. *Classification and Regression Trees*. 1st ed. Chapman & Hall/CRC, 1984. ISBN: 0412048418.

[5] C. Castiello, G. Castellano, and A. Fanelli. "Meta-data: Characterization of input features for meta-learning". In: *Modeling Decisions for Artificial Intelligence* (2005), pp. 457–468.

[6] David A. Cieslak and Nitesh V. Chawla. "Learning Decision Trees for Unbalanced Data". In: *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*. ECML PKDD '08. Antwerp, Belgium: Springer-Verlag, 2008, pp. 241–256. ISBN: 978-3-540-87478-2. DOI: 10.1007/978-3-540-87479-9_34.

[7] W.W. Cohen and Y. Singer. "A simple, fast, and effective rule learner". In: *Proceedings of the National Conference on Artificial Intelligence*. John Wiley & sons Ltd. 1999, pp. 335–342.

[8] J.C.W. Debuse and V.J. Rayward-Smith. "Feature subset selection within a simulated annealing data mining algorithm". In: *Journal of Intelligent Information Systems* 9.1 (1997), pp. 57–81.

[9] J. Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine Learning Research* 7 (2006), p. 30.

[10] T. Dietterich. "Ensemble methods in machine learning". In: *Multiple classifier systems* (2000), pp. 1–15.

[11] T.G. Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[12] José Diez-Pastor et al. "GRASP Forest: A New Ensemble Method for Trees". In: *Multiple Classifier Systems*. Ed. by Carlo Sansone, Josef Kittler, and Fabio Roli. Vol. 6713. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, pp. 66–75. ISBN: 978-3-642-21556-8.

[13] O. Dunn. "Multiple comparisons among means". In: *Journal of the American Statistical Association* 56 (1961), pp. 52–64.

[14] T.A. Feo and M.G.C. Resende. "A probabilistic heuristic for a computationally difficult set covering problem". In: *Operations Research Letters* 8 (1989), pp. 67–71.

[15] T.A. Feo and M.G.C. Resende. "Greedy randomized adaptive search procedures". In: *Journal of Global Optimization* 6.2 (1995), pp. 109–133. ISSN: 0925-5001.

[16] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. 2010.

[17] Y. Freund and R.E. Schapire. "Experiments with a new boosting algorithm". In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer. 1996, pp. 148–156.

[18] Z. Fu et al. "Diversification for better classification trees". In: *Computers & operations research* 33.11 (2006), pp. 3185–3202.

[19] Mikel Galar et al. "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 42.4 (2012), pp. 463–484. URL: `http://dx.doi.org/10.1109/TSMCC.2011.2161285`.

[20] J. Gama. "Functional trees". In: *Machine Learning* 55.3 (2004), pp. 219–250.

[21] Salvador García et al. "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization". In: *J. Heuristics* 15.6 (2009), pp. 617–644.

[22] S. Geman, E. Bienenstock, and R. Doursat. "Neural networks and the bias/variance dilemma". In: *Neural computation* 4.1 (1992), pp. 1–58.

[23] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely randomized trees". In: *Machine Learning* 63.1 (2006), pp. 3–42. ISSN: 0885-6125.

[24] F. Glover. "Tabu search-part II". In: *ORSA Journal on computing* 2.1 (1990), pp. 4–32.

[25] F. Glover et al. "Tabu search-part I". In: *ORSA Journal on computing* 1.3 (1989), pp. 190–206.

[26] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Artificial Intelligence. Addison-Wesley Pub. Co., 1989. ISBN: 9780201157673.

[27] Mark Hall et al. "The WEKA data mining software: an update". In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: `10.1145/1656274.1656278`.

[28] T.K. Ho. "The random subspace method for constructing decision forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[29] T.K. Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.3 (2002), pp. 289–300. ISSN: 0162-8828.

[30] Y. Hochberg. "A sharper Bonferroni procedure for multiple tests of significance". In: *Biometrika* 75 (1988), pp. 800–803.

[31] J. Hühn and E. Hüllermeier. "FURIA: an algorithm for unordered fuzzy rule induction". In: *Data Mining and Knowledge Discovery* 19.3 (2009), pp. 293–319.

[32] R.L. Iman and J.M. Davenport. "Approximations of the critical region of the fbietkan statistic". In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[33] YeongSeog Kim, W. Nick Street, and Filippo Menczer. "Feature selection in unsupervised learning via evolutionary search". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '00. Boston, Massachusetts, United States: ACM, 2000, pp. 365–369. ISBN: 1-58113-233-6. URL: http://doi.acm.org/10.1145/347090.347169.

[34] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), p. 671.

[35] Anders Krogh and Jesper Vedelsby. "Neural Network Ensembles, Cross Validation, and Active Learning". In: *Advances in Neural Information Processing Systems*. MIT Press, 1995, pp. 231–238.

[36] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[37] Ludmila I. Kuncheva and Juan José Rodríguez. "Classifier Ensembles with a Random Linear Oracle". In: *IEEE Trans. Knowl. Data Eng.* 19.4 (2007), pp. 500–508.

[38] Niels Landwehr, Mark Hall, and Eibe Frank. "Logistic Model Trees". In: *Proceeding of the 14th European Conference on Machine Learning*. 2003, pp. 241–252. URL: http://dx.doi.org/10.1007/978-3-540-39857-8_23.

[39] P. Larrañaga et al. "Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.9 (1996), pp. 912–926.

[40] Wei Liu et al. "A Robust Decision Tree Algorithm for Imbalanced Data Sets". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2010*. 2010, pp. 766–777.

[41] Brenda Mak, Robert Blanning, and Susanna Ho. "Genetic algorithms in logic tree decision modeling". In: *European Journal of Operational Research* 170.2 (2006), pp. 597–612. ISSN: 0377-2217. URL: http://dx.doi.org/10.1016/j.ejor.2004.09.030.

[42] Dragos D. Margineantu and Thomas G. Dietterich. "Pruning Adaptive Boosting". In: *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*. 1997, pp. 211–218.

[43] J. Maudes, J. J. Rodríguez, and C. García-Osorio. "Disturbing neighbors diversity for decision forests". In: *Applications of Supervised and Unsupervised Ensemble Methods*. Ed. by Oleg Okun and Giorgio Valentini. Vol. 245. Studies in Computational Intelligence. Springer, 2009, pp. 113–133. ISBN: 978-3-642-03998-0. URL: http://dx.doi.org/10.1007/978-3-642-03999-7_7.

[44] J. Maudes et al. "Random Feature Weights for Decision Tree Ensemble Construction". In: *Information Fusion* In Press, Accepted Manuscript (2010). ISSN: 1566-2535. DOI: DOI:10.1016/j.inffus.2010.11.004.

[45] N. Mladenovic and P. Hansen. "Variable neighborhood search". In: *Computers & Operations Research* 24.11 (1997), pp. 1097–1100.

[46] Claude Nadeau and Yoshua Bengio. "Inference for the Generalization Error". In: *Mach. Learn.* 52.3 (Sept. 2003), pp. 239–281. ISSN: 0885-6125. URL: http://dx.doi.org/10.1023/A:1024068626366.

[47] S. Olafsson, X. Li, and S. Wu. "Operations research and data mining". In: *European Journal of Operational Research* 187.3 (2008), pp. 1429–1448.

[48] J. Pacheco et al. "A GRASP method for building classification trees". In: *Expert Systems with Applications* 39.3 (2012), pp. 3241–3248. URL: http://dx.doi.org/10.1016/j.eswa.2011.09.011.

[49] J. Pacheco et al. "Uso del metaheurístico GRASP en la construcción de árboles de clasificación". In: *Rect@* 11 (2010), pp. 139–154. ISSN: 1575605X.

[50] J.A. Pacheco. "A scatter search approach for the minimum sum-of-squares clustering problem". In: *Computers & operations research* 32.5 (2005), pp. 1325–1335.

[51] R. Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[52] J.R. Quinlan. "Bagging, boosting, and C4. 5". In: *Proceedings of the National Conference on Artificial Intelligence*. 1996, pp. 725–730.

[53] R. J. Quinlan. *C4.5: Programs for Machine Learning*. 1st ed. Morgan Kaufmann, 1993. ISBN: 1558602380.

[54] Marko Robnik-Šikonja and Igor Kononenko. "Theoretical and Empirical Analysis of ReliefF and RReliefF". In: *Machine Learning* 53.1-2 (Oct. 2003), pp. 23–69. ISSN: 0885-6125. URL: http://dx.doi.org/10.1023/A:1025667309714.

[55] K. Sorensen and G.K. Janssens. "Data mining with genetic algorithms on binary trees". In: *European Journal of Operational Research* 151.2 (2003), pp. 253–264.

[56] Y. Wang and I. H. Witten. "Inducing Model Trees for Continuous Classes". In: *Proceedings of the 9th European Conference on Machine Learning Poster Papers*. 1997, pp. 128–137.

[57] G.I. Webb. "Multiboosting: A technique for combining boosting and wagging". In: *Machine learning* 40.2 (2000), pp. 159–196.

[58] P. Winker and M. Gilli. "Applications of optimization heuristics to estimation and modelling problems". In: *Computational statistics & data analysis* 47.2 (2004), pp. 211–223.

[59] J. Yang and V. Honavar. “Feature subset selection using a genetic algorithm”. In: *Intelligent Systems and Their Applications, IEEE* 13.2 (1998), pp. 44–49.

Chapter 5

Random Balance: Ensembles of Variable Priors Classifiers for Imbalanced Data

Authors Jose F Diez-Pastor; Juan J. Rodriguez; Cesar I Garcia-Osorio; Ludmila I Kuncheva

Type Journal

Published in Knowledge-Based Systems. (In press)

Year Review 2015.

Abstract

In machine learning, a data set is imbalanced when the class proportions are highly skewed. Imbalanced data sets arise routinely in many application domains and pose a challenge to traditional classifiers. We propose a new approach to building ensembles of classifiers for two-class imbalanced data sets, called Random Balance. Each member of the Random Balance ensemble is trained with data sampled from the training set and augmented by artificial instances obtained using SMOTE. The novelty in the approach is that the proportions of the classes for each ensemble member are chosen randomly. The intuition behind the method is that the proposed diversity heuristic will ensure that the ensemble contains classifiers that are specialised for different operating points on the ROC space, thereby leading to larger AUC compared to other ensembles of classifiers. Experiments have been carried out to test the Random Balance approach by itself, and also in combination with standard ensemble methods. As a result, we propose a

new ensemble creation method called RB-Boost which combines Random Balance with AdaBoost.M2. This combination involves enforcing random class proportions in addition to instance re-weighting. Experiments with 86 imbalanced data sets from two well known repositories demonstrate the advantage of the Random Balance approach.

Index terms— Classifier ensembles, imbalanced data sets, Bagging, AdaBoost, SMOTE, Undersampling

5.1 Introduction

The class-imbalance problem occurs when there are many more instances of some classes than others [9]. Imbalanced data sets are common in fields such as bioinformatics (translation initiation site (TIS) recognition in DNA sequences [25], gene recognition [5]), engineering (non-destructive testing in weld flaws detection through visual inspection [39]), finance (predicting credit card customer churn [2]), fraud detection [46] and many more.

Bespoke methods are needed for imbalanced classes for at least three reasons [56]. Firstly, standard classifiers are driven by accuracy so the minority class may be ignored. Secondly, standard classification methods operate under the assumption that the data sample is a faithful representation of the population of interest, which is not always the case with imbalanced problems. Finally, the classification methods for imbalanced problems should allow for errors coming from different classes to have different costs.

Galar et al. [23] systemize the wealth of recent techniques and approaches into four categories:

a) *Algorithm level approaches.* This category contains variants of existing classifier learning algorithms biased towards learning more accurately the minority class. Examples include decision tree algorithms insensitive to the class sizes, like Hellinger Distance Decision Tree (HDDT) [12], Class Confidence Proportion Decision Tree (CCPDT) [41] and a SVM classifier with different penalty constants for different classes [55].

b) *Data level approaches.* The main idea in this category is to pre-process the data so as to transform the imbalanced problem into a balanced one by manipulating the distribution of the classes. These algorithms are often used in combination with ensembles of classifiers. This category can be further subdivided into methods that increase the number of minority class examples: Oversampling [4], SMOTE [10], Borderline-SMOTE [28] and Safelevel-SMOTE [8] among others; and methods that reduce the size of the majority class, such as random undersampling, this approach has been used both with and without replacement [3]. These techniques can be jointly applied to increase the size of the minority class while simultaneously decreasing the majority class.

c) *Cost-sensitive learning.* While traditional algorithms aim at increasing the accuracy by giving equal weights to the examples of any class, cost-sensitive methods, such as cost-sensitive decision trees [40] or cost-sensitive neural networks [36], assign a different cost to each class. The best known methods in this category are the cost-sensitive versions of AdaBoost: AdaCost [17], [33], AdaC1, AdaC2 and AdaC3 [53].

d) *Ensemble learning.* Classifier ensembles have often offered solutions to challenging problems where standard classification methods have been insufficient. One approach for constructing ensembles for imbalanced data is based on using data level approaches: each base classifier is trained with a pre-processed data set. As data level approaches usually use random values, the pre-processed data sets and the corresponding classifiers will be different. Another strategy is based on combining conventional ensemble methods (i.e., not specific for imbalance) with data level approaches. Examples of this strategy are SMOTEBagging [57], SMOTEBoost [11] and RUSBoost [49]. It is also possible to have ensembles that combine classifiers obtained with different methods [34].

In general, according to [23], algorithm level and cost-sensitive approaches are more data-dependent, whereas data level and ensemble learning methods are more versatile.

Here we propose a new preprocessing technique that can be used to build ensembles, for two-class imbalanced learning tasks, based on a simple randomisation heuristic. The data for training an ensemble member is sampled from the training data using random class proportions. The classes are either undersampled or augmented with artificial examples to make up such a sample.

The rest of the paper is structured as follows. Section 5.2 presents the performance measures used in the experimental evaluation. Section 5.3 briefly overviews some of the most relevant methods in imbalanced learning, those used in the experimental study. Section 5.4 explains the proposed method. In Section 5.5 we provide a simulation example that tries to give some insight in why the method works. An experimental study is reported in Section 6.3, and, finally, Section 6.5 contains our conclusions and several future research lines.

5.2 Measures of performance for imbalanced data

When working with binary classification problems instances can be labelled as positive (p) or negative (n). In binary imbalanced data sets usually the minority class is considered positive while the majority class is considered negative. For a prediction there are 4 possible outcomes: a) True Positive: prediction is p and the real label is p. b) True Negative: prediction is n and the real label is n. c) False Positive: prediction is p and the real label is n. d) False Negative: prediction is n and the real label is p. Given a test dataset, containing P examples of the positive class and N examples of the negative class, TP is the number of True Positives, FP is the number of False Positives, TN is the number of True Negatives and FN the number of False Negatives.

The True Positive Rate (TPR), also called Sensitivity or Recall, is defined as TP/P and False Positive Rate (FPR) is defined as FP/N. The precision is defined as $TP/(TP+FP)$.

Commonly used measures of performance for imbalanced data are the Area Under the ROC (Receiver Operation Characteristic) curve [18], the F-Measure [54] and the Geometric Mean [35]. The F-Measure is defined as $2 \times \frac{precision \times recall}{precision + recall}$. The Geometric Mean is defined as $\sqrt{TP/P \times TN/N}$. The ROC Curve is a two-dimensional representation of classifier performance, it is created by plotting the TPR against the FPR for different decision thresholds. The Area Under the ROC curve (AUC) is a way to represent the performance of a binary classifier using a scalar.

5.3 Classification methods for imbalanced problems

In recent years, numerous techniques have been developed to deal with the problem of class-imbalance datasets. This section is a sort summary of the subset of methods tested in this article. The methods are organized using the same classification presented in the introduction:

- *Data level approaches.*
  - Random Undersampling. This technique will randomly drop some of the examples of the majority class. When it comes to sampling without replacement, an example of the minority class can appear only once in the sub-sampling; with replacement, the same example can appear multiple times.

  - Random oversampling [4] consists of adding exact copies of some minority class examples. With this technique overfitting is more common than in the prior technique.
  - SMOTE (Synthetic Minority Over-sampling Technique [10]) although this technique has "oversampling" in the name, it does not add copies of existing instances, but creates new artificial examples using the following procedure: a member of the minority class is selected and its k nearest neighbours (from the minority class) are identified. One of them is randomly selected. Then, the new example added to the set is a random point in the line segment defined by the member and its neighbour. A value of $k = 5$ has been recommended and is the one used in this study. This method tries to avoid overfitting using a random procedure to create the new samples, but this can introduce noise or nonsensical samples.

- *Ensemble learning.* One of the keys for good performance of ensembles is the diversity, there are several ways to inject diversity into an ensemble, the most common is the use of sampling. In Bagging [6], each base classifier is obtained from a random sample of the training data. In AdaBoost [22] the resampling is based on a weighted distribution, the weights are modified depending on the correctness of the prediction for the example given by the previous classifier. Bagging and AdaBoost have been modified to deal with imbalanced datasets:
  - SMOTEBagging [57] combines Bagging with different amounts of SMOTE and Oversampling in each iteration, so that the data set is completely balanced and consists of three parts: *i*) a sample with replacement of the majority class, keeping the original size; *ii*) oversampling of the minority class; and *iii*) SMOTE of the minority class. The Oversampling percentage varies in each iteration (ranging from 10% in the first iteration to 100% in the last.) The rest of the positive instances are generated by the SMOTE algorithm.
  - SMOTEBoost [11] and RUSBoost [49] are both modifications of AdaBoost.M2 [22], in each iteration, besides the instance reweighting done according to the algorithm Adaboost.M2, SMOTE or Random undersampling is applied to the training set of the base classifier. Boosting based ensembles tend to perform better than

bagging based ensembles, however, in Boosting based ensembles, the base classifiers are trained in sequence which slows down the training, and they are more sensitive to noise. SMOTEBoost and RUSBoost are more robust to noise because they introduce a high degree of randomness by creating or deleting instances.

- Although the most popular methods are modifications or variations of bagging or boosting, there are methods that do not perform resampling, oversampling or undersampling and, instead of that, they make partitions. One method described in [44], which will be called "Partitioning" in this paper, is similar to undersampling based ensembles, it breaks the majority class into several disjoint partitions and constructs several models which use one partition from the majority class and the entire minority class.
- Most of the above methods, at the same time they increase accuracy in minority class, they decrease overall accuracy compared to traditional learning algorithms. Some approaches combine both types of classifiers, one trained with the original skewed data and other trained according one of the previous approaches in an attempt to cope with the imbalance. Reliability Based Classifier [50] trains two classifiers and then chooses between the output of the classifier trained on the original skewed distribution and the output of the classifier trained according to a learning method addressing the curse of imbalanced data. This decision is guided by a parameter whose value maximizes, on a validation set, the accuracy and a measure designed to evaluate the performance of a classifier in imbalanced classifiers, such as the geometric mean.

5.4 Random Balance and RB-Boost ensembles

This section presents the main contribution of the paper. In this section we present a new preprocessing technique called Random Balance, this technique can be used within an ensemble to increase the diversity and deal with imbalance. We also describe a new ensemble method for imbalanced learning called RB-Boost (Random Balance Boost) which is a a Random Balance modification of AdaBoost.M2. We also explain the intuition behind the method in an aside subsection.

When dealing with imbalanced dataset the three common data-level approaches to balancing the classes are listed below[1]:

- The new data set is formed by taking the entire minority class and a random subsample from the majority class. The method has a parameter N that is the desired percentage of instances that belongs to the majority class in the processed dataset. For example, consider a data set with 20 instances in the minority class and 480 instances in the majority class. For $N = 40$, the desired number of instances from the majority class is 30 so that the 20 instances of the minority class make up 40% of the data.
- The new data set is formed by adding to it $(M/100) \times sizeMinority$ synthetic instances of the minority class using the SMOTE method. The amount of artificial instances is expressed as a percentage M of the size of the minority class, and is again a parameter of the algorithm. In the example above, if we choose $M = 200$, 40 examples from the minority class will be generated through SMOTE.
- Use both undersampling and oversampling through SMOTE to reach a desired new size of the data and proportions of the classes within.

The problem with these data-level approaches is that the optimal proportions depend on the data set and are hard to find, it is known that this proportions have a substantial influence on the performance of the classifier. The proposed method relies completely on randomness and repetition to try to overcome this problem.

5.4.1 Random Balance

While preprocessing techniques are commonly used to restore the balance of the class proportions to a given extent, Random Balance relies on a completely random ratio. This includes the case where the minority class is over-represented and the imbalance ratio is inverted.

An example of the sampling procedure can be seen in Figure 5.1. Given a data set, a different data set of the same size is obtained for each member of ensemble where the imbalance ratio is chosen randomly. In this example, the initial proportions of both classes appears on the top. Classifiers $1, 2 \ldots T$

[1]Note that although random undersampling and SMOTE are mentioned because they are the most used techniques, more sophisticated techniques could be used resulting in variants of the proposed method.

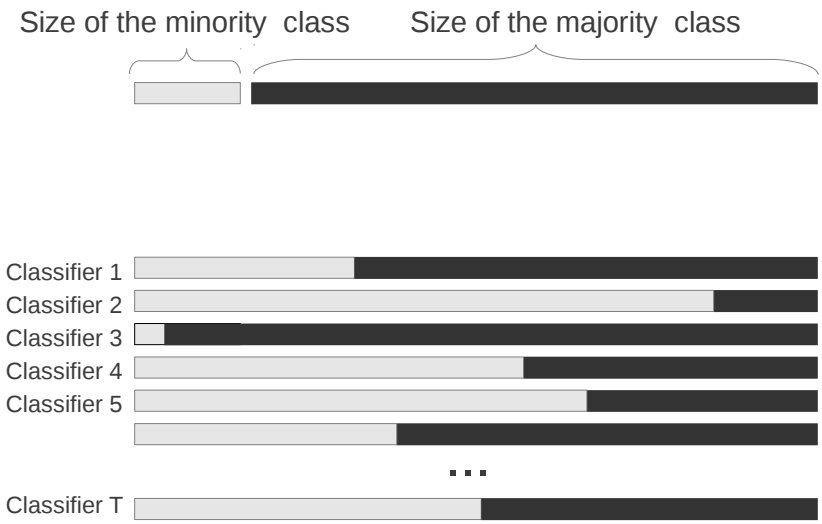


Figure 5.1: Example of data sets used to train a Random Balance ensemble, note that the imbalance ratio is different for each dataset (even in favor of the minority class, for example, for the second classifier).

are trained with variants of this data set where the ratio between classes varies randomly. In iteration 1, the imbalance ratio has been slightly reduced. In iteration 2, the ratio is reversed, the size of the previous minority class exceeds the size of the previous majority class. And in iteration 3, the minority class has become even smaller. All these cases are possible since the procedure is random.

The procedure is described in the pseudocode in Figure 5.2. The fundamental step is to randomly set the new size of the majority and minority classes (lines 6-7). Then SMOTE and Random Undersampling (resampling without replacement) are used to respectively increase or reduce the size of the classes to match the desired size (lines 8-11 or lines 12-15 as required.) We call this generic ensemble method *Ensemble-RB*. Additionally, it can be combined with Bagging, resulting in what we call *Bagging-RB*.

Pre-procesing strategies can have important drawbacks. Undersampling can throw out potentially useful data, while SMOTE increases the size of the dataset and hence the training time. Random-Balance maintains the size of the training set and because it is a process which is repeated several times, the problem of removing important examples is reduced.

RANDOM BALANCE

Require: Set S of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathbf{Y} = \{-1, +1\}$ (+1: positive or minority class, −1: negative or majority class), neighbours used in SMOTE, k

Ensure: New set S' of examples with random balance

$totalSize \leftarrow |S|$
$S_N \leftarrow \{(\mathbf{x}_i, y_i) \in S \mid y_i = -1\}$
$S_P \leftarrow \{(\mathbf{x}_i, y_i) \in S \mid y_i = +1\}$
$majoritySize \leftarrow |S_N|$
$minoritySize \leftarrow |S_P|$
$newMajoritySize \leftarrow$ Random integer between 2 and $totalSize$-2
// Resulting classes will have at least 2 instances
$newMinoritySize \leftarrow totalSize$ - $newMajoritySize$
if $newMajoritySize < majoritySize$ **then**
    $S' \leftarrow S_P$
    Take a random sample of size $newMajoritySize$ from S_N, add the sample to S'.
    Create $newMinoritySize - minoritySize$ artificial examples from S_P using SMOTE, add these examples to S'.
else
    $S' \leftarrow S_N$
    Take a random sample of size $newMinoritySize$ from S_P, add the sample to S'.
    Create $newMajoritySize - majoritySize$ artificial examples from S_N using SMOTE, add these examples to S'.
end if
return S'

Figure 5.2: Pseudocode for the Random Balance ensemble method.

5.4.1.1 Instance inclusion probability

The data sets generated in random balance have instances from the original training data and artificial instances. For Random Balance, the probability of including an instance is different for minority and majority instances. Given p positive instances and n negative instances with $m = n + p$ and assuming that $p \geq 2$ the probability of including an instance of the minority class is:

$$P_{mino} = \frac{1}{m-3}\left(\sum_{i=2}^{p-1}\frac{i}{p} + \sum_{i=p}^{m-2} 1\right) = \frac{1}{m-3}\left(m - \frac{p+3}{2} - \frac{1}{p}\right)$$

In the generated data set, each class has at least two instances. Then, there are $n-3$ possible sizes of the minority class in the generated data sets (from 2 to $m-2$). The summation $\sum_{i=2}^{p-1}\frac{i}{p}$ is for the cases when the number of instances in the minority class is reduced (from p instances we

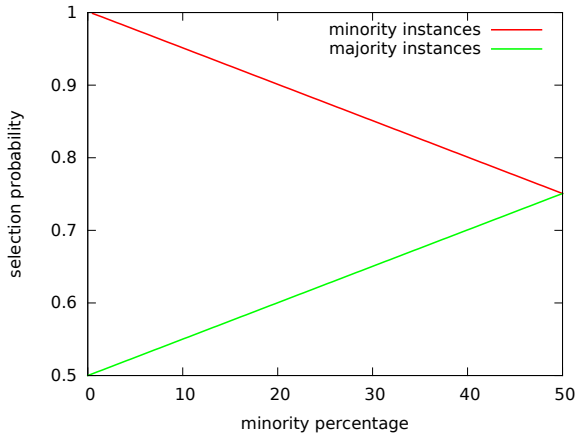


Figure 5.3: Probabilities of including an instance in the transformed dataset, when the number of instances is $m = 1000$.

randomly take i so the selection probability is i/p), while $\sum_{i=p}^{m-2} 1$ is for the cases when the minority size is increased (the selection probability is 1).

Analogously, the probability of selecting an instance of the majority class will be:

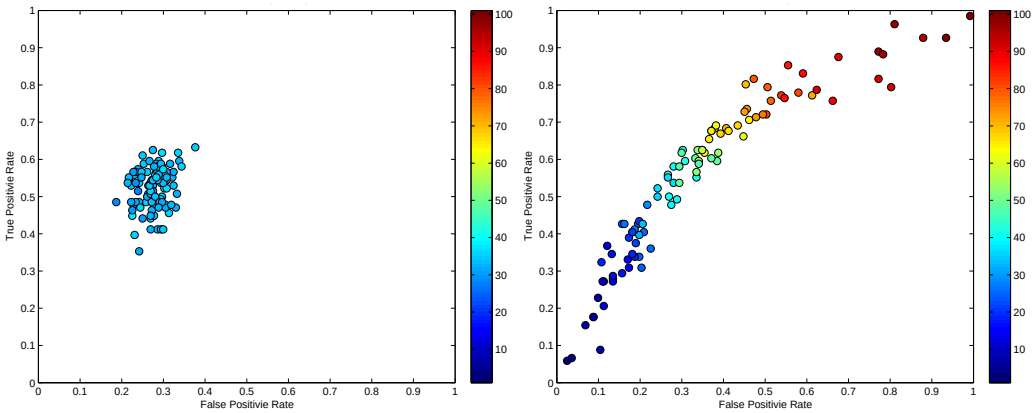
$$P_{majo} = \frac{1}{m-3}\left(m - \frac{n+3}{2} - \frac{1}{n}\right)$$

Figure 5.3 shows the probabilities of selecting an instance in the generated data set as a function of the percentage of instances from the minority class, for a data set with 1000 instances. The probability of selecting an instance of the minority class decreases when the data set is more balanced.

It can be seen that if $p \leq n$ then $P_{majo} \leq P_{mino}$, $P_{mino} \geq 0.75$ and $P_{majo} \geq 0.5$. For a perfectly balanced data set, the probability of selecting an instance is a bit greater than 0.75 because there will be at least two instances of each class. The problem of discarding important instances of the majority class is ameliorated because the expected number of base classifiers that are trained with a given instance of the majority class is greater than 50%. Moreover, some of the instances included in the data set will also used to generate artificial instances.

5.4.1.2 Intuition behind the method

The ROC space is defined by FPR and TPR as x and y axes respectively because there is a trade-off between this two values. A classifier can be represented as a point in this space and all base classifiers in an ensemble can be represented as a cloud of points. Figure 5.4a shows the cloud of points for a Bagging ensemble trained with the credit-g dataset, the color of each point represents the percentage of the instances than belong to the positive class in the dataset used for training that base classifier. It is easy to



(a) Bagging **(b)** Ensemble of Random Balance

Figure 5.4: Base classifiers in the ROC Space (credit-g dataset). The color of each point represents the percentage of the instances than belong to the positive class in the dataset used for training that base classifier. Higher values (in red) represent that the imbalance ratio has been changed in favour of the minority class, values around 50 (in light blue/cyan) are for balanced cases, and lower values (in dark blue) the minority class has been made even smaller (originally credit-g dataset has 2.33 times more negative instances than positive).

appreciate that all of the members of the ensemble are trained with samples which vary very slightly the proportion between classes. In contrast, Figure 5.4b shows the cloud for an ensemble of Random Balance classifiers, the large variability in the ratio between classes in the datasets used to train each of the base classifiers, including cases in which the positive class becomes larger than the negative, makes the base classifiers of the ensemble spread out over the ROC space.

In the proposed method, the base classifiers are forced to learn different points on the ROC space and thereby expected to be more diverse and to improve the ensemble performance (see Figure 5.4). Diversity is generally considered beneficial for ensemble methods, including the imbalanced case [58].

5.4.2 RB-Boost

There are several modification of AdaBoost.M2 for imbalanced problems. The best known of these methods is SMOTEBoost [11].

As in AdaBoost.M2, the examples of the training data have weights that are updated according to a pseudo-loss function. For each base classifier the weighted training data is augmented with artificial examples generated by SMOTE.

RUSBoost [49], as SMOTEBoost, is also an AdaBoost.M2 modification,

but in this case instances of the majority class are removed using random undersampling in each iteration. No new weights are assigned; the weights of the remaining instances are normalized according to the new sum of weights of the data set. The rest of the procedure is as in AdaBoost.M2 and SMOTEBoost.

Both methods apply a preprocessing technique to the data and simultaneously alter the weights. Following this philosophy we propose *RB-Boost*, whose pseudocode is described in Figure 5.5. It is also a modification of AdaBoost.M2, in which line 3 is changed to generate a data set according to the procedure shown in Figure 5.2. The number of instances removed by undersampling is equal to the number of instances introduced by SMOTE. The algorithm works as follows: for each of the T rounds (lines 2-11) a data set $S_t^{'}$ is generated according to the Random Balance procedure (line 3). Distribution $D_t^{'}$ is updated, maintaining for each instance of the original data set its associated weight and assigning a uniform weight to the artificial examples (line 4). Then a weak learning algorithm is trained using $S_t^{'}$ and $D_t^{'}$ (line 5), this classifier will give a probability between 0 and 1 to each class[2] . The pseudo-loss ε_t of the weak classifier h_t is computed according to the formula presented in line 6. The distribution $D_t^{'}$ is updated to make the weights associated with wrong classifications higher than the weights given to correct classifications (line 7-9). Finally, the different classifiers outputs are combined (line 11) taking into account their respective β_t (obtained in line 7).

5.5 A simulation experiment

To test-run the idea we carried out experiments with generated data. By contrasting the Random Balance with Bagging, we intend to gain more insight and support for our hypothesis that the Random Balance heuristic improves diversity in a way which leads to larger AUC[3]. We generated two 2-dimensional Gaussian classes centred at (0,0) and (3,3), both with identity covariance matrices. To simulate unbalanced classes, 450 points

[2]In experiments, J48 classification tree with Laplace smoothing has been used as a weak classifier. The prediction returned by the classifier is the probability calculated taking into the instances that end in the leaf. With Laplace smoothing this is $(a_i+1)/(A+c)$, where a_i is the number of instances of class i in the leaf, A is the total number of instances in the leaf, and c the number of classes.

[3]The varying parameter for the ROC curve is the threshold on the class membership probability estimated by the whole ensemble, not a particular base classifier.

RB-BOOST

Require: Set S of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathbf{Y} = \{-1, +1\}$ ($+1$: positive or minority class, -1: negative or majority class),
Weak learner, *weakLearn*
Number of iterations, T
Number of neighbours used in SMOTE, k

Ensure: RB-Boost is built

// Initialize distribution D_1

$D_1(i) \leftarrow \frac{1}{m}$ **for** $i = 1, \ldots, m$

for $t = 1, 2 \ldots T$ **do**

  $S'_t \leftarrow$ **RandomBalance**(S, k)

  $D'_t(i) \leftarrow D_t(j)$ **if** $S'_t(i) = S_t(j)$ **else** $\frac{1}{m}$, **for** $i = 1, \ldots, m$

  // If the example is from the sample it maintains its weight, if the example is artificial it has the initial weight.

  Using S'_t and weights D'_t, train *weakLearn* h_t: $\mathbf{X} \times \mathbf{Y} \rightarrow [0, 1]$,

  Compute the pseudo-loss of hypothesis h_t:

$$\varepsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(\mathbf{x}_i, y_i) + h_t(\mathbf{x}_i, y))$$

  $\beta_t \leftarrow \varepsilon_t / (1 - \varepsilon_t)$

  Update D_t:

  $D_{t+1}(i) \leftarrow D_t(i) \cdot \beta_t^{\frac{1}{2}(1 + h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y))}$

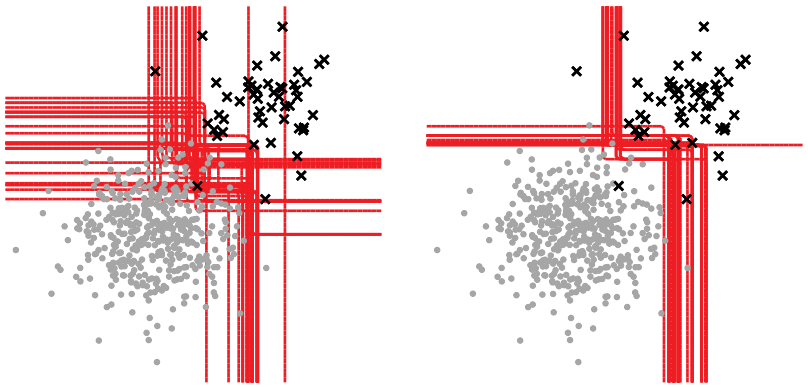
  Normalize D_{t+1}: Let $Z_t \leftarrow \sum_i D_{t+1}(i)$

  $D_{t+1}(i) \leftarrow \frac{D_{t+1}(i)}{Z_t}$

end for

return $h_f(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(\mathbf{x}, y)$

Figure 5.5: Pseudocode for the RB-Boost ensemble method.



(a) Random Balance (b) Bagging

Figure 5.6: An unbalanced data set and examples of the classification boundaries generated by two ensemble methods.

were sampled from the first class, and 50 points from the second class (10%). Each ensemble was composed of 50 decision tree classifiers[4]. The ensemble output was calculated as the average of the the individual outputs. An example of the classification boundaries for the Random Balance ensemble and the Bagging ensemble is shown in Figure 5.6. To evaluate the individual and ensemble accuracies as well as the AUC, we sampled a new data set from the same distribution and of the same size. The numerical results for this illustrative example are given in Table 5.1.

It can be observed that the boundary lines for the Random Balance ensemble are more widely scattered compared to these for the Bagging ensemble, stepping well into the region of the majority class. Table 5.1 shows also the average results from 200 iterations, each iteration with freshly sampled training and testing data. The results indicate that: *i*) individual errors of the decision trees for the ensemble-RB are larger than these for the Bagging ensemble, *ii*) RB has a higher classification error than Bagging, and *iii*) RB has a better AUC than Bagging. All differences were found to be statistically significant (two-tailed paired t-test, $p < 0.005$). This suggests that the better AUC produced by the ensemble-RB may come at the expense of slightly reduced classification accuracy. Since AUC is often viewed as the primary criterion for problems with unbalanced classes, the results of this simulation favour the ensemble-RB.

Kappa-error diagrams are often used for comparing classifier ensembles [43, 37]. Consider a testing set with N examples and the contingency

[4]MATLAB's Statistic Toolbox was used for training the decision trees and estimating AUC.

Data sets	Ensemble	Individual error	Ensemble error	AUC
1 simulation	RB	0.0272	0.0180	0.9979
(Figure 5.6)	Bagging	0.0250	0.0220	0.9373
200 simulations	RB	0.0307	0.0162	**0.9963**
(average values)	Bagging	**0.0192**	**0.0133**	0.9917

Table 5.1: Comparison of Random Balance and Bagging ensembles.

table of two classifiers, C_1 and C_2

	C_2 correct	C_2 wrong
C_1 correct	a	b
C_1 wrong	c	d

where the table entries are the number of examples jointly classified as indicated, and $a+b+c+d=N$. Diversity between the two classifiers is measured by κ [19] as

$$\kappa = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{5.1}$$

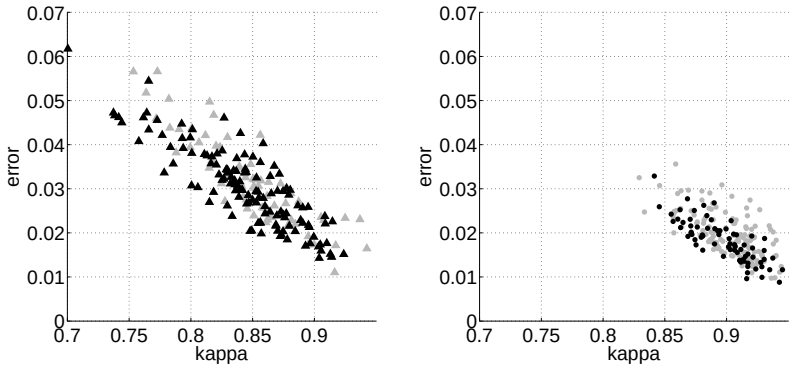
Kappa is plotted on the x-axis on the diagram. Smaller kappa indicates more diverse classifiers. The averaged individual error for the pair of classifiers is

$$e = \frac{1}{2}\left(\frac{c+d}{N}+\frac{b+d}{N}\right) = \frac{b+c+2d}{2N} \tag{5.2}$$

The error is plotted on the y-axis of the diagram. An ensemble with L classifier generates a "cloud" of $L(L-1)/2$ points on the kappa-error diagram, one point for a pair of classifiers.

We calculated the centroid points of 200 RB and 200 Bagging ensemble clouds following the simulation protocol described above. Figure 5.7 shows the kappa-error diagrams with the centroids, 200 in each subplot. The black points correspond to ensembles whose AUC is larger than the respective AUC of the rival ensemble. Out of the 200 ensembles, RB had larger AUC in 127 cases, which is seen as the larger proportion of black triangles in the left subplot compared to the proportion of black dots in the right subplot.

As expected, the ensemble-RB generates substantial diversity compared to Bagging, which is indicated by the stretch to the left of the set of points in the left subplot. The cloud of points is tilted, showing that the larger diversity



(a) Random Balance (b) Bagging

Figure 5.7: Kappa-error diagrams for the two ensemble methods. Black points indicate ensembles for which the AUC was larger than that for the rival method.

is paid for by larger individual error. An interesting observation from the figure is that the black markers (triangles and dots) are spread uniformly along the point clouds, suggesting that there is no specific diversity-accuracy pattern which is symptomatic of better AUC.

5.6 Experimental setup and results

Two collections of data sets were used. The HDDT collection[5] contains the binary imbalanced data sets used in [13]. Table 6.1 shows the characteristics of the 20 data sets in this collection.

The KEEL collection[6] contains the binary imbalanced data sets from the repository of KEEL [1]. Table 6.2 shows the characteristics of the 66 data sets in this collection[7].

In both tables, the first column is the name of the data set, the second the number of examples, the third the number of attributes and the last is the imbalance ratio (the number of instances of the majority class for each instance of the minority class).

Many data sets in these two collections are available or are modifications of data sets in the UCI Repository [20].

Weka [27] was used for the experiments. The ensemble size was set to 100, for some methods this is not the exact size, but it is the maximum,

[5]Available at `http://www.nd.edu/~dial/hddt/`.

[6]Available at `http://sci2s.ugr.es/keel/imbalanced.php`.

[7]Notice that several of the data sets come from data sets that were originally multiclass, the 66 datasets have been derived from 16 original sets.

data set	Examples	Attributes (Numeric / Nominal)	IR
boundary	3505	(0/175)	27.50
breast-y	286	(0/9)	2.36
cam	18916	(0/132)	19.08
compustat	13657	(20/0)	25.26
covtype	38500	(10/0)	13.02
credit-g	1000	(7/13)	2.33
estate	5322	(12/0)	7.37
german-numer	1000	(24/0)	2.33
heart-v	200	(5/8)	2.92
hypo	3163	(7/18)	19.95
ism	11180	(6/0)	42.00
letter	20000	(16/0)	24.35
oil	937	(49/0)	21.85
optdigits	5620	(64/0)	9.14
page	5473	(10/0)	8.77
pendigits	10992	(16/0)	8.63
phoneme	5404	(5/0)	2.41
PhosS	11411	(480/0)	17.62
satimage	6430	(36/0)	9.29
segment	2310	(19/0)	6.00

Table 5.2: Characteristics of the data sets from the HDDT collection.

data set	Examples	Attributes (Numeric / Nominal)	IR	data set	Examples	Attributes (Numeric / Nominal)	IR
abalone19	4174	(7/1)	129.44	glass5	214	(9/0)	22.78
abalone9-18	731	(7/1)	16.40	glass6	214	(9/0)	6.38
cleveland-0_vs_4	177	(13/0)	12.62	haberman	306	(3/0)	2.78
ecoli-0-1-3-7_vs_2-6	281	(7/0)	39.14	iris0	150	(4/0)	2.00
ecoli-0-1-4-6_vs_5	280	(6/0)	13.00	led7digit-0-2-4-5-6-7-8-9_vs_1	443	(7/0)	10.97
ecoli-0-1-4-7_vs_2-3-5-6	336	(7/0)	10.59	new-thyroid1	215	(5/0)	5.14
ecoli-0-1-4-7_vs_5-6	332	(6/0)	12.28	new-thyroid2	215	(5/0)	5.14
ecoli-0-1_vs_2-3-5	244	(7/0)	9.17	page-blocks-1-3_vs_4	472	(10/0)	15.86
ecoli-0-1_vs_5	240	(6/0)	11.00	page-blocks0	5472	(10/0)	8.79
ecoli-0-2-3-4_vs_5	202	(7/0)	9.10	pima	768	(8/0)	1.87
ecoli-0-2-6-7_vs_3-5	224	(7/0)	9.18	segment0	2308	(19/0)	6.02
ecoli-0-3-4-6_vs_5	205	(7/0)	9.25	shuttle-c0-vs-c4	1829	(9/0)	13.87
ecoli-0-3-4-7_vs_5-6	257	(7/0)	9.28	shuttle-c2-vs-c4	129	(9/0)	20.50
ecoli-0-3-4_vs_5	200	(7/0)	9.00	vehicle0	846	(18/0)	3.25
ecoli-0-4-6_vs_5	203	(6/0)	9.15	vehicle1	846	(18/0)	2.90
ecoli-0-6-7_vs_3-5	222	(7/0)	9.09	vehicle2	846	(18/0)	2.88
ecoli-0-6-7_vs_5	220	(6/0)	10.00	vehicle3	846	(18/0)	2.99
ecoli-0_vs_1	220	(7/0)	1.86	vowel0	988	(13/0)	9.98
ecoli1	336	(7/0)	3.36	wisconsin	683	(9/0)	1.86
ecoli2	336	(7/0)	5.46	yeast-0-2-5-6_vs_3-7-8-9	1004	(8/0)	9.14
ecoli3	336	(7/0)	8.60	yeast-0-2-5-7-9_vs_3-6-8	1004	(8/0)	9.14
ecoli4	336	(7/0)	15.80	yeast-0-3-5-9_vs_7-8	506	(8/0)	9.12
glass-0-1-2-3_vs_4-5-6	214	(9/0)	3.20	yeast-0-5-6-7-9_vs_4	528	(8/0)	9.35
glass-0-1-4-6_vs_2	205	(9/0)	11.06	yeast-1-2-8-9_vs_7	947	(8/0)	30.57
glass-0-1-5_vs_2	172	(9/0)	9.12	yeast-1-4-5-8_vs_7	693	(8/0)	22.10
glass-0-1-6_vs_2	192	(9/0)	10.29	yeast-1_vs_7	459	(7/0)	14.30
glass-0-1-6_vs_5	184	(9/0)	19.44	yeast-2_vs_4	514	(8/0)	9.08
glass-0-4_vs_5	92	(9/0)	9.22	yeast-2_vs_8	482	(8/0)	23.10
glass-0-6_vs_5	108	(9/0)	11.00	yeast1	1484	(8/0)	2.46
glass0	214	(9/0)	2.06	yeast3	1484	(8/0)	8.10
glass1	214	(9/0)	1.82	yeast4	1484	(8/0)	28.10
glass2	214	(9/0)	11.59	yeast5	1484	(8/0)	32.73
glass4	214	(9/0)	15.46	yeast6	1484	(8/0)	41.40

Table 5.3: Characteristics of the data sets from the KEEL collection.

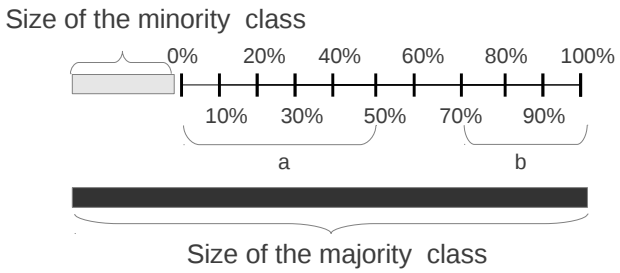


Figure 5.8: The figure shows how to interpret the parameters used for SMOTE and Undersampling. These parameters can be thought of as a percentage of the difference of class sizes. For SMOTE, in this case, a value of 50% (*a* in the figure) indicates that the number of artificial instances to be created are the 50% of the number needed to match the size of the majority class. For Undersampling a value of 70% (*b* in the figure) indicates that the number of removed instances in the majority class will be 30% of the size of the difference.

Data-processing-only Based Ensembles		
Abbr.	*Method*	*Details*
ESM100	Ensemble, SMOTE=100%	Amount of SMOTE in each iteration equal to 100% size of the minority class
ESM200	Ensemble, SMOTE=200%	Amount of SMOTE in each iteration equal to 200% size of the minority class
ESM500	Ensemble, SMOTE=500%	Amount of SMOTE in each iteration equal to 500% size of the minority class
ESM	Ensemble, SMOTE	SMOTE in each iteration until 50% of the data belongs to minority class
ERUS	Ensemble, RUS	Random Undersampling in each iteration until 50% of the data belongs to minority class
ERUSR	Ensemble, RUS with replacement	Random Undersampling with replacement in each iteration until 50% of the data belongs to minority class
EPart	Ensemble, Partitioning	Build balanced training sets by splitting the majority class into subsets.
EopS	Ensemble, optimized SMOTE	Amount of SMOTE selected by cross validation
EopU	Ensemble, optimized Undersampling	Amount of Random Undersampling selected by cross validation
EopB	Ensemble, optimized Both	Amounts of SMOTE and Undersampling selected by cross validation
E-RB	Ensemble, Random Balance	Random Balance in each iteration

Table 5.4: Algorithms used in the experimental study: Data-processing family

since some method have a stopping criteria. J48 was chosen as the base classifier in all ensembles[8]. As recommended for imbalanced data [13], it was used without pruning and collapsing but with Laplace smoothing at the leaves. C4.5 with this options is called C4.4 [47].

The results were obtained with a 5×2-fold cross validation [15]. The data set is halved in two folds. One fold is used for training and the other for testing, and then the roles of the folds are reversed. This process is repeated five times. The results are the averages of these ten experiments. Cross validation was stratified: the class proportions was approximately preserved for each fold.

Given the large number of methods and variants tested, the comparisons are divided into families. Each family includes different types of classi-

[8] J48 is the Weka's re-implementation of C4.5 [48].

Bagging Based Ensembles		
Abbr.	*Method*	*Details*
SMBAG	SMOTEBagging	
BAG	Bagging	
BAGSM100	Bagging, SMOTE=100%	Amount of SMOTE in each iteration equal to 100% size of the minority class
BAGSM200	Bagging, SMOTE=200%	Amount of SMOTE in each iteration equal to 200% size of the minority class
BAGSM500	Bagging, SMOTE=500%	Amount of SMOTE in each iteration equal to 500% size of the minority class
BAGSM	Bagging, SMOTE	SMOTE in each iteration until 50% of the data belongs to minority class
BAGRUS	Bagging, RUS	Random Undersampling in each iteration until 50% of the data belongs to minority class
ŘbB:IC+BAGSM	Reliability-based Balancing with SMOTE	Miniensemble formed by Bagging and Bagging + SMOTE in each iteration until 50% of the data belongs to minority class
ŘbB:IC+BAGRUS	Reliability-based Balancing with UnderSampling	Miniensemble formed by Bagging and Bagging + Random Undersampling in each iteration until 50% of the data belongs to minority class
BAGopS	Bagging, optimized SMOTE	Amount of SMOTE selected by cross validation
BAGopU	Bagging, optimized Undersampling	Amount of Random Undersampling selected by cross validation
BAGopB	Bagging, optimized Both	Amounts of SMOTE and Undersampling selected by cross validation
BAG-RB	Bagging, Random Balance	Random Balance in each iteration

Table 5.5: Algorithms used in the experimental study: Bagging family

Boosting Based Ensembles		
Abbr.	*Method*	*Details*
AdaM1W	AdaBoost.M1 using reweighting	
AdaM1S	AdaBoost.M1 using resampling	
MultiW	MultiBoost using reweighting	Number of subcommittees=10
MultiS	MultiBoost using resampling	Number of subcommittees=10
SB100	SMOTEBoost, SMOTE=100%	Amount of SMOTE in each iteration equal to 100% size of the minority class
SB200	SMOTEBoost, SMOTE=200%	Amount of SMOTE in each iteration equal to 200% size of the minority class
SB500	SMOTEBoost, SMOTE=500%	Amount of SMOTE in each iteration equal to 500% size of the minority class
RUSB	RUSBoost	Random Undersampling in each iteration until 50% of the data belongs to minority class
RB-B	RB-Boost	Random Balance in each iteration

Table 5.6: Algorithms used in the experimental study: Boosting family

fier ensembles depending on the main diversity-generating strategy. We distinguished three such families: Data-preprocessing-only, Bagging and Boosting. The names, abbreviations and descriptions of the methods can be found in tables 5.4, 5.5 and 5.6.

The scores obtained by the proposed methods: E-RB, BAG-RB and RB-B are shown in Table 5.7, the reader is encouraged to consult the full table of results in the supplementary material[9]. Some of the methods obtain low result in certain datasets. The reason is that some of the performance measures are a geometric mean (the G-mean) and a harmonic mean (the F-measure) so the results are biased towards the lower of the two values that are combined in the measure. With a classifier that always predict the majority class the accuracy will be very high (depending on the imbalance ratio), the AUC will be 0.5 if all the instances are given the same confidence; but for these two means the value will be 0.

Table 5.7: Scores of the proposed methods according to de AUC, F-Measure and Geometric Mean

	AUC			F-Measure			Geometric Mean		
Dataset	E-RB	BAG-RB	RB-B	E-RB	BAG-RB	RB-B	E-RB	BAG-RB	RB-B
hddt_boundary	0.6748	0.6945	0.7085	0.1421	0.1132	0.0388	0.3552	0.2730	0.1320
hddt_breast-y	0.6414	0.6460	0.6223	0.4417	0.4496	0.4023	0.5805	0.5872	0.5454
hddt_cam	0.7277	0.7631	0.7665	0.1922	0.1916	0.1356	0.3715	0.3610	0.2916
hddt_compustat	0.9072	0.9107	0.9320	0.3404	0.3632	0.4538	0.7924	0.7780	0.5965
hddt_covtype	0.9933	0.9934	0.9959	0.8517	0.8586	0.9055	0.9587	0.9555	0.9439
hddt_credit-g	0.7508	0.7695	0.7546	0.5536	0.5790	0.5173	0.6723	0.6941	0.6334
hddt_estate	0.6239	0.6239	0.6138	0.2425	0.2373	0.0813	0.5320	0.5266	0.2176
hddt_german-numer	0.7750	0.7856	0.7626	0.5819	0.6002	0.5334	0.6965	0.7134	0.6438
hddt_heart-v	0.6907	0.7067	0.7056	0.4250	0.4350	0.4107	0.5822	0.5871	0.5617
hddt_hypo	0.9911	0.9905	0.9925	0.8685	0.8793	0.8863	0.9610	0.9635	0.9432
hddt_ism	0.9394	0.9421	0.9130	0.5359	0.5660	0.6804	0.8860	0.8836	0.8308
hddt_letter	0.9990	0.9994	0.9999	0.9569	0.9618	0.9768	0.9747	0.9719	0.9779
hddt_oil	0.9128	0.9201	0.9281	0.4510	0.5254	0.5504	0.7642	0.7454	0.6679
hddt_optdigits	0.9986	0.9980	0.9999	0.9793	0.9811	0.9925	0.9901	0.9902	0.9937
hddt_page	0.9918	0.9918	0.9911	0.8498	0.8568	0.8792	0.9581	0.9569	0.9340
hddt_pendigits	0.9995	0.9996	1.0000	0.9725	0.9775	0.9892	0.9859	0.9869	0.9921
hddt_phoneme	0.9339	0.9379	0.9502	0.7837	0.7905	0.8149	0.8636	0.8663	0.8675
hddt_PhosS	0.7183	0.7502	0.7276	0.1753	0.1204	0.0045	0.3432	0.2598	0.0300
hddt_satimage	0.9513	0.9517	0.9620	0.6354	0.6427	0.6916	0.8513	0.8440	0.7929
hddt_segment	0.9991	0.9989	0.9999	0.9727	0.9753	0.9912	0.9873	0.9880	0.9932
keel_abalone19	0.7427	0.7685	0.7154	0.0535	0.0607	0.0284	0.4729	0.3001	0.1099
keel_abalone9-18	0.7919	0.8081	0.8070	0.3077	0.3487	0.3769	0.6512	0.6237	0.5716
keel_cleveland-0_vs_4	0.9377	0.9539	0.9572	0.5551	0.6396	0.5681	0.7246	0.7622	0.6754
keel_ecoli-0-1-3-7_vs_2-6	0.9278	0.9321	0.9204	0.6382	0.6226	0.5110	0.8256	0.7971	0.6880
keel_ecoli-0-1-4-6_vs_5	0.9654	0.9637	0.9892	0.6883	0.7310	0.8031	0.8336	0.8442	0.8912
keel_ecoli-0-1-4-7_vs_2-3	0.9308	0.9333	0.9325	0.6390	0.6721	0.6978	0.8355	0.8236	0.8243
keel_ecoli-0-1-4-7_vs_5-6	0.9521	0.9603	0.9668	0.7227	0.7195	0.8016	0.8634	0.8359	0.8635
keel_ecoli-0-1_vs_2-3-5	0.9480	0.9507	0.9495	0.6878	0.7247	0.7508	0.8726	0.8763	0.8544
keel_ecoli-0-1_vs_5	0.9579	0.9709	0.9853	0.6755	0.7272	0.7602	0.8287	0.8547	0.8506
keel_ecoli-0-2-3-4_vs_5	0.9690	0.9729	0.9833	0.6741	0.7135	0.7529	0.8717	0.8830	0.8746
keel_ecoli-0-2-6-7_vs_3-5	0.9261	0.9285	0.9305	0.7111	0.7537	0.7589	0.8573	0.8636	0.8553
keel_ecoli-0-3-4-6_vs_5	0.9568	0.9667	0.9789	0.7124	0.7425	0.7791	0.8683	0.8700	0.8841
keel_ecoli-0-3-4-7_vs_5-6	0.9474	0.9497	0.9622	0.7236	0.7103	0.7983	0.8718	0.8405	0.8706
keel_ecoli-0-3-4_vs_5	0.9619	0.9671	0.9815	0.7103	0.7475	0.7433	0.8441	0.8495	0.8495

Continued on next page

[9] https://github.com/joseFranciscoDiez/research/wiki/Supplementary-Material

Table 5.7: Scores of the proposed methods according to de AUC, F-Measure and Geometric Mean

	AUC			F-Measure			Geometric Mean		
Dataset	E-RB	BAG-RB	RB-B	E-RB	BAG-RB	RB-B	E-RB	BAG-RB	RB-B
keel_ecoli-0-4-6_vs_5	0.9677	0.9721	0.9840	0.7450	0.7638	0.7453	0.8824	0.8804	0.8254
keel_ecoli-0-6-7_vs_3-5	0.9213	0.9346	0.9237	0.6796	0.7124	0.6996	0.8402	0.8405	0.8059
keel_ecoli-0-6-7_vs_5	0.9541	0.9610	0.9612	0.7534	0.7614	0.8079	0.9023	0.9034	0.8953
keel_ecoli-0_vs_1	0.9954	0.9925	0.9909	0.9765	0.9728	0.9691	0.9814	0.9793	0.9771
keel_ecoli1	0.9543	0.9573	0.9456	0.7876	0.7847	0.7650	0.8936	0.8886	0.8538
keel_ecoli2	0.9429	0.9473	0.9639	0.7915	0.7910	0.8128	0.8825	0.8839	0.8694
keel_ecoli3	0.9391	0.9379	0.9209	0.6214	0.6174	0.5567	0.8574	0.8519	0.7162
keel_ecoli4	0.9630	0.9724	0.9855	0.6585	0.6947	0.7911	0.8196	0.8385	0.8860
keel_glass-0-1-2-3_vs_4-5	0.9724	0.9747	0.9767	0.8412	0.8508	0.8363	0.9135	0.9183	0.8867
keel_glass-0-1-4-6_vs_2	0.7662	0.7510	0.7737	0.2970	0.3398	0.2646	0.5575	0.5606	0.4106
keel_glass-0-1-5_vs_2	0.7551	0.7466	0.7489	0.3464	0.2671	0.2688	0.6330	0.4768	0.4377
keel_glass-0-1-6_vs_2	0.7335	0.7148	0.7582	0.2650	0.2425	0.1841	0.5334	0.4649	0.3315
keel_glass-0-1-6_vs_5	0.9948	0.9938	0.9909	0.7913	0.7882	0.6867	0.9856	0.9756	0.8161
keel_glass-0-4_vs_5	0.9957	0.9964	0.9956	0.9505	0.9505	0.8519	0.9939	0.9939	0.9185
keel_glass-0-6_vs_5	0.9843	0.9837	0.9907	0.8946	0.8946	0.7988	0.9493	0.9493	0.8719
keel_glass0	0.8593	0.8694	0.8833	0.7216	0.7206	0.7172	0.7976	0.7967	0.7868
keel_glass1	0.8146	0.8264	0.8592	0.6354	0.6791	0.6997	0.7105	0.7466	0.7602
keel_glass2	0.8214	0.8020	0.7502	0.2984	0.2500	0.2469	0.6008	0.5043	0.3848
keel_glass4	0.9117	0.9322	0.9628	0.4748	0.5512	0.5128	0.7349	0.7836	0.6551
keel_glass5	0.9922	0.9905	0.9864	0.7606	0.7571	0.6602	0.9759	0.9754	0.7761
keel_glass6	0.9530	0.9602	0.9565	0.8239	0.8423	0.8551	0.9167	0.9235	0.9109
keel_haberman	0.7090	0.7130	0.6735	0.5002	0.4943	0.3409	0.6518	0.6454	0.4957
keel_iris0	1.0000	1.0000	1.0000	0.9813	0.9813	0.9813	0.9816	0.9816	0.9816
keel_led7digit-0-2-4-5-6-	0.9577	0.9605	0.9653	0.7541	0.7779	0.7667	0.8925	0.8960	0.8714
keel_new-thyroid1	0.9936	0.9949	0.9971	0.9077	0.9124	0.9270	0.9413	0.9494	0.9546
keel_new-thyroid2	0.9950	0.9953	0.9983	0.8960	0.8993	0.9455	0.9482	0.9420	0.9637
keel_page-blocks-1-3_vs_4	0.9997	0.9995	0.9998	0.9284	0.9271	0.9610	0.9700	0.9698	0.9837
keel_page-blocks0	0.9913	0.9912	0.9904	0.8455	0.8530	0.8692	0.9519	0.9537	0.9302
keel_pima	0.8185	0.8214	0.8018	0.6654	0.6721	0.6225	0.7387	0.7451	0.7025
keel_segment0	0.9983	0.9986	0.9999	0.9683	0.9700	0.9881	0.9847	0.9847	0.9919
keel_shuttle-c0-vs-c4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
keel_shuttle-c2-vs-c4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
keel_vehicle0	0.9885	0.9896	0.9958	0.8803	0.8803	0.9335	0.9391	0.9394	0.9588
keel_vehicle1	0.8452	0.8510	0.8511	0.6243	0.6197	0.5608	0.7643	0.7541	0.6830
keel_vehicle2	0.9936	0.9945	0.9981	0.9329	0.9404	0.9665	0.9653	0.9665	0.9772
keel_vehicle3	0.8478	0.8475	0.8458	0.6162	0.6140	0.5442	0.7626	0.7524	0.6647
keel_vowel0	0.9965	0.9965	0.9997	0.8733	0.8787	0.9697	0.9623	0.9681	0.9817
keel_wisconsin	0.9921	0.9924	0.9931	0.9521	0.9501	0.9526	0.9661	0.9635	0.9646
keel_yeast-0-2-5-6_vs_3-7	0.8449	0.8533	0.8427	0.5531	0.5957	0.5896	0.7745	0.7731	0.7195
keel_yeast-0-2-5-7-9_vs_3	0.9483	0.9444	0.9436	0.7434	0.7775	0.8049	0.8956	0.9022	0.8799
keel_yeast-0-3-5-9_vs_7-8	0.7573	0.7638	0.7565	0.3717	0.3869	0.3635	0.6575	0.6694	0.5319
keel_yeast-0-5-6-7-9_vs_4	0.8931	0.8963	0.8795	0.4982	0.5246	0.4876	0.7714	0.7433	0.6452
keel_yeast-1-2-8-9_vs_7	0.7373	0.7592	0.7477	0.1868	0.1785	0.2663	0.6400	0.4947	0.4452
keel_yeast-1-4-5-8_vs_7	0.6477	0.6617	0.6655	0.1644	0.1565	0.1135	0.5561	0.4222	0.2365
keel_yeast-1_vs_7	0.8096	0.8184	0.8059	0.3310	0.3350	0.3824	0.6851	0.6455	0.5663
keel_yeast-2_vs_4	0.9799	0.9799	0.9705	0.7149	0.7292	0.7514	0.9104	0.9070	0.8547
keel_yeast-2_vs_8	0.8167	0.8204	0.8216	0.4098	0.5572	0.5942	0.7089	0.7286	0.7238
keel_yeast1	0.7949	0.7992	0.7768	0.5920	0.6027	0.5309	0.7107	0.7212	0.6461
keel_yeast3	0.9741	0.9745	0.9641	0.7788	0.7811	0.7649	0.9320	0.9294	0.8603
keel_yeast4	0.9335	0.9381	0.9148	0.3336	0.3884	0.3790	0.8075	0.8055	0.5807
keel_yeast5	0.9897	0.9901	0.9766	0.7311	0.7269	0.6899	0.9461	0.9391	0.8438
keel_yeast6	0.9137	0.9168	0.8965	0.3685	0.4575	0.4997	0.7823	0.7869	0.6878

We used the most common configurations of SMOTE where the number of synthetic instances was set to 100%, 200% and 500% of the minority class. In the variants called ESM and BAGSM, the minority class was oversampled to match the size of the majority class. For the undersampling ensembles, the size of the majority class was reduced to match the size of

the minority class.

In addition, optimized versions of some the ensemble methods were tried. In the Data-preprocessing-only and the Bagging families we included three versions: optimizing the amount of SMOTE oversampling, optimizing the amount of Undersampling and optimizing both simultaneously. In all these variants we used a 5-fold internal cross-validation [10] and tested 10 different amounts of SMOTE and Undersampling, which means that the version that optimizes both parameters simultaneously has evaluated 100 possible combinations. These amounts are expressed in terms of the difference between the majority and minority class sizes, as shown in Figure 5.8, for SMOTE, in this case, a value of 0% means not to add any instance, a value of 100% means to create as many as necessary to match the size of the majority. For Undersampling a value of 0% means not to delete any instance, a value of 100% means remove instances to match the original size of the minority class. Once found, the parameters that maximize the AUC for a single decision tree are used for constructing the ensemble.

The Data-preprocessing-only also includes the Partitioning (or Random Splitting) method, described in [44]. In that work, the ensemble size was the Imbalanced Ratio, while in this work it is 100 [11], as for the other methods in this section, in order to make a fair comparison.

The Bagging family includes the Reliability-based Balancing (RbB) method [50]. The classifiers obtained with this method can be seen as a mini-ensemble of two classifiers, the first one using the original imbalanced class distribution (IC), the second one using a classifier with balanced data (BC). In order to have ensembles of 100 classifiers, two ensembles of 50 classifiers are combined. The first classifier is obtained with Bagging. For the second classifier two configurations are considered: Bagging with SMOTE and Bagging with Undersampling. RbB uses a threshold to determine which label return, when the reliability provided by IC is larger than the threshold, the final label corresponds to the label returned by IC, in the opposite case the label corresponds to the label returned by BC. This threshold is selected for each dataset, considering the values from 0.0 to 1.0 in steps of size 0.05, the threshold chosen is the one for which the sum of accuracy and geometric mean is maximized over a validation dataset.

[10] That means that the training set is repeatedly divided into train and validation sets to find the optimal parameter value, and then the classifier is finally built using the complete training set.

[11] To achieve this size, as many partitions as necessary are created. e.g. if the imbalance ratio is 5, the 100 classifiers are created using 20 times the partitioning technique.

The Data-preprocessing-only family includes the *Random Balance* ensemble (E-RB), while Bagging family includes the combination of Bagging and *Random Balance* (BAG-RB).

In the Boosting family, we have compared the most popular algorithms. For completeness, we included the standard boosting variants AdaBoost.M1 and MultiBoost. Both were tested with reweighting as well as with weighted resampling [21]. The main contenders in this family were the boosting variants especially designed for unbalanced data sets: SMOTEBoost, with 3 different rates of SMOTE, and RUSBoost. The proposed method: *RB-Boost* was also added to the boosting family.

For comparison between multiple algorithms for each family and multiple data sets we used average ranks [14]. For a given data set, the methods are sorted from best to worst. The best method receives rank 1, the second best receives rank 2, and so on. In case of a tie, average ranks are assigned. For instance, if two methods tie for the top rank, they both receive rank 1.5. Average ranks across all data sets are then obtained.

The first question is whether there are any significant differences between the ranks of the compared methods. The Friedman test and the subsequent version of Iman and Davenport [31] test were applied.

To detect pairwise differences between a designated method and the remaining methods, we used the Hochberg test [30], which was found to be more powerful than the Bonferroni-Dunn test [16, 24].

Table 5.8a shows the results of the comparison of the algorithms of the Data-preprocessing-only family in the form of average ranking calculated from the area under the curve. The second column shows the average rank of each method. The Iman and Davenport test gives a p-value of 6.1904e-86, which means that it rejects the hypothesis that the compared algorithms are equivalent. The last column shows the adjusted Hochberg p-value between E-RB and the respective method of that row. An adjusted p-value less than 0.05 means that the two methods are significantly different with a significance of $\alpha = 0.05$. The table shows that the Random Balance ensemble (E-RB) has a demonstrably better AUC than all the other ensembles in this family.

Table 5.8b shows the average ranks for the Bagging family calculated using the same measure. With p-value of 8.1240e-56, the Iman and Davenport test discards the hypothesis of equivalence between the algorithms. The combination of Bagging with the proposed method obtains the best

Algorithm	Average Rank	*p*-Hochberg
E-RB	2.2674	
ERUSR	3.8256	0.0021
EPart	4.1337	4.4869e-004
ERUS	4.4767	3.7599e-005
EopB	5.5930	1.9442e-010
ESM200	6.7442	4.3307e-018
ESM500	7.0291	2.8543e-020
ESM100	7.1861	1.6549e-021
EopU	7.6744	9.0344e-026
ESM	8.2674	1.6612e-031
EopS	8.8023	3.4537e-037

(a) data-processing family

Algorithm	Average Rank	*p*-Hochberg
BAG-RB	3.0930	
BAGopB	5.9302	1.7768e-006
BAGSM500	6.0465	1.3180e-006
BAGSM200	6.1456	8.2646e-007
BAGSM100	6.3081	2.4709e-007
BAGRUS	6.3256	2.4709e-007
BAGopS	6.4826	6.8877e-008
BAGSM	6.5058	6.3800e-008
BAGopU	6.6977	1.0265e-008
SMBAG	8.3663	6.0674e-018
BAG	8.5233	6.0521e-019
RbB:IC+BAGSM	10.1919	6.8898e-032
RbB:IC+BAGRUS	10.3837	1.4618e-033

(b) Bagging Family

Algorithm	Average Rank	*p*-Hochberg
RB-B	2.9884	
RUSB	3.6628	0.10634
SB200	4.4419	0.00100
SB500	4.5116	7.9490e-004
SB100	4.8139	4.9417e-005
MultiS	4.9128	2.0338e-005
AdaM1S	6.0407	1.6196e-012
MultiW	6.1977	1.0754e-013
AdaM1W	7.4302	1.6253e-025

(b) Boosting Family

Table 5.8: Average ranks (AUC)

ranking and also presents significant differences with the other methods.

Table 5.8c shows the average ranks for the Boosting family. With *p*-value of 1.0638e-37 the Iman and Davenport test discards the hypothesis of equivalence. The proposed algorithm *RB-Boost* takes the top spot for the AUC criterion, and there are significant differences with all other algorithms, except RUSBoost, which occupies the second position (adjusted Hochberg's *p*-value of 0.10634).

Table 5.9a shows the average ranks for the data-processing according to the F-Measure. In this case the Iman and Davenport test gives a *p*-value of 2.3794e-44, so the compared algorithms are not equivalent. The Random Balance ensemble gets the best ranking, but this time there are no statistically significant differences with the next three algorithms.

Table 5.9b shows the average ranks for the Bagging family according to the F-Measure. The Iman and Davenport test discards the hypothesis of equivalence between the algorithms with *p*-value of 1.2896e-23. The proposed method obtains the second highest ranking, but there are no significant differences from the first method.

Finally, Table 5.9c shows the average ranks for the Boosting family. With *p*-value of 6.912e-11 the Iman and Davenport test discards the hypothesis of equivalence between the algorithms. The proposed algorithm has the best place in the ranking with significant differences with all remaining algorithms in this family.

Figure 5.9 shows scatter plots with the average ranks for the three families of methods. The best methods according to the AUC appear at the left, and the best methods according to the F-Measure appear at the bottom.

Similar patterns appear in the left and center plots:

Algorithm	Average Rank	*p*-Hochberg
E-RB	4.0639	
ESM200	4.3023	0.6374
ESM500	4.3954	0.6374
ESM100	4.7674	0.4928
EopB	5.5116	0.0168
ESM	5.7326	0.0049
EopS	5.9070	0.0016
EopU	6.7790	5.5677e-007
ERUS	7.9419	1.4066e-013
EPart	7.9535	1.3225e-013
ERUSR	8.6454	1.3278e-018

(a) data-processing family

Algorithm	Average Rank	*p*-Hochberg
BAGSM500	5.3081	
BAG-RB	5.5930	0.6315
BAGSM200	5.6686	0.6315
BAGSM	5.9651	0.6315
BAGopS	6.3023	0.3765
BAGSM100	6.4302	0.2942
RbB:IC+BAGSM	6.7791	0.0796
RbB:IC+BAGRUS	7.1977	0.0103
SMBAG	7.2035	0.0103
BAGopB	7.2616	0.0090
BAGopU	8.5639	4.2025e-007
BAG	8.7849	5.2748e-008
BAGRUS	9.9419	7.2984e-014

(b) Bagging Family

Algorithm	Average Rank	*p*-Hochberg
RB-B	3.3372	
RUSB	4.5639	0.00331
AdaM1S	4.6337	0.00331
SB500	4.7326	0.00250
MultiS	4.9942	2.9049e-004
SB200	5.2267	3.0287e-005
SB100	5.6919	1.0318e-007
AdaM1W	5.7733	3.8116e-008
MultiW	6.0465	6.9931e-010

(b) Boosting Family

Table 5.9: Average ranks (F-Measure)

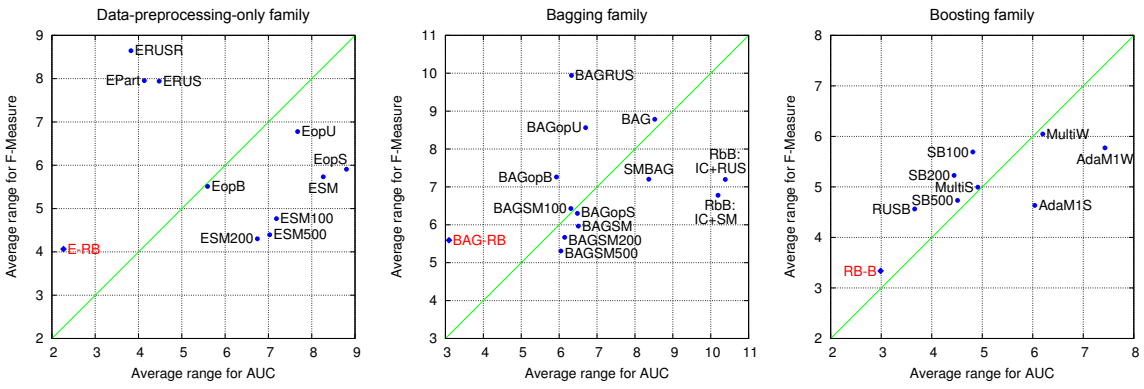


Figure 5.9: Average ranks for the ensemble methods, according to te AUC and F-Measure

Algorithm	Average Rank	*p*-Hochberg
E-RB	3.3779	
ERUS	3.7500	4.6192e-001
EPart	3.9767	4.6192e-001
ERUSR	4.3488	1.6470e-001
EopB	5.2442	8.9737e-004
ESM	6.4070	1.0563e-008
ESM500	6.5116	3.4791e-009
ESM200	7.8140	1.2430e-017
EopU	7.8372	9.4328e-018
EopS	8.2093	1.1410e-020
ESM100	8.5233	2.6149e-023

(a) data-processing family

Algorithm	Average Rank	*p*-Hochberg
BAG-RB	3.2326	
BAGRUS	4.1686	1.1500e-001
SMBAG	4.7267	2.3746e-002
BAGopB	5.3837	8.7661e-004
BAGSM	5.8430	4.4210e-005
BAGSM500	6.0640	9.3268e-006
BAGSM200	7.6570	5.6084e-013
BAGopU	7.6686	5.6084e-013
BAGopS	7.7791	1.5418e-013
RbB:IC+BAGRUS	8.3081	1.1445e-016
BAGSM100	8.9942	2.9739e-021
RbB:IC+BAGSM	9.3547	7.1062e-024
BAG	11.8198	2.6359e-046

(b) Bagging Family

Algorithm	Average Rank	*p*-Hochberg
RUSB	2.0872	
SB500	3.5233	5.8489e-004
RB-B	3.6686	3.0550e-004
SB200	4.7384	6.5421e-010
AdaM1S	5.4709	2.1606e-015
SB100	5.5756	3.3363e-016
MultiS	5.9477	1.4294e-019
AdaM1W	6.5872	3.1635e-026
MultiW	7.4012	3.4831e-036

(b) Boosting Family

Table 5.10: Average ranks (Geometric Mean)

- In the case of data processing family, ensembles which only use Random Undersampling (ERUS, ERUSR and EPart) obtain the three worst results for the F-Measure but according to the AUC criterion they are much better, only surpassed by E-RB.

- The ensembles that apply only SMOTE (ESM/100/200/500, EopS) are grouped into a cluster and methods that combine bagging and SMOTE (BAGSM/ 100/200/500, BAGopS) are grouped into another cluster.

- The proposed method appears far ahead of the other methods on the AUC criterion and is the best or the second best on the F-Measure criterion.

The right plot, showing the Boosting family, reveals that the methods are much closer to the diagonal line where the ranks for the AUC and the F-Measure are identical. The proposed method *RB-Boost* is located at a considerable distance from the other methods on both axes, which indicates its advantage.

Table 5.10 shows the rankings of the three families according to the geometric mean. The proposed methods get the best positions in the data processing and bagging families, in both cases significantly according to Hochberg's Test. But it gets the third position in the boosting family ranking.

Although accuracy is not usually considered an adequate performance measure for imbalanced data, for the sake of completeness, Table 5.11 shows the average ranks for the considered ensemble methods according to this measure. As it could be expected, the methods that do not consider

Data-level family	
Algorithm	Rank
ESM100	2.4826
ESM200	3.4593
EopS	4.2326
ESM500	4.7500
E-RB	5.0407
ESM	5.2267
EopB	6.0000
EopU	6.1570
EPart	9.1512
RUS	9.2209
ERUSR	10.2791

Bagging family	
Algorithm	Rank
BAG	3.5233
BAGSM100	3.9942
RbB:IC+BAGSM	4.6221
BAGSM200	5.1337
BAGopS	5.3198
RbB:IC+BAGRUS	6.3314
BAGSM500	6.6337
BAGSM	6.9128
BAG-RB	8.3721
BAGopU	8.7151
SMBAG	9.4884
BAGopB	9.5116
BAGRUS	12.4419

Boosting family	
Algorithm	Rank
MultiS	3.3314
AdaS	3.4826
MultiW	3.8430
RB-B	4.1337
AdaW	4.2093
SB100	6.0174
SB200	6.3023
RUSB	6.6337
SB500	7.0465

Table 5.11: Average ranks for the considered ensemble methods, obtained from the accuracies.

imbalance (i.e., Bagging, AdaBoost and MultiBoost) have the top ranks for their respective families.

In this paper we have used several different measures to evaluate the performance of various methods. Some measures such as AUC, F-Measure and Geometric Mean are specific to unbalanced datasets, while accuracy is not specific to unbalanced. A combined average rank has been calculated to show the overall performance of the four measures. This time the average rank for each method is the average of their average ranks for each measure. Table 5.12 shows the average ranks for the considered ensemble methods according to the combination of measures. In all families, the proposed method obtains the best position. In this case it is not appropriate to apply any test to detect equivalence between methods or pairwise differences because the values are not independent, for each dataset-algorithm pair there are several values (one per measure).

After comparing the methods within their own families, we performed a comparison between the methods that have achieved first place in their respective rankings.

Table 5.13 shows the average ranks for the best methods in each family, calculated for each different measure. With p-values of 8.113e-6 and 0.04933 the Iman and Davenport test discards the hypothesis of equivalence between the algorithms in AUC and F-Measure. By contrast a p-value of 0.91474, in the case of ranking calculated with the best methods according to their geometric means, indicates that there is not significant differences between methods, RUSBoost obtains the top position but it is equivalent to the next two methods.

In the ranking calculated from the AUC, the best position is for *Bagging-*

Algorithm	Average Rank
E-RB	3.6948
ESM200	5.5828
EopB	5.5858
ESM500	5.6744
ESM100	5.7427
EPart	6.3009
RUS	6.3503
ESM	6.4113
ERUSR	6.7645
EopS	6.7820
EopU	7.1105

(a) data-processing family

Algorithm	Average Rank
BAG-RB	5.0727
BAGSM500	6.0131
BAGSM200	6.1512
BAGSM	6.3067
BAGSM100	6.4346
BAGopS	6.4695
BAGopB	7.0218
SMBAG	7.4462
RbB:IC+BAGSM	7.7384
BAGopU	7.9113
RbB:IC+BAGRUS	8.0552
BAG	8.1599
BAGRUS	8.2195

(b) Bagging Family

Algorithm	Average Rank
RB-B	3.5320
RUSB	4.2369
MultiS	4.7965
AdaS	4.9070
SB500	4.9535
SB200	5.1773
SB100	5.5247
MultiW	5.8721
AdaW	6.0000

(b) Boosting Family

Table 5.12: Average ranks (combined)

RB, which shows significant differences with *Ensemble-RB*. In the ranking calculated with the F-Measure, the best position is for *RB-Boost*. In this case, despite the p-value given by the Iman and Davenport test, the post-hoc Hochberg test found no significant differences between the methods at $\alpha = 0.05$; the p-value of Hochberg between the first ranking method and the last one is 0.05954. The method which obtains the best rank according to accuracy is Multiboost with resampling, but we emphasize that accuracy is not the best measure to evaluate classification methods in imbalanced dataset. And finally, the method that obtains the best average ranking considering all measures is one of the proposed methods: Random Balance Boost (*RB-B*).

5.6.1 Fusion Rules

The outputs of the classifiers in an ensemble can be combined in several ways [38]. For Ensemble-RB and Bagging-RB, the outputs are combined using the simple average of probabilities. For RB-Boost, the outputs are combined using a weighted average (line 11 in Figure 5.5), because it is the method used in AdaBoost.M2 and its variants for imbalance (RUSBoost, SMOTEBoost).

This section considers other combination methods for Ensemble-RB and Bagging-RB: majority voting and product of probabilities. Tables 5.14 and 5.15 show the average ranks for the considered fusion rules. Iman and Davenport Test discards the hypothesis of equivalence between the

Algorithm	Average Rank	*p*-Hochberg
BAG-RB	1.76163	
RB-B	1.82558	0.67494
E-RB	2.41279	0.00004

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
RB-B	1.88372	
BAGSM500	1.90116	0.90894
E-RB	2.21512	0.05954

(b) F-Measure

Algorithm	Average Rank	*p*-Hochberg
RUSB	1.9651	
E-RB	2.0058	7.8957e-001
BAG-RB	2.0291	7.8957e-001

(c) G-Mean

Algorithm	Average Rank
MultiS	1.6395
BAG	1.7209
ESM100	2.6395

(d) Acurracy

Algorithm	Average Rank
RB-B	1.8488
BAG-RB	1.8532
E-RB	2.2980

(e) Combined

Table 5.13: Average ranks (Best algorithms)

Algorithm	Average Rank	*p*-Hochberg
Average	1.08721	
Product	1.95349	1.34e-8
Majority Voting	2.9593	2.43e-34

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
Majority Voting	1.49419	
Average	1.94186	3.33e-003
Product	2.56395	4.60e-012

(b) F-Measure

Table 5.14: Average ranks for Ensemble-RB Fusion Rules.

Algorithm	Average Rank	*p*-Hochberg
Average	1.11047	
Product	1.91860	1.16e-007
Majority Voting	2.97093	6.23e-034

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
Majority Voting	1.56395	
Average	1.94186	0.01321
Product	2.49419	2.12e-009

(b) F-Measure

Table 5.15: Average ranks for Bagging-RB Fusion Rules.

algorithms in all cases. Ensemble-RB and Bagging-RB show the same behaviour: for AUC the order of fusion rules is average, product and majority voting, while for F-Measure the order is majority voting, average and product. When comparing the best method with the remaining methods, the adjusted *p*-values for Hochberg's procedure are small (<0.015). Hence, which fusion rule is used gives significant differences.

5.6.2 Base Classifiers

Decision trees are usually used as base classifiers, since they are simple and fast to compute, and they are unstable (small variations in the training set can result in different trees and different predictions), which contributes to the diversity of the ensemble. This section considers the performance of the proposed ensemble methods with other two base classifiers: nearest neighbour (1-NN) and SVM with Gaussian kernel. Due to the high computational cost of SVM classifiers, the size of all ensembles used in the comparison of this section was set to 50.

Tables 5.16, 5.17 and 5.18 show the average ranks for, respectively, Ensemble-RB, Bagging-RB and RB-Boost. These tables show, for AUC and F-Measure, the average ranks of the three considered base classifiers with the corresponding ensemble methods. Decision trees work better in these ensembles than the other two considered alternatives: in all the ranks, decision trees have the top position. The differences are larger for AUC than for F-Measure.

Algorithm	Average Rank	*p*-Hochberg
J48	1.74419	
1-NN	2.04070	0.0519
SVM	2.21512	0.0040

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
J48	1.63953	
1-NN	1.88953	0.1011
SVM	2.47093	9.97e-008

(b) F-Measure

Table 5.16: Average ranks of base classifiers for Ensemble-RB.

Algorithm	Average Rank	*p*-Hochberg
J48	1.72674	
1-NN	2.05233	0.0328
SVM	2.22093	0.0024

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
J48	1.72674	
1-NN	1.91279	0.2225
SVM	2.36047	6.49e-005

(b) F-Measure

Table 5.17: Average ranks of base classifiers for Bagging-RB.

Algorithm	Average Rank	*p*-Hochberg
J48	1.27326	
SVM	2.17442	3.44e-009
1-NN	2.55233	9.94e-017

(a) AUC

Algorithm	Average Rank	*p*-Hochberg
J48	1.89535	
1-NN	2.01744	0.4234
SVM	2.08721	0.4167

(b) F-Measure

Table 5.18: Average ranks of base classifiers for RB-Boost.

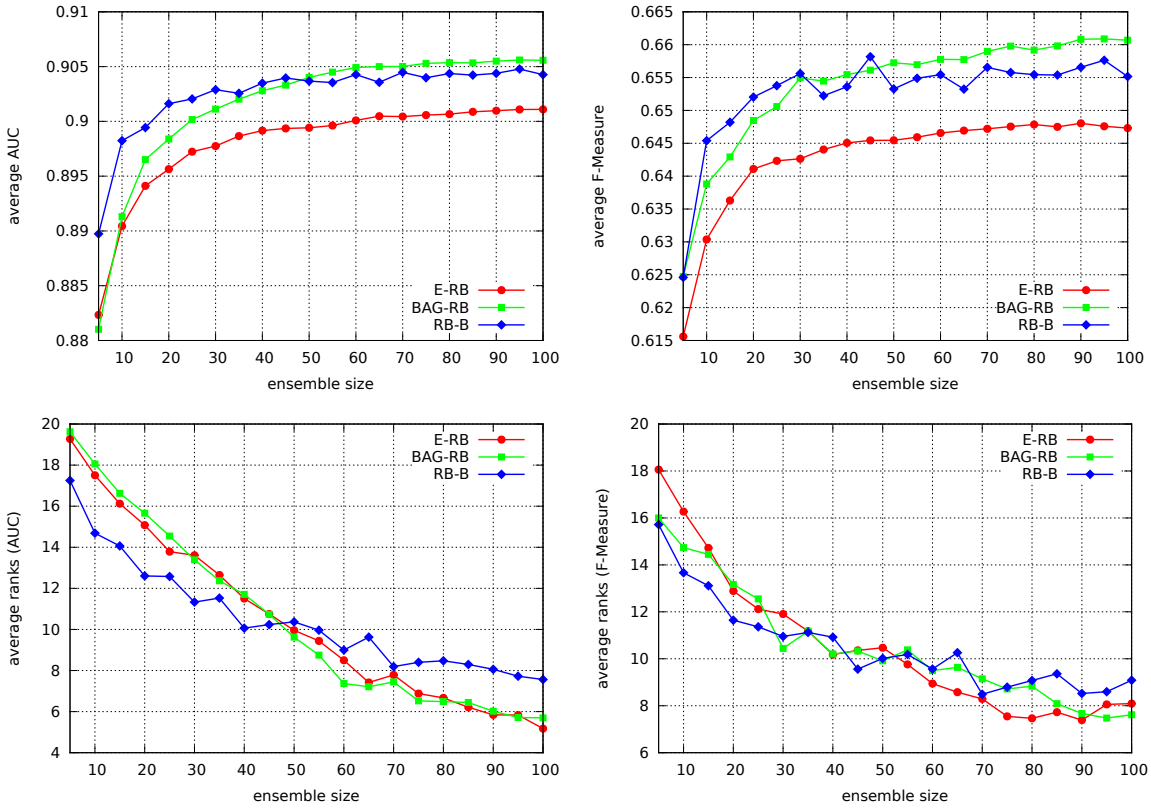


Figure 5.10: Performance measures as a function of the ensemble size.

5.6.3 Ensemble Size

In all the previous experiments the ensemble size was 100. This section considers the effect of the ensemble size in the performance. Figure 5.10 shows the performance as a function of the ensemble size. The ensemble sizes vary from 5 to 100 in steps of size 5. The top two plots show the average value of the performance measure (AUC or F-Measure) across all the data sets, for Ensemble-RB, Bagging-RB and RB-Boost. As usual with ensembles, the performance improves with size, but the improvements are smaller as the size grows. The bottom two plots shows the performance but using average ranks instead of the mean across all the data sets. For each method, 20 sizes are considered $(5, 10, 15, \ldots, 100)$, and the average ranks are computed for them. The values are in the interval [1,20], and smaller values represent better performance. The average ranks are better as the ensemble size increases.

Figure 5.11 shows six plots, each one compares a pair of methods across the considered ensemble sizes. Each plot shows, as a function of the ensemble size, the percentage of data sets with the best performance in

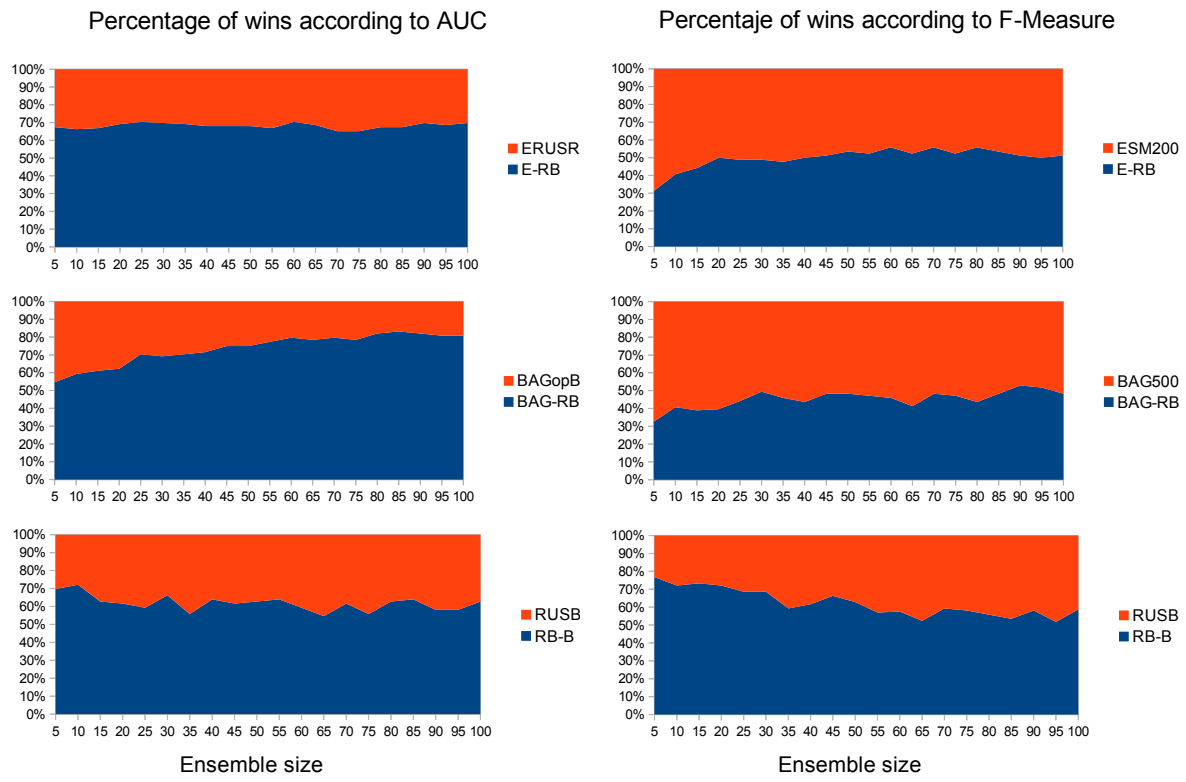


Figure 5.11: Comparison of methods as a function of the ensemble size.

terms of AUC (left plots) or F-Measure (right plots). The selected pairs are the two methods with best average ranks in each family (tables 5.8 and 5.9).

According to the AUC, the methods based on RB have a percentage of victories around 70%. In the left center plot, when comparing Bagging-RB with BAGopB (Bagging with the amount of SMOTE and Undersampling selected by cross validation), the initial percentage is smaller but it increases to greater values with the ensemble size

For the F-Measure (right plots), Ensemble-RB and Bagging-RB are not better than the corresponding pairs (ESM200 and BAG500), this was expected as in tables 5.8 and 5.9 the differences for the considered pairs were not significant. When comparing RB-Boost with RUSBoost the difference is greater, although it decreases with the ensemble size.

5.7 Conclusion

We propose a new preprocessing technique adequate to balance datasets within ensemble methods: Random Balance, based on the idea of varying randomly the proportions of the classes, and applied it to design a new ensemble method: RB-Boost. In addition to boosting the AUC, this in-

tuitive heuristic bypasses the need to tune the sensitive class proportion parameter, common to most methods for imbalanced classification. Despite their simplicity, the two proposed methods have proved competitive when compared with other state-of-the-art ensembles, including those specifically devised for imbalanced data classification.

There are several future research lines:

- Study the performance of Random Balance in presence of several data intrinsic characteristics which have been proven to have a strong influence on imbalanced classification [42, 29]. Some of these problems are overlapping [51], noisy examples [7], small disjuncts [59] or borderline examples [45]. On several occasions these problems have been addressed with preprocessing techniques, these techniques could be combined using the same strategy that Random Balance uses to combine SMOTE and undersampling resulting in new methods. For example, the resampling strategy CBO [32] has been used successfully with small disjuncts; cleaning techniques such as ENN [60] or CNN [26] have been used with noisy datasets; and variants of SMOTE, such as, Safe-Level-SMOTE [8] or SPIDER [52] with borderline examples.
- Extend the ideas in this article to multiple-class unbalanced problems.
- Test other combinations of classifiers beyond the average or the weighted average.

Acknowledgments

This work was partially supported by the project TIN2011-24046 of the Spanish Ministry of Economy and Competitiveness. We wish to thank the developers of Weka [27], the KEEL Experimental Analysis Framework [1] and the donors of the different data sets.

References

[1] J. Alcala-Fdez et al. “KEEL Data-Mining Software Tool: Data Set Repository and Integration of Algorithms and Experimental Analysis Framework”. In: *Journal of Multiple-Valued Logic and Soft Computing* 17.2-3 (2011), pp. 255–287.

[2] D. Anil Kumar and V. Ravi. “Predicting credit card customer churn in banks using data mining”. In: *International Journal of Data Analysis Techniques and Strategies* 1.1 (2008), pp. 4–28.

[3] R Barandela, RM Valdovinos, and JS Sánchez. "New applications of ensembles of classifiers". In: *Pattern Analysis & Applications* 6.3 (2003), pp. 245–256.

[4] G.E. Batista, R.C. Prati, and M.C. Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 20–29.

[5] R. Batuwita and V. Palade. "microPred: effective classification of pre-miRNAs for human miRNA gene prediction". In: *Bioinformatics* 25.8 (2009), pp. 989–995.

[6] L. Breiman. "Bagging predictors". In: *Machine Learning* 24 (1996), pp. 123–140.

[7] Carla E Brodley and Mark A Friedl. "Identifying Mislabeled Training Data". In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 131–167.

[8] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD09)*. Vol. 5476. Lecture Notes on Computer Science. Springer-Verlag, 2009, pp. 475–482.

[9] N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6.

[10] N.V. Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16.1 (2002), pp. 321–357.

[11] N.V. Chawla et al. "SMOTEBoost: Improving prediction of the minority class in boosting". In: *7th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD 2003)*. 2003, pp. 107–119.

[12] David A. Cieslak and Nitesh V. Chawla. "Learning Decision Trees for Unbalanced Data". In: *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*. ECML PKDD '08. Antwerp, Belgium: Springer-Verlag, 2008, pp. 241–256. ISBN: 978-3-540-87478-2. DOI: 10.1007/978-3-540-87479-9_34.

[13] David A. Cieslak et al. "Hellinger distance decision trees are robust and skew-insensitive". In: *Data Min. Knowl. Discov.* 24.1 (Jan. 2012), pp. 136–158. ISSN: 1384-5810. DOI: 10.1007/s10618-011-0222-1.

[14] Janez Demsar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[15] T.G. Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[16] O. Dunn. "Multiple comparisons among means". In: *Journal of the American Statistical Association* 56 (1961), pp. 52–64.

[17] Wei Fan et al. "AdaCost: Misclassification Cost-Sensitive Boosting". In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 97–105. ISBN: 1-55860-612-2.

[18] T. Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[19] Joseph L. Fleiss. *Statistical methods for rates and proportions*. Wiley series in probability and mathematical statistics. Applied probability and statistics. John Wiley & Sons, 1981. ISBN: 9780471064282.

[20] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. 2010. URL: http://archive.ics.uci.edu/ml.

[21] Y. Freund and R. E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[22] Yoav Freund and Robert E. Schapire. “Experiments with a New Boosting Algorithm”. In: *Machine Learning, Proceedings of the Thirteenth International Conference (ICML ’96), Bari, Italy, July 3-6, 1996*. 1996, pp. 148–156.

[23] M. Galar et al. “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42.4 (2012), pp. 463 –484. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2011.2161285.

[24] Salvador García et al. “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 Special Session on Real Parameter Optimization”. In: *J. Heuristics* 15.6 (2009), pp. 617–644.

[25] Nicolás García-Pedrajas et al. “Class imbalance methods for translation initiation site recognition in DNA sequences”. In: *Knowl.-Based Syst.* 25.1 (2012), pp. 22–34.

[26] K. Gowda and G. Krishna. “The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (Corresp.)” In: *Information Theory, IEEE Transactions on* 25.4 (1979), pp. 488–490.

[27] Mark Hall et al. “The WEKA data mining software: an update”. In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[28] H. Han, W.Y. Wang, and B.H. Mao. “Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning”. In: *2005 International Conference on Intelligent Computing (ICIC05)*. Vol. 3644. Lecture Notes on Computer Science. Springer-Verlag, 2005, pp. 878–887.

[29] Haibo He and Edwardo A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Trans. on Knowl. and Data Eng.* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. URL: http://dx.doi.org/10.1109/TKDE.2008.239.

[30] Y. Hochberg. “A sharper Bonferroni procedure for multiple tests of significance”. In: *Biometrika* 75 (1988), pp. 800–803.

[31] R.L. Iman and J.M. Davenport. “Approximations of the critical region of the fbietkan statistic”. In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[32] Taeho Jo and Nathalie Japkowicz. "Class imbalances versus small disjuncts". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 40–49.

[33] M.V. Joshi, V. Kumar, and R.C. Agarwal. "Evaluating boosting algorithms to classify rare classes: Comparison and improvements". In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE. 2001, pp. 257–264.

[34] S. B. Kotsiantis and P. E. Pintelas. "Mixture of expert agents for handling imbalanced data sets". In: *Annals of Mathematics, Computing & Teleinformatics* 1.1 (2003), pp. 46–55.

[35] M. Kubat and Matwin. "Addressing the Curse of Imbalanced Training Sets : One-Sided Selection". In: *Proceedings of the 14th International Conference on Machine Learning*. 1997, pp. 179–186.

[36] M. Kukar and I. Kononenko. "Cost-sensitive learning with neural networks". In: *Proceedings of the 13th European conference on artificial intelligence (ECAI-98)*. Citeseer. 1998, pp. 445–449.

[37] L. Kuncheva. "A Bound on Kappa-Error Diagrams for Analysis of Classifier Ensembles". In: *Knowledge and Data Engineering, IEEE Transactions on* PP.99 (2011), p. 1. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.234.

[38] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[39] T. Warren Liao. "Classification of weld flaws with imbalanced class data". In: *Expert Systems with Applications* 35.3 (2008), pp. 1041 –1052. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2007.08.044.

[40] C.X. Ling, V.S. Sheng, and Q. Yang. "Test strategies for cost-sensitive decision trees". In: *Knowledge and Data Engineering, IEEE Transactions on* 18.8 (2006), pp. 1055–1067.

[41] Wei Liu et al. "A Robust Decision Tree Algorithm for Imbalanced Data Sets". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2010*. 2010, pp. 766–777.

[42] Victoria López et al. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* 250.0 (2013), pp. 113 –141. ISSN: 0020-0255. DOI: http://dx.doi.org/10.1016/j.ins.2013.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0020025513005124.

[43] Dragos D. Margineantu and Thomas G. Dietterich. "Pruning Adaptive Boosting". In: *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*. 1997, pp. 211–218.

[44] M Molinara, MT Ricamato, and F Tortorella. "Facing imbalanced classes through aggregation of classifiers". In: *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE. 2007, pp. 43–48.

[45] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. "Learning from imbalanced data in presence of noisy and borderline examples". In: *Rough Sets and Current Trends in Computing*. Springer. 2010, pp. 158–167.

[46] C. Phua, D. Alahakoon, and V. Lee. “Minority report in fraud detection: classification of skewed data”. In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 50–59.

[47] F. Provost and P. Domingos. “Tree induction for probability-based ranking”. In: *Machine Learning* 52.3 (2003), pp. 199–215.

[48] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[49] C. Seiffert et al. “RUSBoost: A hybrid approach to alleviating class imbalance”. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40.1 (2010), pp. 185–197.

[50] Paolo Soda. “A multi-objective optimisation approach for class imbalance learning”. In: *Pattern Recognition* 44.8 (2011), pp. 1801–1810.

[51] Jerzy Stefanowski. “Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data”. In: *Emerging Paradigms in Machine Learning*. Springer, 2013, pp. 277–306.

[52] Jerzy Stefanowski and Szymon Wilk. “Selective pre-processing of imbalanced data for improving classification performance”. In: *Data Warehousing and Knowledge Discovery*. Springer, 2008, pp. 283–292.

[53] Y. Sun et al. “Cost-sensitive boosting for classification of imbalanced data”. In: *Pattern Recognition* 40 (2007), pp. 3358–3378.

[54] C.J. Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[55] K. Veropoulos, C. Campbell, and N. Cristianini. “Controlling the Sensitivity of Support Vector Machines”. In: *Proceedings of the International Joint Conference on AI*. 1999, pp. 55–60.

[56] Sofia Visa and Anca Ralescu. “Issues in mining imbalanced data sets - a review paper”. In: *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*. 2005, pp. 67–73.

[57] S. Wang and X. Yao. “Diversity analysis on imbalanced data sets by using ensemble models”. In: *IEEE Symposium Series on Computational Intelligence and Data Mining(IEEE CIDM 2009)*. 2009, pp. 324–331.

[58] Shuo Wang and Xin Yao. “Relationships between diversity of classification ensembles and single-class performance measures”. In: *Knowledge and Data Engineering, IEEE Transactions on* 25.1 (2013), pp. 206–219.

[59] Gary M Weiss. “The impact of small disjuncts on classifier learning”. In: *Data Mining*. Springer. 2010, pp. 193–226.

[60] D.L. Wilson. “Asymptotic properties of nearest neighbor rules using edited data”. In: *Systems, Man and Cybernetics, IEEE Transactions on* 2.3 (1972), pp. 408–421.

Chapter 6

Diversity techniques improve the performance of the best imbalance learning ensembles

Authors Jose F Diez-Pastor; Juan J. Rodriguez; Cesar I Garcia-Osorio; Ludmila I Kuncheva

Type Journal

Published in Information Sciences

Year Review (Minor changes).

Abstract

Many real-life problems can be described as unbalanced, where the number of instances belonging to one of the classes is much larger than the numbers in other classes. Examples are spam detection, credit card fraud detection or medical diagnosis. Ensembles of classifiers have acquired popularity in this kind of problems for their ability to obtain better results than individual classifiers. The most commonly used techniques by those ensembles specially designed to deal with imbalanced problems are for example re-weighting, oversampling and undersampling. Other techniques, originally intended to increase the ensemble diversity, have not been systematically studied for their effect on imbalanced problems. Among these are Random Oracles, Disturbing Neighbors, Random Feature Weights or Rotation Forest. This paper presents an overview and an experimental study of various ensemble-based methods for imbalanced problems, the methods have been tested in its original form and in conjunction with several diversity-increasing

techniques, using 84 imbalanced data sets from two well known repositories. This paper shows that these diversity-increasing techniques significantly improve the performance of ensemble methods for imbalanced problems and provides some ideas about when it is more convenient to use these diversifying techniques.

Index terms— Classifier ensembles, imbalanced data sets, SMOTE, Undersampling, Rotation Forest, diversity

6.1 Introduction

The class imbalance problem[1] arises when one class has much more examples than the others [10].

Imbalance learning has attracted much attention because imbalanced data sets are common in real world problems like those related to security: spam detection [29], fraud detection [17], software defect detection [64]; biomedical data: finding the transition between coding and non-coding DNA in genes [27], mining cancer gene expression [69]; or financial data, for example, risk predictions in credit data [25].

Classification of imbalanced data is difficult because standard classifiers are driven by accuracy, hence the minority class may simply be ignored [62], besides generally all classifiers present some performance loss when the data is unbalanced [49]. In addition, many imbalanced datasets suffer problems related to its intrinsic characteristic. According to [44] there are at least six of these problems: overlapping [57], lack of density and information [65], noisy examples [8], small disjuncts [67], the significance of borderline instances to discriminate between positive and negative classes [47] and differences in the data distributions between training and test stages [53].

In [23], the approaches to dealing with unbalanced datasets are sorted into four categories[2]:

• **The algorithm-level** category encompasses modifications of existing general learning algorithms which bias the learning toward the minority class. Examples of this category are Hellinger Distance Decision Trees (HDDT) [14], Class Confidence Proportion Decision Tree (CCPDT) [42] and Significant, Positively Associated and Relatively Class Correlated Classification Trees (SPARCCC) [61], as well as other class-size insensitive decision trees. In other occasions misclassification costs are different for different examples, [70] presents decision tree and Naïve Bayesian learning methods that learns with unknown costs.

• **The data-level** category includes pre-processing algorithms that change the prior distribution of the classes either by increasing the number of

[1] Being aware of the terminological debate about "*unbalanced*" versus "*imbalanced*", we will use both words interchangeably. Our reason is that a keyword look-up should be able to retrieve this study, whichever word has been picked.

[2] Notice that these categories are not mutually exclusive, for example some cost-sensitive methods can be included in the classifier ensembles category

minority class examples or by reducing the size of the majority class. In the first category of algorithms the simplest technique is to randomly add examples, without caring about neighbors from other class or the overlap between classes, some examples are Oversampling [4], SMOTE [11]. Other methods creates artificial instances taking into account these issues: Borderline-SMOTE [31], Safe-level SMOTE [9], ADASYN [32] or Cluster Based Oversampling [38]. In the second category Random Undersampling [3] removes random examples from the majority class and other methods like Edited Nearest Neighbor (ENN) [68] and Tomek Links [59] are based on data cleaning techniques.

• **The cost-sensitive** category contains methods that assign different costs for each class. Examples include AdaCost [20], AdaC1, AdaC2, and AdaC3 [58].

• **Classifier ensembles** [40, 48] are combinations of several classifiers which are called base classifiers or member classifiers. Ensembles often give better results than individual classifiers. Although ensembles were not designed to work with imbalanced data, they have been successfully applied to this task through combination with processing techniques from the data-level category.

According to [23], the algorithm level and cost-sensitive approaches are more problem-dependent, whereas data level and ensemble learning approaches based on data processing are more versatile.

Ensemble methods for imbalanced learning tackle the imbalance problem using techniques like re-weighting, oversampling and undersampling. These preprocessing techniques attempt to train base classifiers with a less unbalanced dataset. These preprocessing techniques not only address the problem of imbalance, but add diversity, since each base classifier is trained on a different version of the data set. Diversity is one of the cornerstones of ensembles. An ideal ensemble system should have accurate individual classifiers and at the same time their errors should be in different instances. Several techniques have been developed to increase the diversity of an ensemble (see Section 6.2.4). In this paper we argue that techniques specially designed to increase diversity impact the performance of imbalance learning, significantly improving even the specific techniques.

To prove this claim, we conducted an experimental study where both classifiers specially designed for unbalanced sets and standard classifiers

were tested in its original form and in conjunction with several diversity-increasing techniques. Finding that ensembles combined with diversity-increasing techniques ranked better than their original counterpart, even though the original version was specifically designed to work for imbalanced data. We also try to provide some clues when it is more appropriate to use diversity techniques using meta-learning, and evaluate the performance of the techniques in the presence of noisy and borderline examples.

The rest of the paper is structured as follows: Section 6.2 presents some background of ensemble learning, state-of-the-art techniques for imbalanced data, and our research hypothesis concerning diversity enhancing techniques. Section 6.3 shows the experimental study and results. Section 6.4 enumerate each one of the findings extracted in the experimental study. and finally, in Section 6.5 and 6.6 the conclusions and several future lines of research are presented.

6.2 Ensemble learning for imbalanced problems

In this section, the concept of ensemble and the important of diversity will be introduced, then the preprocessing techniques and the ensembles methods for imbalanced problems used in this paper will be described. Finally, several techniques to increase the diversity in ensembles will be explained

6.2.1 Ensembles of classifiers

Ensemble of classifiers are combinations of multiple classifiers, referred as base classifiers. Ensembles usually achieve better performance than any of the single classifiers [40]. In order to build a good ensemble, it is necessary not only to build good base classifiers, also the base classifiers must be diverse, this means that for the same instance, the base classifiers return different outputs and their errors should be in different instances. Ensemble methods differ in the way they induce diversity between the base classifiers. The most common approach is modifying the training set for each member of the ensemble. In Bagging [6], each base classifier is obtained from a random sample of the training data. In the resampling version of AdaBoost [22], the data set for each subsequent ensemble member is drawn according to a distribution of weights over the data. The weights are modified depending on the correctness of the prediction given to the

example by the previous classifier. In this way, the next classifier will give more importance to the difficult examples. MultiBoost [66] combines AdaBoost with Wagging (Weighted Bagging) [5], a variant of Bagging which, instead of creating samples from the original dataset, it randomly modifies the weight associated to each instance.

6.2.2 Preprocessing techniques for imbalance learning

Preprocessing techniques aiming at balancing the class proportions can be easily embedded into an ensemble. The strategies are usually to increase the size of the minority class, to reduce the size of the majority class, or do both at the same time. While there are many variants of such preprocessing techniques, we will focus on:

- **Random Undersampling**, that is done by eliminating random examples from the majority class. A drawback of this method is that, potentially, it can discard useful data. However, this adverse effect is minimized when using ensembles, since instances discarded in one iteration can remain in others. The most common implementation is to remove as much majority instances as necessary to match the size of the minority class. When Random Undersampling is used in this way in the experimental study, we name it RUS in the abbreviated names.

- **Random Oversampling**, that creates copies of minority class instances randomly chosen. This method can lead to overfitting, since it creates copies of existing instances.

- **SMOTE**, that creates artificial instances for the minority class. To create an instance from an existing one, it randomly generates a synthetic example along the imaginary line that connects the instance with one of its k nearest neighbors from the same class. The amount of SMOTE, the number of generated synthetic instances, is a parameter of the method, whose value differs from one study to another. In this paper, we have used the two most commonly used configurations. These are to create as much artificial instances of the minority class as needed to double the minority class size (we call it 100% of SMOTE and we use the abbreviation SM100 in figures and tables), and to create as much artificial instances of the minority class as necessary to match the size of the majority class (we simply call this SMOTE, shortened as SM).

SM100 only improves the balance ratio while SM equalizes the class proportions.

- **Random Balance**, that takes into account the fact that the optimal amount of undersampling and oversampling/SMOTE is problem-specific and has considerable influence on the performance of the classifier. Random Balance [19] is designed to be used within an ensemble, to solve the above problem, it relies on the randomness and repetition. Random Balance conserves the size of the dataset but varies the class proportions in the training sample of each base classifier using a random ratio. This includes the case where the minority class is over-represented and the imbalance ratio is inverted. SMOTE and Random Undersampling (resampling without replacement) are used to respectively increase or reduce the size of the classes to achieve the desired ratios. The procedure is simple, having a dataset S, with minority class S_P (subset of positive instances) and majority class S_N (subset of negative instances), it can be described as follows:

  1. A random number between 2 and $|S|-2$ is obtained. This number is going to be the new size of the majority class, *newMajSize*, and accordingly the new size of the minority class, *newMinSize*, will be $|S|-newMajSize$, so that the size of the new set, S', will be identical to the initial set ($|S|=|S'|$).
  2. If $newMajSize < |S_N|$, the new majority class, S'_N, is created by random sampling without replacement the original S_N so that its final size is $|S'_N| = newMajSize$, and the new minority class, S'_P, is obtained from S_P using SMOTE to get $newMinSize-|S_P|$ new artificial instances.
  3. Otherwise, S'_P is the class created as a random sample of S_P, and S'_N is the class grown by SMOTE from S_N, so that the final sizes are $|S'_P| = newMinSize$ and $|S'_N| = newMajSize$.

6.2.3 Ensemble methods specially designed for imbalance

Sometimes ensembles not designed specifically for imbalance are used in unbalanced problems. A very common way of doing this is to combine them with one of the previous preprocessing techniques, for example, combining Bagging with Undersampling as it is done in [3]. Other times the method is specially designed to deal with unbalanced datasets. This section lists

some of the most prominent methods specifically designed to deal with unbalanced.

- **SMOTEBagging** [63] is similar to Bagging except that each classifier is built with a data set with classes of equal size. The data set in each iteration is composed as follows. A bootstrap sample is taken from the majority class, keeping the original size, say N. The sample of size N for the minority class is created through a combination of Oversampling and SMOTE. The oversampling percentage varies in each iteration, ranging from 10% in the first iteration to 100% in the last, always being multiple of ten. The rest of the positive instances are generated by SMOTE.

- **SMOTEBoost** [12] is a modification of the re-weighting version of AdaBoost.M2. After each boosting round, SMOTE is applied in order to create new synthetic examples from the minority class. The synthetic instances always have the same weight, the weight of the instances in the original dataset, while the originals have weights that are updated according to a pseudo-loss function. Those instances that are difficult for the previous classifiers have bigger weights.

- **RAMOBoost** [13] is inspired by SMOTEBoost and ADASYN [32] algorithms. The main difference with SMOTEBoost is the way it creates positive instances. While SMOTE, used within SMOTEBoost, creates instances uniformly, in RAMOBoost there exists a sampling distribution based in the underlying data distribution. As a result, the artificial instances are created on the difficult regions of the decision boundary.

- **RUSBoost** [56] works similarly to SMOTEBoost. This time a Random Undersampling is applied after each boosting round, removing instances from the majority class. In this case, there are no new instances to assign weights to. It is only necessary to normalize the weights of the instances in the processed dataset with regard to the total sum of weights in the original dataset.

- **EUSBoost** [24] is based on RUSBoost and it aims to improve the original method by using evolutionary undersampling. EUSBoost also tries to promote diversity using different subsets of majority class instances to train each base classifier in each iteration.

- **EasyEnsemble** [43] samples several subsets from the majority class, trains an AdaBoost ensemble using repeatedly each of these subsets, and combines the outputs of those classifiers.

- **Random Balance-Boost** [19] follows the same philosophy as SMOTE-Boost and RUSBoost. Each base classifier is trained with a dataset obtained through Random Balance. The number of instances removed by undersampling is equal to the number of instances introduced by SMOTE. As in SMOTEBoost, synthetic instances are generated with a weight proportional to the total number of instances. The combination of SMOTE, UnderSampling and re-weighting provides more diversity which generally leads to better performance in ensemble learning.

6.2.4 Diversity-enhancing techniques

Diversity is essential in order to build an accurate ensemble of classifiers.

Figure 6.1 summarizes some widely used diversifying heuristics for building classifier ensembles based on manipulating of the training data.

Diversity is naturally promoted by oversampling, undersampling or re-weighting. Approaches which do not specifically target imbalance are often overlooked in imbalanced learning, in spite of their marked success in general multi-class classification or regression. Among these, the vertical approach methods (A) and (B) are commonly used to introduce diversity in ensembles. Here we propose that ensemble creation methods may offer more to imbalanced learning than what has already been achieved. To this end, we examine four further techniques illustrated in the lower part of Figure 6.1, called diversity-enhancing techniques; these are detailed below.

We decided on the following approaches:

- **Guided Random Sampling.** In the Random Oracle ensemble [41], for each iteration, the instances are divided into two groups using a random hyperplane, and then a classifier is built for each group. Ensemble methods based on random partitioning the training set into several samples are often used for scaling up classifiers in large databases. [51].

- **New Attributes.** Some methods, like Disturbing neighbors [45], expand the feature space with attributes that are not originally present in the dataset. For each base classifier in the ensemble, Disturbing neighbors uses N randomly selected instances, the disturbing neighbors, to

train a 1-NN (Nearest Neighbors) classifier, then it creates N binary attributes for each instance (with value 1 if the corresponding disturbing neighbor is the closest to the instance, 0 otherwise) and an additional attribute whose value is the class predicted by the 1-NN classifier, hence, the feature space is expanded with N+1 new attributes.

- **Random Weights.** It is possible to introduce diversity by giving a different importance/weight to every attribute for each member of the ensemble. Random Subspaces [34] can be viewed as a special case of this approach. A set of binary random weights is used, where a weight of value 0 means that the attribute is not included in the subset for the respective ensemble member. The result is that different classifiers are constructed using different subsets of the attributes. The Random Forest ensemble [7] is a bagging ensemble with random trees. A random tree differs from the standard tree only by its training. In random trees, a random subset of attributes is considered for the splitting of each node. This can be seen as a variation of Random Subspaces but instead of using the same subset for the whole tree the set is different in each node. Proposed more recently, the Random Feature Weights ensemble [46] associates a vector of weights with each tree of the ensemble. This vector is used to modify the way in which the merit function of the attributes is calculated. For a training set D and a weight vector $\mathbf{w}$, the new merit function for attribute a_i is defined as $f_{\mathbf{w}}(a_i, D) = w_i f(a_i, D)$, where $f(a_i, D)$ denotes the original merit function of a_i for D. Thus, the method introduces a bias which favours the selection and use of attributes with higher associated weight. The vector of weights is randomly drawn for each tree in the ensemble, thereby introducing diversity. These weights are real-valued (not just 0 and 1, as in Random Subspaces and Random Forest) so it is possible to draw a parallel with Wagging, but using features instead of instances.

- **Projections.** These are frequently used to reduce the dimensionality of the data. In the context of ensemble learning, projecting can be construed as a diversifying heuristic. A well known example is Rotation Forest [54], an ensemble method for decision trees which uses principal component analysis (PCA) to project different groups of attributes for each base classifier. A similar approach but using

supervised projections is Boosting Projections [26, 28]. Random Projections have been used to provide an extra diversity and embed the original set into a space of lower dimension [55]. Random Subspaces can be also seen as a special case of Random Projections.

The preprocessing techniques surrounded by a dashed line in the upper part of Figure 6.1 are specifically designed for dealing with imbalanced data. Each of these techniques alone can be used for creating an ensemble (indicated by the dashed arrows).

Here we are interested in finding out whether the preprocessing techniques (a)–(d) can be enhanced by (A)–(B), as well as other diversity heuristics (1)–(4), to produce better ensembles. Many of the techniques can be used in combination, both within their own group (for example, technique (d) described in 6.2.2 combines (a) with (c)) and with techniques from different groups (subsection 6.2.3 describes some ensemble learning methods that combine preprocessing techniques (a)–(d) with (A)–(B) sampling and re-weighting techniques). In the experimental study we have used a representative from each diversity heuristic: from (1), Random Oracle, from (2), Disturbing neighbors, from (3), Random Feature Weights, and from (4), Rotation Forest.

6.3 Experimental Set-up and Results

We intend to demonstrate that techniques for promoting diversity in classifier ensembles enhance the performance of bespoke state-ot-the-art ensembles for imbalance learning. The structure of this section is: firstly, the ensemble methods used in the experiments, along with their basic parameters and some clarifications of its operation are listed (Section 6.3.1). Secondly, the datasets used, along with their basic characteristics are shown (Section 6.3.2). Then (In Section 6.3.3), will be carried out a comparison between the ensembles and the same group of ensembles combined with the techiques used to increase the diversity listed in Section 6.2.4. The effect of the size ensemble is studied inf Section 6.3.4.

Some ideas about when it is more appropriate to use diversity techniques taking into account complexity measures of the datasets are provided in Section 6.3.5, and finally Section 6.3.6 explores the effect of Disturbing Neigbours in presence of noisy and borderline instances.

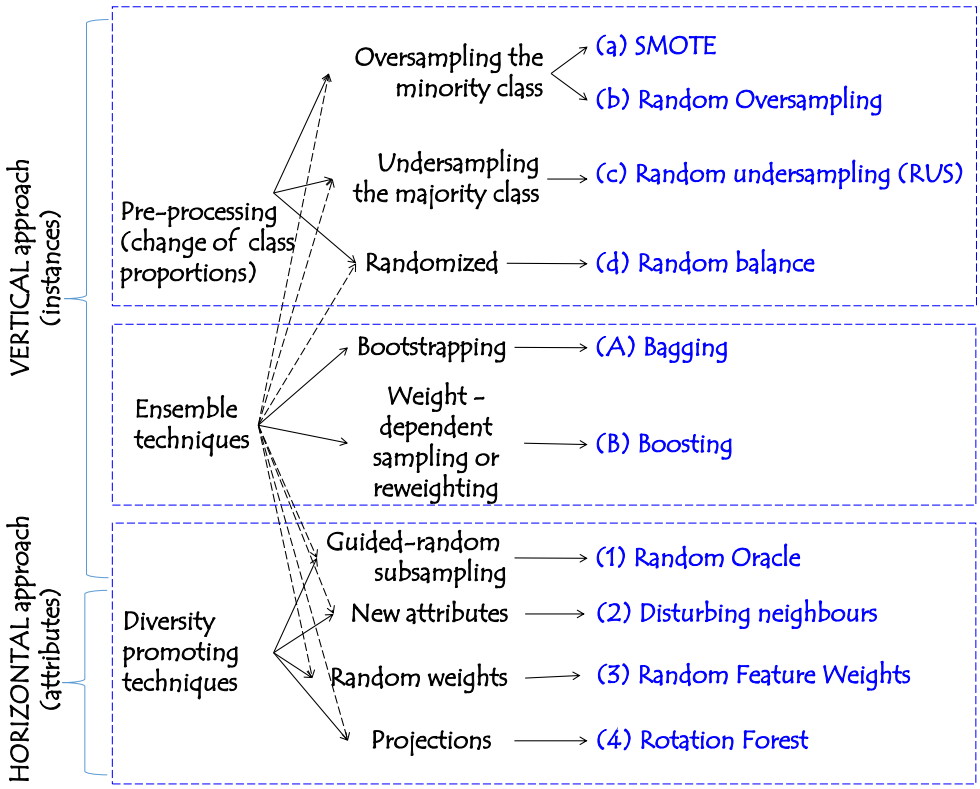


Figure 6.1: Ensemble diversifying heuristics based on data manipulation.

6.3.1 Ensemble methods tested in the experimental set-up

Ensembles which only use techniques like reweighing, resampling, oversampling and undersampling will be called *basic ensembles*. Figure 6.2 displays the collection of basic ensemble methods examined in this study. Each of the following methods will be combined with each of the diversity-enhancing techniques described in Section 6.2.4, these methods will be called *enhanced ensembles*, for example we refer to RUSBoost (RUSBo) as *basic ensemble* and we refer to RUSBoost combined with Disturbing Neighbors (DN+RUSBo) as a *enhanced ensemble*. Finally ensembles using only the diversity-enhancing techniques (no resampling or other preprocessing) were also tested.

These *basic* methods can be grouped into three different categories: x) baseline methods which are not modified in any way to cope with imbalanced data, y) ensemble methods especially designed to cope with imbalanced data, and z) baseline methods combined with preprocessing techniques to improve its performance in unbalanced datasets. In these methods, the abbreviation used contains the name of the family of ensembles (e.g. E:

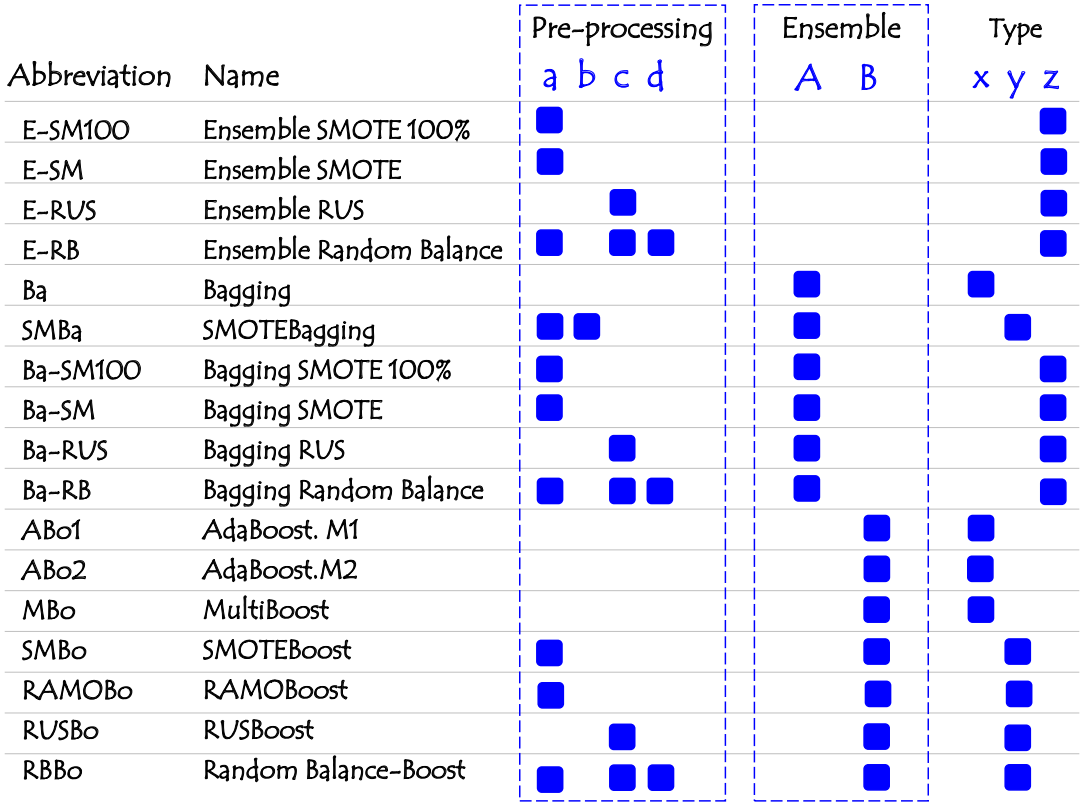


Figure 6.2: Ensemble methods compared in this study.

Simple Ensemble, Ba: Bagging and Bo: Boosting)[3], and the preprocessing techniques (SM/SM100: SMOTE, RUS: Random Undersampling, RB: Random Balance). These methods are listed below:

1. E-SM100: SMOTE is used, in each iteration, to double the size of the minority class. The number of neighbors is 5 for all methods that use SMOTE.

2. E-SM: SMOTE is used, in each iteration, to get a minority class of the same size as the majority class.

3. E-RUS: Random Undersampling is used, in each iteration, to reduce the majority class so that equal in size to that of the minority class.

4. E-RB: Radom Balance is used in each iteration.

[3]In a method belonging to *Simple Ensemble family* the dataset used to build each member of the ensemble is built using only the preprocessing technique (The source of diversity in these ensembles is the preprocessing method, since SMOTE, RUS and Random Balance are randomized methods that produce a different dataset at each iteration) , while a method that is a member of the *Bagging family* the dataset used to build each base classifier comes from a resampling and then the preprocessing technique.

5. Ba: Bagging.
6. SMBa: SMOTEBagging.
7. Ba-SM100: Bagging in which in each iteration SMOTE is used to double the size of the minority class.
8. Ba-SM: Bagging in which in each iteration SMOTE is used to get a minority class of the same size as the majority class.
9. Ba-RUS: Bagging in which in each iteration Random Undersampling is used to reduce the size of the majority class to the size of the minority class. This method is called UnderBagging in [23, 3].
10. Ba-RB: Bagging with Random Balance in each iteration.
11. ABo1: AdaBoost.M1.
12. ABo2: AdaBoost.M2.
13. MBo: MultiBoost.
14. SMBo: SMOTEBoost with the following settings: SMOTE 100% in each iteration.
15. RAMOBo: RAMOBoost with the following settings: number of synthetic instances equal to the size of the minority class, number of neighbors used to create synthetic instances equal to 5, number of neighbors used to compute probabilities equal to 10.
16. RUSBo: RUSBoost.
17. RBBo: Random Balance Boost.

The size of the ensembles was set to 100. Default Weka parameters were used in all the ensemble methods provided by the library, except the number of sub committes in Multiboost that was set to 10. In a further section (Section 6.3.4) the effect of the size of the ensemble is studied.

The classifier used as a base classifier in all ensembles was J48, the Java implementation of Quinlan's C4.5 [52]. As recommended for imbalanced data [15], it was used with Laplace smoothing at the leaves, but without pruning and collapsing. When C4.5 is used with this configuration, it is called C4.4 [50].

Table 6.1: Characteristics of the 20 data sets from the HDDT collection. Column #E shows the number of examples in the dataset, column #A the number of attributes, both numeric and nominal in the format (numeric/nominal), and column IR the imbalance ratio (the number of instances of the majority class per instance of the minority class).

Data set	#E	#A	IR	Data set	#E	#A	IR
boundary	3505	(0/175)	27.50	ism	11180	(6/0)	42.00
breast-y	286	(0/9)	2.36	letter	20000	(16/0)	24.35
cam	18916	(0/132)	19.08	oil	937	(49/0)	21.85
compustat	13657	(20/0)	25.26	optdigits	5620	(64/0)	9.14
covtype	38500	(10/0)	13.02	page	5473	(10/0)	8.77
credit-g	1000	(7/13)	2.33	pendigits	10992	(16/0)	8.63
estate	5322	(12/0)	7.37	phoneme	5404	(5/0)	2.41
german-numer	1000	(24/0)	2.33	PhosS	11411	(480/0)	17.62
heart-v	200	(5/8)	2.92	satimage	6430	(36/0)	9.29
hypo	3163	(7/18)	19.95	segment	2310	(19/0)	6.00

6.3.2 Datasets and Tools

Two collections of data sets were used. The HDDT collection[4] contains 20 binary imbalanced data sets used in [15]. Table 6.1 shows the characteristics of this dataset collection.

The KEEL collection[5] contains 66 binary imbalanced data sets (we have used 64 of them in the experiments, because two of them were almost identical to two in the other repository) from the repository of KEEL [1]. Datasets in the KEEL collection are not completely independent of each other. Several of them are variants of the same original dataset. Starting with a single multiclass dataset, several binary datasets are created by grouping their classes in different ways. Table 6.2 shows the characteristics of this dataset collection.

Many data sets in these two collections are available or are modifications of data sets in the UCI Repository [2].

Weka 3.7.10 [30] was used for the experiments. The results were obtained with a 5×2-fold cross validation [18].

Three criteria were used for evaluating the ensemble performance: The F-measure [60], the Geometric Mean [39] and the Area Under the ROC Curve (AUC) [21].

Given a test dataset, containing P examples of the positive class and N examples of the negative class. The confusion matrix is shown in Table 6.3.

The True Positive Rate (TPR) also named Sensitivity or Recall in some

[4] Available at `http://www.nd.edu/~dial/hddt/`.

[5] Available at `http://sci2s.ugr.es/keel/imbalanced.php`.

Table 6.2: Characteristics of the data sets from the KEEL collection. Column #E shows the number of examples in the dataset, column #A the number of attributes, both numeric and nominal in the format (numeric/nominal), and column IR the imbalance ratio (the number of instances of the majority class for each instance of the minority class).

Data set	#E	#A	IR	Data set	#E	#A	IR
abalone19	4174	(7/1)	129.44	glass4	214	(9/0)	15.46
abalone9-18	731	(7/1)	16.40	glass5	214	(9/0)	22.78
cleveland-0_vs_4	177	(13/0)	12.62	glass6	214	(9/0)	6.38
ecoli-0-1-3-7_vs_2-6	281	(7/0)	39.14	haberman	306	(3/0)	2.78
ecoli-0-1-4-6_vs_5	280	(6/0)	13.00	iris0	150	(4/0)	2.00
ecoli-0-1-4-7_vs_2-3-5-6	336	(7/0)	10.59	led7digit-0-2-4-5-6-7-8-9_vs_1	443	(7/0)	10.97
ecoli-0-1-4-7_vs_5-6	332	(6/0)	12.28	new-thyroid1	215	(5/0)	5.14
ecoli-0-1_vs_2-3-5	244	(7/0)	9.17	new-thyroid2	215	(5/0)	5.14
ecoli-0-1_vs_5	240	(6/0)	11.00	page-blocks-1-3_vs_4	472	(10/0)	15.86
ecoli-0-2-3-4_vs_5	202	(7/0)	9.10	pima	768	(8/0)	1.87
ecoli-0-2-6-7_vs_3-5	224	(7/0)	9.18	shuttle-c0-vs-c4	1829	(9/0)	13.87
ecoli-0-3-4-6_vs_5	205	(7/0)	9.25	shuttle-c2-vs-c4	129	(9/0)	20.50
ecoli-0-3-4-7_vs_5-6	257	(7/0)	9.28	vehicle0	846	(18/0)	3.25
ecoli-0-3-4_vs_5	200	(7/0)	9.00	vehicle1	846	(18/0)	2.90
ecoli-0-4-6_vs_5	203	(6/0)	9.15	vehicle2	846	(18/0)	2.88
ecoli-0-6-7_vs_3-5	222	(7/0)	9.09	vehicle3	846	(18/0)	2.99
ecoli-0-6-7_vs_5	220	(6/0)	10.00	vowel0	988	(13/0)	9.98
ecoli-0_vs_1	220	(7/0)	1.86	wisconsin	683	(9/0)	1.86
ecoli1	336	(7/0)	3.36	yeast-0-2-5-6_vs_3-7-8-9	1004	(8/0)	9.14
ecoli2	336	(7/0)	5.46	yeast-0-2-5-7-9_vs_3-6-8	1004	(8/0)	9.14
ecoli3	336	(7/0)	8.60	yeast-0-3-5-9_vs_7-8	506	(8/0)	9.12
ecoli4	336	(7/0)	15.80	yeast-0-5-6-7-9_vs_4	528	(8/0)	9.35
glass-0-1-2-3_vs_4-5-6	214	(9/0)	3.20	yeast-1-2-8-9_vs_7	947	(8/0)	30.57
glass-0-1-4-6_vs_2	205	(9/0)	11.06	yeast-1-4-5-8_vs_7	693	(8/0)	22.10
glass-0-1-5_vs_2	172	(9/0)	9.12	yeast-1_vs_7	459	(7/0)	14.30
glass-0-1-6_vs_2	192	(9/0)	10.29	yeast-2_vs_4	514	(8/0)	9.08
glass-0-1-6_vs_5	184	(9/0)	19.44	yeast-2_vs_8	482	(8/0)	23.10
glass-0-4_vs_5	92	(9/0)	9.22	yeast1	1484	(8/0)	2.46
glass-0-6_vs_5	108	(9/0)	11.00	yeast3	1484	(8/0)	8.10
glass0	214	(9/0)	2.06	yeast4	1484	(8/0)	28.10
glass1	214	(9/0)	1.82	yeast5	1484	(8/0)	32.73
glass2	214	(9/0)	11.59	yeast6	1484	(8/0)	41.40

Table 6.3: Confusion matrix in binary problems

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

fields, is defined as TP/P, and False Positive Rate (FPR) is defined as FP/N. The precision is defined as $TP/(TP+FP)$.

Using these previous measures it is posible to define the F-Measure as

$$FMeasure = 2 \times \frac{precision \times recall}{precision + recall}$$

The Geometric Mean is defined as

$$GMean = \sqrt{TP/P \times TN/N}$$

In this work the ROC curve is obtained from the probabilities assigned to the instances by the classifier, each probability threshold gives a TPR and FPR that defines a point in the curve. The AUC is computed from Wilcoxon Rank Sum test statistic.

6.3.3 Comparison between basic and enhanced ensembles

This section will show a summary of the results for each of the basic ensembles and their improved versions and a comparison between these methods is performed. The summary is done by averaging, for each method the score across the 84 data sets.

Averaging the results is not the best way to compare multiple methods, since a big difference in a dataset can mask a general trend in the rest. To compare multiple methods, we used average ranks [16]. Each method was assigned a rank for each data set based on its performance, separately for each criterion. The best method obtained rank 1, the second best obtained rank 2, etc. When there was a tie, the ranks were shared out. For example, if the top three methods for a given data set tied, each one of them would receive rank $(1+2+3)/3 = 2$ for this data set. The ranks are averaged across all datasets. The methods were arranged by their ranks, where the best methods (the ones with the lowest average ranks) were at the top of the list. Iman and Davenport's [37] test was applied to check whether there are any significant differences between the ranks of the compared methods. Subsequently, Hochberg's test [36] was carried out next to identify all methods which were not significantly different from the winner.

This section of the experiments consisted of two parts:

1. **Basic ensembles versus enhanced variants and enhanced variants among themselves** Each basic ensemble was compared with its 4 enhanced variants, where each variant was obtained by applying the respective diversity-enhancing method. The average ranks were calculated using five methods (four when comparing diversity techniques among themselves).

   Moreover, due to most of the enhanced versions make use of some type of preprocessing, average ranks were calculated using all of the methods that use the same diversity technique. The 17 ensemble variants and a basic ensemble that only use the diversity technique.

2. **The overall winner.** In one final comparison, we select the method with best average rank for each row in figures 6.5.a, 6.6.a and 6.7.a, and calculated the ranks anew using these best methods.

6.3.3.1 Basic ensembles versus enhanced variants and enhanced variants among themselves.

The way in which the combination of the ensemble methods and the diversity technique is performed is always the same. In each iteration a modified training set resulting from the application of sampling, oversampling, undersampling, reweighting or the combination of two of these techniques (depending on the ensemble type) is created. This modified training set is modified again using a diversity technique for Random Linear Oracles, Disturbing neighbors and Rotation Forest. The method Random Feature Weights does not modify the dataset, it uses a modified decision tree.

Some of the diversity techniques have parameters. Rotation Forest has been used with the default parameters found in Weka package. In the case of Disturbing Neighbors, as the authors of the method proposed, the dimensions used to compute the nearest neighbor are ramdomly selected, choosing 50% of the attributes. The number of “Disturbing Neighbors" selected in each iteration is $N = 10$. The Random Feature Weights implementation is based on J48 and is used with the C4.4 configuration. The exponent value, which control the level of randomness, was set to 1 (the value used by the authors when combined with another methods). Random Linear Oracles do not have any parameter.

Figure 6.3: Comparison of the basic ensemble methods with their combinations with Disturbing Neighbors and Rotation Forest. The numbers in the corners indicate the percentage of points above or below the diagonal.

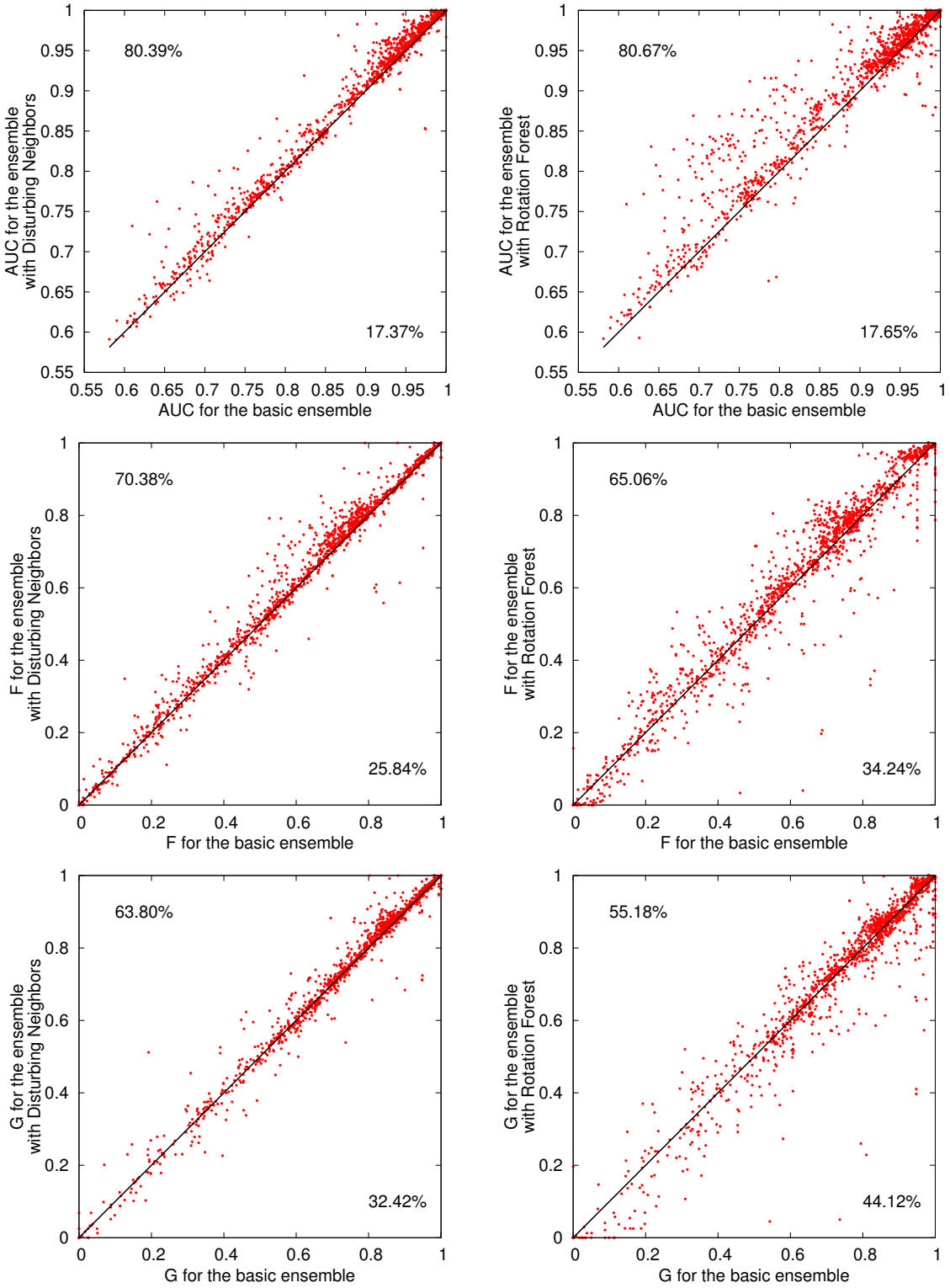
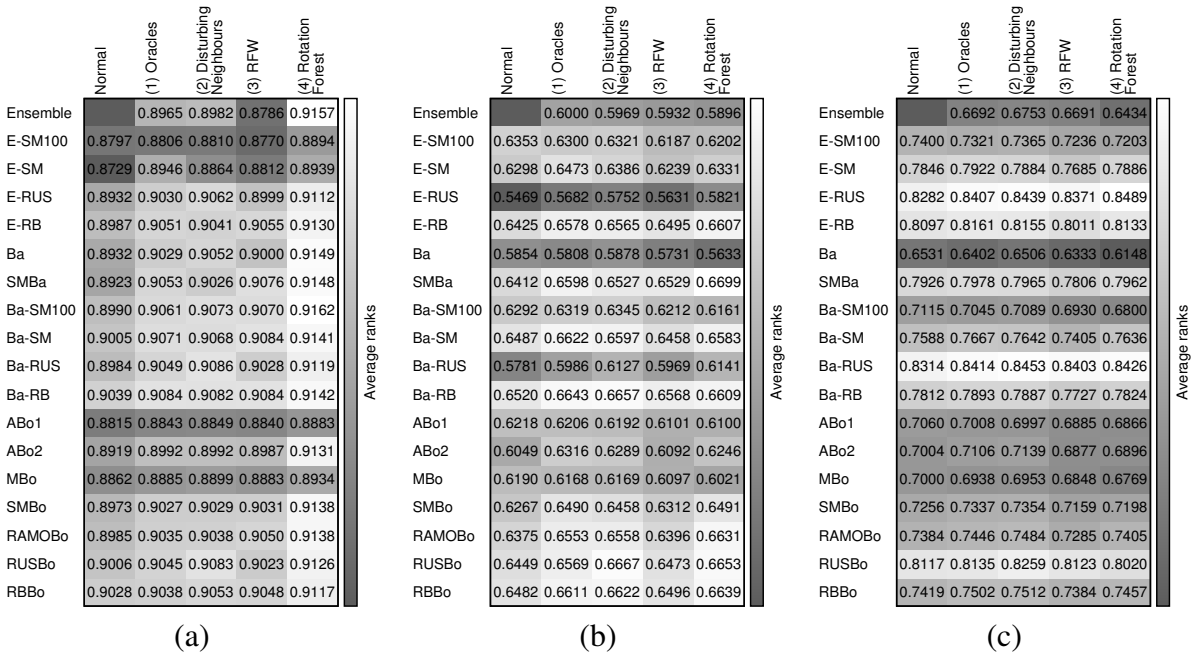


Figure 6.4: Average scores in terms of the AUC (a) F-Measure (b) and G-Mean (c). The intensity of the cell reflects the score. Cells with higher values (better) are filled in light gray and those with lower values (worse) are filled in dark gray.



(a)

	Normal	(1) Oracles	(2) Disturbing Neighbours	(3) RFW	(4) Rotation Forest
Ensemble		0.8965	0.8982	0.8786	0.9157
E-SM100	0.8797	0.8806	0.8810	0.8770	0.8894
E-SM	0.8729	0.8946	0.8864	0.8812	0.8939
E-RUS	0.8932	0.9030	0.9062	0.8999	0.9112
E-RB	0.8987	0.9051	0.9041	0.9055	0.9130
Ba	0.8932	0.9029	0.9052	0.9000	0.9149
SMBa	0.8923	0.9053	0.9026	0.9076	0.9148
Ba-SM100	0.8990	0.9061	0.9073	0.9070	0.9162
Ba-SM	0.9005	0.9071	0.9068	0.9084	0.9141
Ba-RUS	0.8984	0.9049	0.9086	0.9028	0.9119
Ba-RB	0.9039	0.9084	0.9082	0.9084	0.9142
ABo1	0.8815	0.8843	0.8849	0.8840	0.8883
ABo2	0.8919	0.8992	0.8992	0.8987	0.9131
MBo	0.8862	0.8885	0.8899	0.8883	0.8934
SMBo	0.8973	0.9027	0.9029	0.9031	0.9138
RAMOBo	0.8985	0.9035	0.9038	0.9050	0.9138
RUSBo	0.9006	0.9045	0.9083	0.9023	0.9126
RBBo	0.9028	0.9038	0.9053	0.9048	0.9117

Average ranks

(b)

	Normal	(1) Oracles	(2) Disturbing Neighbours	(3) RFW	(4) Rotation Forest
Ensemble		0.6000	0.5969	0.5932	0.5896
E-SM100	0.6353	0.6300	0.6321	0.6187	0.6202
E-SM	0.6298	0.6473	0.6386	0.6239	0.6331
E-RUS	0.5469	0.5682	0.5752	0.5631	0.5821
E-RB	0.6425	0.6578	0.6565	0.6495	0.6607
Ba	0.5854	0.5808	0.5878	0.5731	0.5633
SMBa	0.6412	0.6598	0.6527	0.6529	0.6699
Ba-SM100	0.6292	0.6319	0.6345	0.6212	0.6161
Ba-SM	0.6487	0.6622	0.6597	0.6458	0.6583
Ba-RUS	0.5781	0.5986	0.6127	0.5969	0.6141
Ba-RB	0.6520	0.6643	0.6657	0.6568	0.6609
ABo1	0.6218	0.6206	0.6192	0.6101	0.6100
ABo2	0.6049	0.6316	0.6289	0.6092	0.6246
MBo	0.6190	0.6168	0.6169	0.6097	0.6021
SMBo	0.6267	0.6490	0.6458	0.6312	0.6491
RAMOBo	0.6375	0.6553	0.6558	0.6396	0.6631
RUSBo	0.6449	0.6569	0.6667	0.6473	0.6653
RBBo	0.6482	0.6611	0.6622	0.6496	0.6639

Average ranks

(c)

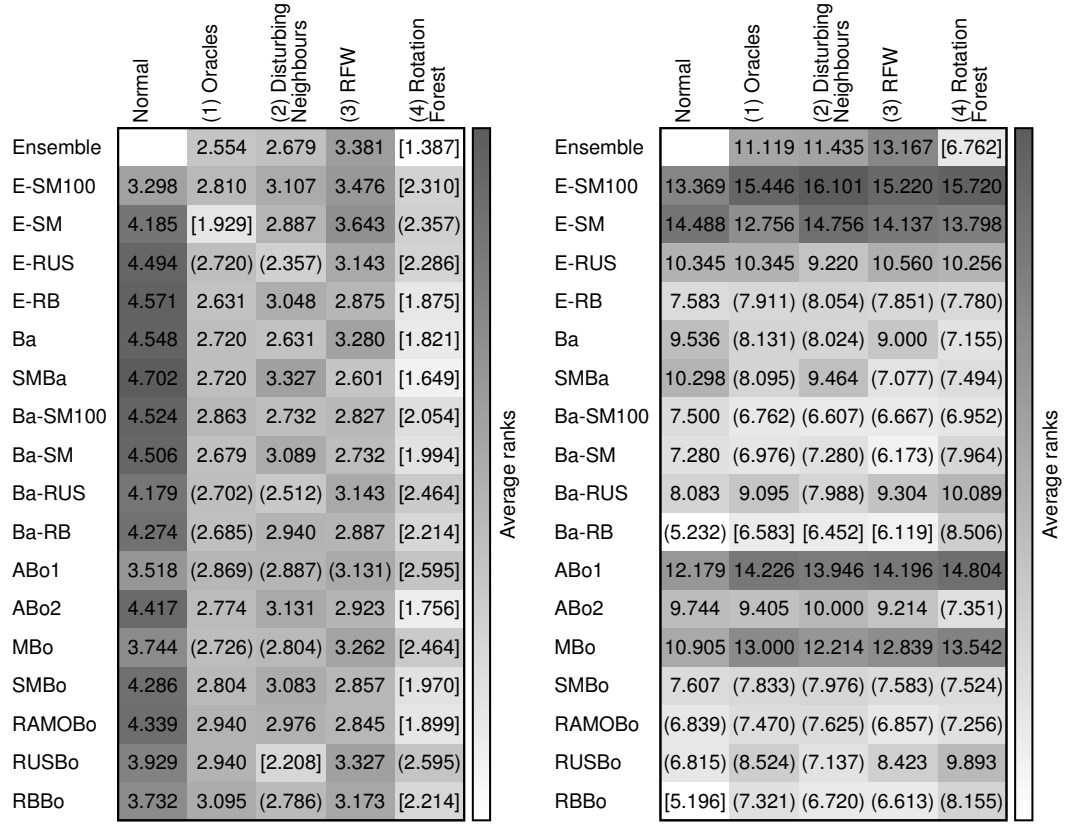
	Normal	(1) Oracles	(2) Disturbing Neighbours	(3) RFW	(4) Rotation Forest
Ensemble		0.6692	0.6753	0.6691	0.6434
E-SM100	0.7400	0.7321	0.7365	0.7236	0.7203
E-SM	0.7846	0.7922	0.7884	0.7685	0.7886
E-RUS	0.8282	0.8407	0.8439	0.8371	0.8489
E-RB	0.8097	0.8161	0.8155	0.8011	0.8133
Ba	0.6531	0.6402	0.6506	0.6333	0.6148
SMBa	0.7926	0.7978	0.7965	0.7806	0.7962
Ba-SM100	0.7115	0.7045	0.7089	0.6930	0.6800
Ba-SM	0.7588	0.7667	0.7642	0.7405	0.7636
Ba-RUS	0.8314	0.8414	0.8453	0.8403	0.8426
Ba-RB	0.7812	0.7893	0.7887	0.7727	0.7824
ABo1	0.7060	0.7008	0.6997	0.6885	0.6866
ABo2	0.7004	0.7106	0.7139	0.6877	0.6896
MBo	0.7000	0.6938	0.6953	0.6848	0.6769
SMBo	0.7256	0.7337	0.7354	0.7159	0.7198
RAMOBo	0.7384	0.7446	0.7484	0.7285	0.7405
RUSBo	0.8117	0.8135	0.8259	0.8123	0.8020
RBBo	0.7419	0.7502	0.7512	0.7384	0.7457

Average ranks

Figure 6.3 shows a comparison of the basic ensemble methods with the same methods augmented with Disturbing Neighbors (graphs at the left) or Rotation Forest (at the right). There are three rows of graphs, one for each performance measure. Each point in the graph corresponds to a pair of basic ensemble method and dataset. The *x* coordinate is the value of the performance measure for the basic ensemble while the *y* coordinate is the value for that method combined with the diversity technique. Points above the diagonal are the cases when the combination is better than the basic ensemble. The graphs also show the percentage of points above and below the diagonal. In the six graphs the majority of points are above the diagonal. Nevertheless, the performance measures show different behaviours, specially for Rotation Forest: for AUC the advantage of Rotation Forest is clear. For F-measure and G-mean, although the majority of the points are above the diagonal, there are some points were Rotation Forest is clearly worse. These points are relatively few, if we consider that each graph has 1428 points (17 basic ensemble methods multiplied by 84 datasets).

Figure 6.4 shows the average scores across the 84 datasets for the AUC, F-Measure and G-Mean. The way to interpret the tables is as follows: Each line, from the second onwards, contains the values of a *basic ensemble* and

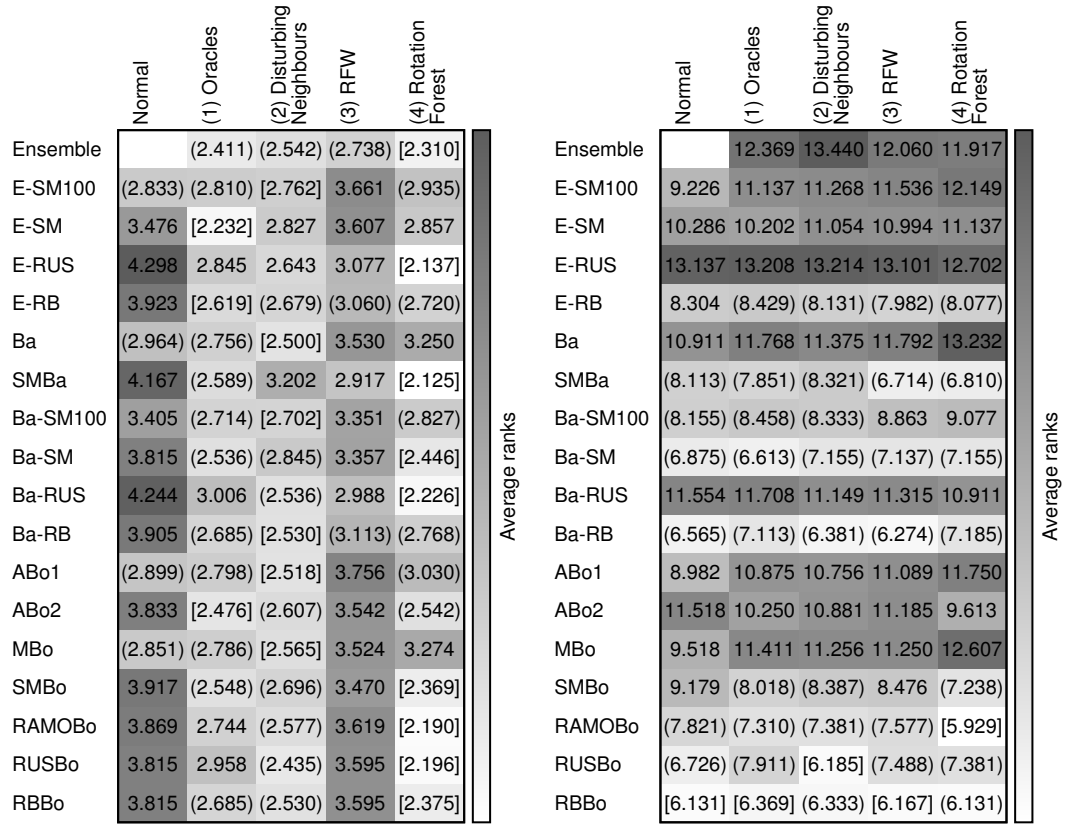
Figure 6.5: Average Ranks per row (a) and column (b) (AUC). The best method in each row (a) or in each column (b) has it rank in brackets. Those methods that are equivalent to the best one at significance 0.05 are delimited in parentheses.



the enhanced versions of this ensemble. The column indicates the diversity-enhancing strategy used. So, for example the intersection of column RFW and row Ba contains the scores of Bagging combined with Random Feature Weights. Ensembles using only the diversity-enhancing techniques (no resampling or other preprocessing) are shown in the first row. The full table of results can be consulted in the supplementary material[6]. For the AUC it is clear that enhanced methods, specially those that use Rotation Forest obtain better average results. One trend that also happens when considering the F-Measure. For the G-Mean, the best average results are obtained by methods that use Random Undersampling as a preprocessing technique (Ba-RUS and E-RUS). For this measure, although the basic methods are improved with the diversity techniques, the selection of the basic ensemble method (row) has more influence than the diversity-enhancing technique (column).

[6] `https://github.com/joseFranciscoDiez/research/wiki/Supplementary-Material`

Figure 6.6: Average Ranks per row (a) and column (b) (F-Measure). The best method in each row (a) or in each column (b) has it rank in brackets. Those methods that are equivalent to the best one at significance 0.05 are delimited in parentheses.

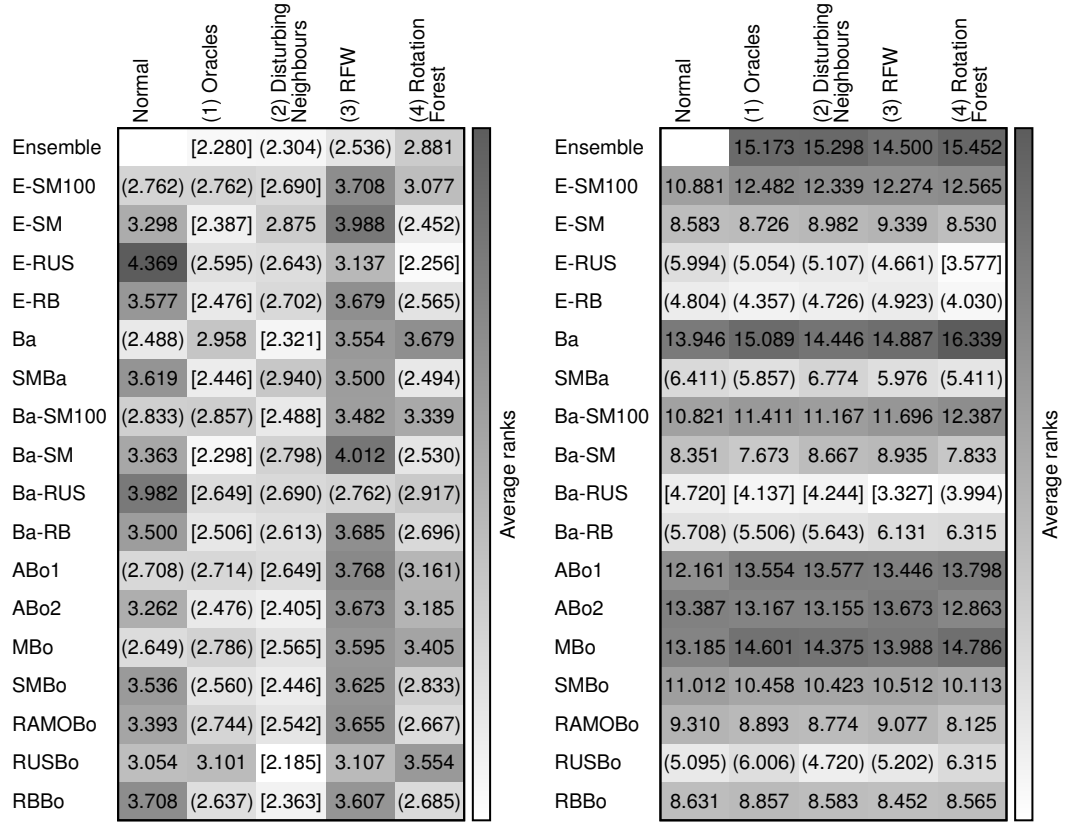


Figures 6.5, 6.6 and 6.7 show the average ranks calculated from the area under the curve, the F-Measure and the G-Mean, respectively.

The structure of the three figures is the same, on the left, the ranks are calculated by rows. The possible ranks were from 1 to 5 (the basic ensemble method and its four combinations with diversity techniques) or 1 to 4 when the diversity techniques are not combined with any other ensemble or preprocessing technique. Again, rank 1 would correspond to the best alternative, and rank 5, to the worst. The intensity of the cell reflects the average rank. Cells with lower ranks (better) are filled in light gray and those with higher ranks (worse) are filled in dark gray. It is easy to see that, in general, the methods that use a diversity enhancing techniques perform better than those without.

The best method in each row has it ranking in brackets. Those methods that are equivalent to the best one at significance 0.05 (Hochberg's test [36]) are delimited in parentheses.

Figure 6.7: Average Ranks per row (a) and column (b) (G-Mean). The best method in each row (a) or in each column (b) has it rank in brackets. Those methods that are equivalent to the best one at significance 0.05 are delimited in parentheses.



	Normal	(1) Oracles	(2) Disturbing Neighbours	(3) RFW	(4) Rotation Forest	Average ranks
Ensemble		[2.280]	(2.304)	(2.536)	2.881	
E-SM100	(2.762)	(2.762)	[2.690]	3.708	3.077	
E-SM	3.298	[2.387]	2.875	3.988	(2.452)	
E-RUS	4.369	(2.595)	(2.643)	3.137	[2.256]	
E-RB	3.577	[2.476]	(2.702)	3.679	(2.565)	
Ba	(2.488)	2.958	[2.321]	3.554	3.679	
SMBa	3.619	[2.446]	(2.940)	3.500	(2.494)	
Ba-SM100	(2.833)	(2.857)	[2.488]	3.482	3.339	
Ba-SM	3.363	[2.298]	(2.798)	4.012	(2.530)	
Ba-RUS	3.982	[2.649]	(2.690)	(2.762)	(2.917)	
Ba-RB	3.500	[2.506]	(2.613)	3.685	(2.696)	
ABo1	(2.708)	(2.714)	[2.649]	3.768	(3.161)	
ABo2	3.262	(2.476)	[2.405]	3.673	3.185	
MBo	(2.649)	(2.786)	[2.565]	3.595	3.405	
SMBo	3.536	(2.560)	[2.446]	3.625	(2.833)	
RAMOBo	3.393	(2.744)	[2.542]	3.655	(2.667)	
RUSBo	3.054	3.101	[2.185]	3.107	3.554	
RBBo	3.708	(2.637)	[2.363]	3.607	(2.685)	

(a)

	Normal	(1) Oracles	(2) Disturbing Neighbours	(3) RFW	(4) Rotation Forest	Average ranks
Ensemble		15.173	15.298	14.500	15.452	
E-SM100	10.881	12.482	12.339	12.274	12.565	
E-SM	8.583	8.726	8.982	9.339	8.530	
E-RUS	(5.994)	(5.054)	(5.107)	(4.661)	[3.577]	
E-RB	(4.804)	(4.357)	(4.726)	(4.923)	(4.030)	
Ba	13.946	15.089	14.446	14.887	16.339	
SMBa	(6.411)	(5.857)	6.774	5.976	(5.411)	
Ba-SM100	10.821	11.411	11.167	11.696	12.387	
Ba-SM	8.351	7.673	8.667	8.935	7.833	
Ba-RUS	[4.720]	[4.137]	[4.244]	[3.327]	(3.994)	
Ba-RB	(5.708)	(5.506)	(5.643)	6.131	6.315	
ABo1	12.161	13.554	13.577	13.446	13.798	
ABo2	13.387	13.167	13.155	13.673	12.863	
MBo	13.185	14.601	14.375	13.988	14.786	
SMBo	11.012	10.458	10.423	10.512	10.113	
RAMOBo	9.310	8.893	8.774	9.077	8.125	
RUSBo	(5.095)	(6.006)	(4.720)	(5.202)	6.315	
RBBo	8.631	8.857	8.583	8.452	8.565	

(b)

Note that there are no entries in parentheses in the first column in Figure 6.5.a. This means that, for the AUC, all *basic ensembles* perform significantly worse than the best enhanced variant. For the F-Measure and G-Mean criteria, improvement does not happen for all the methods but it is clearly noticeable for the methods of interest, the best methods according to this measure.

The way to know which are the best basic ensembles is through the average ranks calculated column-wise instead of row-wise, on the first column in the right table (figures 6.5.b, 6.6.b and 6.7.b)). Again the best method in each column has it rank in brackets.

The best basic methods according the F-Measure are RBBo, Ba-RB, RUSBo, and Ba-SM and these methods are clearly improved when combined with Rotation Forest strategy, as seen in Figure 6.6.a.

For average ranks computed using G-Mean, the top methods are Ba-RUS, E-RB and RUSBo and these methods are improved when combined with

Table 6.4: Average ranks for best methods. a) According to the AUC b) According to the F-Measure c) According to the G-Mean. The combination of diversity techniques with other ensemble methods will be named using the prefix O in the case of Random Linear Oracles, DN for Disturbing neighbors and RF for Rotation Forest.

(a) AUC

Method	Rank
Rotation Forest	6.774
RF+Ba-SM100	6.929
RF+Ba	7.155
RF+RAMOBo	7.256
RF+ABo2	7.506
RF+SMBa	7.542
RF+SMBo	7.679
RF+E-RB	7.780
RF+Ba-SM	7.940
RF+RBBo	8.304
RF+Ba-RB	8.458
DN+RUSBo	9.536
RF+Ba-RUS	10.125
RF+E-RUS	10.280
RF+MBo	13.482
O+E-SM	13.780
RF+ABo1	14.780
RF+E-SM100	15.696

(b) F-measure

Method	Rank
RF+RAMOBo	6.250
RF+RBBo	6.375
RF+SMBa	6.946
RF+Ba-SM	7.375
DN+Ba-RB	7.512
RF+RUSBo	7.577
RF+SMBo	7.744
DN+Ba-SM100	9.012
O+E-RB	9.107
O+ABo2	10.679
RF+Ba-RUS	10.762
O+E-SM	10.857
DN+ABo1	11.214
DN+E-SM100	11.518
DN+MBo	11.738
Rotation Forest	11.857
DN+Ba	11.935
RF+E-RUS	12.542

(c) G-Mean

Method	Rank
O+Ba-RUS	4.607
DN+RUSBo	4.750
RF+E-RUS	4.768
O+E-RB	4.798
O+Ba-RB	5.982
O+SMBa	6.417
O+Ba-SM	8.214
DN+RBBo	8.768
DN+RAMOBo	8.833
O+E-SM	8.923
DN+SMBo	10.560
DN+Ba-SM100	11.232
DN+E-SM100	12.244
DN+ABo2	13.107
DN+ABo1	13.530
DN+MBo	14.399
DN+Ba	14.464
Random Oracles	15.405

Random Oracles and Disturbing Neighbors.

6.3.3.2 The overall winner.

A new average rank has been calculated from the best combinations in each row in figures 6.5.a, 6.6.a and 6.7.a.

The results are shown in Table 6.4. The classic Rotation Forest on its own, alongside several methods combined with Rotation Forest monopolize the best positions in the AUC table.

For the F-Measure, the basic Rotation Forest does not achieve a good position, although various combinations of Rotation Forest with other ensembles still occupy the top positions. The best combinations use oversampling strategies trying to obtain more balance.

This difference may be due to the fact that these two measures consider different aspects: the AUC only takes into account the probabilities given by the classifier for each instance, while the F-Measure takes into account whether the instances are correctly labeled or not. One classifier could achieve the maximum possible AUC and simultaneously the minimum F-Measure, that happens when none of the instances of the positive class

are correctly labeled, but the probabilities assigned to the instances of the positive class are higher than the probabilities of instances belonging to the negative class. This could be the reason why balancing strategies have less impact when considering the AUC as an evaluation measure.

For the G-Mean, there is no clear trend. Although it seems that this time ensembles enhanced with Random Linear Oracles and Disturbing neighbors outperform those enhanced with Rotation Forest.

6.3.4 Ensemble size.

In the results presented until now, the ensemble size was 100. Nevertheless, different sizes can be more adequate for different ensemble methods. In [23] two ensemble sizes were considered, 10 and 40, and some ensemble methods have better results with a smaller size.

In order to study the behaviour of this size, the five non-enhanced methods with best ranks in Figures 6.5.b, 6.6.b and 6.7.b were selected for each performance measure. For the selected methods, experiments were carried out using the values 10, 20, 30, ..., 100 as for the ensemble size. For each ensemble method, the ten considered sizes were compared using average ranks. Figures 6.8.a–c show these average ranks for the three measures. Each graph has five lines, one for each considered ensemble method. The values in the lines are in the range [1,10], as they are average ranks from 10 configurations.

In general, bigger ensemble sizes give better average ranks. Then, using 100 for the ensemble size instead of a smaller value is justified. There is one clear exception, the behaviour of RUSBo for the G-mean, for this method the best sizes are 20 and 30. Interestingly, this method does not have this behaviour for AUC and F-measure.

Given the unsual behaviour of RUSBo with G-mean, its performance with the diversity-enhancing methods was analysed. Figures 6.8.d shows, for the considered ensemble sizes, the average ranks of the five RUSBo ensemble configurations. As there are five configurations, the average ranks are in [1,5]. The behaviour is rather uniform: for instance, for all ensemble sizes, DN+RUSBo has better rank than RUSBo and RUSBo is better than RF+RUSBo. Hence, the possibility of improving an ensemble method with a diversity-enhancing method is not restricted to a particular ensemble size.

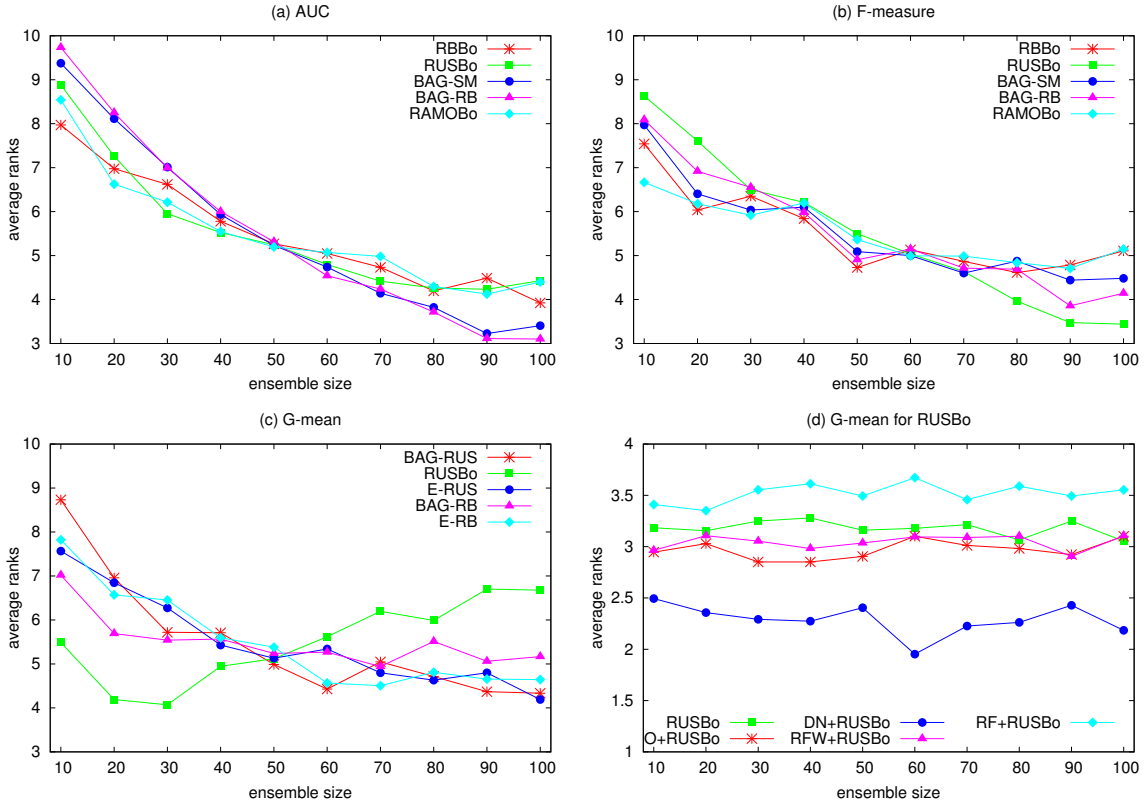


Figure 6.8: Average ranks for different ensemble sizes.

6.3.5 Trying to predict when to apply diversity techniques

In the majority of the cases the diversity-enhancing technique improves the basic ensemble method, but as this improvement is not guaranteed, in this section a study is performed that attempts to relate the meta-feature of dataset with the convenience of whether to apply or not the techniques to increase diversity.

The meta-features were obtained using *the data complexity library*[7] (DCoL). This software computes the list of fourteen features shown in Table 6.6, which are designed to characterize the complexity of data sets for supervised learning and that were first defined in [35, 33].

In order to learn the relationship between meta-features and the best combination of ensemble and diversity technique, three datasets, one per performance measure, were built with the following attributes:

1. The fourteen features that characterize the dataset.

2. The name of the basic ensemble.

[7]This software is available at `http://dcol.sourceforge.net`.

Table 6.5: Meta-features

ID	Measure	ID	Measure
$F1$	Maximum Fisher's discriminant ratio	$L3$	Nonlinearity of a linear classifier
$F1v$	Directional-vector maximum Fisher's discriminant ratio	$N1$	Fraction of points on the class boundary
$F2$	Overlap of the per-class bounding boxes	$N2$	Ratio of average intra/inter class nearest neighbor distance
$F3$	Maximum (individual) feature efficiency	$N3$	Leave-one-out error rate of the one-nearest neighbor classifier
$F4$	Collective feature efficiency	$N4$	Nonlinearity of the one-nearest neighbor classifier
$L1$	Minimized sum of the error distance of a linear classifier	$T1$	Fraction of maximum covering spheres
$L2$	Training error of a linear classifier	$T2$	Average number of points per dimension

3. The name of the technique used to increase the diversity.

4. And the class, which encodes if, for the given dataset (the one from which the first fourteen features were obtained), the combination of the ensemble and the diversity technique gives better results than the ensemble alone. The value of this attribute depends on the performance measure, and hence is usually different in each of the three datasets. If it is '*yes*', the combination of diversity technique and ensemble give better results (measured in terms of the AUC, the G-mean or the F-measure) than using the ensemble alone, on the contrary, its value is '*no*'.

First we want to know if it is possible to establish any relationship between the meta-features and the fact that the diversity technique improves the ensemble. To do this, we compared the performance of a weak classifier, that just predicts the mode of the class, with others much stronger, as J48 and Rotation Forest. Results are listed in Table 6.6.

The statistical improvement obtained by using a classifier as J48 or Rotation Forest rather than simply predicting the mode is an indication that there is a relationship between the meta-features of a dataset and the fact that the combination of a diversity technique with an ensemble can give better results than the ensemble alone.

Next step is to try to learn this relationship and extract some general rules that could help us to find if a diversity technique will improve a basic ensemble method for a given dataset. We use HotSpot [30] to learn this set

Table 6.6: Success percentage of the three classifiers evaluated on three meta-learning datasets (the symbol ○ indicates the cases where there strong classifier is statistically better than the mode)

Dataset	Mode	J48	Rotation Forest
Metadata for AUC	78.78	78.41	84.02 ○
Metadata for F-measure	65.00	72.97 ○	78.92 ○
Metadata for G-mean	56.29	68.26 ○	76.56 ○

of rules in a tree structure, they identify for which meta-features there is a high probability that a certain diversity technique improves the ensemble. As well, this rule could help us to discard diversity techniques since they do not give any improvement to the ensemble used alone.

The following are the rules found for the dataset where the improvement is measure in terms of the AUC:

```
Class=yes (78.78% [4500/5712])
|__F2 > 0.0008 (84.09% [1601/1904])
|   |__N1 <= 0.448 (86.65% [1532/1768])
|   |__N3 <= 0.304 (86.65% [1532/1768])
|__F1v <= 1.235 (82.46% [1514/1836])
    |__N1 <= 0.448 (84.79% [1499/1768])
    |__N3 <= 0.304 (84.79% [1499/1768])
```

The way of interpreting this rule is the following, there are 4500 instances out of 5712[8] for which the diversity technique improves the ensemble (that is in 78.78% of the instances). But the percentage is even bigger (86.65%) if we consider only those instances corresponding with datasets for which $F2$ is bigger than 0.0008 and $N1$ lower than 0.448. So the argument in favor of using diversity techniques together with ensembles is stronger for datasets with values of $F2$ and $N1$ in these ranges. With the dataset created using the F-Measure the following rules were obtained:

[8] 5712 instances: = 17 ensemble methods × 4 diversity techniques × 84 datasets

```
Class=yes (65% [3713/5712])
|__F1v <= 0.657 (78.98% [1289/1632])
|   |__F2 <= 0.271 (80.75% [1263/1564])
|   |   |__L1 > 0.143 (82.15% [1229/1496])
|   |   |__T2 <= 1250 (82.15% [1229/1496])
|   |__N2 <= 0.634 (80.75% [1263/1564])
|       |__L1 > 0.143 (82.15% [1229/1496])
|       |__T2 <= 1250 (82.15% [1229/1496])
|__F2 > 0.0008 (72.64% [1383/1904])
    |__F1 > 0.185 (76.53% [1249/1632])
    |__F2 <= 0.664 (76.23% [1244/1632])
```

That is, considering all datasets, the instances corresponding with configurations in which diversity improves the ensembles alone are 65% (improvement considering the F-measure). If we consider only dataset for which $F1v$ is less than 0.657 the percentage increase to 78.98%. If the dataset has also a $F2$ value lower than 0.271, there is an extra increase to 80.75%. The percentage is 82.15 if we further restrict the dataset considering only those that have also a $L1$ greater than 0.143 or a $T2$ value lower or equal to 1250. So if we have a new dataset with values of $F1v$, $F2$, $L1$ and $T2$ verifying this inequalities, we better do not use the ensemble alone, but better combined with a diversity technique (at least is the performance measure we want to improve is the F-measure). The rules for the dataset created using the G-Mean are:

```
Class=yes (56.29% [3215/5712])
|__F1v <= 0.657 (67.34% [1099/1632])
|   |__F2 <= 0.271 (68.41% [1070/1564])
|   |__N2 <= 0.634 (68.41% [1070/1564])
|__L1 > 0.287 (65.88% [1344/2040])
    |__F3 <= 0.809 (69.49% [1323/1904])
    |   |__N1 <= 0.372 (73.4% [1098/1496])
    |   |   |__T2 > 15.385 (75.28% [1075/1428])
    |   |   |__N4 <= 0.356 (74.44% [1063/1428])
    |   |__N3 <= 0.258 (73.4% [1098/1496])
    |       |__T2 > 15.385 (75.28% [1075/1428])
    |       |__N4 <= 0.356 (74.44% [1063/1428])
    |__N1 <= 0.365 (69.12% [1081/1564])
        |__F1v <= 25.483 (71.99% [1077/1496])
        |__T1 > 0.21 (71.99% [1077/1496])
```

In view of the rules obtained, if we have a dataset with low values for the directional-vector maximum Fisher's discriminant ratio, $F1v$, it would be a good idea to apply some diversity techniques[9].

[9] A high $F1v$ indicates that there is a vector that can separate well the classes once instances are projected

In the selected rules do not appear the basic ensemble nor the diversity-enhancing technique. This means that to determine if the basic ensemble could be improved, the meta-features are more relevant than the specific ensemble methods.

Of course all the above analysis would need further investigation, for example to find relations between the ensembles and the diversity technique more suitable when the dataset meta-features verify certain values. We include this analysis here just to give general insights about for which datasets the use of diversity techniques has a higher expectation to improve the use the ensemble alone.

6.3.6 The impact of noisy and borderline examples

Now we will test the suitability of a specific diversity technique, Disturbing Neigbours, for dealing with datasets that have presence of noisy and borderline examples. The repository used for this purpose comes from [47]. It is a repository that contains 30 different synthetic imbalance datasets[10].

The artificial data sets are all 2-dimensional datasets, so the increasing diversity technique more appropriate is Disturbing Neigbours, because it increases the diversity by adding new features to the dataset.

To summarize the results, we used average ranks. They are calculated using all the ensemble methods together with their enhanced version using Disturbing Neigbours.

Table 6.7 shows the results, the first column in each subtable contains the name of the method, the second its average rank and the third value is the improvement (difference between the enhanced ensemble method and its counterpart without additional diversity).

It is clearly seen that the methods which have been combined with Disturbing Neigbours occupy the top positions of the ranking for the three measures. In the ranking calculated with the AUC and the other calculated with the F-Measure, all the methods combined with Disturbing Neigbours obtains a better rank than their equivalent. This is also true for 14 of the 18 methods when the average rank is calculated with the G-Mean. The extra dimensions added by Disturbing Neigbours help in the classification task when there is presence of noisy and borderline examples.

on it, a low value indicates the opposite.

[10]It can be downloaded from `http://sci2s.ugr.es/keel/imbalanced.php#sub50`

Table 6.7: Average ranks and Δ (increment/decrement) of rank between basic and enhanced ensembles in presence of noisy and borderline instances

(a) AUC			(b) F-measure			(c) G-Mean		
Method	Rank	Δ	Method	Rank	Δ	Method	Rank	Δ
DN+Ba-SM100	5.333	12.767	DN+Ba-SM	8.500	4.000	DN+E-RB	4.300	4.467
DN+Ba-SM	5.600	10.033	DN+RAMOBo	8.667	3.567	DN+E-RUS	6.067	8.933
DN+SMBa	6.700	10.433	DN+SMBa	9.367	4.933	SMBa	6.433	
DN+E-RB	7.167	13.567	DN+E-SM	9.600	6.733	DN+Ba-RUS	7.433	5.567
DN+Ba-RB	8.600	10.100	DN+RUSBo	10.533	3.767	DN+SMBa	7.867	-1.433
DN+Ba	9.067	14.050	DN+Ba-RB	11.133	5.433	E-RB	8.767	
DN+RUSBo	9.533	7.967	DN+RBBo	11.967	5.033	DN+E-SM	9.133	0.333
DN+RAMOBo	12.467	4.067	RAMOBo	12.233		RUSBo	9.167	
DN+E-RUS	13.767	14.833	Ba-SM	12.500		E-SM	9.467	
DN+E-SM	14.133	11.700	DN+SMBo	12.767	8.133	Ba-SM	12.233	
DN+Ba-RUS	14.900	11.633	DN+E-RB	12.800	8.267	DN+Ba-SM	12.967	-0.733
DN+RBBo	14.933	1.933	DN+E-SM100	13.233	6.367	Ba-RUS	13	
Ba-SM	15.633		RUSBo	14.300		DN+RUSBo	13.100	-3.933
DN+E-SM100	15.633	12.233	SMBa	14.300		DN+Ba-RB	13.433	0.200
DN+MBo	16	11.400	DN+Ba-SM100	15.067	2.833	Ba-RB	13.633	
RAMOBo	16.533		E-SM	16.333		E-RUS	15	
RBBo	16.867		Ba-RB	16.567		DN+RAMOBo	15.767	2.433
SMBa	17.133		RBBo	17		E-SM100	18.067	
DN+SMBo	17.300	5.067	Ba-SM100	17.900		RAMOBo	18.200	
RUSBo	17.500		DN+ABo2	18.333	9.833	DN+E-SM100	18.400	-0.333
Ba-SM100	18.100		DN+ABo1	18.867	7.467	DN+RBBo	20.100	3.467
Ba-RB	18.700		DN+MBo	19.133	7.067	DN+SMBo	20.667	4.467
E-RB	20.733		DN+Ba-RUS	19.400	6.100	Ba-SM100	21.300	
DN+ABo2	21.500	4.817	E-SM100	19.600		DN+Ba-SM100	22.867	-1.567
SMBo	22.367		SMBo	20.900		RBBo	23.567	
Ba	23.117		E-RB	21.067		DN+ABo2	24.100	6.167
DN+ABo1	23.533	6.067	DN+E-RUS	22.067	5.633	SMBo	25.133	
E-SM	25.833		Ba-RUS	25.500		DN+ABo1	25.433	3.933
ABo2	26.317		MBo	26.200		DN+MBo	26.667	3.000
Ba-RUS	26.533		ABo1	26.333		ABo1	29.367	
MBo	27.400		DN+Ba	26.567	3.833	MBo	29.667	
E-SM100	27.867		E-RUS	27.700		ABo2	30.267	
E-RUS	28.600		ABo2	28.167		DN+Ba	31.300	0.833
ABo1	29.600		Ba	30.400		Ba	32.133	

6.4 Lessons learned

The paper includes a vast experimental study: 17 of the state of the art methods for imbalance learning: RAMOBoost, Random Balance Boost, RUSBoost, SMOTEBoost and many more are tested in its *basic* form and *enhanced* in combination with four different diversifying techniques: Random Oracles, Random Feature Weigths, Disturbing Neighbors and Rotation Forest. Experiments were conducted using datasets from KEEL repository and the HDDT collection. Five different analysis are conducted in this paper and this section enumerates some of the findings of each an analysis.

1. Effect of diversity techniques in combination with ensembles in imbalanced classification. Average ranks are used for multiple method comparisons. The ranks were computed over AUC, F-Measure and Geometric Mean, comparing each ensemble method with its diversity enhanced variants. This is the summary of findings:

    - In the average rank computed using the AUC and according to Hochberg's test, all basic ensembles are significaly worse than at least one of its enhanced counterpart.
    - In the case of using the F-measure to compute the average ranks, the improvement is not statistically significant for all cases, but the improvement exists for the best methods. The best non-enhanced methods are RBBo, Ba-RB and RUSBo, and these methods are improved significantly when combined with Rotation Forest strategy.
    - Something similar happens when examining the rank calculated with the G-mean. Ba-RUS, E-RB and RUSBoare improved significantly when combined with Random Oracles and Disturbing Neighbors.

2. Determination of which is the best combination according different metrics of performance.

    - According to the AUC the overall winner is Rotation Forest.
    - According to the F-measure the best method is RF+RAMOBo (RAMOBo combined with Rotation Forest).

- According to the G-mean the best method is O+Ba-RUS (Ba-RUScombined with Random Oracles.

Another interesting finding is that the best combinations according to the F-measure use oversampling strategies while the best combinations according to the G-mean use undersampling. The method that gets the top position in the rank according to the AUC do not uses any balancing strategy.

3. Impact of ensemble size in their performance. We wanted to ensure that the performance of the ensemble increases as its size increases, up to a point. It is found that in general, bigger ensemble sizes give better average ranks. This trend is also seen in the enhanced ensembles. So the use of ensembles of size 100 is justified. Among the evaluated methods it was found one with a different behavior, RUSBo best performance according to the G-mean is obtained when the size of the ensemble is in 20 to 30 range.

4. Prediction of the convenience of applying diversity techniques using complexity metrics.

   - It was checked whether it is possible to establish relationships between complexity metrics and the fact that the combination of a diversity technique with an ensemble can give better results than the ensemble alone.
   - The rule-learner algorithm HotSpot was used to identify for which meta-features exists a high probability that a certain diversity technique improves the ensemble.
   - From the set of rules generated it was found some pieces of knowledge, for example when the Overlap of the per-class bounding boxes is high or the Directional-vector maximum Fisher's discriminant ratio is low (which indicates the classes are hardly separables when projected into the maximum separability vector) it would be a good idea to apply diversity techniques.

5. Suitability of Disturbing Neigbours for dealing with noisy and borderline examples. Average ranks were used and it was found that methods which have been combined with Disturbing Neigbours perform better than their non-enhanced counterpart for all performance measures

6.5 Concluding remarks

This article presents an exhaustive experimental study that combines techniques especially designed to work with imbalanced data with ensemble diversifying techniques. Examining 17 ensembles on their own and with four diversifying techniques, using 84 imbalanced data sets, we found that enhancing diversity pays off. Diversity-enhanced ensembles ranked better than their original counterpart. This is a curious finding because all diversity-enhancing techniques that we applied are "imbalance-blind". The method with best ranking in our experiments was the basic Rotation Forest according to AUC, and Rotation Forest combined with balancing techniques according to the F-measure. When the G-Mean is used, there is not a technique of increasing diversity that highlights so clearly, but the techniques of increasing diversity clearly improve the ensembles to which they are applied. One interesting conclusion of this study is that the results obtained for one measure can not be extrapolated to others, and one method that is the best according to a measure, not necessarily is the best according to others.

In order to justify the ensemble size used in the experiments, it has been checked whether the ensemble size can influence the results on the improvement that can be achieved by the diversity-enhancing techniques. In general, bigger ensemble sizes give better average ranks. The RUSBo, according to the G-mean, is an exception to this general tendency, as its best results are for sizes 20 and 30. However, the general tendency is observed if AUC and F-measure are considered. This reinforces the previous observation, the results obtained using one measure not necessarily are obtained when the others are used. Another conclusion is that it does not matter the size of the ensemble, it is always possible to improve the results by using a diversity-enhacing technique.

A preliminary study has been made that attempts to characterize for which datasets the use of diversity-enhancing techniques could be beneficial.

6.6 Future research directions

One interesting future line of research would be to take this study further to investigate whether it is possible to find an optimal combination of balancing and diversity techniques or which is the balancing technique and diversity strategy best suited to a problem based on certain meta-features.

Another future research line could be to investigate further the effect of the diversity techniques on the class overlapping and small disjuncts, two of the characteristics that make imbalance problems so difficult to solve. The aim will not only be to provide insight on why these techniques work, but also could inspire the development of new diversity strategies specially designed to deal with imbalanced datasets.

Acknowledgements

This work was supported by the Project TIN2011-24046 of the Spanish Ministry of Economy and Competitiveness.

References

[1] J. Alcala-Fdez et al. "KEEL Data-Mining Software Tool: Data Set Repository and Integration of Algorithms and Experimental Analysis Framework". In: *Journal of Multiple-Valued Logic and Soft Computing* 17.2-3 (2011), pp. 255–287.

[2] K. Bache and M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[3] R Barandela, RM Valdovinos, and JS Sánchez. "New applications of ensembles of classifiers". In: *Pattern Analysis & Applications* 6.3 (2003), pp. 245–256.

[4] G.E. Batista, R.C. Prati, and M.C. Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 20–29.

[5] E. Bauer and R. Kohavi. "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants". In: *Machine learning* 36.1 (1999), pp. 105–139.

[6] L. Breiman. "Bagging predictors". In: *Machine Learning* 24 (1996), pp. 123–140.

[7] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125.

[8] Carla E Brodley and Mark A Friedl. "Identifying Mislabeled Training Data". In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 131–167.

[9] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD09)*. Vol. 5476. Lecture Notes on Computer Science. Springer-Verlag, 2009, pp. 475–482.

[10] N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6.

[11] N.V. Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of Artificial Intelligence Research* 16.1 (2002), pp. 321–357.

[12] N.V. Chawla et al. “SMOTEBoost: Improving prediction of the minority class in boosting”. In: *7th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD 2003)*. 2003, pp. 107–119.

[13] Sheng Chen, Haibo He, and Edwardo A Garcia. “Ramoboost: Ranked minority oversampling in boosting”. In: *Neural Networks, IEEE Transactions on* 21.10 (2010), pp. 1624–1642.

[14] David A. Cieslak and Nitesh V. Chawla. “Learning Decision Trees for Unbalanced Data”. In: *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*. ECML PKDD ’08. Antwerp, Belgium: Springer-Verlag, 2008, pp. 241–256. ISBN: 978-3-540-87478-2. DOI: 10.1007/978-3-540-87479-9_34.

[15] David A. Cieslak et al. “Hellinger distance decision trees are robust and skew-insensitive”. In: *Data Min. Knowl. Discov.* 24.1 (Jan. 2012), pp. 136–158. ISSN: 1384-5810. DOI: 10.1007/s10618-011-0222-1.

[16] Janez Demsar. “Statistical Comparisons of Classifiers over Multiple Data Sets”. In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[17] Matias Di Martino et al. “Improving Electric Fraud Detection using Class Imbalance Strategies.” In: *ICPRAM (2)*. 2012, pp. 135–141.

[18] T.G. Dietterich. “Approximate statistical tests for comparing supervised classification learning algorithms”. In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[19] José F Díez-Pastor et al. “Random Balance: Ensembles of Variable Priors Classiffiers for Imbalanced Data”. In: *Under Submision* (2014), pp. –.

[20] Wei Fan et al. “AdaCost: Misclassification Cost-Sensitive Boosting”. In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML ’99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 97–105. ISBN: 1-55860-612-2.

[21] T. Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[22] Yoav Freund and Robert E. Schapire. “Experiments with a New Boosting Algorithm”. In: *Machine Learning, Proceedings of the Thirteenth International Conference (ICML ’96), Bari, Italy, July 3-6, 1996*. 1996, pp. 148–156.

[23] M. Galar et al. “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42.4 (2012), pp. 463 –484. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2011.2161285.

[24] Mikel Galar et al. “Eusboost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling”. In: *Pattern Recognition* 46.12 (2013), pp. 3460–3471.

[25] Vicente García, Ana Isabel Marqués, and Jose Salvador Sánchez. "Improving risk predictions by preprocessing imbalanced credit data". In: *Neural Information Processing*. Springer. 2012, pp. 68–75.

[26] Nicolás García-Pedrajas and César García-Osorio. "Constructing ensembles of classifiers using supervised projection methods based on misclassified instances". In: *Expert Systems with Applications* 38.1 (2011), pp. 343 –359. ISSN: 0957-4174.

[27] Nicolás García-Pedrajas et al. "Class imbalance methods for translation initiation site recognition in DNA sequences". In: *Knowl.-Based Syst.* 25.1 (2012), pp. 22–34.

[28] Nicolás García-Pedrajas et al. "Supervised subspace projections for constructing ensembles of classifiers". In: *Information Sciences* 193.0 (2012), pp. 1 –21. ISSN: 0020-0255.

[29] Guang-Gang Geng et al. "Boosting the performance of web spam detection with ensemble under-sampling classification". In: *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*. Vol. 4. IEEE. 2007, pp. 583–587.

[30] Mark Hall et al. "The WEKA data mining software: an update". In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[31] H. Han, W.Y. Wang, and B.H. Mao. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *2005 International Conference on Intelligent Computing (ICIC05)*. Vol. 3644. Lecture Notes on Computer Science. Springer-Verlag, 2005, pp. 878–887.

[32] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE. 2008, pp. 1322–1328.

[33] Tin Kam Ho, Mitra Basu, and Martin Hiu Chung Law. "Measures of geometrical complexity in classification problems". In: *Data complexity in pattern recognition*. Springer, 2006, pp. 1–23.

[34] T.K. Ho. "The random subspace method for constructing decision forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[35] T.K. Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.3 (2002), pp. 289–300. ISSN: 0162-8828.

[36] Y. Hochberg. "A sharper Bonferroni procedure for multiple tests of significance". In: *Biometrika* 75 (1988), pp. 800–803.

[37] R.L. Iman and J.M. Davenport. "Approximations of the critical region of the fbietkan statistic". In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[38] Taeho Jo and Nathalie Japkowicz. "Class imbalances versus small disjuncts". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 40–49.

[39] M. Kubat and Matwin. "Addressing the Curse of Imbalanced Training Sets : One-Sided Selection". In: *Proceedings of the 14th International Conference on Machine Learning*. 1997, pp. 179–186.

[40] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[41] L.I. Kuncheva and J.J. Rodriguez. "Classifier ensembles with a random linear oracle". In: *IEEE Transactions on Knowledge and Data Engineering* 19.4 (2007), p. 500.

[42] Wei Liu et al. "A Robust Decision Tree Algorithm for Imbalanced Data Sets". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2010*. 2010, pp. 766–777.

[43] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. "Exploratory undersampling for class-imbalance learning". In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39.2 (2009), pp. 539–550.

[44] Victoria López et al. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* 250.0 (2013), pp. 113 –141. ISSN: 0020-0255. DOI: http://dx.doi.org/10.1016/j.ins.2013.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0020025513005124.

[45] J. Maudes, J. J. Rodríguez, and C. García-Osorio. "Disturbing neighbors diversity for decision forests". In: *Applications of Supervised and Unsupervised Ensemble Methods*. Ed. by Oleg Okun and Giorgio Valentini. Vol. 245. Studies in Computational Intelligence. Springer, 2009, pp. 113–133. ISBN: 978-3-642-03998-0. URL: http://dx.doi.org/10.1007/978-3-642-03999-7_7.

[46] Jesús Maudes et al. "Random feature weights for decision tree ensemble construction". In: *Information Fusion* 13.1 (2012), pp. 20–30.

[47] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. "Learning from imbalanced data in presence of noisy and borderline examples". In: *Rough Sets and Current Trends in Computing*. Springer. 2010, pp. 158–167.

[48] Robi Polikar. "Ensemble learning". In: *Ensemble Machine Learning*. Springer, 2012, pp. 1–34.

[49] Ronaldo C Prati, Gustavo EAPA Batista, and Diego F Silva. "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods". In: *Knowledge and Information Systems* (2014), pp. 1–24.

[50] F. Provost and P. Domingos. "Tree induction for probability-based ranking". In: *Machine Learning* 52.3 (2003), pp. 199–215.

[51] Foster Provost and Venkateswarlu Kolluri. "A survey of methods for scaling up inductive algorithms". In: *Data mining and knowledge discovery* 3.2 (1999), pp. 131–169.

[52] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[53] Joaquin Quionero-Candela et al. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN: 0262170051, 9780262170055.

[54] JJ Rodriguez, LI Kuncheva, and CJ Alonso. "Rotation forest: A new classifier ensemble method". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (2006), pp. 1619–1630.

[55] Alon Schclar and Lior Rokach. "Random Projection Ensemble Classifiers". In: *Enterprise Information Systems*. Ed. by Joaquim Filipe and Jose Cordeiro. Vol. 24. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2009, pp. 309–316. ISBN: 978-3-642-01346-1.

[56] C. Seiffert et al. "RUSBoost: A hybrid approach to alleviating class imbalance". In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40.1 (2010), pp. 185–197.

[57] Jerzy Stefanowski. "Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data". In: *Emerging Paradigms in Machine Learning*. Springer, 2013, pp. 277–306.

[58] Y. Sun et al. "Cost-sensitive boosting for classification of imbalanced data". In: *Pattern Recognition* 40 (2007), pp. 3358–3378.

[59] Ivan Tomek. "Two modifications of CNN". In: *Systems, Man and Cybernetics, Transactions on* 6 (1976), pp. 769–772.

[60] C.J. Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[61] F. Verhein and S. Chawla. "Using Significant, Positively Associated and Relatively Class Correlated Rules for Associative Classification of Imbalanced Datasets". In: *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. 2007, pp. 679–684.

[62] Sofia Visa and Anca Ralescu. "Issues in mining imbalanced data sets - a review paper". In: *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*. 2005, pp. 67–73.

[63] S. Wang and X. Yao. "Diversity analysis on imbalanced data sets by using ensemble models". In: *IEEE Symposium Series on Computational Intelligence and Data Mining(IEEE CIDM 2009)*. 2009, pp. 324–331.

[64] Shuo Wang and Xin Yao. "Relationships between diversity of classification ensembles and single-class performance measures". In: *Knowledge and Data Engineering, IEEE Transactions on* 25.1 (2013), pp. 206–219.

[65] M. Wasikowski and Xue wen Chen. "Combating the Small Sample Class Imbalance Problem Using Feature Selection". In: *Knowledge and Data Engineering, IEEE Transactions on* 22.10 (2010), pp. 1388–1400. ISSN: 1041-4347.

[66] Geoffrey I. Webb. "MultiBoosting: A Technique for Combining Boosting and Wagging". In: *Machine Learning* 40.2 (2000), pp. 159–196.

[67] Gary M Weiss. "The impact of small disjuncts on classifier learning". In: *Data Mining*. Springer. 2010, pp. 193–226.

[68] D.L. Wilson. "Asymptotic properties of nearest neighbor rules using edited data". In: *Systems, Man and Cybernetics, IEEE Transactions on* 2.3 (1972), pp. 408–421.

[69] Hualong Yu et al. “Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets”. In: *Tsinghua Science and Technology* 17.6 (2012), pp. 666–673.

[70] Bianca Zadrozny and Charles Elkan. “Learning and making decisions when costs and probabilities are both unknown”. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2001, pp. 204–213.

Chapter 7

Boosting Projections to improve surface roughness prediction in high-torque milling operations

Authors José-Francisco Díez-Pastor, Andrés Bustillo, Guillem Quintana, Cesar García-Osorio

Type Journal

Published in Soft Computing. 16(8): 1427-1437

Year 2012

Abstract

Industrial solutions for surface roughness prediction are in great demand, especially in high-torque milling operations, owing to the exponential expansion of wind power energy generation over the last decade. In this paper, we use Boosting Projections to predict surface roughness in high-torque, high-power face milling operations. A data set is generated from experiments performed under industrial conditions, using a milling machine with a high working volume, in order to train and validate the new algorithm. The experimental data comprise a very extensive set of parameters that influence surface roughness: cutting tool properties, machining parameters and cutting phenomena. The proposed method is based on Non-linear Boosting Projections (although it uses linear projections to speed up the training process). To the best of our knowledge this is the first time it has been used in an industrial context. It demonstrates a higher prediction accuracy when

compared with single multilayer perceptrons, decision trees, and classical ensemble methods .

Index terms— high-torque milling, surface roughness, ensemble methods, linear projections, ordinal classification

7.1 Introduction

The manufacture of many industrial metal components such as moulds and dies is a complex process, usually involving a trade-off between various parameters, in which the following constraints play an important role: product quality specifications, pre-defined equipment, cost and time constraints, among others. Unfortunately, there are certain quality characteristics of a product, such as surface roughness, that can not be accurately evaluated before the manufacturing process is completed [5]. Surface roughness is quantified by the vertical deviations of a real surface from its ideal form. Surface roughness plays an important role in the performance of a finished part. It has an enormous influence on several relevant characteristics of the final product such as dimensional accuracy, friction coefficient, wear, thermal resistance, electric resistance, fatigue limit and behaviour, corrosion, post-processing requirements, appearance and cost [30].

The final roughness of a milled workpiece will depend on many parameters and the interactions or cause-effect relationships between them. Benardos and Vosniakos [5] divided these factors into 4 main groups: 1) cutting tool properties such as tool material, tool shape and nose radius; 2) machining parameters such as process kinematics, feed rate, cutting speed, cooling fluid, step over, depth of cut and tool angle; 3) cutting phenomena such as cutting force variation, vibrations, friction in the cutting zone and chip formation; and 4) workpiece properties such as length, diameter and hardness.

The complexity of the roughness formation mechanism means it is difficult to find a simple and general solution to optimize the milling process parameters. Therefore, the process engineer prefers to select conservative cutting conditions in a real industrial context, which neither guarantee a desired surface quality, nor guarantee high metal removal rates [5]. In the moulds and dies industry, for instance, it is standard practice to apply a final manual polishing operation, in order to achieve the required surface roughness. Obviously, this solution is a time-consuming process that radically increases the cost of the mould and depends greatly on the skills of the workers that perform the manual polishing operation [41].

The optimization of surface quality under industrial conditions entails a further difficulty: surface roughness can only be measured off-line after the part has been machined. This is an out-of-process quality evaluation

Table 7.1: Roughness levels according to ISO Standard 4288:1996.

Level	Ra (μm)
N1	[0.000,0.025)
N2	[0.025,0.050)
N3	[0.050,0.100)
N4	[0.100,0.200)
N5	[0.200,0.400)
N6	[0.400,0.800)
N7	[0.800,1.600)
N8	[1.600,3.200)

process that inevitably results in a loss of time and money, as there is no other alternative than to remove defective parts from the production line.

Surface quality is usually evaluated using the average surface roughness parameter (Ra). In accordance with ISO standard 4287:1997 [36], Ra is calculated as an arithmetic average of absolute values of the vertical deviations from the nominal surface (y) for a certain distance over which the surface deviations are measured (Lm), as shown in Eq.7.1.

$$Ra = \int_0^{Lm} \frac{|y|}{Lm} dx \tag{7.1}$$

Although surface roughness could take any continuous value, its evaluation is usually done in a discrete way to assure the functionality of the manufactured component. For example, if holes for ejector pins or surface of gear's teeth do not have a roughness quality of N6 (ISO 4288:1996 [37]), they fail to guarantee correct ejection of mould components following plastic injection or smooth movement of the gears in a gearbox. ISO standard 4288:1996 [37] is the international reference for measured roughness quality in machining processes, establishing 12 roughness levels from 0.006 to 50 μm. Table 7.1 summarizes the first 8 levels of this scale.

In consequence, a great deal of literature has been generated on the surface generation process, due to its high complexity and the evident industrial interest in its improvement. There are four broad lines of research into surface roughness prediction [5] based on: 1) machining theory [45, 35, 48, 2]; 2) experimental investigations [4, 52]; 3) designed experiments [15] and 4) artificial intelligence [17, 6, 9, 18, 49, 42, 33, 34, 50, 16, 10, 47, 11]. Nevertheless, the classifications in much of the literature, where no one single methodology is followed, are not always straightforward.

The first category, based on machining theory, considers process kinematics, cutting tool properties and chip formation mechanisms, among others. An essential reference is the approach to peripheral milling operations proposed by Martellotti, in 1941, [44] whose mathematical treatment of peripheral milling operations showed that a cutting tool path is a trochoid arc. A trochoid arc is described by an equation that is formulated from known cutting variables in such a way that, when considering a rigid tool and a rigid workpiece system, a maximum feed mark height may be easily calculated. With regard to the second category, experimental investigations usually examine the effects of various factors through results obtained from experimentation and their subsequent analysis. Designed experiments, which form the third category, apply systematic methodologies such as response surface methodology or Taguchi techniques for planning and analyzing experimentation.

The fourth group is composed of those approaches that use artificial intelligence methods: Artificial Neural Network (ANN) models, genetic algorithms, fuzzy logic or expert systems. These techniques simulate the way in which human beings process information and take decisions. Different alternative approaches to the industrial task of roughness prediction in milling operations have been tested with good results: neuro-fuzzy inference system [42, 33, 34, 50], Bayesian networks [17, 16], genetic algorithms [10, 9] and support vector machines [47]. However, the neural networks approach [49, 15, 6, 16, 11] is the most widely used solution. Its most common configuration is a Multilayer Perceptron (MLP) with a single hidden layer [30]. Unfortunately, the results obtained are highly dependent on the parameters of the neural networks [11]. The process of fine-tuning these parameters requires a lot of work and experience and there are no general rules that may be followed as a guide, which means the scientific process is somewhat of an art.

All of the above soft-computing techniques developed over the past 20 years are becoming standard practice in manufacturing and are incorporated in commercial software. New techniques have also been developed over the past 10 years, although their suitability for this kind of industrial task has still to be demonstrated. Their potential to improve process optimization accuracy is an active research line [13]. One of these techniques is ensemble learning: a learning paradigm where multiple learners (or classifiers, or, in the ensemble learning context, *base classifiers*) are combined to solve a

problem. A classifier ensemble can significantly improve the generalization ability of a single classifier and can provide better results than an individual classifier [46] in many applications. Algorithms are used to construct ensembles from the generation of the base classifiers (different models, or more usually a single model with different initialization parameters) as well as from combinations of their results to give the final prediction model (for example, by majority voting) [40]. There are a few examples of the use of ensembles to improve milling processes due to the novelty of this technique. These examples refer to breakage detection [14, 7] and to roughness prediction in laser polishing [12] and in end-mill finishing [11] of milled surfaces.

Most of the existing literature describes finishing operations using ball-end mills that only require a low torque. This work refers to face mills working under high torque and high-power finishing operations on steel, which are nowadays widely employed for the manufacture of wind turbine components. Face mills achieve a better roughness surface if no strong vibrations appear during the milling process, and if the mill is able to machine all the required surface in one pass, avoiding third order deviations from the nominal surface in surface roughness [5]. These requirements are possible on the critical surfaces of many different wind turbines components, such as the main frame and the hub, if face mills with diameters greater than 35 mm are used.

The aim of this work is to demonstrate that new algorithms of ensembles could improve the accuracy of surface roughness prediction models and also avoid the time-consuming task of tuning ANN models to a certain data set. Ensemble modelling has been applied as an interpolator for the calculation of the average surface roughness parameter, considering the complex phenomena that influences surface roughness generation and the large number of factors that interact in the milling process. The data set used for training and validation of the new algorithm was obtained from experimental tests performed under industrial conditions, varying all of the cutting conditions that the machine operator would usually adjust during the manufacturing process.

The paper is structured as follows: Section 7.2 explains the experimental procedure and the data set generated to validate the new algorithm proposed in this work; Section 7.3 reviews the ensemble learning techniques tested for the prediction model; Section 7.4 provides the background on

Boosting Projections; Section 7.5 explains some questions about ordered classification; Section 7.6 shows the results obtained with these techniques and includes an in-depth comparison with an ANN approach for the same data set. Finally, the conclusions and future lines of work are summarized in Section 7.7.

7.2 Experimental procedure and data set description

All of the four groups of parameters that influence surface roughness - *machining parameters*, *cutting tool properties*, *workpiece properties*, and *cutting phenomena*- can not be modified under real workshop conditions. In a workshop, the *workpiece properties* will not usually be modified, because they will be specified by the client in the order for the machined workpiece. *Cutting phenomena* include all the phenomena that can not be controlled by the operator, such as vibrations, cutting forces, etc. These can only be measured during the cutting process but not directly changed. *Cutting tool properties* depend on the cutting tool selected from those available in the workshop, therefore the operator only has a limited possibility of influencing workpiece roughness changing this third group of parameters, and none at all when the cutting tool is already selected. The *machining parameters* include all the parameters that may be changed when the operator designs the machining CAM (Computer Aided Manufacturing) program. The main machining parameters are cooling fluid, depth of cut, feed rate, cutting speed, and stepover [5]. The experiments presented in this paper were designed to take the majority of these parameters into account. This approach will allow us to include the main factors that affect surface roughness in the prediction system, which may be controlled by the machine operator; thereby guaranteeing a reliable prediction system clearly focused on industrial needs.

The experimental procedure and data collection has been already described elsewhere for ball-end milling [49]. The experiments consisted of a simple raster along the machine tool's Y axis. During the experiments, process vibrations were captured with piezoelectric unidirectional accelerometers. The following parameters were considered: milling head speed, feed rate, feed per tooth and axial depth of cut. The experimental design aims to evaluate the wide range of values for cutting parameters under real industrial conditions. A sufficiently large number of parametric

Table 7.2: Cutting conditions selected for the experimental tests.

Cutting parameter (units)	*Range*
Coolant system	Dry milling
Tool radius (mm)	20
Radial depth of cut (mm)	40
Axial depth of cut (mm)	0.1, 1, 1.5, 2, 3,
Milling head speed (rpm)	1655, 1759, 1862, 1966, 2069, 2172, 2276, 2379, 2483
Feed per tooth ($\times 10^{-2}$ mm/tooth)	10, 10.4, 10.6, 10.9, 11.1, 11.2, 11.4, 11.6, 11.7, 11.9, 12, 12.1, 12.2, 12.4, 12.5, 12.6, 12.8

combinations were considered in this research and data gathered on surface roughness proved useful. A total of 141 experiments were designed and performed. Cutting parameters and levels are summarized in Table 7.2.

The experimental data were taken from a Nicolas Correa Axis floor-type moving column milling machine. It was equipped with a universal automatic milling head specially designed for high-torque operations: 32 kW power up to 6000 rpm and 1045 Nm maximum torque. The cutting tool was a Walter F4042.B.040.Z04.15 face mill with ADMT160608R-D56 WKP25-type inserts. This model has 4 inserts (Z) and a diameter of 40 mm. The cast iron (EN-GJS-400-18U-LT) blank used for the tests is typically used for the manufacture of wind turbines components such as the hub and the main frame.

Vibrations were captured by two unidirectional piezoelectric accelerometers placed along the X and Y axis of the machine tool, both of which were placed on the milling head. The sampling frequency was 10 kHz. After experimentation, the vibration signals were analyzed to split the data between the rapid traverses along the Y axis, where acceleration is maximum, and during the effective cut when the tool removes material. Only those vibrations that occurred when the tool was engaged in the workpiece were included in the data sets. Different vibration variables were considered: low, medium and high-frequency vibration amplitudes, temporal domain vibration amplitude and tooth passing frequency amplitude, all of which on the X and Y axes.

Once the experiments had been performed, a Diavite DH5 roughness tester with a nominal 2 μm stylus tip was used to measure surface roughness. The evaluation length was 4.8 mm composed of 6 basic lengths of 0.8mm. The *Ra* parameter was calculated as a mean value of measured roughness, as shown in Eq.7.1, in accordance with ISO standard 4287:1997 [36]. The calculated roughness was then discretized according to ISO standard 4288:1996 [37], as is often done in the literature before applying any artificial intelligence technique [17, 16, 49, 11]. Table 7.1 shows the intervals in which the final roughness was discretized; only four of these intervals, from level 5 to level 8, are presented in the data set with the following distribution: 17 experiments had N5 level final roughness, 136 experiment N6 level, 250 experiments N7 and 20 experiments N8 level. Surface roughness was measured 3 times for each experiment at different locations to increase the size of the data set: at the beginning, midway along its length and close to the end of the simple raster. The vibrations varied in these 3 measurements, even though the cutting conditions did not vary.

The experiments generated a data set of 423 records that contain information on the following 18 variables: axial depth of cut (A_p), feed rate (f), milling head speed (N), feed per tooth (f_z), cutting speed (V_c), tooth passing frequency (f_t), cutting section (Cs), low frequency vibration amplitude on X and Y axes (l_x, l_y), medium frequency vibration amplitude on X and Y axes (m_x, m_y), high frequency vibration amplitude on X and Y axes (h_x, h_y), temporal domain vibration amplitude on X and Y axes (td_x, td_y), tooth passing frequency amplitude on X and Y axes (tp_x, tp_y), and material removal rate (MRR). It should be underlined that many of these input variables are determined by the others. Table 7.3 also shows which of the input variables are non-deterministic and which are deterministic and their relation with the non-deterministic variables, based on the definition given in [48] and after some ordering described in detail in [11].

7.3 Introduction to ensembles

The choice of which film to see at the cinema may be influenced by film reviews, by Oscar nominations, and even by box-office statistics. The purchase of a new vehicle may involve quizzing friends who already have one, reading specialist automobile magazines, or consulting specialized webpages. Indeed, prior to the publication of a scientific article, it will

Table 7.3: Ranges of the input and output variables and relationship between them (ndpp: non-deterministic-process parameter, ndctc: non-deterministic-cutting tool characteristic, ndipmd: non-deterministic-in-process measured data, det: deterministic).

Variable (Units)	Range	Relationship
A_p (mm)	0.1–3	ndpp
f (mm/min)	993–2978	ndpp
N (rpm)	1655–2483	ndpp
td_x (mm/s^2)	0.095–0.105	ndipmd
td_y (mm/s^2)	0.130–0.150	ndipmd
tp_x ($\times 10^{-3}$ mm/s^2)	0.02–0.19	ndipmd
tp_y ($\times 10^{-3}$ mm/s^2)	0.02–0.19	ndipmd
l_x ($\times 10^{-3}$ mm/s^2)	0.20–0.36	ndipmd
l_y ($\times 10^{-3}$ mm/s^2)	0.10–0.27	ndipmd
m_x ($\times 10^{-3}$ mm/s^2)	0.25–0.40	ndipmd
m_y ($\times 10^{-3}$ mm/s^2)	0.10–0.26	ndipmd
h_x ($\times 10^{-3}$ mm/s^2)	0.51–0.73	ndipmd
h_y ($\times 10^{-3}$ mm/s^2)	0.24–0.41	ndipmd
f_t (Hz)	110–165	det ($N{\cdot}Z/60$)
Cs (mm^2)	4–120	det (see [11])
MRR (mm^3/min)	3970–89360	det ($= f{\cdot}Cs$)
f_z (mm/tooth)	0.1–0.127	det ($= f/N{\cdot}Z$)
Vc (m/min)	208–312	det ($= 2\pi{\cdot}R{\cdot}N/10^{-3}$)
Ra (μm)	0.24–1.96	output

be reviewed by various experts and in accordance with their opinions, the editor will decide either to accept or to reject it. These examples, taken from everyday life, all apply the same raison-d'être that justifies the use of ensembles: the assimilation of various expert opinions and/or sources of information for the purpose of taking a decision.

Formally, an ensemble of classifiers consisted of a combination of different, homogeneous or heterogeneous classifiers that jointly performed a classification task. Ensemble construction is one of the fields of machine learning that is receiving most research attention at present, which is mainly due to the significant way in which ensemble methods are reported to outperform single classifiers [3, 53, 40, 43].

A classification problem of K classes and n training observations consists of a labelled data set

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

where, each instance $\mathbf{x}_i$ belongs to a domain X and each class label, y_i, is an integer from the set $Y = \{1, \ldots, K\}$. A classifier is a function $f : X \to Y$ that maps an instance $\mathbf{x} \in X \subseteq \mathbb{R}^D$ onto an element of Y. The difficulty consists in finding a definition for the unknown function $f(\mathbf{x})$. In the case of ensembles we have a set of base classifiers $\mathscr{C} = C_1, C_2, \ldots, C_m$, each of which map $\mathbf{x} \in \mathbb{R}^D$ onto a label in the set $Y = \{1, \ldots, K\}$. The construction of an ensemble must fulfil two main tasks: the construction of the individual classifiers C_i, and the design of the combination rule that allows the ensemble to get the label of the instance $\mathbf{x}$ from the outputs of the individual classifiers $\{C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_m(\mathbf{x})\}$. A more detailed description of the process may be found in [53, 40, 43, 19, 22].

There are several reasons for using a classifier ensemble instead of a single classifier, as described in [19]:

- *Statistical reason*: to obtain a classifier only one subset of all possible instances that can occur in practice is used. Only one part from that subset is used for training purposes, while the other part is used for estimating the generalization error. This error, though, is an estimate that only partially guarantees good classifier performance when dealing with new instances. Using this estimate could lead us to choose a non-optimal classifier, a problem that is even more acute when the set of instances is too small. The combination of the outputs from several classifiers may reduce this risk.

- *Computational reason*: many algorithms used to obtain classifiers may often be trapped in local optima. For example, neural networks use gradient descent to minimize the error function over the training set, while decision trees use a greedy algorithm to grow the tree. The final classifier is strongly dependent on the initial conditions. Also, in some cases, convergence will gradually slow down, adding considerably to the time needed for computation of the last phase of the learning process. In such cases, it is difficult to find the best classifier. An ensemble constructed from classifiers that have started their training under different initial conditions may lead to a classifier that approximates optimal conditions in better way than any of the individual classifiers.

- *Representational reason*: the decision boundary that separates data from different classes may be too complex for a single classifier, but can be approximated from an appropriate combination of different individual classifiers. For example, non-linear data may not be correctly classified by a single linear classifier. However, a group of several linear decision classifiers can be combined, in order to solve the problem. Also, for certain problems, an ensemble of simple classifiers will perform faster and is more easily implemented than a single complex classifier.

A combination of diverse multiple classifiers makes sense wherever the instances in which they make erroneous predictions differ. The underlying logic is that if they all gave the same output, the whole ensemble could be replaced by a single classifier. In contrast, the correct predictions in a proper combination of classifiers will be able to compensate its incorrect predictions, when the ensemble classifier predictions are wrong for different instances. Hence, the most common strategy for building ensembles is to ensure diversity among the base classifiers in the ensemble [39, 19].

Two of the most successful and widely used heuristics are: *i*) train each classifier with a different sample of the data set, *ii*) train each classifier with different subsets of the data set features. Bagging and Boosting are two strategies that use the first approach, Bagging [8] trains each base classifier in the ensemble with a different bootstrap sample of the original training. Bootstrap sampling is sampling with replacement. Boosting [24] is similar to bagging, in that each classifier is trained with a different bootstrap sample of the original training set, but with two differences: *i*) each sample is built

in order to attribute greater importance to the instances that have been misclassified by the previous classifiers (that is, the instance probability of being chosen as part of next sample is higher when the instance has been misclassified by previous classifiers); *ii*) in the final ensemble, each classifier is given a weight associated with its specific accuracy. Initially, all the instances have the same weight and have the same probability of being chosen to form the training set of a classifier. During the boosting procedure, the weights are adjusted after the training of each classifier. The weights of the misclassified instances are increased, whereas they are reduced for correctly classified instances. Random Subspace method [32] is an example of the second heuristic. These methods have been quite successful and their comparision with new algorithms is common practice.

7.4 Introduction to boosting projections

In a previous work [29], we presented a method based on the use of the misclassified instances to obtain a supervised projection of the data set, which assisted correct classification of these instances, but without putting too much pressure on their correct classification.

This approach is able to incorporate the advantages of boosting without its main drawbacks; boosting is very sensitive to noise and does not usually perform well on small data sets [20] (noise and small data set sizes are both characteristics of data sets in the field of machining).

The construction of the projection taking into account only instances that have been misclassified by a previous classifier enables the new classifier to focus on difficult instances. Nevertheless, as this classifier receives a uniform distribution of the training instances, sensitivity to noise and the effect of small data sets is greatly reduced. The proposed method at each step t considers only the subset of instances, $S' \subset S$, misclassified by the classifier added in step $t-1$. It uses the instances in S' to obtain a supervised projection that is focused only on misclassified instances. In [28], the supervised projection was non linear and used the hidden layers of a Multilayer Perceptron(MLP). In [27], the projection was linear, the results were equally good, but the projection was obtained much faster, because there was no need to train an MLP. The proposed method is shown in Algorithm 9. The next section explains how the supervised linear projections are obtained.

Algorithm 9: Linear Boosting Projections algorithm.

Input: A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a base learning algorithm, $\mathbb{L}$, and the number of iterations T.

Output: The final classifier: $C^*(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t:C(\mathbf{x})=y} 1$.

1 $C_0 = \mathbb{L}(S)$

for $t = 1$ *to* $T-1$ **do**

2 $S' \subset S$, $S' = \{\mathbf{x}_i \in S : C_{t-1}(\mathbf{x}_i) \neq y_i\}$.

3 Obtain supervised linear/non-linear projection $\mathbf{P}(\mathbf{x})$ using S'.

4 $C_t = \mathbb{L}(\mathbf{P}(S))$

end

7.4.1 Linear Supervised Projections

One of the most widely used methods for linear projection is Principal Component Analysis (PCA). PCA projects the data set onto the directions which explain most of the variance in the data set. Assuming Gaussian distribution, these are the directions with more information. The main drawback of PCA is that it is an unsupervised technique and does not take into account the class labels of the data set. PCA tries to find the linear transformation W that maximizes

$$J(W) = W^T \widehat{\Sigma} W \tag{7.2}$$

where $\widehat{\Sigma}$ is the sample covariance matrix for the whole data set

$$\widehat{\Sigma} = \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(x_i - \mu)^T$$

It can be shown that PCA solves the eigenproblem

$$\widehat{\Sigma}\mathbf{v} = \lambda \mathbf{v}$$

PCA is more a method for efficient representation rather than a method for efficient discrimination. A typical example where PCA fails is shown in Figure 7.1. In such cases we need a supervised technique such as Linear Discriminant Analysis.

7.4.1.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) was first used for two classes. It identifies a linear subspace that maximizes the class separability. The

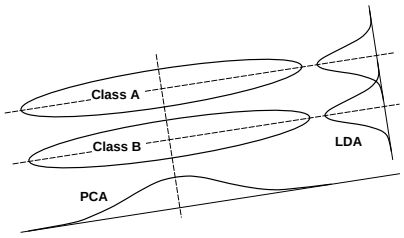


Figure 7.1: PCA fails when the class labels are not used.

objective is to find a projection W that maximizes the ratio of between-class scatter S_B as against within-class scatter S_W [26].

$$\arg\max_W = \frac{\left|WS_BW^T\right|}{|WS_WW^T|} \tag{7.3}$$

where

$$S_W = \sum_{i=1}^{L} P_i\widehat{\Sigma}_i \tag{7.4}$$

$$S_B = \sum_{i=1}^{L} P_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{7.5}$$

P_i is the *a priori* probability of class $_i$, $\widehat{\Sigma}_i$ is the sample covariance matrix for class i, L is the number of classes, μ_i is the sample mean vector of class i, and μ is the global sample mean vector $\mu = \sum_{i=1}^{L} P_i\mu_i$.

The maximization can be performed solving the generalized eigenproblem

$$S_B\mathbf{v} = \lambda S_W\mathbf{v}$$

LDA has three important disadvantages: *i*) it assumes a Gaussian distribution over the class distribution of the data samples; and *ii*) the dimensionality of the subspaces obtained is limited by the number of classes; for an L classes data set, at most $L-1$ dimensional, *iii*) if S_W is singular (which often occurs in high dimensional problems), LDA performs poorly for classification.

One simple method to overcome the constraints of LDA is a method called Hybrid Discriminant Analysis, which is explained in the next subsection.

7.4.1.2 Hybrid Discriminant Analysis

Tian et al. presented Hybrid Discriminant Analysis (HDA) in [51] as a framework that unifies PCA and LDA. The ratio to maximise in HDA is

$$\arg\max_{W} \frac{\left|W^T \left((1-\lambda)S_B + \lambda\widehat{\Sigma}\right) W\right|}{\left|W^T \left((1-\eta)S_W + \eta I\right) W\right|} \tag{7.6}$$

where I is the identity matrix and $\widehat{\Sigma}$ the covariance matrix. The combination of values $(\lambda = 0, \eta = 0)$ gives LDA. PCA can be obtained with values $(\lambda = 1, \eta = 1)$. For other values, equation (7.6) provides a set of alternatives between PCA and LDA.

The main advantages of the method are: *i*)the matrix $(1-\lambda)S_B + \lambda\widehat{\Sigma}$ is full rank with values $\lambda \neq 0$, such that HDA overcomes one of the limitations of LDA that is restricted to projections of $L-1$ dimensions at most; and *ii*) values $\eta \neq 0$ give us a simple regularization scheme that avoids the singularity of $(1-\eta)S_W + \eta I$.

7.5 Ordinal Classification

In recent years, classifier ensembles have been studied and have developed considerably. In the majority of these methods, it is assumed that the class values are not ordered, however many problems exist in which the class values are ordered. For example, a person may be classified by age as a child, adolescent, adult or elderly person. It is easy to appreciate that these classes have an order in which the adolescent comes before the adult and the adult before the elderly person.

Multiclass classification methods are commonly used to work with data in which the class is ordered, but the information that relates to the order is simply not taken into account. However, in some cases this information that is habitually ignored could be used to improve the precision of the classifier [23] [1]. In this work, the method described in [23] is used. It is a classic method that is based on a transformation of data to convert an ordered k class problem to a k-1 binary classification problem. It may be thought of as an ensemble in which the acceptable transformations are firstly made to obtain the k-1 binary classification problems, and then a base classifier is used, which moves as a parameter to the ensemble, in order to resolve each of these k-1 binary problems. In the training stage, the k-1 new data sets are obtained. Each one of them is formed of all the instances and the value of their class has simply been modified. In the example of age, 3 data sets will be generated:

1. (target $>$ child) Instances labelled as child will be labelled as 0 and the rest as 1.

2. (target $>$ adolescent) Instances labelled as child and adolescent will be labelled as 0 and the rest as 1.

3. (target $>$ adult) Instances labelled as child, adolescent and adult will be labelled as 0 and the instances labelled as elderly will be labelled as 1.

In the following stage, a classifier is constructed for each one of the data sets that are generated. And finally, at the prediction stage, for each new instance, the probabilities are obtained of the k-1 base classifiers in the following way:

$$Pr(V_1) = 1 - Pr(Target > V_1)$$

$$Pr(V_i) = Pr(Target > V_{i-1}) - Pr(Target > V_i), 1 < i < k$$

$$Pr(V_k) = 1 - Pr(Target > V_{k-1})$$

And it returns the class with the highest probability.

7.6 Results and discussion

We compared Boosting Projections with the most commonly used types of ensembles and, we evaluated both the classical and the ordinal approach for each method:

1. Random Committee: An ensemble of randomizable base classifiers. Each base classifier is built using the same data but a different seed for the generation of randomness. The final predicted probabilities are simply the average of the probabilities generated by the individual base classifiers.

2. Bagging [8].

3. Boosting: AdaBoost.M1 [25] and Multiboost [53]. The variant with reweighting was used. For Multiboost the approximate number of subcommittees was 3 and 10.

4. Random Subspaces [32]: with two different configurations, with 50% and 75% of the original set of attributes.

5. Boosting Projections, using HDA projection with parameters $\lambda = 0.5$ and $\eta = 0.1$.

Base classifiers are neural networks, because they are the most widely used type of classifier in this type of problems, and decision trees, because they are efficient and unstable. The training and testing time and memory requirements of an ensemble are a direct function of the corresponding values of the base classifier. It is not practical to construct an ensemble of many classifiers if they are slow or need too much memory. Instability means that, with relatively small changes in the data set, it is possible to obtain very different classifiers, which is desirable because successful ensembles need diverse base classifiers [38].

Table 7.4: Accuracy for the different base classifiers: (a) Multilayer perceptron with default training parameters, (b) Multilayer perceptron with optimal training parameters, (c) J48 decision tree with prunning, (d) J48 decision tree without prunning. (A) Ordinal-Multilayer perceptron with default training parameters, (B) Ordinal-Multilayer perceptron with optimal training parameters, (C) Ordinal-J48 decision tree with prunning, (D) Ordinal-J48 decision tree without prunning

MLP		Decision trees		ORD-MLP		ORD-Decision trees	
Default (a)	Optimal (b)	Pruned (c)	Unpruned (d)	Default(A)	Optimal (B)	Pruned (C)	Unpruned (D)
69.976	71.183	68.758	68.236	68.868	70.870	69.604	68.488

The evaluated method, Boosting Projections, was implemented in the Weka library [31][1] and the other base classifiers and ensembles are also taken from this library.

Both the base classifiers as well as the ensembles were used by themselves and as base classifiers within the *Ordinal Class Classifier*, which is the name given in Weka to the method that implements the ideas of [23] outlined in section 7.5.

The parameter for the base classifiers is the default training parameter for the classifier in Weka; in the case of MLP: learning rate=0.3, momentum=0.2, number of epochs=500, number of neurons in the hidden layer=(number of attributes+number of classes)/2. As was done in [11] for a similar industrial task (Roughness prediction for ball-end milling operations), in order to obtain a further reference value for comparison of the accuracy of the ensembles, we searched within 255 different training parameter combinations for the optimal neural network parameters for this data set: number of neurons in the hidden layer=18, learning rate=0.1, momentum=0.075 (although we do not use the optimal neural network as base classifiers in the ensembles). That same search procedure was performed to optimize the parameters of the neural network used within the Ordinal Class Classifier, finding the following optimal values: number of neurons in the hidden layer=18, learning rate=0.5, momentum=0.375.

The results for the base classifiers and the optimal MLP are shown in Table 7.4.

The size of the ensembles (including the Random Committee) was set to 50. All experiments were performed using 10×10 cross validation [21] (each prediction method is trained on nine tenths of the total data and tested on the remaining tenth. This process was repeated a further nine times until all 10 subsamples had been used once for training, and the whole process was repeated with 10 different partitions, in order to account for the variance between partitions).

Table 7.5 shows the results for the different ensembles.

Various conclusions may be drawn from the contents of the table. For this data set in particular, the ordered classification contributes no improvements in the performance of the classifiers. The variants of the methods that perform the classification without taking the order of classes into account outperform their equivalent methods with ordering on 15 occasions, 10 of which significantly. Many ensembles yield better results than the optimal neuronal network, among which all those marked with the symbol $\circ$ obtained significantly better precision than the optimal neuronal network in accordance with the paired t-test to a level of significance of 10%.

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Table 7.5: The following table presents the results of the different ensembles in terms of percentage precision. In the rows we can see the types of ensembles and in the columns, the different base classifiers: Multilayer Perceptron, Decision Trees and Unpruned Decision Trees. The data situated below columns headings of "Ordinal", correspond to the results of the Ordinal Classifier which takes as the base classifier the ensemble and the corresponding base classifier. All the results marked ∘ are significantly better than the optimal neuronal network. Equally, all the results marked • are significatively better than the best of the traditional ensembles (in this case Random SubSpaces 50%) the value of which is underlined. When comparing the classic and ordered versions of each method, $< y >$ is used to identify simple achievements and $\ll y \gg$ to identify significant achievements in accordance with the paired Student's *t* test. The best value appears in bold.

Type Ensemble	MLP		Decision trees		Decision trees (U)	
		Ordinal		Ordinal		Ordinal
Random Commite	71.634	$\gg$ 70.217				
Bagging	∘72.667	$\gg$ 71.824	70.929	$>$ 70.787	70.952	$>$ 70.667
RS 50	∘72.295	$<$ 72.414∘	72.110	$<$ 72.767∘	71.990	$<$ 72.274∘
RS 75	∘<u>73.520</u>	$\gg$ 71.916	69.678	$\ll$ 70.671	69.673	$<$ 70.340
Adaboost	70.931	$\gg$ 68.611	71.321	$\gg$ 70.539	71.414	$>$ 70.989
Multiboost C=3	71.588	$\gg$ 69.273	71.488	$>$ 70.971	71.089	$<$ 71.274
Multiboost C=10	∘72.508	$\gg$ 71.164	71.229	$<$ 71.607	71.578	$>$ 71.466
Boosting Projections	∘74.162	$\gg$ 72.180∘	•75.462	$\gg$ 73.976∘	•**75.534**	$\gg$ 73.947∘

It was noted that Boosting Projections achieves an accuracy that is superior to other traditional ensembles, especially when the base classifiers are decision trees. Using the results from the cross-validations and a paired *t*-test with 10% significance level, the accuracy of the Boosting Projections using trees as base classifiers significantly outperforms the accuracy of the best of the classical ensemble methods (highlighted in bold in the table).

7.7 Conclusions

Surface roughness has a strong influence on the performance of a finished part. It is a characteristic that is nowadays evaluated out-of-process when the defective part has been already completely machined and may no longer be removed from the production chain. If the machined parts are very large, such as the hub or the main frame of a wind turbine, poor surface quality will imply costly losses owing to the manufacture of defective components, a situation which may be avoided through improvements to data mining algorithms, leading to more accurate on-line process information.

A recent ensemble method, Boosting Projection, has been used for surface roughness prediction in a vertical milling machine. The method has been applied to high-torque, high-power milling operations; operations that are nowadays in great demand by wind turbine manufacturers. The data set was obtained from 141 experiments performed under real industrial conditions. The experiments consider 18 parameters that play a central role in surface generation under industrial conditions and include cutting tool properties,

machining parameters and cutting phenomena.

In this data set, Boosting Projection obtained better results than the classical ensemble methods. The best method was Boosting Projection with unpruned trees as the base classifier. Regarding to the use of ordinal classification, in this data set, the technique has not shown any advantage compare to the use of multiclass classification which does not take into account the order of the class labels.

Future work will focus on the application of Boosting Projections to other materials of industrial interest, especially aeronautical aluminium because of the immense number of face milling tasks involved in the manufacture of different structural aeronautical components, such as aeroplane ribs. Moreover, this model will be applied to optimize slightly different industrial problems such as the contouring of multicomponent plates for the aeronautical industry. In this paper, roughness prediction has been addressed as a classification problem because, as explained in the introduction, roughness quality in an industrial context is always given in terms of the discrete values defined in ISO standard 4288:1996 [37]. However, an interesting future line of work would also be to compare the results of this paper with the results of using ensembles of regressors. Instead of directly predicting the level of the standard, they would first predict the exact roughness value and would then use this value to determine the roughness level.

7.8 Acknowledgements

This investigation has been partially supported by the CENIT project MAGNO (Ref. 2008–1028) funded by the Spanish Ministry of Science and Innovation. The authors would especially like to thank Mr. Desiderio Sutil from Nicolas Correa S.A. for his kind-spirited and useful advice.

References

[1] A. Agresti. *Analysis of Ordinal Categorical Data*. Wiley Series in Probability and Statistics. Wiley, 2010. ISBN: 9780470082898. URL: http://books.google.es/books?id=VVIe4BPDR7kC.

[2] M. Arizmendi et al. “Effect of tool setting error on the topography of surfaces machined by peripheral milling”. In: *International Journal of Machine Tools and Manufacture* 49.1 (2009), pp. 36 –52. ISSN: 0890-6955. DOI: DOI:10.1016/j.ijmachtools.2008.08.004.

[3] E. Bauer and R. Kohavi. “An empirical comparison of voting classification algorithms: Bagging, boosting, and variants”. In: *Machine learning* 36.1 (1999), pp. 105–139.

[4] C. Beggan et al. “Using Acoustic Emission to Predict Surface Quality”. In: *The International Journal of Advanced Manufacturing Technology* 15 (10 1999), pp. 737–742. ISSN: 0268-3768.

[5] P. G. Benardos and G. C. Vosniakos. "Predicting surface roughness in machining: a review". In: *International Journal of Machine Tools and Manufacture* 43.8 (2003), pp. 833 –844. ISSN: 0890-6955. DOI: DOI:10.1016/S0890-6955(03)00059-2. URL: http://www.sciencedirect.com/science/article/B6V4B-48BKNN7-8/2/67f1ad26e976f978e073fb0c6ff513ef.

[6] PG Benardos and GC Vosniakos. "Prediction of surface roughness in CNC face milling using neural networks and Taguchi's design of experiments". In: *Robotics and Computer-Integrated Manufacturing* 18.5-6 (2002), pp. 343–354. ISSN: 0736-5845.

[7] S. Binsaeid et al. "Machine ensemble approach for simultaneous detection of transient and gradual abnormalities in end milling using multisensor fusion". In: *Journal of Materials Processing Technology* 209.10 (2009), pp. 4728 –4738. ISSN: 0924-0136. DOI: DOI:10.1016/j.jmatprotec.2008.11.038. URL: http://www.sciencedirect.com/science/article/B6TGJ-4V34D5P-4/2/73586df93b0247b4275db4288e1b1a3e.

[8] L. Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[9] M. Brezocnik and M. Kovacic. "Integrated Genetic Programming and Genetic Algorithm Approach to Predict Surface Roughness". In: *Materials and Manufacturing Processes* 18.3 (2003), pp. 475–491. ISSN: 1042-6914.

[10] M. Brezocnik, M. Kovacic, and M. Ficko. "Prediction of surface roughness with genetic programming". In: *Journal of Materials Processing Technology* 157-158 (Dec. 2004), pp. 28–36. ISSN: 0924-0136. URL: http://www.sciencedirect.com/science/article/B6TGJ-4DHXJR6-3/2/a7ca428ffbf6c10027da075b63009b2a.

[11] A. Bustillo et al. "Avoiding neural network fine tuning by using ensemble learning: application to ball-end milling operations". In: *The International Journal of Advanced Manufacturing Technology* 57 (5 2011). 10.1007/s00170-011-3300-z, pp. 521–532. ISSN: 0268-3768. URL: http://dx.doi.org/10.1007/s00170-011-3300-z.

[12] A. Bustillo et al. "Modelling of process parameters in laser polishing of steel components using ensembles of regression trees". In: *International Journal of Computer Integrated Manufacturing* 24.8 (2011), pp. 735–747. ISSN: 0951-192X,1362-3052. DOI: http://dx.doi.org/10.1080/0951192X.2011.574155.

[13] M. Chandrasekaran et al. "Application of soft computing techniques in machining performance prediction and optimization: a literature review". In: *The International Journal of Advanced Manufacturing Technology* 46.5 (Jan. 2010), pp. 445–464. URL: http://dx.doi.org/10.1007/s00170-009-2104-x.

[14] S. Cho, S. Binsaeid, and S. Asfour. "Design of multisensor fusion-based tool condition monitoring system in end milling". In: *The International Journal of Advanced Manufacturing Technology* 46.5 (Jan. 2010), pp. 681–694. URL: http://dx.doi.org/10.1007/s00170-009-2110-z.

[15] S. K. Choudhury and G. Bartarya. “Role of temperature and surface finish in predicting tool wear using neural network and design of experiments”. In: *International Journal of Machine Tools and Manufacture* 43.7 (2003), pp. 747 – 753. ISSN: 0890-6955. DOI: DOI : 10 . 1016 / S0890 - 6955(02) 00166 - 9. URL: http : / / www . sciencedirect . com / science / article / B6V4B - 484V7NB - 1/2/1f70e50137375b665510badcc773d63a.

[16] M. Correa, C. Bielza, and J. Pamies-Teixeira. “Comparison of bayesian networks and artificial neural networks for quality detection in a machining process”. In: *Expert Systems with Applications* 36.3 (2009), pp. 7270–7279. DOI: http://dx.doi.org/10.1016/j.eswa.2008.09.024.

[17] M. Correa et al. “A Bayesian network model for surface roughness prediction in the machining process”. In: *Intern. J. Syst. Sci.* 39.12 (2008), pp. 1181–1192. DOI: http://dx.doi.org/10.1080/00207720802344683.

[18] V. G. Dhokia et al. “An intelligent approach for the prediction of surface roughness in ball-end machining of polypropylene”. In: *Robotics and Computer-Integrated Manufacturing* 24.6 (2008). FAIM 2007, 17th International Conference on Flexible Automation and Intelligent Manufacturing, pp. 835 –842. ISSN: 0736-5845. DOI: DOI:10.1016/j.rcim.2008.03.019.

[19] T. Dietterich. “Ensemble methods in machine learning”. In: *Multiple classifier systems* (2000), pp. 1–15.

[20] T.G. Dietterich. “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization”. In: *Machine learning* 40.2 (2000), pp. 139–157. ISSN: 0885-6125.

[21] T.G. Dietterich. “Approximate statistical tests for comparing supervised classification learning algorithms”. In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[22] S. Dzeroski and B. Zenko. “Is Combining Classifiers with Stacking Better than Selecting the Best One?” In: *Machine Learning* 54.3 (2004), pp. 255–273.

[23] E. Frank and M. Hall. “A simple approach to ordinal classification”. In: *Machine Learning: ECML 2001* (2001), pp. 145–156.

[24] Y. Freund and R.E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997). cited By (since 1996) 2237, pp. 119–139. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0031211090&partnerID=40&md5=6aa263ad916f3130742d61a6bf8337c3.

[25] Y. Freund and R.E. Schapire. “Experiments with a new boosting algorithm”. In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer. 1996, pp. 148–156.

[26] K. Fukunaga and JM Mantock. “Nonparametric discriminant analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5.6 (1983), pp. 671–678.

[27] C. García-Osorio and N. García-Pedrajas. “Constructing ensembles of classifiers using linear projections based on misclassified instances”. In: *16th European Symposium on Artificial Neural Networks (ESANN 2008)*. Ed. by Michel Verleysen. Bruges, Belgium: d-side publications, 2008, pp. 283–288. ISBN: 2-930307-08-0.

[28] N. Garcıa-Pedrajas, C. Garcıa-Osorio, and C. Fyfe. “Nonlinear boosting projections for ensemble construction”. In: *Journal of Machine Learning Research* 8 (2007), pp. 1–33.

[29] N. García-Pedrajas and C. García-Osorio. “Constructing ensembles of classifiers using supervised projection methods based on misclassified instances”. In: *Expert Systems with Applications* 38.1 (2011), pp. 343–359. ISSN: 0957-4174. DOI: DOI:10.1016/j.eswa.2010.06.072.

[30] M. P. Groover. *Fundamentals of modern manufacturing: materials, processes, and systems*. 3rd. ISBN: 0471744859; ISBN-13: 9780471744856, 978-0471744856; Binding: Hardcover; Number of Pages: 1022. John Wiley & Sons, 2006.

[31] Mark Hall et al. “The WEKA data mining software: an update”. In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[32] T.K. Ho. “The random subspace method for constructing decision forests”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[33] W. H. Ho et al. “Adaptive network-based fuzzy inference system for prediction of surface roughness in end milling process using hybrid Taguchi-genetic learning algorithm”. In: *Expert Systems with Applications* 36.2 (2009), pp. 3216–3222. ISSN: 0957-4174. DOI: http://dx.doi.org/10.1016/j.eswa.2008.01.051.

[34] A. Iqbal et al. “A fuzzy expert system for optimizing parameters and predicting performance measures in hard-milling process”. In: *Expert Systems with Applications* 32.4 (May 2007), pp. 1020–1027. ISSN: 0957-4174.

[35] F. Ismail et al. “Generation of Milled Surfaces Including Tool Dynamics and Wear”. In: *Journal of Engineering for Industry* 115.3 (1993), pp. 245–252. DOI: 10.1115/1.2901656. URL: http://link.aip.org/link/?MSE/115/245/1.

[36] *ISO-4287. Geometrical Product Specifications (GPS) — Surface texture: Profile method — Terms, definitions and surface texture parameters*. International Organization for Standardization. 1997.

[37] *ISO-4288. Geometrical Product Specifications (GPS): Rules and procedures for the assessment of surface texture*. International Organization for Standardization. 1996.

[38] L. I. Kuncheva. “Diversity in multiple classifier systems”. In: *Information Fusion* 6.1 (2005), pp. 3–4.

[39] L.I. Kuncheva. “Combining classifiers: Soft computing solutions”. In: *Pattern Recognition: From Classical to Modern Approaches* (2001), pp. 427–451.

[40] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[41] H.S. Lee et al. “Systematic finishing of dies and moulds”. In: *International Journal of Machine Tools and Manufacture* 46.9 (2006), pp. 1027 –1034. ISSN: 0890-6955. DOI: DOI:10.1016/j.ijmachtools.2005.07.049.

[42] S. P. Lo. "An adaptive-network based fuzzy inference system for prediction of workpiece surface roughness in end milling". In: *Journal of Materials Processing Technology* 142.3 (2003), pp. 665 –675. ISSN: 0924-0136. DOI: DOI:10.1016/S0924-0136(03)00687-3. URL: http://www.sciencedirect.com/science/article/B6TGJ-495VKPK-H/2/c3f09b7f079528786a234bd4986cc6c6.

[43] O. Maimon and L. Rokach, eds. *Data Mining and Knowledge Discovery Handbook, 2nd ed*. Springer, 2010. ISBN: 978-0-387-09822-7.

[44] M.E Martellotti. "An analysis of the milling process". In: *Transaction of ASME* 63 (1941), pp. 667–700.

[45] D. Montgomery and Y. Altintas. "Mechanism of Cutting Force and Surface Generation in Dynamic Milling". In: *Journal of Engineering for Industry* 113.2 (1991), pp. 160–168. DOI: 10.1115/1.2899673. URL: http://link.aip.org/link/?MSE/113/160/1.

[46] N. Oza and K. Tumer. "Classifier ensembles: Select real-world applications". In: *Information Fusion* 9.1 (2008), pp. 4–20. ISSN: 15662535.

[47] C. Prakasvudhisarn, S. Kunnapapdeelert, and P. Yenradee. "Optimal cutting condition determination for desired surface roughness in end milling". In: *The International Journal of Advanced Manufacturing Technology* 41.5 (Mar. 2009), pp. 440–451. URL: http://dx.doi.org/10.1007/s00170-008-1491-8.

[48] G. Quintana, J.de Ciurana, and J. Ribatallada. "Surface roughness generation and material removal rate in ball end milling operations". In: *Materials and Manufacturing Processes* 25.6 (2010), pp. 386–398. URL: http://www.informaworld.com/10.1080/15394450902996601.

[49] G. Quintana, M. Garcia-Romeu, and J. Ciurana. "Surface roughness monitoring application based on artificial neural networks for ball-end milling operations". In: *Journal of Intelligent Manufacturing* (2009). 10.1007/s10845-009-0323-5, pp. 1–11. ISSN: 0956-5515. URL: http://dx.doi.org/10.1007/s10845-009-0323-5.

[50] B. Samanta, W. Erevelles, and Y. Omurtag. "Prediction of workpiece surface roughness using soft computing". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 222.10 (Oct. 2008), pp. 1221–1232. URL: http://dx.doi.org/10.1243/09544054JEM1035.

[51] Q. Tian, J. Yu, and T. S. Huang. "Boosting Multiple Classifiers Constructed by Hybrid DiscriminantAnalysis". In: *Multiple Classifier Systems*. Ed. by N. C. Oza et al. Vol. 3541. Lecture Notes in Computer Science. Springer, 2005, pp. 42–52. ISBN: 3-540-26306-3.

[52] J. Vivancos et al. "Analysis of factors affecting the high-speed side milling of hardened die steels". In: *Journal of Materials Processing Technology* 162-163 (2005). AMPT/AMME05, pp. 696 –701. ISSN: 0924-0136. DOI: DOI:10.1016/j.jmatprotec.2005.02.155. URL: http://www.sciencedirect.com/science/article/B6TGJ-4FPN9W8-4/2/974eb1172f01767d744204e04c554b31.

[53] G.I. Webb. "Multiboosting: A technique for combining boosting and wagging". In: *Machine learning* 40.2 (2000), pp. 159–196.

Chapter 8

Imbalanced Learning Ensembles for Defect Detection in X-ray Images

Authors José-Francisco Díez-Pastor, Cesar García-Osorio, Víctor Barbero-García, Alan Blanco-Álamo

Type Conference

Published in The 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems. (IEA/AIE 2013), pages 654-663.

Year 2013

Abstract

This paper describes the process of detection of defects in metallic pieces through the analysis of X-ray images. The images used in this work are highly variable (several different pieces, different views, variability introduced by the inspection process such as positioning the piece). Because of this variability, the sliding window technique has been used, an approach based on data mining. Experiments have been carried out with various window sizes, several feature selection algorithms and different classification algorithms, with a special focus on learning unbalanced data sets. The results show that Bagging achieved significantly better results than decision trees by themselves or combined with SMOTE or Undersampling.

Index terms— Non Destructive testing, ensemble learning, X-ray, Bagging, Undersampling, SMOTE

8.1 Introduction

The inspection of defects is a very important task to ensure the quality of industrial processes. Quality control has become an essential prerequisite for companies to remain competitive. The Non-Destructive Testings [6](NDT) are among the most commonly used tests, because they do not destroy or alter the material on which they are applied. The Radiography is one of the oldest NDT methods but it remains the most widely used. The inspection performed by human operators have the advantage that can be adapted to many different situations, many of which have not been previously seen [28]. Unfortunately, human inspection is not as consistent as an automated process since it is a process that requires high concentration, besides it is affected by the occurrence of fatigue, different levels of skill, experience or ways of working of each operator. In an extreme case, the inspection of a casting by an operator may determine that a pore or bubble is large enough to cause the breakage of the casting and another operator may determine that the casting is correct. There is a need of objective automatic inspection systems. It is considered that there are two types of problems, the detection of defects (defect or non-defect) and the classification of defects (porosity, lack of penetration, etc.)[20], in this case it is the first.

8.2 Problem description and methodology

This paper describes the process of detecting defects in magnesium alloys castings. Magnesium alloys are approximately 60% lighter than aluminium casting alloys and 80% lighter than steel, in spite of this advantage its adoption is slowed due to the internal porosity in the magnesium casting components [26]. In this paper, we analize the problem of defect detection in the context of high variability images: pieces of different types, multiple views for each piece, variability between views of the same piece due to the manual process of positioning the piece in the X-ray inspection system and the mechanical imprecision of the positioning system. In this regard the images used in this work are very different from the images used in previous works, as seen in Figure 8.2. Along recent years, a large number of methods for automatic detection of defects in X-ray images have been developed, but many of them do not work properly when the variability in the images is too high. The defect detection methods can be classified into: a) methods based on the subtraction of a reference image (this reference image can be

obtained by applying specific image processing operations to the image where defects are being sought [17], obtaining the reference image can automatically be obtained from a set of images [26]), b) methods based on digital image processing: automatic thresholding [23, 27], Mathematical morphology [1], or watershed [30, 2]. The first approach was tested, obtaining a reference image from the median of a set of images of the same view, to mitigate the problem that the images are not taken from exactly the same prespective due to manual placement process and positioning system imprecision a stitching process was attempted, using the algorithm SURF [3], however the quality expected in the alignment was not achieved. The methods in the second approach were also considered, but these methods are highly dependent on the type of images and could not find one that worked well for processing all the pieces and views used in this work.

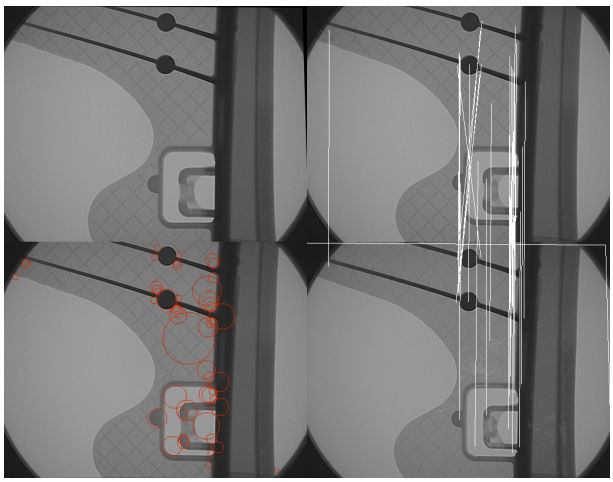


Figure 8.1: Top left: x-ray image of a piece, Botton left: points of interest found using SURF, Botton right: new image aligned with the first

In many computer vision problems, the solution has evolved from an approach based on image processing tailored to the specific characteristics of the problem at hand, to a more general approach called "Appearance-based method" in which learning methods are applied to a data set, transforming the detection problem into a binary classification problem. This approach allows to work with images of poor quality and can deal with more generic problems. For example, a system can detect vehicle license plates using image processing and heuristics techniques as [13] or detect faces using a skin color model [19] or it can detect license plates [11] or faces [29] transforming the problem into a classification problem. This Appearance-based approach is receiving considerable attention and has been applied to the

detection of defects in radiography images in [31] and [21].

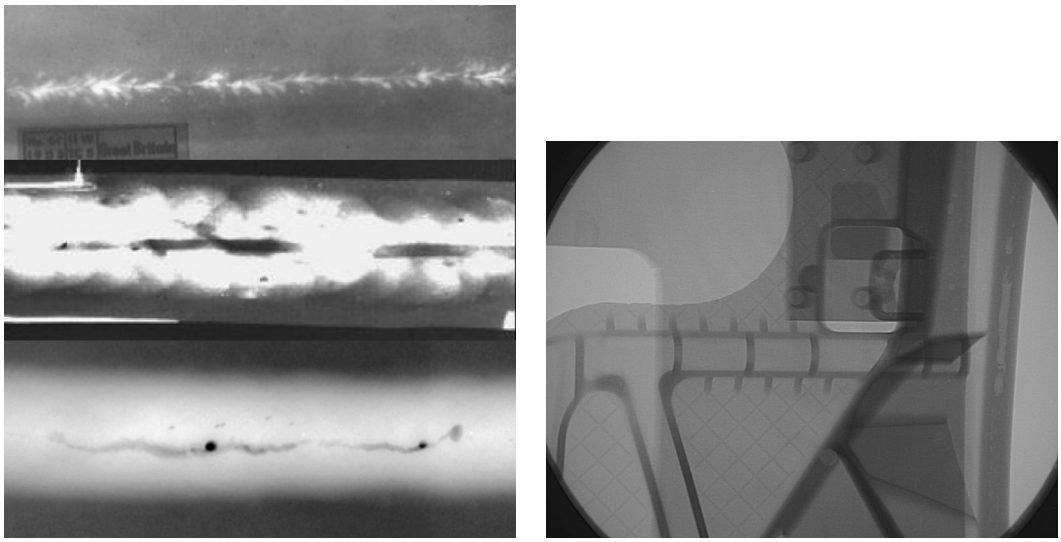


Figure 8.2: Differences between images: a) Images used in previous works [2, 4, 21] b) Images used in this work

8.2.1 Sliding Window

The sliding window technique consists of extracting features from a subimage, the window, of the image that is being procesed. This window is sistematically moved along the image. The features extracted together with a label, the presence or not of defects inside the window, are used to train a classifier. In the predictions stage the classifier is used to detect windows with defect inside them. Figure 8.3 describes how the technique works, there is an image I, which is scanned with a window of size $N \times N$, starting from an initial position it moves horizontally at intervals of $step\_h$ and vertically at intervals of $step\_v$. This window is passed to one or more feature extractors. The set of one or more feature extractors returns a vector of features for each of the windows. This technique can be applied on the raw grayscale image or on the result of applying some processing to this image. The window at coordinates x, y can be represented as the concatenation of the feature vector extracted from the original grayscale image and the feature vector extracted from images obtained from processing the original image. As in [21], in this work characteristics have been extracted both on the original image in grayscale and on the saliency Map [22] (an image transformation based on a biologically inspired attention system) of the grayscale image.

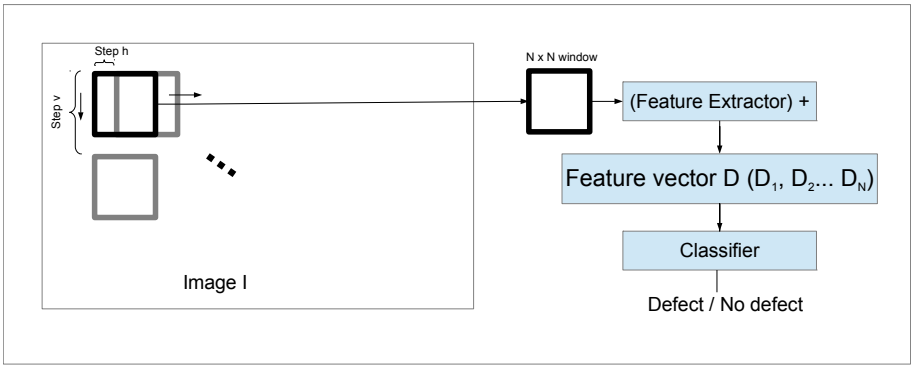


Figure 8.3: Sliding window

Type	Names	Number
Standard	1. mean, 2. standard desviation, 3. first derivative, 4. second derivative	4
Haralick	1. Angular Second Moment 2. Contrast, Correlation, 4. Sum of squares, 5. Inverse Difference Moment, 6. Sum Average, 7. Sum Entropy, 8. Sum Variance, 9. Entropy, 10. Difference Variance, 11. Difference Entropy, 12-13. Information Measures of Correlation,	13 features x 5 different pixel distances x 2 (mean and range of each vectors) =130
LBP	LBP(1) ... LBP(59)	59
		193 features per channel
		2 channel (normal and saliency) x 193 = 387 features

Table 8.1: Features used in the experiments.

8.2.2 Features

In artificial vision problems is unusual to train classifiers directly with the intensity values of the pixels. Typically, different types of features are extracted from these intensity values. These characteristics are often dependent on the images and type of problem. In this work, a subset of the characteristics used in [21] has been selected (see Table 8.1).

Standard features includes average and standard deviation of the intensities of the region and the average of the first and second derivative of the region. Haralick features [18], represent textural information, these features are computed from the co-occurrence matrix that represents second order texture information, these features are 14, but Maximal Correlation Coefficient was excluded due to its elevated computation time and because the system is desired to operate in near real-time. The local Binary Patterns

Dataset	FCBF						
	Total	Standard	Haralick	LBP	Standard(S)	Haralick(S)	LBP(S)
Size 8	4	1	1	1	1	0	0
Size 16	7	0	1	5	0	1	0
Size 24	4	0	0	1	0	1	2
Size 32	4	0	0	1	0	1	2
Size 40	5	1	0	2	0	1	1
Dataset	SVM Att Eval						
	Total	Standard	Haralick	LBP	Standard(S)	Haralick(S)	LBP(S)
Size 8	30	0	8	0	3	18	1
Size 16	30	0	11	0	2	15	2
Size 24	30	0	8	3	2	13	4
Size 32	30	0	11	2	2	13	2
Size 40	30	0	7	3	1	17	2
Dataset	CFS-Best First						
	Total	Standard	Haralick	LBP	Standard(S)	Haralick(S)	LBP(S)
Size 8	52	1	17	0	0	20	14
Size 16	39	0	15	0	1	13	10
Size 24	31	2	11	0	1	11	6
Size 32	33	1	13	1	0	11	7
Size 40	31	0	11	6	1	9	4

Table 8.2: Features selected characteristics of each type

texture descriptors are extracted from an histogram elaborated from the relationship between each pixel intensity value with its eight neighbors.

8.2.3 Atribute selection

Due to the large number of attributes, an attribute selection process is necessary to reduce the classifiers training time and for improving its performance. We tested three methods:

1. Correlation Feature Selection (CFS)[15] in conjunction with best first search. Evaluates a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred.

2. FCBF [32] (Fast Correlation-Based Filter Solution) is a feature selection method based on correlation measure, relevance and redundancy analysis especially oriented to sets of high dimensionality.

3. SVM attribute evaluator [14] evaluates attributes by using an SVM

classifier.

The first two methods return a subset of attributes. The third elaborates a ranking, so it is necessary to specify N, the number of selected attributes; $N = 30$ was used in the experiments. To perform the atribute selection, a dataset for each window size was elaborated, this dataset was obtained from 10 different images. In each image 500 samples were obtained through a random window. Half the windows including defects, the other half without any defect inside them. Table 8.2 show the total number of selected attributes and the number of selected attributes for each type of features and attribute selection algorithm. The suffix (S) indicates that these characteristics have been calculated on the saliency Map. It can be seen that FCBF is a very aggressive attribute selector that selects very small subsets. Attributes selected in a greater number according to the CFS and SVM Att EVal algorithms are those of Haralick. It is generally observed that the characteristics calculated on the saliency map are selected slightly more times.

8.2.4 Ensemble learning for inbalanced datasets

The class-imbalance problem occurs when there are many more instances of some classes than others [7]. X-ray images are unbalanced datasets because the proportion of regions with defects is much smaller than regions without them. This proportion must also exist in the training set, because it is not a good practice to train a classifier with a dataset with a very different proportion of classes that the proportion that will be found in the exploitation phase.

In unbalanced data sets, precision should not be used, since this measure provides the same value to the hits and misses, regardless of the distribution of classes. Commonly used measure of performance for imbalanced data is the Area Under the ROC (Receiver Operation Characteristic) curve [12]. There are several strategies for dealing with unbalanced sets, the most used is to preprocess the data set to reduce its imbalance, either by removing random instances of the majority class (Random undersampling) or by adding artificial instances of the majority class the technique is more representative of the latter is SMOTE [8]. In Machine Learning, the ensembles of classifiers are known to increase the performance of single classifiers by combining several of them. The ensemble of classifiers can be combined

Dataset	Non-defect Windows	Defect Windows	Imbalance Ratio
Size 8	205702	10618	19,3730
Size 16	201566	14754	13.6618
Size 24	197600	18720	10.5556
Size 32	193647	22673	8.5409
Size 40	189774	26546	7.1489

Table 8.3: Instances of the datasets

with the previous preprocessing techniques, to handle the problem of the imbalance better than any individual classifier.

8.2.5 Experimental setup

A total of 15 different data sets were elaborated using 5 different window sizes and 3 attribute selection algorithms for each of them. These data sets were obtained from 10 different images, using the sliding window procedure, with $step\_h$ y $step\_v = 4$. Table 8.3 shows the number of instances of each class and the imbalance ratio for each window size. And as previously mentioned, the tables 8.2 show the number of attributes depending on the attributes selection algorithm used.

Weka [16] was used for the experiments. J48, the Weka's re-implementation of C4.5 [25], was chosen as the base classifier in all ensembles. As recommended for imbalanced data [9], it was used without pruning and collapsing but with Laplace smoothing at the leaves. C4.5 with this options is called C4.4 [24]. We tested various ensembles methods such as Bagging [5], Bagging + undersampling (eliminating as many instances of the majority class as necessary to achieve an imbalance ratio IB = 1 and IB = 2), Bagging + SMOTE (generating an artificial number of instances equal to 100% and 200% of the size of the minority class). The size of the ensembles was 20. We also tested the performance of J48 by itself and in combination with undersampling and SMOTE, with the same settings as those used in the ensembles.

The Area Under the ROC results were obtained with a 5×2-fold cross validation [10]. The data set is halved in two folds. One fold is used for training and the other for testing, and then the roles of the folds are reversed. This process is repeated 5 times. The results are the averages of these 10 experiments. Cross validation was stratified.

8.2.6 Results

The results of the area under the ROC for J48 and Bagging of J48, with the features selected by different feature selection methods and different window sizes are shown in Figure 8.4 It can be seen how the classifiers built with features extracted by CFS + Best First and SVM Att Eval perform significantly better than those built with features extracted by FCBS, which is understandable since FCBS is a very strict attribute selector which selects a very small number of attributes.

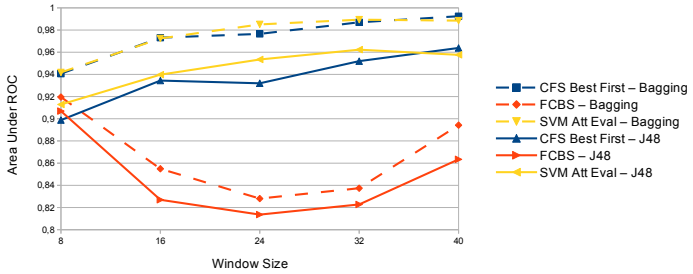


Figure 8.4: Area Under ROC vs. Window Size for Bagging and J48 classifiers

It is also noted that for window size 8, Bagging of J48 built with FCBS features get surprisingly good results despite using only 4 attributes, beating J48 built with CFS and SVM Att Eval features.

In general, it is observed that, as the window size increases the area under the ROC also increases, which can be explained in part because as seen in Table 8.3, as window size increases the imbalance between classes is reduced. Given that by increasing window size, the chances of a window covering a defect is greater. However increasing the window size leads to a less accurate defect detection.

Figure 8.5 show the difference (improvement or deterioration) between the area under ROC obtained by J48 with respect to that obtained by Undersampling + J48 (A and B), SMOTE (C and D), Bagging (E), Bagging + Undersampling (F and G) and Bagging + SMOTE (H and I). Bagging of J48 is significantly better than J48 in each of the data sets. Contrary to what would be expected, neither SMOTE nor undersampling, J48 improve performance for virtually any of the combinations of window size and the set of attributes used. Combinations of Bagging + Undersampling and Bagging + SMOTE outperforms Bagging in some datasets, Bagging + smote 100% is in general the classifier which obtains better results, although the differences are not significant compared to Bagging.

The results of the predictions of the classifiers are combined as shown

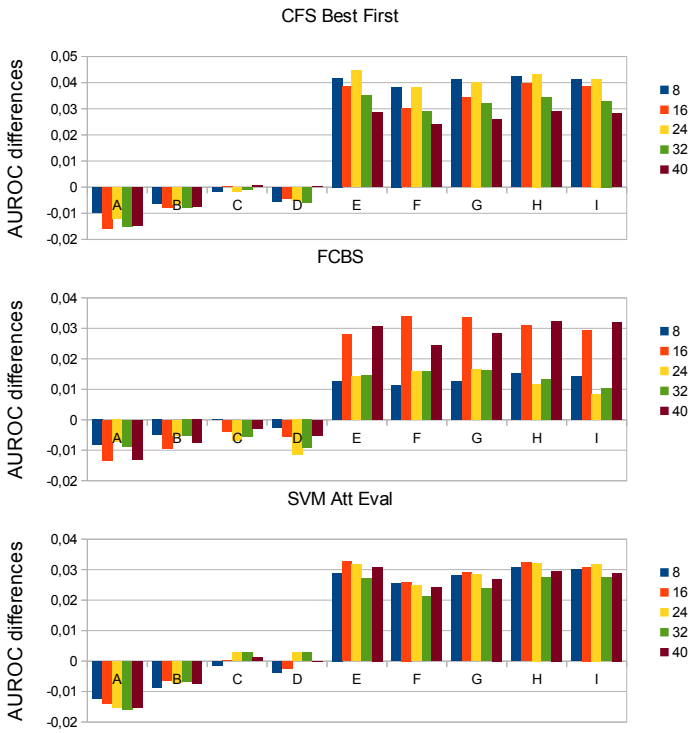


Figure 8.5: Area under ROC Difference of between J48 and the other classifiers

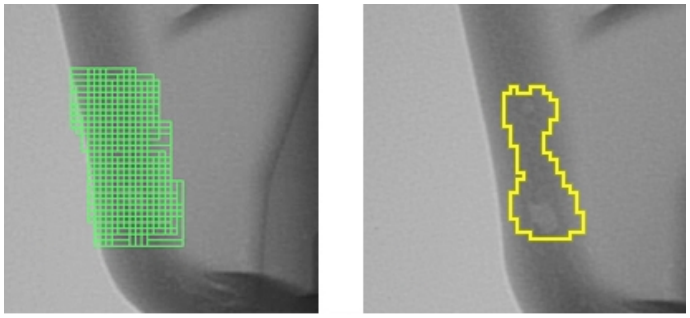


Figure 8.6: Results of the detection process

in Figure 8.6. When a pixel is covered by a window which the classifier predicts as defect, this window receives a vote. Following this, a threshold is set to determine the minimum number of votes required to consider that a pixel actually belongs to a defect. This threshold depends on the size of the window, the larger the window, the higher the threshold, and in the current implementation is not calculated automatically, instead, the user can adjust it, making the predicted region widens or fit to the true defect.

8.3 Conclusions and future lines

The results show that Bagging achieved significantly better results than decision trees by themselves or combined with SMOTE or undersampling. Contrary to expectations, no improvements are obtained by combining Bagging with preprocessing techniques as SMOTE or Undersampling.

Since the real-time operation is a relatively important constraint for the system, a possible future line would be to incorporate the computation time of each attribute to the attribute selection algorithm, in order to obtain subsets of attributes with a good trade-off between their usefulness in predicting the class and its speed to be calculated. Next, add the following stages of the inspection process: quantifying the characteristics of the defects, classifying the type of defect in terms of its features, obtaining defect statistics for each piece and finally using all previous data, designing a system that is capable of providing a measure of the quality of the piece from the analysis of this x-ray image.

References

[1] RS Anand, P. Kumar, et al. "Flaw detection in radiographic weld images using morphological approach". In: *NDT & E International* 39.1 (2006), pp. 29–33.

[2] RS Anand, P. Kumar, et al. "Flaw detection in radiographic weldment images using morphological watershed segmentation technique". In: *NDT & E International* 42.1 (2009), pp. 2–8.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features". In: *Computer Vision–ECCV 2006* (2006), pp. 404–417.

[4] S. Belaifa, M. Tridi, and N. Nacereddine. "Weld defect classification using EM algorithm for Gaussian mixture model". In: *SETIT Tunisia 2005*. 2005.

[5] L. Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[6] Louis Cartz. *Nondestructive Testing: Radiography, Ultrasonics, Liquid Penetrant, Magnetic Particle, Eddy Current*. Asm International, 1995. ISBN: 9780871705174. URL: http://books.google.es/books?id=pnk0AzwNSEsC.

[7] N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6.

[8] N.V. Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16.1 (2002), pp. 321–357.

[9] David A. Cieslak et al. "Hellinger distance decision trees are robust and skew-insensitive". In: *Data Min. Knowl. Discov.* 24.1 (Jan. 2012), pp. 136–158. ISSN: 1384-5810. DOI: 10.1007/s10618-011-0222-1.

[10] T.G. Dietterich. “Approximate statistical tests for comparing supervised classification learning algorithms”. In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[11] L. Dlagnekov. “License plate detection using adaboost”. In: *Computer Science and Engineering Department, San Diego* (2004).

[12] T. Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[13] Cesar García-Osorio et al. “License Plate Number Recognition - New Heuristics and a Comparative Study of Classifiers”. In: *ICINCO 2008, Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics, Robotics and Automation 1*. 2008, pp. 268–273.

[14] I. Guyon et al. “Gene selection for cancer classification using support vector machines”. In: *Machine learning* 46.1 (2002), pp. 389–422.

[15] M.A. Hall. “Correlation-based feature selection for machine learning”. PhD thesis. The University of Waikato, 1999.

[16] Mark Hall et al. “The WEKA data mining software: an update”. In: *SIGKDD Explor. Newsl.* 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[17] RF Hanke, U. Hassler, and K. Heil. “Fast automatic X-ray image processing by means of a new multistage filter for background modelling”. In: *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*. Vol. 1. IEEE. 1994, pp. 392–396.

[18] R.M. Haralick, K. Shanmugam, and I.H. Dinstein. “Textural features for image classification”. In: *Systems, Man and Cybernetics, IEEE Transactions on* 6 (1973), pp. 610–621.

[19] M.J. Jones and J.M. Rehg. “Statistical color models with application to skin detection”. In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* Vol. 1. IEEE. 1999.

[20] TW Liao. “Classification of welding flaw types with fuzzy expert systems”. In: *Expert Systems with Applications* 25.1 (2003), pp. 101–111.

[21] Domingo Mery. “Automated Detection of Welding Discontinuities without Segmentation”. In: *Materials Evaluation* June (2011), pp. 657–663. URL: http://web.ing.puc.cl/~dmery/Prints/ISI-Journals/2011-MatEval-Welding.pdf.

[22] S. Montabone and A. Soto. “Human detection using a mobile platform and novel features derived from a visual saliency mechanism”. In: *Image and Vision Computing* 28.3 (2010), pp. 391–402.

[23] H.F. Ng. “Automatic thresholding for defect detection”. In: *Pattern recognition letters* 27.14 (2006), pp. 1644–1649.

[24] F. Provost and P. Domingos. “Tree induction for probability-based ranking”. In: *Machine Learning* 52.3 (2003), pp. 199–215.

[25] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[26] V. Rebuffel, S.C. Sood, and B. Blakeley. "Defect Detection Method in Digital Radiography for Porosity in Magnesium Casting". In: *Materials Evaluation*. ECNDT 2006. 2006.

[27] T. Saravanan et al. "Segmentation of defects from radiography images by the histogram concavity threshold method". In: *Insight-Non-Destructive Testing and Condition Monitoring* 49.10 (2007), pp. 578–584.

[28] F.W. Spencer. *Visual Inspection Research Project Report on Benchmark Inspections*. Tech. rep. Office of Aviation Research Washington, D.C. 20591: U.S. Department of Transportation, Federal Aviation Administration,Washington, DC, 1996.

[29] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.

[30] M. Wang and L. CHAI. "Application of an improved watershed algorithm in welding image segmentation". In: *Transactions China Welding Institution* 28.7 (2007), p. 13.

[31] Y. Wang et al. "Detection of line weld defects based on multiple thresholds and support vector machine". In: *NDT & E International* 41.7 (2008), pp. 517–524.

[32] L. Yu and H. Liu. "Feature selection for high-dimensional data: A fast correlation-based filter solution". In: *Machine Learning-International Workshop Then Conference*. Vol. 20. 2. 2003, pp. 856–863.

Chapter 9

Segmentación de defectos en piezas de fundido usando umbrales adaptativos y ensembles

Authors José-Francisco Díez-Pastor, Álvar Arnaiz-González, Cesar García-Osorio, Juan J. Rodriguez

Type Conference

Published in XVII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2014), pages 345-349

Year 2014

Abstract

La inspección de imágenes de rayos X es uno de los métodos más utilizados para la detección de poros y burbujas de aire en piezas de fundido. En este artículo de describe el proceso de localización y segmentación de este tipo de defectos en imágenes que presentan una gran variabilidad: múltiples piezas con formas complejas y varias perspectivas para cada una de ellas. Es un proceso dividido en dos etapas. La primera de ellas identifica regiones susceptibles de ser un defecto y la segunda clasifica éstas regiones en defecto/no defecto, mediante la extracción de características morfológicas y de textura. El proceso ha sido validado en un estudio experimental con imágenes reales, evaluando la semejanza entre el «ground truth» o máscara de defectos reales y la imagen de defectos predicha, obteniendo buenos resultados en cuanto a precisión y tiempo.

Index terms— Test No-Destructivos, rayos X, segmentación de imágenes, PSO, umbrales adaptativos, rotation forest.

9.1 Introducción

Este artículo describe el proceso de detección y segmentación de defectos en piezas de fundido de magnesio. El magnesio es más ligero que el acero y el aluminio, pero puede presentar porosidad interna en los fundidos [17]. El control de calidad es imprescindible para que una empresa permanezca competitiva y la radiografía es uno de los TND (Test No-Destructivos) más ampliamente utilizados. En el proceso de inspección de imágenes de rayos X, habitualmente un operador humano examina la calidad, pero desafortunadamente este proceso es caro e inconsistente, los resultados pueden variar según la fatiga, la experiencia y la subjetividad de cada operador. Existe una necesidad de sistemas de inspección automáticos y objetivos.

A lo largo de los últimos años se han desarrollado una gran cantidad de métodos para la detección automática de defectos en imágenes de rayos X.

En el caso del análisis de imágenes de rayos X, al igual que en otros problemas de visión artificial, se pueden clasificar los métodos en:

a) Métodos dependientes del problema basados en procesamiento de imágenes, son soluciones a medida para un problema especifico, por lo que no funcionan correctamente cuando la variabilidad en las imágenes es muy grande, como es el caso. Utilizan técnicas muy variadas como por ejemplo: Sustracción de una imagen de referencia obtenida a partir de aplicar procesamientos a la imagen actual [10] u obtenida a partir de un conjunto de imágenes[17], Binarización automática [15, 21], operaciones morfológicas [1], *watershed* [23, 2].

b) Métodos basados en minería de datos utilizando la técnica de ventana deslizante [14, 24], consistente en recorrer la imagen usando una ventana rectangular, extrayendo para cada posición una serie de características dentro de dicha ventana. En la fase de entrenamiento se extraen todas las ventanas y se etiquetan como positivas si cubren un defecto y como negativas en caso contrario. En la etapa de predicción se combinan los resultados usando un umbral, dado que cada pixel esta cubierto por múltiples ventanas. Esta técnica de la ventana deslizante ha demostrado robustez frente a distintos tipos de imágenes y ha sido utilizada en [6] con resultados satisfactorios con las mismas imágenes que las utilizadas en este trabajo, pero es un proceso muy lento, dado que involucra la evaluación de miles de instancias para cada imagen.

En este artículo se presenta un método que es combina ambos enfoques: la simplicidad y velocidad de los métodos basados en procesamiento de imágenes y la robustez frente a la variabilidad de los métodos basados en minería de datos.

9.2 Descripción del método

Visualmente un defecto en una imagen de radiografía se puede definir como una región que muestra una gran disimilaridad con su vecindad. El método propuesto está compuesto por dos etapas, en la primera de ellas se detectan todas aquellas regiones candidatas de ser un defecto mediante del uso de algoritmos de umbrales locales. Debido a la gran complejidad y variabilidad de las imágenes, en esta etapa se detectan gran cantidad de elementos como bordes o ruido que no son defectos. En la segunda se realiza la extracción de características y clasificación de estas regiones candidatas en defectos y no defectos.

9.2.1 Umbrales locales para la detección de regiones candidatas

Los métodos de binarización pueden ser agrupados en dos categorías: Binarización con umbrales globales y con umbrales locales. Los métodos de binarización global calculan un único umbral para toda la imagen, son métodos rápidos y obtienen buenos resultados en gran cantidad de problemas. Sin embargo existen casos en los que debido a la complejidad de la imagen, su degradación, la presencia de ruido etc los métodos globales no funcionan correctamente. Como solución a este problema se han propuesto los métodos locales, en los que se calcula un umbral diferente para cada pixel usando información de los pixels que forman una vecindad, en función de un determinado radio [20].

Existen varios algoritmos de binarización local, que además del radio pueden tener uno o dos parámetros. Los métodos probados en este trabajo han sido Bernsen [3], Mean [8], Niblack [16] y Sauvola [22], todos estos métodos tiene un tamaño de radio y uno o dos parámetros adicionales.

La binarización de una radiografía de una pieza de fundido da como resultado una imagen en blanco y negro, con las zonas susceptibles de ser defectos en blanco (Figura 9.1 b). Para separar regiones que hayan quedado pegadas a otras adyacentes y para eliminar ruido se ha utilizado la operación morfológica *Closing*, una dilatación seguida de una erosión [12].

La selección del método y sus parámetros se hizo con optimización por enjambre de partículas o PSO (Particle Swarm Optimization) [7], la función de ajuste a maximizar fue la precisión $\frac{TP}{TP+FP}$, donde TP son los pixels marcados como defecto en el «ground truth» y también blancos en la imagen binarizada y FP los blancos en la imagen binarizada pero no en el «ground truth».

El resultado de esta primera etapa es una imagen con una gran cantidad de regiones señaladas como potenciales defectos (Figura 9.1 c), a continuación se utilizará un proceso de extracción de características y de clasificación para seleccionar entre estas regiones aquellas que efectivamente son un defecto.

9.2.2 Clasificación de regiones candidatas en defecto/no-defecto

En esta etapa se va a crear una instancia para cada región candidata. En los sistemas de visión artificial es necesario obtener un conjunto de características con los que trabajar, dado que no es una buena práctica utilizar directamente los valores de los pixels, dado los valores de los pixels no describen texturas ni otras propiedades de las imágenes. En este trabajo se han utilizado las siguientes características:

- Básicas: Este grupo incluye la media y desviación de las intensidades de la región y las medias de la primera y segunda derivada de la región.

- Haralick: Características de textura. Son 14 características calculadas a partir de la matriz de co-occurrencia [11]. En este trabajo no se usa el coeficiente de correlación máximo debido a que su alto coste computacional no lo hace apropiado para una aplicación que funcione en tiempo real. La matrix de co-ocurrencia tiene como parámetros una orientación y una distancia. En este trabajo se han usado 4 orientaciones 0^o, 45^o, 90^o y 135^o y 5 distancias (de 1 a 5). Con lo cual para cada una de las 13 características usadas existen 5 vectores, de los que se ha obtenido la media y el rango. En total se tienen 130 características de Haralick (13 x 5 x 2 = 130).

- Medidas geométricas y de tamaño: área, perímetro, Major, Minor (longitud del eje mayor y menor de la elipse que circunscribe a la región), altura y anchura de la región, Feret (4 características), redondez, circularidad, solidez y razón de aspecto. [18].

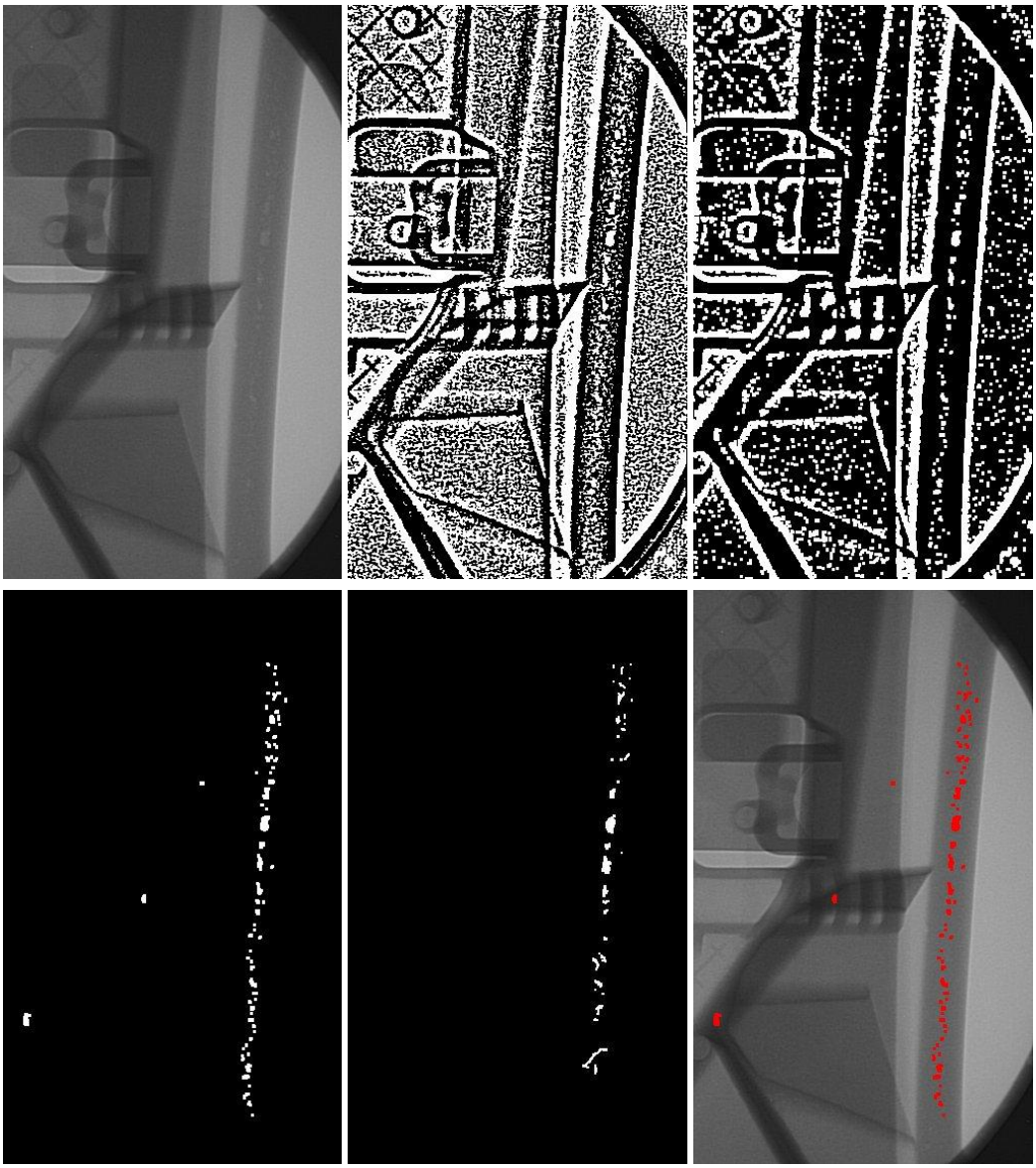


Figure 9.1: Proceso de detección de defectos. De arriba a abajo y de izquierda a derecha (a) Imagen original, (b) Binarización con umbrales adaptativos, (c) Closing de b, (d) predicción de defectos, (e) máscara o «ground truth» , (f) original con predicción de defectos superpuesta.

Al igual que en [14], las características básicas y de Haralick, se extraen de la imagen original en escala de grises y de su *Saliency Map* [14], una transformación basada en el sistema biológico de atención visual. Se han obtenido un total de 282 características: ((4 básicas + 130 Haralick) $\times$ 2 canales) + 14 medidas geométricas y de tamaño = 282 características.

Al trabajar con grandes conjuntos de características puede ocurrir el problema de que algunas de ellas sean irrelevantes, redundates o bien el clasificador no puede funcionar apropiadamente o sea lento de entrenar. Para ello se suelen usar algoritmos de selección de características. En este caso se ha utilizado el algoritmo *SVM attribute evaluator* [9], que evalúa los atributos usando un SVM para elegir los 50 mejores atributos de entre todos.

Hay otro aspecto que hay que tener en cuenta. Este es un problema desequilibrado, el problema de los conjuntos desequilibrados ocurre cuando existen muchas más instancias de una clase que de las demás [4].

Este es un problema desequilibrado desde 2 puntos de vista:

1. El número de regiones candidatas que realmente son defectos es mucho menor que el número de falsos positivos.

2. Considerando una imagen de radiografía como un conjunto de datos, el número de pixels pertenecientes a un defecto es mucho menor que el número de pixels correspondiente a zonas sin defecto.

Teniendo en cuenta el primer punto, se debe usar un clasificador especialmente preparado para lidiar con el problema del desequilibrio. Y teniendo en cuenta el segundo punto se tiene que usar una medida que tenga en cuenta el desequilibrio para evaluar el funcionamiento del sistema.

El clasificador y la metodología experimental utilizada se describen en la siguiente sección.

9.3 Metodología experimental y resultados

En esta sección se describe el proceso de validación experimental realizado. Se tiene un conjunto de diez imágenes en escala de grises con su correspondiente *ground truth* o máscara de defectos. Se va a evaluar la calidad final de la predicción de defectos en términos de la media geométrica, ya que esta medida es independiente de la distribución de ejemplos entre clases [13]. Considerando los pixels pertenecientes a un

Image nº	Media G	Image nº	Media G
Image 1	0.687	Image 6	0.255
Image 2	0.888	Image 7	0.721
Image 3	0.619	Image 8	0.641
Image 4	0.619	Image 9	0.438
Image 5	0.587	Image 10	0.510

Table 9.1: Media geométrica de la predicción para cada imagen

defecto como positivos y el resto como negativos, la media geométrica es $\sqrt{\text{Acierto positivos} \times \text{Acierto negativos}}$

Se ha realizado una validación cruzada a nivel de imagen: De las diez imágenes se toman 9 para extraer las regiones candidatas, construir el conjunto de datos, realizar la selección de atributos y entrenar el clasificador y se usa la imagen restante para evaluar la media geométrica comparando la predicción con la máscara. Este proceso se repite diez veces, utilizando para test una imagen distinta cada vez.

El clasificador utilizado ha sido Rotation Forest dado que en varios estudios experimentales con conjuntos desequilibrados ha resultado ser el mejor método [19]. El número de iteraciones fue 100. Como clasificador base se ha usado J48, la implementación de Weka de C4.5, se ha usado sin poda, sin colapsar pero con suavizado de Laplace, tal y como se recomienda para problemas desequilibrados [5].

Los resultados obtenidos aparecen en la Tabla 9.1.

9.4 Conclusiones y líneas futuras

Este artículo presenta un método para la detección de defectos en imágenes de fundido. Se basa en la detección de zonas candidatas mediante un proceso de binarización adaptativa y la posterior clasificación de estas zonas en defecto/no-defecto.

Si bien el método presentado es rápido y obtiene unos buenos resultados para la mayoría de las imágenes, existen varias líneas futuras. En la primera etapa del método se ha utilizado un PSO para la elección del algoritmo de binarización y sus parámetros. Esta etapa se podría sofisticar, en lugar de optimizar únicamente los parámetros de una operación, se podrían optimizar varias operaciones y su orden. Con la intención de obtener en esta etapa una imagen con más TP y menos FP, facilitando el trabajo posterior del clasificador y mejorando la calidad total del sistema.

References

[1] RS Anand, P. Kumar, et al. "Flaw detection in radiographic weld images using morphological approach". In: *NDT & E International* 39.1 (2006), pp. 29–33.

[2] RS Anand, P. Kumar, et al. "Flaw detection in radiographic weldment images using morphological watershed segmentation technique". In: *NDT & E International* 42.1 (2009), pp. 2–8.

[3] J. Bernsen. "Dynamic Thresholding of Grey level Images". In: *Proceedings of the 8th International Conference on Pattern Recognition*. 1986, pp. 1251–1255.

[4] N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6.

[5] David A. Cieslak et al. "Hellinger distance decision trees are robust and skew-insensitive". In: *Data Min. Knowl. Discov.* 24.1 (Jan. 2012), pp. 136–158. ISSN: 1384-5810. DOI: 10.1007/s10618-011-0222-1.

[6] J. F. Diez-Pastor et al. "Imbalanced Learning Ensembles for Defect Detection in X-Ray Images". In: *Proceedings of the 26th International Conference on Industrial, Engineering & other Applications of Applied Intelligent Systems (IEA/AIE)*. 2013, 654–663.

[7] Russell Eberhart and James Kennedy. "A new optimizer using particle swarm theory". In: *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE. 1995, pp. 39–43.

[8] R.C. Gonzalez and R.E. Woods. *Digital image processing*. Pearson/Prentice Hall, 2008. ISBN: 9780131687288. URL: http://books.google.es/books?id=8uGOnjRGEzoC.

[9] I. Guyon et al. "Gene selection for cancer classification using support vector machines". In: *Machine learning* 46.1 (2002), pp. 389–422.

[10] RF Hanke, U. Hassler, and K. Heil. "Fast automatic X-ray image processing by means of a new multistage filter for background modelling". In: *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*. Vol. 1. IEEE. 1994, pp. 392–396.

[11] R.M. Haralick, K. Shanmugam, and I.H. Dinstein. "Textural features for image classification". In: *Systems, Man and Cybernetics, IEEE Transactions on* 6 (1973), pp. 610–621.

[12] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. "Image analysis using mathematical morphology". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 4 (1987), pp. 532–550.

[13] Miroslav Kubat, Stan Matwin, et al. "Addressing the curse of imbalanced training sets: one-sided selection". In: *ICML*. Vol. 97. 1997, pp. 179–186.

[14] Domingo Mery. "Automated Detection of Welding Discontinuities without Segmentation". In: *Materials Evaluation* June (2011), pp. 657–663. URL: http://web.ing.puc.cl/~dmery/Prints/ISI-Journals/2011-MatEval-Welding.pdf.

[15] H.F. Ng. “Automatic thresholding for defect detection”. In: *Pattern recognition letters* 27.14 (2006), pp. 1644–1649.

[16] Wayne Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, 1985.

[17] V. Rebuffel, S.C. Sood, and B. Blakeley. “Defect Detection Method in Digital Radiography for Porosity in Magnesium Casting”. In: *Materials Evaluation*. ECNDT 2006. 2006.

[18] Séverine Rivollier, Johan Debayle, and J Pinoli. “Shape representation and analysis of 2D compact sets by shape diagrams”. In: *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*. IEEE. 2010, pp. 411–416.

[19] Juan José Rodríguez, José-Francisco Díez-Pastor, and Cesar García-Osorio. “Ensembles of Decision Trees for Imbalanced Data”. In: *MCS*. 2011, pp. 76–85.

[20] Bulent Sankur and Mehmet Sezgin. “Image thresholding techniques: A survey over categories”. In: *Pattern Recognition* 34.2 (2001), pp. 1573–1583.

[21] T. Saravanan et al. “Segmentation of defects from radiography images by the histogram concavity threshold method”. In: *Insight-Non-Destructive Testing and Condition Monitoring* 49.10 (2007), pp. 578–584.

[22] Jaakko Sauvola and Matti Pietikäinen. “Adaptive document image binarization”. In: *Pattern Recognition* 33.2 (2000), pp. 225–236.

[23] M. Wang and L. CHAI. “Application of an improved watershed algorithm in welding image segmentation”. In: *Transactions China Welding Institution* 28.7 (2007), p. 13.

[24] Y. Wang et al. “Detection of line weld defects based on multiple thresholds and support vector machine”. In: *NDT & E International* 41.7 (2008), pp. 517–524.

Bibliography

[1] D. Anil Kumar and V. Ravi. “Predicting credit card customer churn in banks using data mining”. In: *International Journal of Data Analysis Techniques and Strategies* 1.1 (2008), pp. 4–28.

[2] G.E. Batista, R.C. Prati, and M.C. Monard. “A study of the behavior of several methods for balancing machine learning training data”. In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 20–29.

[3] R. Batuwita and V. Palade. “microPred: effective classification of pre-miRNAs for human miRNA gene prediction”. In: *Bioinformatics* 25.8 (2009), pp. 989–995.

[4] E. Bauer and R. Kohavi. “An empirical comparison of voting classification algorithms: Bagging, boosting, and variants”. In: *Machine learning* 36.1 (1999), pp. 105–139.

[5] P. G. Benardos and G. C. Vosniakos. “Predicting surface roughness in machining: a review”. In: *International Journal of Machine Tools and Manufacture* 43.8 (2003), pp. 833 –844. ISSN: 0890-6955. DOI: DOI:10.1016/S0890-6955(03)00059-2. URL: http://www.sciencedirect.com/science/article/B6V4B-48BKNN7-8/2/67f1ad26e976f978e073fb0c6ff513ef.

[6] A. Blum and R. Rivest. “Training a 3-node neural network is NP-complete”. In: *Machine Learning: From Theory to Applications* (1993), pp. 9–28.

[7] L. Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[8] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32. ISSN: 0885-6125.

[9] Carla E Brodley and Mark A Friedl. “Identifying Mislabeled Training Data”. In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 131–167.

[10] G. Brown, J.L. Wyatt, and P. Tiňo. “Managing diversity in regression ensembles”. In: *The Journal of Machine Learning Research* 6 (2005), p. 1650.

[11] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. “Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD09)*. Vol. 5476. Lecture Notes on Computer Science. Springer-Verlag, 2009, pp. 475–482.

[12] Andrés Bustillo and Juan J. Rodríguez. "Online breakage detection of multitooth tools using classifier ensembles for imbalanced data". In: *International Journal of Systems Science* 45.12 (2014), pp. 2590–2602. DOI: 10.1080/00207721.2013.775378. eprint: http://dx.doi.org/10.1080/00207721.2013.775378. URL: http://dx.doi.org/10.1080/00207721.2013.775378.

[13] N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6.

[14] N.V. Chawla et al. "Learning ensembles from bites: A scalable and accurate approach". In: *The Journal of Machine Learning Research* 5 (2004), p. 451.

[15] N.V. Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16.1 (2002), pp. 321–357.

[16] N.V. Chawla et al. "SMOTEBoost: Improving prediction of the minority class in boosting". In: *7th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD 2003)*. 2003, pp. 107–119.

[17] J. Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine Learning Research* 7 (2006), p. 30.

[18] Matias Di Martino et al. "Improving Electric Fraud Detection using Class Imbalance Strategies." In: *ICPRAM (2)*. 2012, pp. 135–141.

[19] T. Dietterich. "Ensemble methods in machine learning". In: *Multiple classifier systems* (2000), pp. 1–15.

[20] Thomas G. Dietterich and Ghulum Bakiri. "Solving multiclass learning problems via error-correcting output codes". In: *Journal of Artificial Intelligence Research* 2 (1995), pp. 263–286.

[21] J. F. Diez-Pastor et al. "Imbalanced Learning Ensembles for Defect Detection in X-Ray Images". In: *Proceedings of the 26th International Conference on Industrial, Engineering & other Applications of Applied Intelligent Systems (IEA/AIE)*. 2013, 654–663.

[22] T. Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[23] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3 (1996), p. 37.

[24] Y. Freund and R. Schapire. "A desicion-theoretic generalization of on-line learning and an application to boosting". In: *Computational learning theory*. Springer. 1995, pp. 23–37.

[25] Y. Freund and R.E. Schapire. "Experiments with a new boosting algorithm". In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer. 1996, pp. 148–156.

[26] Yoav Freund and Robert E. Schapire. "Experiments with a New Boosting Algorithm". In: *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*. 1996, pp. 148–156.

[27] Vicente García, Ana Isabel Marqués, and Jose Salvador Sánchez. "Improving risk predictions by preprocessing imbalanced credit data". In: *Neural Information Processing*. Springer. 2012, pp. 68–75.

[28] Nicolás García-Pedrajas and César García-Osorio. "Constructing ensembles of classifiers using supervised projection methods based on misclassified instances". In: *Expert Systems with Applications* 38.1 (2011), pp. 343 –359. ISSN: 0957-4174.

[29] Nicolás García-Pedrajas and Aida de Haro-García. "Scaling up data mining algorithms: review and taxonomy". In: *Progress in Artificial Intelligence* 1.1 (2012), pp. 71–87.

[30] Nicolás García-Pedrajas et al. "Class imbalance methods for translation initiation site recognition in DNA sequences". In: *Knowl.-Based Syst.* 25.1 (2012), pp. 22–34.

[31] Nicolás García-Pedrajas et al. "Supervised subspace projections for constructing ensembles of classifiers". In: *Information Sciences* 193.0 (2012), pp. 1 –21. ISSN: 0020-0255.

[32] Guang-Gang Geng et al. "Boosting the performance of web spam detection with ensemble under-sampling classification". In: *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*. Vol. 4. IEEE. 2007, pp. 583–587.

[33] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely randomized trees". In: *Machine Learning* 63.1 (2006), pp. 3–42. ISSN: 0885-6125.

[34] K. Gowda and G. Krishna. "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (Corresp.)" In: *Information Theory, IEEE Transactions on* 25.4 (1979), pp. 488–490.

[35] R.W. Hamming. "Error detecting and error correcting codes". In: *Bell System Technical Journal* 29.2 (1950), pp. 147–160.

[36] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Trans. on Knowl. and Data Eng.* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. URL: http://dx.doi.org/10.1109/TKDE.2008.239.

[37] T.K. Ho. "Multiple classifier combination: Lessons and next steps". In: *Hybrid methods in pattern recognition* 74.1 (2002), pp. 171–198.

[38] T.K. Ho. "The random subspace method for constructing decision forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[39] Y. Hochberg. "A sharper Bonferroni procedure for multiple tests of significance". In: *Biometrika* 75 (1988), pp. 800–803.

[40] L. Hyafil and R.L. Rivest. "Constructing optimal binary decision trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), pp. 15–17.

[41] R.L. Iman and J.M. Davenport. "Approximations of the critical region of the fbietkan statistic". In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[42] Taeho Jo and Nathalie Japkowicz. "Class imbalances versus small disjuncts". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 40–49.

[43] M. Kubat and Matwin. "Addressing the Curse of Imbalanced Training Sets : One-Sided Selection". In: *Proceedings of the 14th International Conference on Machine Learning*. 1997, pp. 179–186.

[44] L.I. Kuncheva. "Combining classifiers: Soft computing solutions". In: *Pattern Recognition: From Classical to Modern Approaches* (2001), pp. 427–451.

[45] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[46] L.I. Kuncheva and J.J. Rodriguez. "Classifier ensembles with a random linear oracle". In: *IEEE Transactions on Knowledge and Data Engineering* 19.4 (2007), p. 500.

[47] T. Warren Liao. "Classification of weld flaws with imbalanced class data". In: *Expert Systems with Applications* 35.3 (2008), pp. 1041 –1052. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2007.08.044.

[48] Carlos López, Esperanza Manso, and Yania Crespo. "Evaluación de la eficiencia de métodos de identificación del defecto de diseño GodClass". In: *XVII Jornadas de Ingeniería del Software y Bases de Datos.Almeria. ISBN: 978-84-1587-28-9*. Universidad de Almería. 2012. URL: http://sistedes2012.ual.es/sistedes/jisbd.

[49] Victoria López et al. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* 250.0 (2013), pp. 113 –141. ISSN: 0020-0255. DOI: http://dx.doi.org/10.1016/j.ins.2013.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0020025513005124.

[50] Carlos López Nozal. "Detección de defectos de diseño mediante métricas de código". spa. Departamento de Informática. PhD thesis. 2012. URL: http://uvadoc.uva.es:80/handle/10324/2719 (visited on 05/03/2013).

[51] J. Maudes, J. J. Rodríguez, and C. García-Osorio. "Disturbing neighbors diversity for decision forests". In: *Applications of Supervised and Unsupervised Ensemble Methods*. Ed. by Oleg Okun and Giorgio Valentini. Vol. 245. Studies in Computational Intelligence. Springer, 2009, pp. 113–133. ISBN: 978-3-642-03998-0. URL: http://dx.doi.org/10.1007/978-3-642-03999-7_7.

[52] J. Maudes et al. "Random Feature Weights for Decision Tree Ensemble Construction". In: *Information Fusion* In Press, Accepted Manuscript (2010). ISSN: 1566-2535. DOI: DOI:10.1016/j.inffus.2010.11.004.

[53] Jesús Maudes et al. "Random feature weights for decision tree ensemble construction". In: *Information Fusion* 13.1 (2012), pp. 20–30.

[54] P. Melville and R.J. Mooney. "Constructing diverse classifier ensembles using artificial training examples". In: *Proceedings of the IJCAI*. Citeseer. 2003, pp. 505–510.

[55] M Molinara, MT Ricamato, and F Tortorella. “Facing imbalanced classes through aggregation of classifiers”. In: *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE. 2007, pp. 43–48.

[56] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. “Learning from imbalanced data in presence of noisy and borderline examples”. In: *Rough Sets and Current Trends in Computing*. Springer. 2010, pp. 158–167.

[57] S.J. Nowlan and G.E. Hinton. “Evaluation of adaptive mixtures of competing experts”. In: *Advances in neural information processing systems* 3 (1991), pp. 774–780.

[58] C. Phua, D. Alahakoon, and V. Lee. “Minority report in fraud detection: classification of skewed data”. In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 50–59.

[59] R. Polikar. “Ensemble based systems in decision making”. In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[60] J.R. Quinlan. “Bagging, boosting, and C4. 5”. In: *Proceedings of the National Conference on Artificial Intelligence*. 1996, pp. 725–730.

[61] Joaquin Quionero-Candela et al. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN: 0262170051, 9780262170055.

[62] JJ Rodriguez, LI Kuncheva, and CJ Alonso. “Rotation forest: A new classifier ensemble method”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (2006), pp. 1619–1630.

[63] L. Rokach. “Ensemble-based classifiers”. In: *Artificial Intelligence Review* 33.1 (2010), pp. 1–39.

[64] R.E. Schapire. “The strength of weak learnability”. In: *Machine learning* 5.2 (1990), pp. 197–227.

[65] Alon Schclar and Lior Rokach. “Random Projection Ensemble Classifiers”. In: *Enterprise Information Systems*. Ed. by Joaquim Filipe and Jose Cordeiro. Vol. 24. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2009, pp. 309–316. ISBN: 978-3-642-01346-1.

[66] A. Seewald and J. Furnkranz. “An evaluation of grading classifiers”. In: *Advances in Intelligent Data Analysis* (2001), pp. 115–124.

[67] C. Seiffert et al. “RUSBoost: A hybrid approach to alleviating class imbalance”. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40.1 (2010), pp. 185–197.

[68] Paolo Soda. “A multi-objective optimisation approach for class imbalance learning”. In: *Pattern Recognition* 44.8 (2011), pp. 1801–1810.

[69] Yang Song, Aleksander Kołcz, and C Lee Giles. “Better Naive Bayes classification for high-precision spam detection”. In: *Software: Practice and Experience* 39.11 (2009), pp. 1003–1024.

[70] Jerzy Stefanowski. “Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data”. In: *Emerging Paradigms in Machine Learning*. Springer, 2013, pp. 277–306.

[71] Jerzy Stefanowski and Szymon Wilk. “Selective pre-processing of imbalanced data for improving classification performance”. In: *Data Warehousing and Knowledge Discovery*. Springer, 2008, pp. 283–292.

[72] Giorgio Valentini and Thomas G Dietterich. “Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods”. In: *The Journal of Machine Learning Research* 5 (2004), pp. 725–775.

[73] C.J. Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[74] Sofia Visa and Anca Ralescu. “Issues in mining imbalanced data sets - a review paper”. In: *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*. 2005, pp. 67–73.

[75] S. Wang and X. Yao. “Diversity analysis on imbalanced data sets by using ensemble models”. In: *IEEE Symposium Series on Computational Intelligence and Data Mining(IEEE CIDM 2009)*. 2009, pp. 324–331.

[76] Shuo Wang and Xin Yao. “Relationships between diversity of classification ensembles and single-class performance measures”. In: *Knowledge and Data Engineering, IEEE Transactions on* 25.1 (2013), pp. 206–219.

[77] M. Wasikowski and Xue wen Chen. “Combating the Small Sample Class Imbalance Problem Using Feature Selection”. In: *Knowledge and Data Engineering, IEEE Transactions on* 22.10 (2010), pp. 1388–1400. ISSN: 1041-4347.

[78] Geoffrey I. Webb. “MultiBoosting: A Technique for Combining Boosting and Wagging”. In: *Machine Learning* 40.2 (2000), pp. 159–196.

[79] Gary M Weiss. “The impact of small disjuncts on classifier learning”. In: *Data Mining*. Springer. 2010, pp. 193–226.

[80] D.L. Wilson. “Asymptotic properties of nearest neighbor rules using edited data”. In: *Systems, Man and Cybernetics, IEEE Transactions on* 2.3 (1972), pp. 408–421.

[81] D.H. Wolpert. “Stacked generalization”. In: *Neural networks* 5.2 (1992), pp. 241–259.

[82] Hualong Yu et al. “Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets”. In: *Tsinghua Science and Technology* 17.6 (2012), pp. 666–673.

[83] C.X. Zhang and J.S. Zhang. “RotBoost: A technique for combining Rotation Forest and AdaBoost”. In: *Pattern Recognition Letters* 29.10 (2008), pp. 1524–1536.