Técnicas de preprocesamiento en datos anchos

Preprocessing techniques for wide data

UNIVERSIDAD
DE BURGOS

Ismael Ramos Pérez

Esta memoria ha sido presentada para optar al grado de doctor or la Universidad de Burgos

Programa de Doctorado «Tecnologías Industriales e Ingeniería Civil»

September 2024

The research carried out for the development of this doctoral thesis has been partially funded by the Junta de Castilla y León under project BU055P20 (JCyL/FEDER, UE), by the Spanish Ministry of Science and Innovation (projects PID2020-119894GB-I00 and TED 2021-129485B-C43) and by "la Caixa" Foundation, under agreement LCF/PR/PR18/51130007. The author has been recipient of the predoctoral grant by the Universidad de Burgos.

UNIVERSIDAD
DE BURGOS

Declaración

La tesis «Técnicas de preprocesamiento en datos anchos (wide data)», que presenta D. Ismael Ramos Pérez para optar al título de doctor, ha sido realizada dentro del programa «Tecnologías Industriales e Ingeniería Civil», 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 Dr. D. Jesús Manuel Maudes Raedo y Dr. D. Álvar Arnaiz González. Los directores autorizan la presentación del presente documento como memoria para optar al grado de Doctor por la Universidad de Burgos.

V. B. del Director:	V. B. del Director:	El doctorando:
Dr. D. Jesús Manuel Maudes Raedo	Dr. D. Álvar Arnaiz González	D. Ismael Ramos Pérez

Burgos, 14 de Junio del 2024

Acknowledgements

I enrolled on this PhD programme to learn more about data science and to improve my research skills. I would therefore like to offer my thanks here to those who have helped me to on that journey. Firstly, to my directors, Jesús and Álvar, as well as Juanjo and César, whose supervision has been of great assistance. I would also like to extend my gratitude to Lucy for welcoming me alongside Joselu during our stay in Wales. Last but not least, my thanks goes to all the co-authors and researchers who have helped me in various ways and at different points throughout the preparation of this thesis.

In addition, a special word of thanks to the world of free software and Open Access resources whose tools and resources have been of great utility in this process.

Abstract

In machine learning, wide datasets are common in several fields, especially biology and genomics. Datasets of that sort contain a large number of features compared to the number of their instances. Known as the 'curse of dimensionality', it represents a challenge for learning models, in that the number of dataset features makes it harder to find useful information that might solve a given machine-learning problem. The small number of instances causes overfitting of the data model, preventing it from uncovering a valid general solution for new, unseen data patterns. That same limitation can also imply an uneven number of instances for each label, biasing the model towards the most common label. A phenomenon that is known as class imbalance.

Data preprocessing techniques are effective at dealing with this problem. One of the most common approaches to deal with large datasets is feature selection, which identifies the most relevant features for the learning model and eliminates those deemed irrelevant. Feature reduction or feature extraction transforms the data into a lower-dimensional space, retaining the most relevant information. Data resampling balances the number of instances for each label. The low number of instances may be due to a lack of labels, so the extension of wide data to semi-supervised learning is an interesting solution to this problem.

This thesis is a compendium of three papers, each presented in its own chapter. In the first paper, the optimal order of feature selection and resampling algorithms is investigated. In the second paper, the performance of feature selection algorithms is compared with feature reduction algorithms. In the third paper, a systematic review of semi-supervised feature selection algorithms is presented and a new taxonomy is proposed.

In conclusion, the importance of data preprocessing techniques for wide data is demonstrated. Likewise, the effectiveness of both feature selection and feature reduction algorithms is underlined when used alongside resampling methods and guidance is provided on how to combine both. The feature reduction algorithms that can compete with feature selection algorithms are carefully illustrated. Finally, the review of different semi-supervised algorithms will also serve as a guide for researchers wishing to choose the most appropriate semi-supervised feature selection algorithm for a specific problem.

Table of contents

List of figures

List of tables

Part I
PhD dissertation

1 Introduction

In this chapter, the necessary concepts to understand the work developed in this thesis are introduced. First, the basics of machine learning, the research field of the thesis, are explained, after which the concept of wide data and the problems that can occur when working with that sort of data are outlined. Subsequently, the most common data-preprocessing techniques are presented, focusing on feature selection, feature reduction, and data resampling. The concept of semi-supervised learning is then introduced, explaining the different types of semi-supervised learning algorithms and the importance of their use with wide data. Finally, the most common model performance evaluation methods are presented.

1.1 Machine learning

Machine learning (ML) [21], a sub-field of computer science, is focused on developing algorithms which can learn the inherent patterns of datasets. These patterns can be used to help professional decision-making in various fields, such as predicting system behaviour, data classification, and process optimisation. The main advantage of ML algorithms is that they can be trained to perform tasks without being explicitly programmed to solve those tasks.

The most common data structure used in ML, the tabular format, is where several measures or features are collected for each instance. An instance is an object or event that is described by a set of features (*e.g.*, a person, a product, or a day). Those features are the individual measurable properties or characteristics of a phenomenon that is under observation. Features can be numerical (height, temperature, price) or categorical (weekday, colour, city). The tabular data, as can be seen in Table 1.1a, is stored in a matrix where each instance is represented by a row and each feature by a column. Images and signals can be stored for analysis in other types of data structures.

One feature of tabular data is the label, which is the target variable that the ML algorithm will attempt to predict. Depending on the variable type, the task is either called regression (numerical) or classification (nominal). The goal of the ML algorithm is to learn the relationship between the features and the label, so it can predict the label of new unseen instances.

Some of the most popular algorithms for regression and classification, respectively, are linear regression and Support Vector Machines (SVM) [29]. Linear regression generates a linear fit of all the data objects, minimising the quadratic distance between the line and the data points, as shown in Formula 1.1. Figure 1.1 shows an example of a fitted linear regression model, where the x-axis represents the feature and the y-axis represents the label to be predicted. On the other hand, the SVM is programmed to find the hyperplane that separates the data into two classes, maximising the margin between the classes, as shown in Figure 1.2.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.1}$$

1.2 Wide data

Wide data [101], sometimes referred to as high dimensionality data, have a large number of features compared to the number of instances; a difference that is visually represented in Table 1.1.

Data of this class are especially common in biology and genomics, some examples of their use being the analysis of type 2 diabetes [20] and the diagnosis of epilepsy using electroencephalography [121, 201]. Wide datasets are also used in engineering, for fault detection to diagnose engine system errors [66], and induction motor faults [6, 84]. High-dimensional data are also analysed in solar radiation estimation [86]. Examples are found

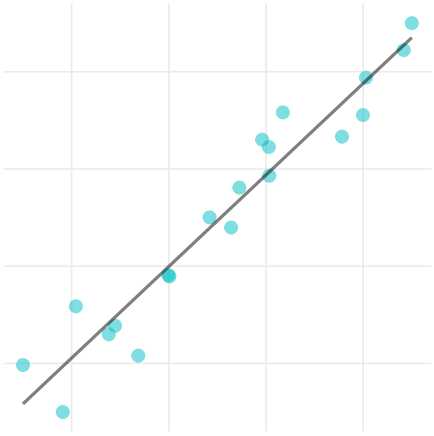


Fig. 1.1 Visual representation of linear regression.

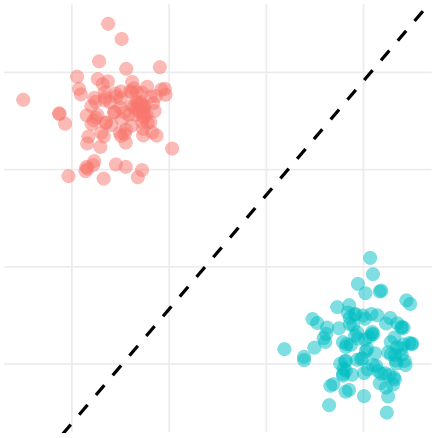


Fig. 1.2 Visual representation of an SVM.

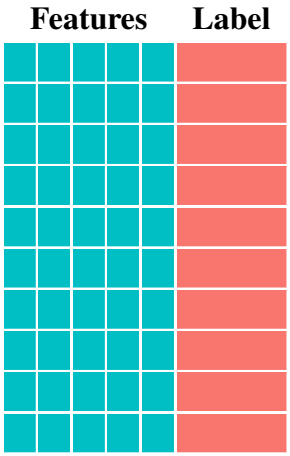
in other areas such as computer security where intrusion detection systems in network environments have to process multi-feature data [211].

Learning models commonly have difficulties when working with wide data. The number of features makes it harder to find the information needed to solve the problem, which is, as previously mentioned, known as the 'curse of dimensionality' [88]. The low number of instances can mean that the model will overfit the data, avoiding the problem of generalisation to new unseen instances. The number of instances for each label is very often not the same in data with a low number of instances, which introduces bias towards the most common label in model. This problem is known as class imbalance [55]. Three data-preprocessing techniques are frequently used to address this issue: feature selection (Section 1.3.1), feature reduction (Section 1.3.2), and resampling (Section 1.3.3).

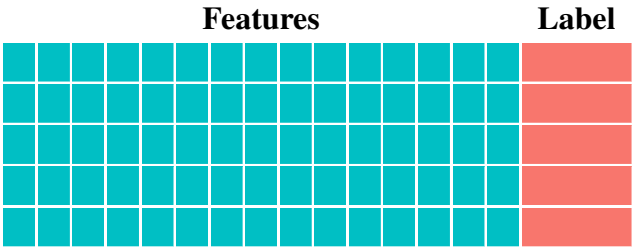
1.3 Data Preprocessing

Data preprocessing [60] is a crucial step in the ML pipeline with great potential for improving model performance. The goal of data preprocessing is to prepare the data, so as to facilitate the learning process of the ML algorithm. A typical data-preprocessing pipeline is shown in Figure 1.3 where the data undergo multiple transformations.

Preprocessing techniques can be divided into two main categories: data preparation and data reduction. Data-preparation techniques include initialising the data, so it can be used as input for the model. Data reduction techniques reduce the dimensionality of the data, making it easier for the model to learn its patterns.



a Normal dataset



b Wide dataset

Table 1.1 Comparison between normal and wide dataset sizes.

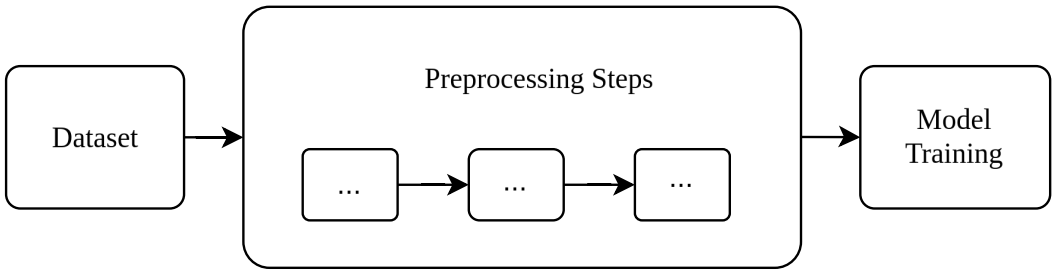


Fig. 1.3 Data preprocessing pipeline.

Some of the most common and useful data-reduction techniques for wide data are feature selection, feature reduction, and data resampling. Feature selection is the process of selecting the most relevant features for the model and discarding the irrelevant ones. Feature reduction transforms the data into a lower dimensionality space, keeping the most relevant information. Data resampling balances the number of instances for each label, reducing the chances of a biased model.

1.3.1 Feature selection

Feature selection is a preprocessing technique that selects the most relevant features, in order to improve model performance [157]. The features that are selected have the most information on the label and any irrelevant and redundant features are discarded.

The correlation between the feature and the label can be calculated, to determine the importance or relevance of a feature. The simplest correlation is the linear correlation or Pearson correlation coefficient. A coefficient that ranges between -1 and 1 where 1 is a perfect positive linear correlation, -1 is a perfect negative linear correlation, and 0 is a non-linear correlation. Figure 1.4 shows an example of several correlation grades between the feature on the horizontal axis and the label on the vertical axis.

However, the linear correlation is not the only kind of possible correlation, as is shown in Figure 1.5. In this case, the logarithmic and quadratic correlations that are shown could not be captured by the linear correlation coefficient. There are different approaches towards capturing different sorts of correlations, including multivariate correlation.

Feature selection algorithms are usually divided into three main categories: filter, wrapper, and embedded. Additionally, more advanced approaches include hybrid and ensemble methods [24]. Their descriptions are listed below and are summarized in Table 1.2.

- **Filter** methods [25] create a ranking of features usually using statistical measures. These methods are the fastest, but the features are selected as a global solution and in no case ever optimise model performance.

- **Wrapper** methods [92] search in the space of possible feature subsets for the one which provides the best model performance. These methods provide the best performance, but the time complexity is higher than the other types, as the performance of the model has to be trained and checked while exploring each subset . This optimisation method can also lead to model overfitting.

- **Embedded** methods [105] evaluate the relevance of the features during a model training. Unlike wrapper methods, the classifier used in embedded methods is not nec-

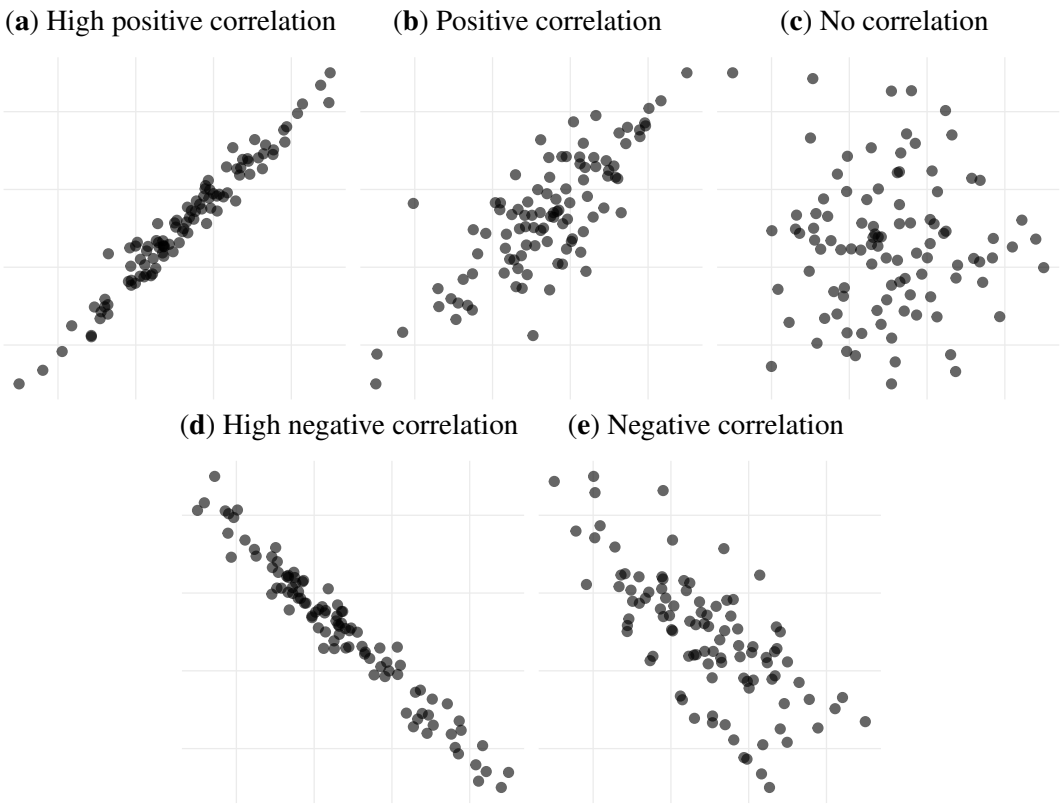


Fig. 1.4 Different grades of linear correlations.

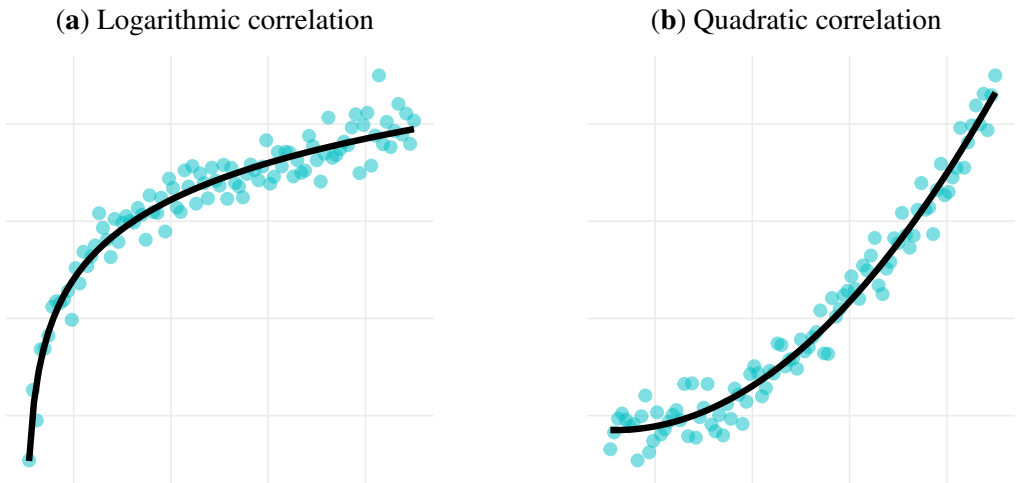


Fig. 1.5 Non linear correlations.

Table 1.2 Advantages and disadvantages of feature selection algorithms.

	Advantages	Disadvantages
Filter	• Generalisation • Speed	• Model-specific optimisation • Lower performance
Wrapper	• Model-specific optimisation • High performance	• Overfitting • Slow • Solution model-specific
Embedded	• Model-specific optimisation • Faster than wrappers • More general than wrappers	• Cannot choose the model freely • Slower than filters • Less general than filters
Hybrid	• Improve filter performance • Faster than wrappers • Improve wrappers generalisation	• Solution model-specific
Ensemble	• Low probability to overfit • Stable solutions	• Hard to interpret • Can be slow

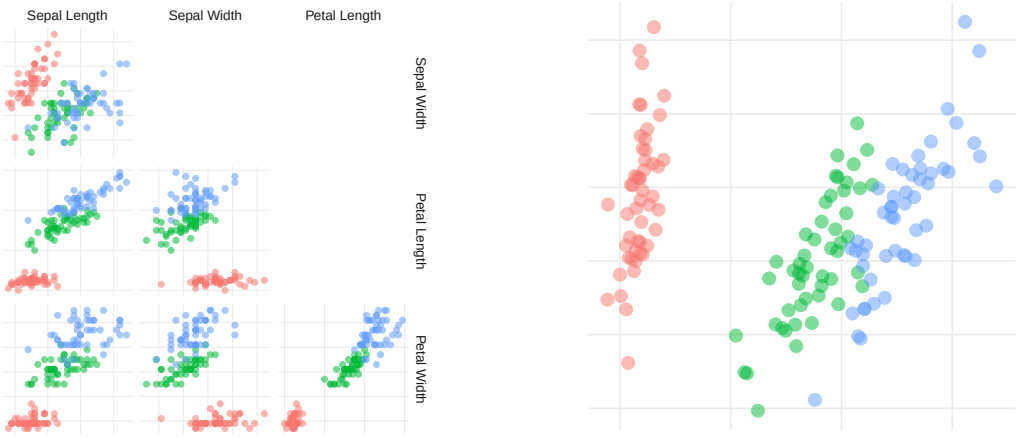
essarily the same as the one used for classification. These methods do not tend towards overfitting as much as wrapper-based methods, and their complexity is somewhere between filter and wrapper methods.

- **Hybrid** methods [158] combine a filter method and a wrapper method. They use the filter method to reduce the number of features and then, the wrapper method to select the best subset of features. This combination can improve the performance of the filter method and reduce the time complexity of the wrapper method.

- **Ensemble** methods [23] combine the results of several feature selection algorithms to provide a more stable solution. These methods can provide a more robust solution by reducing the likelihood of overfitting.

1.3.2 Feature reduction

Feature reduction [10], also known as feature extraction, is an alternative preprocessing technique that transforms the data into a lower dimensionality space, retaining the most relevant information. Rather than removing features as feature selection does, this technique creates a completely new representation of the data that can be used to train the model and to



(**a**) Iris pairs of features. (**b**) PCA projection.

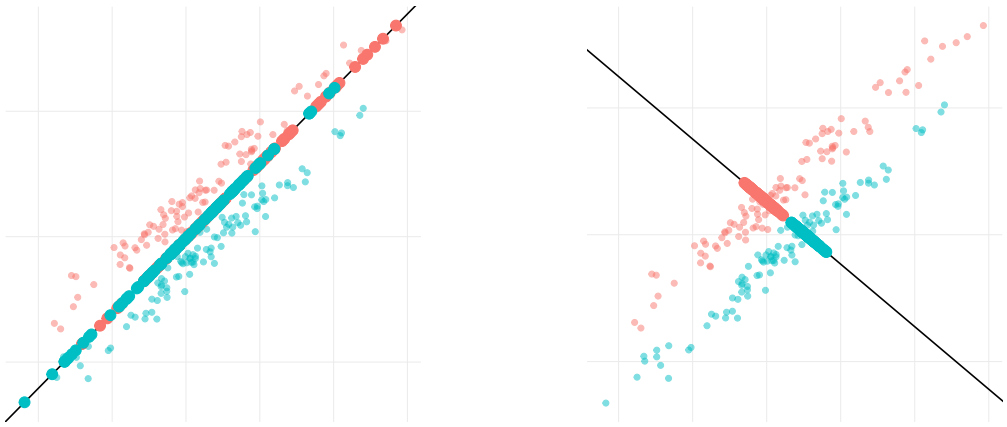
Fig. 1.6 Pairs of features from the Iris dataset plotted in a 2D space and their projection into a 2D space using PCA.

visualise the data. Feature reduction algorithms can be classified into either unsupervised and supervised, or linear and non-linear.

The most common feature reduction algorithm is Principal Component Analysis (PCA) [142]. An unsupervised algorithm that creates a new representation of the data by projecting the data into a lower dimensionality space, retaining the most variance. A typical visualisation of high dimensional data can be in a plot of all the feature pairs within a 2D space. An example of the 4 dimensional iris dataset is shown in Figure 1.6a. PCA can be used to project the data into a 2D space, retaining the most variance, as is shown in Figure 1.6b, and providing a simpler visualisation of the data.

However, since PCA is an unsupervised algorithm, it just projects the data, keeping the most variance, but not the most separability between the classes, which is the goal of the classification algorithms. Supervised feature reduction algorithms such as Linear Discriminant Analysis (LDA) [190] can be used instead. LDA finds the projection that maximises the separability between the classes. An example using artificial data is shown in Figure 1.7 where it can be observed that LDA reacts to the class distribution, unlike PCA.

The previously mentioned algorithms are linear, which means that the new representation of the data is a linear combination of the original features. However, if the manifold is non-linear, then linear algorithms will not be able to capture the most relevant information. The swiss roll dataset shown in Figure 1.8a is a typical example of a non-linear manifold where the data are distributed in a spiral shape. Non-linear algorithms such as Isomap [179] can be used to project the data into a lower dimensionality space. Isomap creates a neighbourhood graph between the instances and calculates the shortest path between them, finally showing



(**a**) PCA projections (**b**) LDA projection

Fig. 1.7 PCA and LDA projections of an artificial dataset. PCA is an unsupervised method, so its data projection is dissimilar to the 'proper' LDA projection.

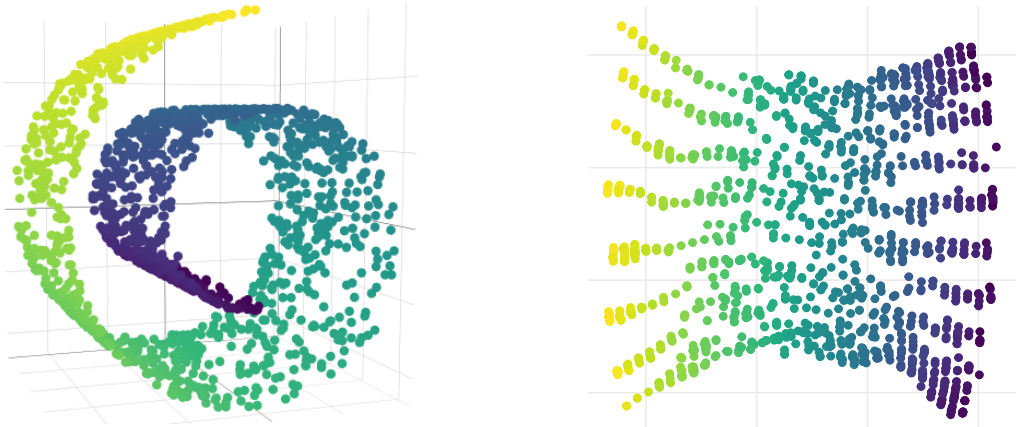
the new data through multidimensional scaling [97]. Figure 1.8a shows the data-projection capabilities of Isomap, through a swiss roll dataset projection into a 2D space.

1.3.3 Data resampling

Resampling is a preprocessing technique that equalises the number of instances for each label. As previously noted, the uneven class distribution affects the classification model, due to its inherent bias towards the most common class. There are two main resampling techniques, oversampling and undersampling:

- Undersampling decreases the number of instances for the majority class. The most common algorithms used for undersampling are Random Undersampling (RUS) [79] and Tomek links [187]. RUS randomly selects instances from the majority class to be removed, while Tomek links removes the instances that are close to the minority class.
- Oversampling, as opposed to undersampling, increases the number of instances for the minority class. The most common algorithms used for oversampling are Random Oversampling (ROS) [79] and Synthetic Minority Over-sampling Technique (SMOTE) [32]. ROS randomly selects instances from the minority class to be duplicated, while SMOTE creates new instances in between the minority class instances.

In Figure 1.9, an example of a dataset with class imbalance is shown where the minority class (red) is hidden by the majority class (blue). Alongside it, a representation of undersampling and oversampling techniques are shown where the majority class is decreased and the minority increased, respectively, so that the class boundaries become more visible.



(**a**) Swiss roll dataset

(**b**) Isomap projection

Fig. 1.8 The Swiss roll dataset is presented in a 3D scatter plot and projection to a 2D space using the non-linear feature reduction algorithm Isomap.

(**a**) Original data (**b**) Undersampling (**c**) Oversampling.

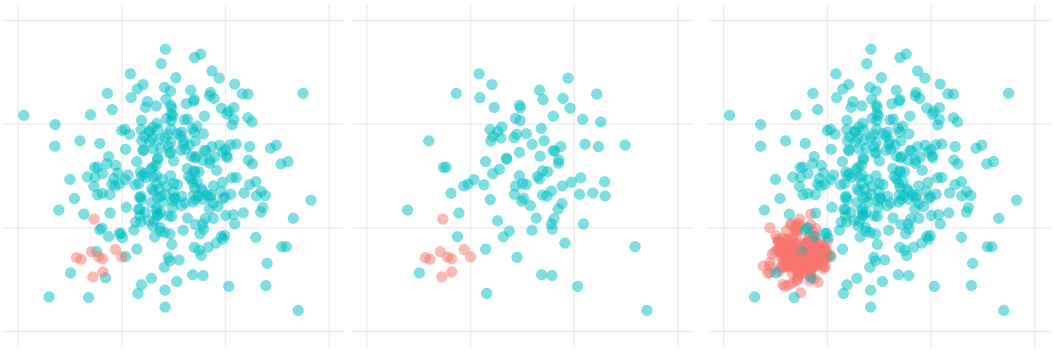
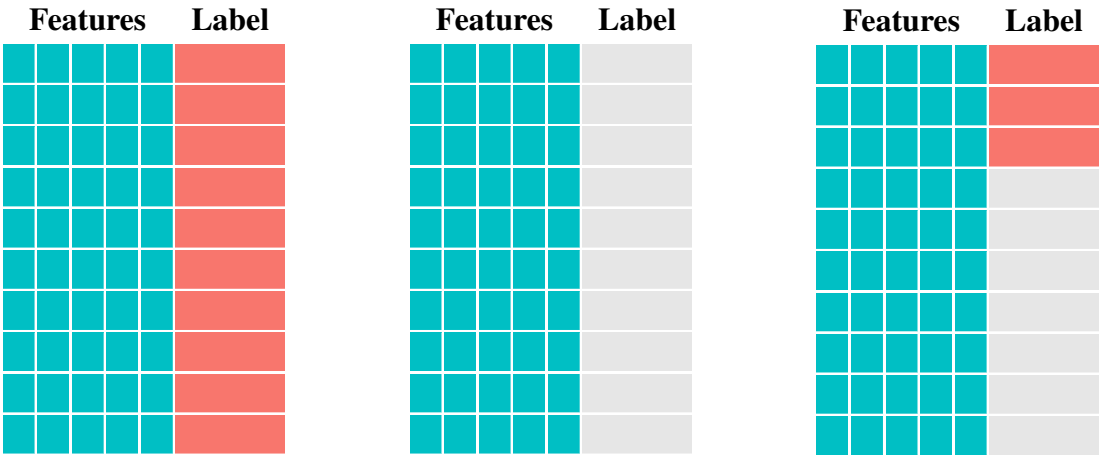


Fig. 1.9 Resampling strategies.



a Supervised b Unsupervised c Semi-supervised

Table 1.3 Comparison of the different datasets tasks.

1.4 Semi-supervised learning

In ML, the supervision defines the availability of the label in the training dataset. This results in three observable categories in Table 1.3: In supervised learning, the instances are fully labelled. In unsupervised learning, there are no labels, and in semi-supervised, only some of the instances are labelled.

Semi-supervised data is common to find in real-world problems where labelling the instances is an expensive, time-consuming process [189]. This lack of instances may be the cause of wide data. Taking advantage of the information retained in both labelled and unlabelled instances and mixing them to improve model performance is the goal of semi-supervised learning. Figure 1.10 shows the error of a linear model trained with a few labelled instances, compared with a better decision boundary obtained when the unlabelled instances are included.

The following data-related conditions are assumed in semi-supervised learning approaches: instances that are close in space have the same label (smoothness assumption), decision boundaries lie in regions of low sample density (low-density assumption), instances that belong to the same manifold belong to the same class (manifold assumption), and each multiple hidden manifold within the data describes the patterns of a class (clustering assumption).

These learning algorithms can be divided into two categories: transductive and inductive [189].

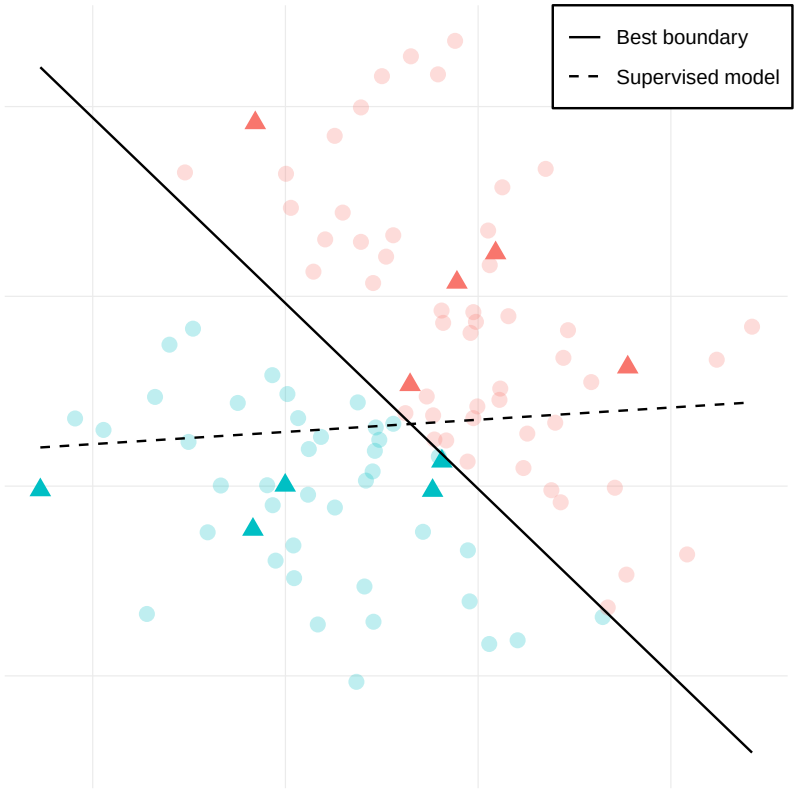


Fig. 1.10 Boundary comparison between a few labelled instances (triangular-shapes) and the optimal boundary.

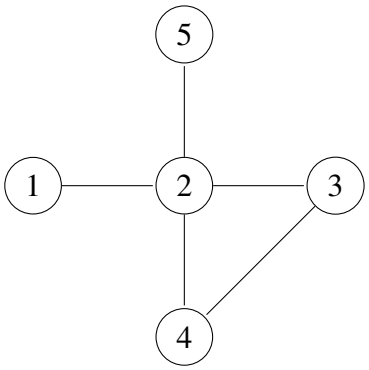
- **Transductive** algorithms, rather than training a model capable of predicting the class of unseen instances, only return the label of the unlabelled instances available at the training stage.
- **Inductive** algorithms train a model using the labelled and the unlabelled instances that can be used to predict the label of new unseen instances. Three main categories can be found within this category:
  - **Wrapper** methods, use one or more models trained with the labelled instances to pseudolabel the unlabelled instances with the most confident predictions. This process is repeated, increasing the number of labelled instances in each iteration. Some examples are self-training[217] and co-training [22].
  - **Unsupervised preprocessing** methods are used to extract useful information from the unlabelled instances as new features, or initial parameters to train a supervised model. PCA and autoencoders are some of the algorithms used [206].
  - **Intrinsically semi-supervised** methods incorporate the unlabelled instances in the training process or optimisation function. Algorithms based on SVM [112] are popular in this field.

1.4.1 Semi-supervised feature selection

In semi-supervised learning, feature selection can be performed using the information retained in both labelled and unlabelled instances [169]. The taxonomy of semi-supervised feature selection is similar to the supervised one, containing the same three main categories: filter, wrapper, and embedded. The most popular tools in this field are the Laplacian matrix and Pairwise constraints.

The Laplacian matrix properties and its eigenvectors are studied through spectral graph theory [43]. The Laplacian matrix is calculated using the adjacency matrix (A) and the degree matrix (D). The adjacency matrix shows the connections between the instances, while the degree matrix shows the number of connections that each instance has. The Laplacian matrix, as it can be seen in Figure 1.11, is calculated as $L = D - A$, showing the connections between the instances by a negative value in the matrix and the degree of each node in the diagonal. Some of the algorithms that use this matrix are the unsupervised feature selector Laplacian score [70] and the semi-supervised Laplacian score [51].

Employing pairwise constraints is another method for integrating information from both labelled and unlabelled instances. These constraints are a set of rules called must-link and cannot-link constraints. Must-link constraints specify that two instances should belong to the



	1	2	3	4	5
1	1	-1	0	0	0
2	-1	4	-1	-1	-1
3	0	-1	2	-1	0
4	0	-1	-1	2	0
5	0	-1	0	0	1

Fig. 1.11 Laplacian matrix of a graph.

same class, whereas cannot-link constraints dictate that two instances should not be in the same class. Pairwise constraints are employed to generate a similarity matrix, wherein the constraints determine the similarity between instances. This similarity matrix is then used to find the feature importance. Some of the algorithms that use this technique are the constraint score 1 (C1) [228] for supervised learning and the pairwise constraint score (C4) [85] for semi-supervised learning.

1.5 Model evaluation

Model evaluation is a critical step in machine learning. It is used to determine the performance of the model and to compare different models. No simple task in itself, it can be divided into performance metrics, validation type and statistical tests.

1.5.1 Performance metrics

Several metrics can be used to evaluate the performance of models. It is recommended to use more than one metric to avoid biased evaluations [125]. Most metrics are based on the confusion matrix for classification tasks, which is a table that shows the real and the predicted values of the instances. This matrix, shown in Table 1.4, measures the following:

- **True positive** (TP): Instances correctly predicted as positive.
- **False positive** (FP): Instances incorrectly predicted as positive.
- **False negative** (FN): Instances incorrectly predicted as negative.
- **True negative** (TN): Instances correctly predicted as negative.

Table 1.4 Confusion matrix: real positive cases (P), real negative cases (N), predicted positive cases (PP), predicted negative cases (PN), true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

		Predicted value	
		Positive (PP)	Negative (PN)
Actual	Positive (P)	TP	FP
	Negative (N)	FN	TN

Based on this matrix, several metrics can be calculated, among the most popular of which are:

- **Accuracy** (ACC): The proportion of correctly classified instances:

$$accuracy = \frac{TP + TN}{P + N} \tag{1.2}$$

- **Recall** (TPR): True positive ratio or the probability of correctly classifying a positive instance:

$$recall = \frac{TP}{P} \tag{1.3}$$

- **Specificity** (TNR): True negative ratio or the probability of correctly classifying a negative instance:

$$specificity = \frac{TN}{N} \tag{1.4}$$

- **F_1-Score**: The harmonic mean of precision and recall:

$$F_1\text{-score} = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.5}$$

- **Matthews correlation coefficient** (MCC): The correlation between the real and predicted values:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{1.6}$$

All data				
Train 80%				Test 20%
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Table 1.5 Example of a 5-fold cross-validation.

1.5.2 Cross validation

After training a model, it is necessary to evaluate its performance with new data. The most common way to do so is to split the data into two sets, the training set and the test set on the basis of an $80-\textbf{to}-20$ ratio. This split should be stratified, so as to ensure that the distribution of classes in the training and test sets is the same as in the original dataset. However, testing several models with the same split can lead to overfitting. Cross-validation can be used to do so. The most common way to use cross-validation is to split the data into k folds, train the model with $k-1$ folds, and test it with the remaining fold. This process is repeated k times, so each fold serves once as the test set. An example of a 5-fold cross-validation is shown in Table 1.5.

1.5.3 Statistical tests

Statistical tests are tools used in machine learning to determine whether the model results are significant in relation to a problem. So, they are used to compare the performance of the models on a problem. The most common tests are the paired t-test [90] and the Wilcoxon signed-rank test.

The paired t-test is a parametric test that supposes that the data follow a normal distribution. It is used to compare the means of two samples. The null hypothesis is that the means are equal, and the alternative hypothesis is that they are different. The t-test is calculated as:

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2 + s_y^2 - 2\rho s_x s_y}{n}}} \tag{1.7}$$

Where $\bar{x}$ and $\bar{y}$ are the means of the samples, s_x^2 and s_y^2 are the variances, ρ is the correlation coefficient for the two variables and n is the number of values.

The Wilcoxon signed-rank test is a non-parametric test that makes no assumptions related to any one data property. It is used to compare the medians of two samples. The null hypothesis is that the medians are equal, and the alternative hypothesis is that they are different. The Wilcoxon signed-rank test is calculated, following the steps listed below:

1. Calculate the differences between paired observations: $d_i = x_i - y_i$, where x and y are the paired observations.
2. Calculate the rank mean of the absolute differences ($|d_i|$) from smallest to largest.
3. Sum the ranks of the positive differences as $T+$ and sum the ranks of the negative differences as $T-$.
4. Calculate the test statistic T as the minimum of $T+$ and $T-$.

Once T is calculated, the Wilcoxon distribution is used to find the p-value, using n degrees of freedom. If the p-value is less or equal to the significance level, T, then the null hypothesis is rejected.

2
Motivation and Goals

The main objective of this thesis is to explore different preprocessing techniques, so as to improve the performance of learning models on wide data. As mentioned in the previous chapter, since wide data contain a large number of features and a low number of instances, the most convenient preprocessing techniques are feature selection, feature reduction, and data resampling.

Studies focused on feature selection commonly use high-dimensional data, but not all include wide data and class imbalance. This is especially true in the feature reduction field where, due to matrix multiplication limitations, datasets with more columns than rows are not expected, and the algorithms commonly used are not valid for datasets of that sort.

Semi-supervised learning is also considered a promising field to explore, since one reason for the wide data set is the lack of labels. Retaining information in both labelled and unlabelled instances can improve model performance, and using semi-supervised feature selection algorithms might be a good approach and first step towards resolving the issue.

Each of the following sections describes the objectives of the three papers compiled in this thesis as chapters.

2.1 Feature selection and resampling pipeline

The aim of investigating the effectiveness of combining balancing techniques with feature selection is to improve classification performance on wide datasets. By using common algorithms for both tasks, the aim is to determine the optimal approach to achieve improved results. Experimenting with wrapper approaches has a high computational cost, excluding it on a cost basis, and making it impractical for wide datasets.

The paper extends the existing literature by providing a comprehensive analysis of different strategies, including comparisons of the different balancing algorithms and exploratory conclusions about when to integrate them, either before or after feature selection.

The results for different percentages of selected features are also evaluated and the performance of different algorithm combinations are compared, providing a more detailed understanding of the problem space.

2.2 Feature reduction and feature selection performance comparison

Appropriate feature reduction and resampling strategies for wide datasets are described in this study, in terms of performance and computational efficiency. It adds to previous studies, by introducing feature reduction experiments and comparing the new results.

If feature reduction methods are to be used with wide datasets, it is crucial to identify the methods that are compatible with the dataset dimensions. In this study, one aim is to determine the significance of different balancing methods and to find the best feature reduction approach for each classifier.

A significant novelty of this study is its broadly extensive experimentation, with a particular focus on resampling techniques. While previous studies have focused primarily on basic techniques, a wider range of methods are explored in this study, including non-linear approaches.

Suitable approaches to process out-of-sample instances need to be identified for the application of non-linear feature reduction techniques.

2.3 Systematic review of semi-supervised feature selection

The aim of this review is to address the curse of dimensionality in semi-supervised learning problems through the use of semi-supervised feature selection algorithms. It extends previous studies to provide a new, updated view of the field.

The effectiveness of semi-supervised feature selection is measured in various domains, including video processing, image and multimedia annotation, chemistry, biology, and physics, among others. It demonstrates its applicability and impact on improving model performance with limited labelled data. This preprocessing is relevant for wide datasets where not only can a large number of challenging features be found, but also a low number of instances, which may be due to the cost of labelling in some domains.

3

Discussion of Results

In this chapter, the journal papers published or submitted during the development of the thesis are presented. Additionally, other relevant journal papers not included in the compendium, but related to the main topics and published during the same period are listed. Finally, the repositories containing the experimental data resulting from the experiments and the code used to generate the results are presented.

3.1 Journal Papers

1. **Title:** When is resampling beneficial for feature selection with imbalanced wide data?

   Authors: Ismael Ramos-Pérez, Álvar Arnaiz-González, Juan J. Rodríguez and César García-Osorio.

   Journal: Expert Systems with Applications (JCR: Q1, SJR: Q1).

   Year: 2022

   Reference: [150]

   Repository: https://github.com/Ismael-rp/feature_selection_wide_data

2. **Title:** An Extensive Performance Comparison between Feature Reduction and Feature Selection Preprocessing Algorithms on Imbalanced Wide Data

Authors: Ismael Ramos-Pérez, José Antonio Barbero-Aparicio, Antonio Canepa-Oneto, Álvar Arnaiz-González, and Jesús Maudes-Raedo.

Journal: Information (JCI: Q2, SJR: Q2).

Year: 2024

Reference: [149]

Repository: https://github.com/Ismael-rp/feature_reduction_feature_selection_wide_data_comparison

3. **Title:** Systematic review of semi-supervised feature selection techniques

   Authors: Ismael Ramos-Pérez, Álvar Arnaiz-González, Jesús Maudes-Raedo, Juan J. Rodríguez

   Journal: Under review.

   Year: 2024

   Repository: Not applicable.

3.2 Other Journal Papers

1. **Title:** Analysis of the learning process through eye tracking technology and feature selection techniques

   Authors: María Consuelo Sáiz-Manzanares, Ismael Ramos Pérez, Adrián Arnaiz Rodríguez, Sandra Rodríguez Arribas, Leandro Almeida and Caroline Françoise Martin

   Journal: Applied Sciences (JCR: Q2, SJR: Q2).

   Year: 2021

   Reference: [159]

2. **Title:** An experiment on animal re-identification from video

   Authors: Ludmila I Kuncheva, José Luis Garrido-Labrador, Ismael Ramos-Pérez, Samuel L Hennessey and Juan J Rodríguez

   Journal: Ecological Informatics (JCR: Q1, SJR: Q1).

   Year: 2023

   Reference: [99]

3. **Title:** Semi-supervised classification with pairwise constraints: A case study on animal identification from video

   Authors: Ludmila I Kuncheva, José Luis Garrido-Labrador, Ismael Ramos-Pérez, Samuel L Hennessey and Juan J Rodríguez

   Journal: Information Fusion (JCR: Q1, SJR: Q1).

   Year: 2024

   Reference: [100]

4
Conclusions

A study on preprocessing methods for wide datasets has been presented in this thesis. Each set of results was focused on one of three different preprocessing techniques: feature selection and resampling, feature reduction, and feature selection for semi-supervised learning. The most relevant contributions are listed in the following sections.

4.1 Feature Selection and resampling

The study of the effects of feature selection and resampling on wide data led to a key question that became the topic of the first paper: When is resampling beneficial for feature selection with imbalanced wide data? After a large number of experiments, the following conclusions may be advanced:

- Balancing is beneficial to feature selection for wide data classification.
- The effectiveness of balancing strategies depends on the classifier and feature selection method that is used. Some balancing methods can be counterproductive.
- The preprocessing methods order affects model performance, but the choice depends on the combination of algorithms (Classifier, feature selector, and resampling method) chosen.

- Resampling before feature selection is better with Random Under Sampling while resampling after feature selection works better with Random Over Sampling and SMOTE.
- For the datasets used in the experiments, the best performing classifiers for wide data were KNN and SVM-G, with the best feature selection algorithms being ReliefF, T-test, and SVM-RFE.

4.2 Feature Reduction

The natural question that emerges after applying feature selection to any problem is: What, if instead of feature selection, feature reduction algorithms were applied, to avoid the curse of dimensionality? To answer that question, an extensive and large number of feature reduction models were explored in the second paper for their comparison with the feature selection algorithms. Normally, when a feature reduction algorithm is used, it is compared with other feature reduction and feature selection algorithms. However, the methods selected are basic and limited. Relevant conclusions of this paper may be highlighted here:

- The feature reduction algorithm MMC was of interest, because of its performance and the FSCORE because of its simplicity.
- MMC outperformed the best feature selection algorithm found in the previous paper (SVM-RFE) that performed second best.

4.3 Feature Selection in semi-supervised learning

Expanding the feature selection topic to semi-supervised learning, a systematic review covering 102 studies was performed. As a result of that review, a new taxonomy was proposed, and 103 algorithms were classified. That information is shown alongside other data related to the studies, such as the datasets, authors, and typical experimental setups. The conclusions can be summarized as follows:

- The most common type of semi-supervised feature selection algorithms are the filters based on the Laplacian matrix and pairwise constraints.
- In this field, researchers should explore two critical parameters usually chosen arbitrarily: the number of labelled instances and the number of selected features.

- Some good practices to ensure results validation are not always followed, such as the application of robust metrics to unbalanced data, proper cross-validation techniques, and statistical tests.
- Only 10% of the algorithms have available code, which hinders result reproducibility and delays further research.
- Supervised learning models, particularly SVM and KNN, are the predominant models used to evaluate the performance of semi-supervised feature selection algorithms.

4.4 General conclusions

After a long period of research, the results have shown that preprocessing techniques remain essential to improve model performance and to provide some guidance on the choice of algorithm and the order of preprocessing in the field of wide data. In addition, the systematic review has not only provided an updated comprehensive view of the field for new researchers, but has also been a great exercise to improve critical thinking and scientific research skills. Finally, this thesis has achieved its goals of providing new information on the research topic, as well as offering a great opportunity to learn and to mature as a researcher.

5
Future Lines

This thesis has opened new lines of research that can be explored in the future. Examples include the use of hybrid and ensemble techniques for feature selection.

As mentioned in the conclusions, the choice of dimensionality to reduce the data is a challenging task. The application of dimensionality estimator [134] algorithms may help in making this decision. Applying meta-learning techniques [106] to choose the best type of preprocessing is also an option, as it has been noted that their performance partly depends on the other algorithms with which they are combined. Those algorithms do not need to be limited to the ones already applied, so new preprocessing techniques can be studied individually or combined with others on wide data, such as noise detection [193], which removes irrelevant instances.

The feature selection techniques have been studied in the field of semi-supervised learning, but the scope of these preprocessing methods can be extended to other areas such as regression [202], clustering [127], multi-label [115] or multi-target regression [219].

Part II
Publications

1

When is resampling beneficial for feature selection with imbalanced wide data?

This study explores the influence of combining methods of balancing and feature selection on wide data using different classifiers across multiple datasets.

Authors: Ismael Ramos-Pérez, Álvar Arnaiz-González, Juan J. Rodríguez and César García-Osorio.

Type: Journal

Published in: Expert Systems with Applications (JCR: Q1, SJR: Q1).

Year: 2022

Keywords: Feature selection, Wide data, High dimensional data, Very low sample size, Unbalanced, Machine Learning.

Reference: [150]

Abstract

This paper studies the effects that combinations of balancing and feature selection techniques have on wide data (many more attributes than instances) when different classifiers are used. For this, an extensive study is done using 14 datasets, 3 balancing strategies, and 7

feature selection algorithms. The evaluation is carried out using 5 classification algorithms, analyzing the results for different percentages of selected features, and establishing the statistical significance using Bayesian tests.

Some general conclusions of the study are that it is better to use RUS before the feature selection, while ROS and SMOTE offer better results when applied afterwards. Additionally, specific results are also obtained depending on the classifier used, for example, for Gaussian SVM the best performance is obtained when the feature selection is done with SVM-RFE before balancing the data with RUS.

1.1 Introduction

The term "wide data" has been used to refer to datasets characterized by a high number of features and a low number of instances [101], which severely impairs the smooth performance of learning algorithms. We have not been able to find in the literature a proper definition of the term "wide data". Although strictly speaking a dataset could be considered wide when its number of features (#Features) is just greater than its number of examples (#Examples), #Features $>$ #Examples, we believe that for a dataset to deserve to be called wide, this difference must be substantial, #Features $\gg$ #Examples, for example, of at least one order of magnitude. For reference, in the datasets used in the experimental part of this article, that difference is even greater, with a #Features/#Examples ratio that is greater than 20 for all datasets, and with at least 2 000 features.

Several real-world datasets suffer from these problems, especially biological and genomics datasets, discussed in Liu et al. [121], where data from electroencephalography analyses were used for epilepsy diagnosis, analysis in early detection of type 2 diabetes [20], and the prediction of mortality among patients admitted to an intensive care unit [191]. However, this type of data also arises in other areas such as fault detection in engineering: for example, the diagnosis of engine system faults from measurements taken from the bearing assembly [66], the detection of induction motor failures [6, 84], and in solar radiation estimation [86], among others. It also appears in computer security environments, for intrusion detection in network environments [211]. The presence of a large number of features (high dimensionality) decreases the efficiency of learning algorithms and increases their execution time [128].

As it is well known, the aim of feature selection (FS) algorithms is to find the optimal combination of features that will help to create models that are simpler, faster, and easier to interpret. However, this task is not easy and is, in fact, an NP-hard problem [62]. In addition, this type of data is known as unbalanced data when the number of instances belonging to

each class is very different between classes [55]. When dealing with unbalanced datasets, even if the classifier achieves high global accuracy, it is often the case that the identification of the instances belonging to the minority class is not highly precise. Classifiers work well with instances of the majority classes, though these instances are usually the least interesting, which means that the classifier is largely useless. Some of the most popular algorithms that try to solve this problem are based on oversampling techniques [1] that increase the number of instances of minority classes and undersampling techniques [136] that decrease the number of instances of the majority classes. And there are more recent proposals that use ensembles to combat imbalance [50], or deal with this problem in the context of big data [83].

The objective of this study is to conduct a series of experiments to assess whether balancing techniques improve classification performance when they are used in conjunction with feature selection on wide data. For this purpose, we use the most common algorithms at each of these stages. Moreover, we are not only interested in finding the more suitable balancing algorithm, but we also seek to determine the most appropriate moment to use it: either before or after feature selection. There is too little literature comparing the performance of these two strategies [145, 225], the main novelties of the present paper are:

- The use of a wider variety of algorithms.
- The use of a larger number of datasets.
- The evaluation of the results for different percentages of selected features, avoiding the bias of using a fixed percentage, and hence providing a more complete overview of the problem.
- The performance comparison of different combinations classifier-balancing method.

The *R* code used for the feature selection and to create the figures can be found on *GitHub*[1].

The rest of the paper will be organized as follows. In Section 1.2, the background on the different approaches of the feature selection methods will be given. In Section 1.3, the same will be done for the techniques dealing with unbalanced problems. Information will be provided in Section 1.4 on the experimental setup and the results will be presented and analyzed in Section 1.5. The main insights of the study are discussed in Section 1.6. Finally, the conclusions and future work will be presented in Section 1.7.

[1] https://github.com/Ismael-rp/feature_selection_wide_data.

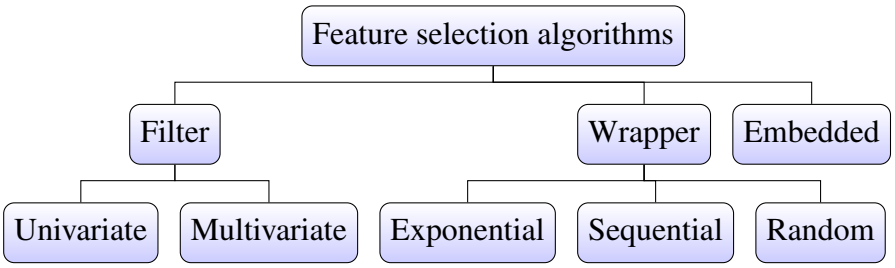


Fig. 1.1 Taxonomy of feature selection algorithms [157].

1.2 Feature selection

In data science, it is often very important to know which features of a dataset are the most relevant for training learning algorithms [157]. The use of certain features may not only make no contribution to the improvement of the learning algorithm, but their use might even worsen its performance. This reason explains why FS algorithms are used to find the subset of features that improves the performance of the models obtained using machine learning algorithms. Moreover, FS algorithms also prevent the learning algorithm from overfitting and speeds up its training. In addition, knowledge of which the selected features actually are can provide useful insight into the datasets.

This problem is even more relevant when dealing with wide data, where the number of features is extremely high. At the same time, this technique is widely used for big data [144] where reducing both data size and execution times are paramount.

Some taxonomies [237, 157] can be found in the literature on the different FS algorithms, the most widely used of which and, from our point of view also the most convenient, classify the features by their relationship with the learning algorithm. This taxonomy is shown in Figure 1.1, in which we can find three main types: filters, wrappers and embedded methods (or nested subset methods):

- **Filters** [25] are used to calculate the relevance of each feature, mainly based on its statistical properties, providing a numerical score for each feature that depends on its contribution to the performance of the algorithm, also called importance.

  Since the operation of these methods will not depend on the use of any particular classifier, it means that the feature sets can be used with any classifier. This approach will reduce overfitting, but it cannot guarantee the best performance, unlike other FS algorithms.

It should be noted that since this type of algorithms cannot determine the optimal number of features, which would provide the best performance. That number is an additional parameter that must be set to select the subset of relevant features.

An advantage of filter methods is that they are only executed once before the training, avoiding having to adjust a lot of hyperparameters, making the training faster and more scalable. This feature makes them specially suitable for big-data problems.

Two kinds of filters may be identified: univariate and multivariate. While the univariate FS algorithms find dependencies between each feature and the output, the multivariate ones try to find dependencies between the features, unfortunately, this is at the cost of more processing time, which makes them less scalable.

- **Wrapper** methods [93] search throughout the entire space of feature subsets looking for the subset that provides the best performance to the specific learning algorithm given as parameter. They usually offer better performance than the filter methods, however the risk of overfitting is higher. They are also slower and less scalable, because the learning algorithm has to be executed every time a subset of features needs to be evaluated.

- **Embedded** methods [207, 64] take advantage of the inner properties of certain learning algorithms, in order to discover the most relevant features in the dataset, as is the case with Random Forest.

  Unlike wrapper methods, the classifier on which the embedded methods are based is not necessarily the same as the one used to classify. These methods have lower risks of overfitting and are faster than the wrapper-based methods, although they are still slower than the filter-based methods.

The strategies of the previous methods can be combined to obtain new algorithms, the two combination approaches are:

- **Hybrid algorithms** take advantage of filter and wrapper methods, sequentially combining their outputs. The output of the filter is given as the input to the wrapper, which reduces the wrapper computation time, by making an initial selection using the filter method and exploiting the efficiency that the wrapper can obtain. Although the sequence of filter-wrapper is generally used, different combinations can also be found [158].

- **Ensemble algorithms** [23] combine the output of several individual methods to improve the results that would have been obtained from using each of them in isolation.

In the same way as ensemble classifiers, ensembles for feature selection can be classified into homogeneous (those that use the same FS method) and heterogeneous (that use different FS methods). The latter are the most widely used.

For this study, we focused on the most cited FS algorithms in the state of the art. To facilitate the comparison of methods, in the experiments we only included those that return a ranking, among which we can find filters (T-test, ANOVA, Information gain, Chi squared, and ReliefF) and embedded methods (Random Forest importance and SVM Recursive Feature Elimination):

- T-test [143] is a popular statistical test that may be used for an individual evaluation of feature relevance. It computes the ratio between the differences of two class means and the variability between them.
- ANOVA [82], an acronym that stands for "ANalysis Of VAriance", is a simple and well-known method that can be used for feature selection. It works in a similar way to the previous method, testing the differences of means between groups.
- Chi squared (Chi-S) is a feature selection algorithm proposed by [116] that uses the χ^2 statistic and works in two phases. The first phase uses the ChiMerge of [89], automatically increasing the χ^2 threshold. The second phase attempts to merge the features, by using the values computed in the previous phase; if two features can be merged, it means that one of them is irrelevant and can be discarded.
- Information gain (Info-Gain) is a measure commonly used in the construction of decision trees for finding the most informative feature to use for splitting each node. The use of information gain for feature selection involves an evaluation of the information gain of each feature with respect to the target feature. More specifically, information gain measures the expected reduction in entropy [132].
- Random Forest importance (RF-Imp) determines the usefulness of each feature by measuring the performance difference of the out-of-bag data when noise is added, as proposed by [27].
- ReliefF [94] is an extended version of the original Relief feature selector of Kira and Rendell [91]. Based on instance-based learning (kNN is used for searching similar instances), it estimates the importance of a feature in relation to other features, and it is non-parametric, that is, no assumption of any distribution is made [188].

- SVM Recursive Feature Elimination (SVM-RFE) [63] is an iterative process that recursively removes features according to the feature weights in a support vector machine classifier (SVM). It is an example of backward feature elimination [92] where the elimination is recursive and the classifier used is the popular SVM.

1.3 Unbalanced data

A dataset is said to be unbalanced when it has a very different number of instances for each class, which affects negatively the classifiers performance, is commonly measured using the imbalance ratio (IR) [125], it is even more common in wide data due to the low number of instances.

A straightforward solution for dealing with unbalanced data is to resample the original data set by adjusting the balance ratio as desired. On the one hand, undersampling methods eliminate instances corresponding to the majority class [136]; on the other hand, oversampling methods create artificial instances for the minority class [1], and hybrid methods combine both approaches.

The resampling methods used in this study are described below:

- *Random undersampling* (RUS) [79] algorithm randomly selects instances from the majority class (*i.e.*, it removes instances of the majority class).
- *Random oversampling* (ROS) [79] algorithm duplicates instances from the minority class.
- *Synthetic Minority Over-sampling Technique* (SMOTE) [32] algorithm creates new instances, interpolating two instances of the original data set, the first one chosen randomly and the second one chosen randomly from among its k closest neighbors where k is a predefined parameter.

1.4 Experimental setup

Evaluating the performance of an FS algorithm is not as simple as with a common classifier, where the evaluation is simply based on the number and the type of correctly classified instances.

We aim to evaluate how good an FS algorithm is at selecting the features. These algorithms will attach greater weight to the more relevant features and less weight to the less relevant ones. Using an artificial dataset, we would be able to compare the real weight

Table 1.1 Datasets used in the experimental study. Datasets 1–9 were previously used in [237]. Datasets 10–14 were used in [109].

	Dataset	#Ex.	#Feat.	$\frac{\#Feat.}{\#Ex.}$	Class (min.; max.)	%min.; %max.	IR
1	Colon[1]	62	2 000	32.26	(Normal; Tumor)	0.35; 0.65	1.86
2	MLL_ALL[1]	72	12 582	174.75	(ALL; rem)	0.33; 0.67	2.03
3	MLL_AML[1]	72	12 582	174.75	(AML; rem)	0.39; 0.61	1.56
4	MLL_MLL[1]	72	12 582	174.75	(MLL; rem)	0.28; 0.72	2.57
5	SRBCT_1[1]	83	2 308	27.81	(1; rem)	0.35; 0.65	1.86
6	SRBCT_4[1]	83	2 308	27.81	(4; rem)	0.30; 0.70	2.33
7	Lung_1[1]	203	12 600	62.07	(rem; 1)	0.32; 0.68	2.12
8	Lung_4[1]	203	12 600	62.07	(rem; 4)	0.10; 0.90	9.00
9	Lung_5[1]	203	12 600	62.07	(rem; 5)	0.10; 0.90	9.00
10	Leukemia_BM[2]	72	7 130	99.03	(BM; rem)	0.29; 0.71	2.45
11	TOX_171_1[2]	171	5 748	33.61	(1; rem)	0.26; 0.74	2.85
12	TOX_171_2[2]	171	5 748	33.61	(2; rem)	0.26; 0.74	2.85
13	TOX_171_3[2]	171	5 748	33.61	(3; rem)	0.23; 0.77	3.35
14	TOX_171_4[2]	171	5 748	33.61	(4; rem)	0.25; 0.75	3.00

[1] https://jundongl.github.io/scikit-feature/datasets.html
[2] http://csse.szu.edu.cn/staff/zhuzx/Datasets.html

of each feature with the one provided by the FS. However, as with the case presented here where real datasets are used, it is common to evaluate the selected features according to the classifier performance [23]. In this section, the steps followed during the experiments are explained, in order to evaluate the performance of the combination of feature selection and balancing on imbalanced wide data.

1.4.1 Datasets

The 14 high dimensional unbalanced datasets used in this study are summarized in Table 1.1. For each dataset, the number of examples, the number of features, the relation between the number of examples and features, the class names, the percentage of examples for each class, the class imbalance ratio (IR), and the reference are shown. All the dataset features were numeric and the original multi-class labels were grouped into a new one, in order to obtain new two-class unbalanced datasets. The dataset name indicates which classes were used (where rem represents the combination of the remaining classes).

1.4.2 Cross validation

The experiments were performed using 5×2-fold cross validation. Having been randomly divided into 2 parts, both parts were used for training and testing and the same step was repeated 5 times. This kind of cross validation is very useful when processing datasets with classes that have a very low number of instances, since using 10-fold cross validation will leave a very low number (or even no one) of instances belonging to the minority class in the test [49].

Moreover, all the datasets were normalized for transforming all the features: the mean to 0 and the standard deviation to 1.

1.4.3 Feature selection and balancing

Since our goal is to assess how data balancing affects feature selection, we will combine the 7 FS algorithms with the 4 balancing strategies explained in sections 1.2 and 1.3 (all the methods are listed in Table 1.2).

There are two possible ways to combine these algorithms: either to perform feature selection first and then to balance the dataset (FS+bal), or in reverse, to balance the dataset first and then to perform feature selection (bal+FS).

All of these combinations (7 FS algorithms, 4 balancing strategies, and 2 ways to combine them) added up to a total of 56 configurations that were all tested.

Since all the FS algorithms used were rankers, they only offered a list of features sorted by importance, without ever indicating the optimal number of features for any algorithm. In other studies [145], the authors selected a specific number of features what could induce a bias in the results. Here, we wished to conduct a broader and more in-depth study of the process, so as to offer a better overview of the behavior of the interactions between the two preprocessing steps, which is why up to 20 different scenarios were considered, each using a different percentage of selected features, with a separation of 5 points between them.

1.4.4 Classifiers

Some of the most popular classifiers were used for testing feature selection: k-nearest neighbors (KNN), SVM-Gaussian, C4.5 trees, Random Forest, and Naive Bayes [23].

1.4.5 Parameters

All the algorithms together with the parameters used and the libraries that were applied are shown in Table 1.2. For the SVM-G classifier we performed a grid parameter search and we

Table 1.2 Resampling algorithms used in this study with their parameters and the R packages used: class (https://cran.r-project.org/web/packages/class/index.html), e1071 (https://cran.r-project.org/web/packages/e1071/index.html), RWeka (https://cran.r-project.org/web/packages/RWeka/index.html), randomForest (https://cran.r-project.org/web/packages/randomForest/randomForest.pdf), naivebayes (https://cran.r-project.org/web/packages/naivebayes/index.html), sigFeature (https://www.bioconductor.org/packages/release/bioc/html/sigFeature.html), mlr3filters (https://cran.r-project.org/web/packages/mlr3filters/index.html), FSelector (https://cran.r-project.org/web/packages/FSelector/index.html), and unbalanced (https://cran.r-project.org/web/packages/unbalanced/index.html).

Algorithms	Parameters	Package
Classifier		
KNN	$K = 1$	class
SVM-G	$c = 1e+09$, $g = 1e-07$	e1071
C4.5	Default	RWeka
Random Forest	Default	randomForest
Naive Bayes	Default	naivebayes
Feature selection		
T-test	-	sigFeature
ANOVA	-	mlr3filters
Chi-Squared	-	FSelector
Info Gain	-	FSelector
RF-Imp	-	FSelector
ReliefF	neighbors = 1	FSelector
SVM-RFE	-	sigFeature
Balancing		
ROS	Ratio 1:1	Own impl.
RUS	Ratio 1:1	Own impl.
SMOTE	Ratio 1:1, k=5	unbalanced

Table 1.3 Confusion matrix

	Actual positive	Actual negative
Predicted true	True positive (TP)	False positive (FP)
Predicted false	False negative (FN)	True negative (TN)

found that using $c = 1e+09$ and $g = 1e-07$, we obtained an optimum performance in all datasets. Regarding ReliefF, as it needs a long time to rank the features, we set the parameter n to 1 in order to reduce its execution time. For SMOTE, we tested a range of values from 1 to 20 for the parameter k, we observed that the performance is very similar for all of them and the recommended parameter 5 gives usually slightly better performance than the others. Finally, with the balancing algorithms, we left the balancing ratio to 1 for all the datasets (*i.e.*, the same number of instances for both the majority and the minority classes).

1.4.6 Metrics

Testing an algorithm with unbalanced data can be problematic. If, for example, the unbalance ratio were 1:10, with 100% majority-class accuracy and 0% minority-class accuracy, the accuracy rate can be set at 90%, however these results are of no use, as the trained classifier cannot distinguish between the two classes.

We selected some of the most accepted metrics in feature selection and data balancing, to evaluate the performance of each configuration: the *Area Under the ROC Curve* (AUC), *Geometric Mean* (G-Mean), and F_1-*Score*.

These metrics are based on the confusion matrix (see Table 1.3) where the following values can be found:

- *True Positive* (*TP*): positive instances correctly classified (minority class in our data).
- *True Negative* (*TN*): negative instances correctly classified.
- *False Positive* (*FP*): positive instances incorrectly classified.
- *False Negative* (*FN*): negative instances incorrectly classified.

Our objective was to maximize the balance between *TP* and *TN* values in the diagonal. Using these values, the three basic ratios can be calculated before computing our three main metrics:

- **Recall** is the probability of considering a positive instance as positive.

$$recall = \frac{TP}{TP + FN} \tag{1.1}$$

- **Specificity**, as opposed to recall, is the probability of classifying a negative instance as negative.

$$specificity = \frac{TN}{TN + FP} \tag{1.2}$$

- **Precision** is the probability of an instance classified as positive.

$$precision = \frac{TP}{TP + TN + FP + FN} \tag{1.3}$$

From these ratios, the measures used to evaluate the results of the experiments can be defined.

- ***Area Under the ROC Curve*** (AUC) the mean between recall and specificity. Note that, although it is often used to evaluate multiple possible classifiers, here we just use it with a single point, which is the mean between recall and specificity. This measure has been used in earlier studies [58].

$$\text{AUC} = \frac{recall + specificity}{2} \tag{1.4}$$

- **$\mathbf{F_1}$-Score** is the harmonic mean between precision and recall.

$$F_1\text{-Score} = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.5}$$

- **G-Mean**, widely used in unbalanced problems, is the geometric mean between sensitivity and specificity.

$$\text{G-Mean} = \sqrt{recall \times specificity} \tag{1.6}$$

1.5 Results

This section summarizes the results of the 6 860 experiments that were performed: the performance of 5 classifiers on 14 wide datasets (Table 1.1) using all possible combinations of 7 strategies for FS algorithms and 7 balancing strategies [2].

For the sake of readability and simplicity, only the results of the *AUC* metric are shown, since the conclusions drawn from the F_1-Score and G-Mean metrics were very similar. Nevertheless, the figures of all the metrics are compiled in the additional material[3].

Figure 1.2 shows all combinations of FS algorithms and classifiers in a matrix of stacked graphs. In the stacked graphs, the ordinate axis is the percentage of victories, and the abscissa axis is the percentage of selected features (20 different percentages with separation steps of 5 percent; the minimum percentage of features used is 5 percent and the maximum is 100 percent). The different shades of blue correspond to different balancing strategies (as shown in the top legend). The rows of the matrix of graphs correspond to different classifiers (shown on the right side), the columns to the selectors (specified on the top of each column).

Moreover, Figure 1.3 is included where the area plots are replaced by line plots: the y axis represents the average ranks obtained for each balancing strategy and the x axis represents the percentage of the features selected by the corresponding FS algorithm (sorted from best to worst).

In Figures 1.2 and 1.3, it can be seen that the results depend more on the classifier than on the other parameters. Broadly speaking, we can see the following insights for each classifier:

- KNN: The best performance was achieved when resampling is performed before the FS algorithm, more specifically, SMOTE+FS and ROS+FS were on the top.
- SVM-G: Although it may not be so clear in the rankings, it can be seen in the area graphs that FS+RUS usually provided the best results. Specially when SVM-RFE was the feature selector in use.
- C4.5: While not using any balancing strategies showed better results in the area graphs and ROS+FS in the average ranks, both RUS balancing strategies were the lowest in the performance rankings.
- RF: The balancing configuration FS+ROS in the area plots, and FS+RUS in the average ranks, showed better performance than the others. Unlike C4.5, the rankings

[2] Each balancing strategy applied in two orders, before or after the feature selection, and the case of not balancing.

[3] Additional material: https://ars.els-cdn.com/content/image/1-s2.0-S0957417421013622-mmc1.pdf

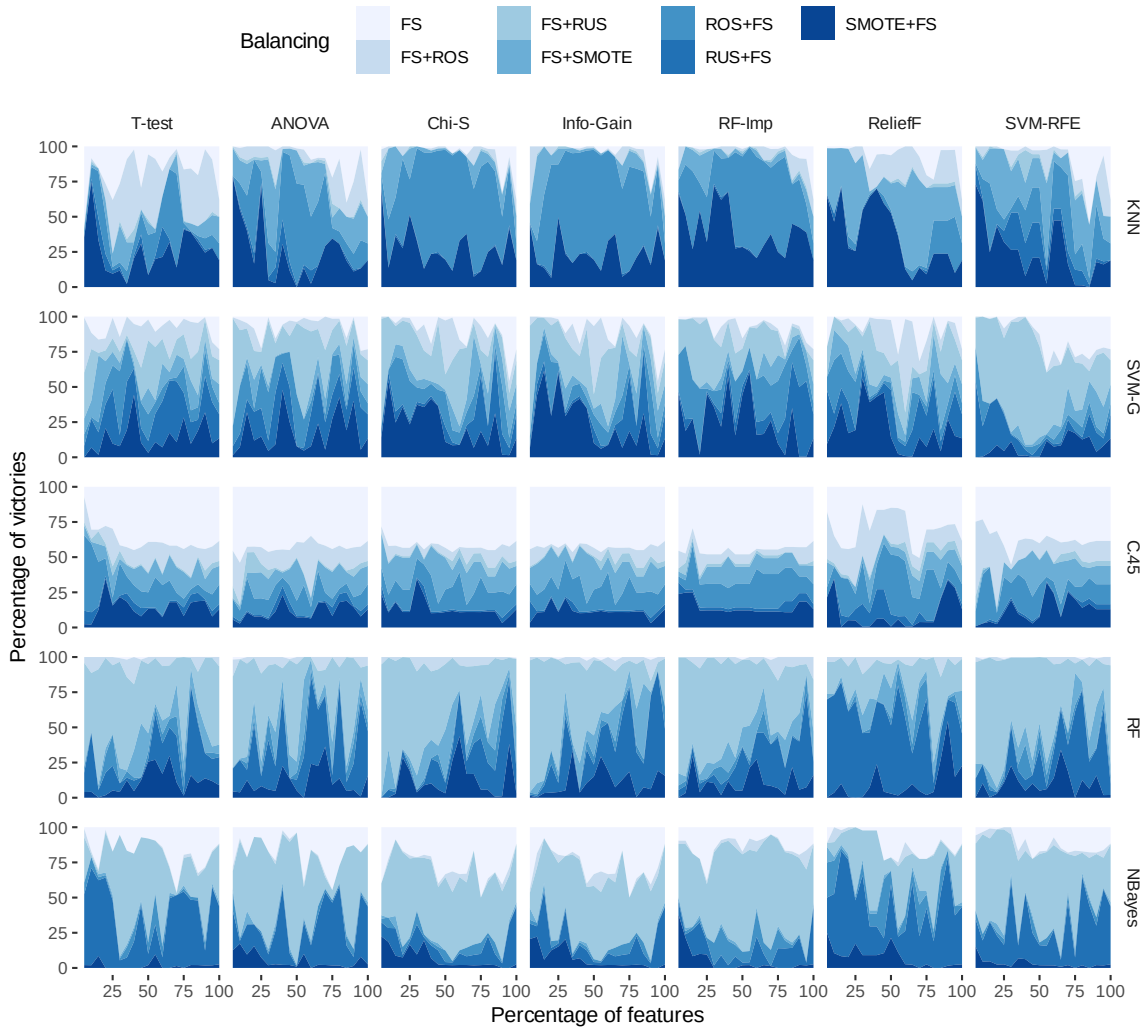


Fig. 1.2 Comparison of balancing methods: each row corresponds to a classifier, each column to an FS algorithm. At the intersection, a stacked graph shows the results of the different balancing methods for the corresponding combination of FS and classifier in shades of blue (the abscissa axis is the percentage of selected characteristics; the ordinate axis is the percentage of victories).

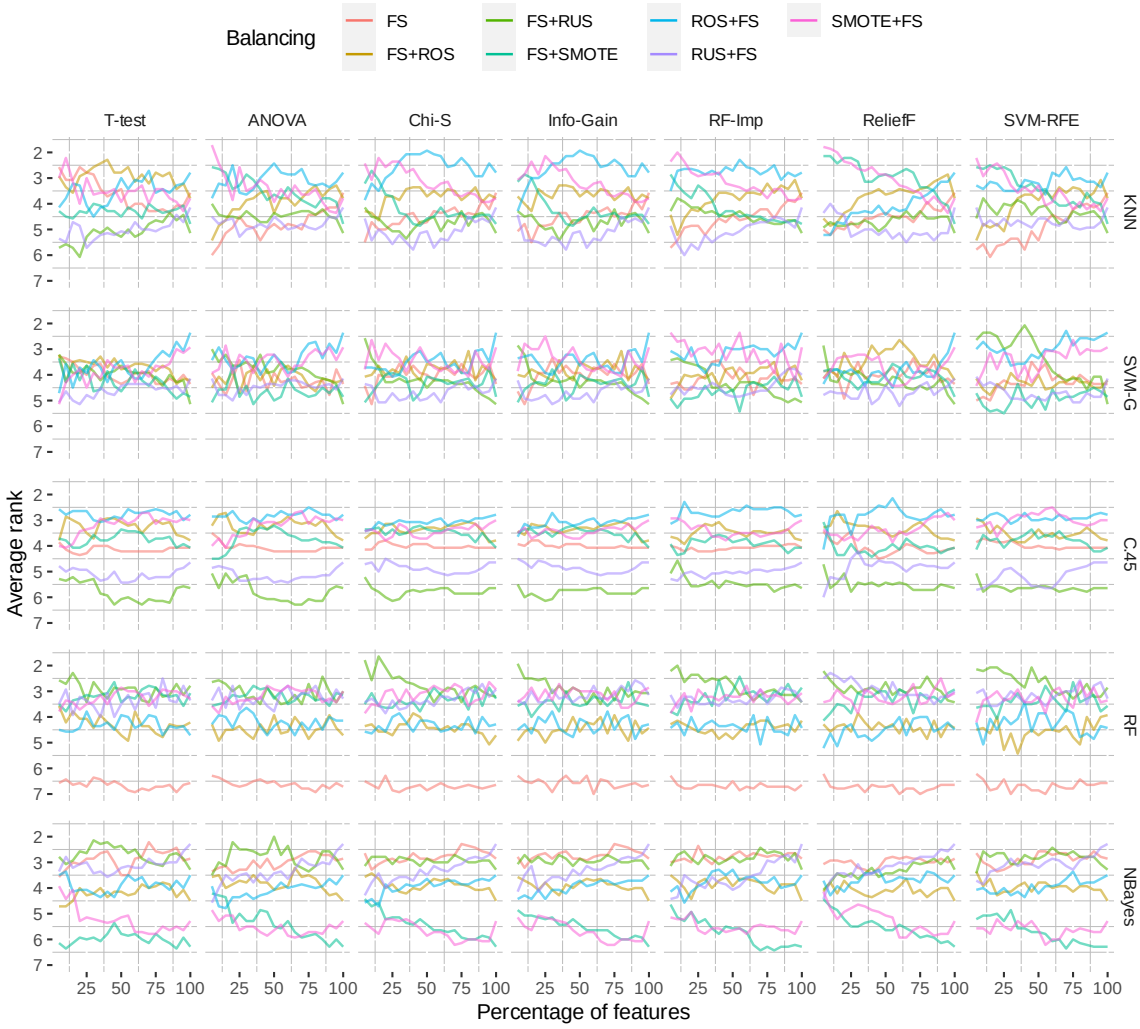


Fig. 1.3 Comparison of balancing methods by means of the average ranks, each row corresponds to a classifier, each column to an FS algorithm. At the intersection, a line graph shows the average rank for the different balancing methods for the corresponding combination of FS and classifier (the abscissa axis is the percentage of selected characteristics; the ordinate axis is the average rank).

Table 1.4 The best balancing configurations (rows) for each classifier (columns).

	KNN	SVM-G	C4.5	RF	NBayes
FS			✓		
FS+ROS				✓	
FS+RUS		✓		✓	✓
FS+SMOTE					
ROS+FS	✓		✓		
RUS+FS					✓
SMOTE+FS	✓				

showed that no use of balancing was by far the worst combination, a behavior also supported by the results of a previous study [145].

- NBayes: The use of RUS appeared to provide the biggest area, especially FS+RUS which can also be seen in the average ranks. According to the rankings, SMOTE was the worst, regardless of when the feature selection was implemented, either before or after its application.

Considering all the percentages of features and the different selectors used, we summarize the most remarkable balancing strategies for each classifier in Table 1.4.

Additionally, it is interesting to note that the number of selected features appeared not to have much impact on the final rankings of the balancing strategies.

So far, the balancing strategies have been compared to show which one worked best for each classifier. In what follows, similar graphs will be used to compare and to determine which the best combinations of classifiers and FS algorithms.

Figure 1.4 compares the FS algorithms which are differentiated by colors. The columns represent each different balancing strategy used, and the rows represent each classifier. It can be seen that among those that provide the best performance are SVM-RFE alongside most of the classifiers, specially with NBayes, KNN, and T-test. It also achieves good results with the KNN, making some appearances with the smallest number of features in SVM-G, C4.5, and RF.

Finally, looking at Figure 1.5, where each graph shows the percentage of times that a method has been the best for different sizes of feature subsets, the classifier that shows the best overall performance was SVM-G, with KNN in second place. However, the behavior of NBayes was also noteworthy, which yielded good results for some combinations of selectors and balancing strategies, when the percentage of selected characteristics was low.

After the general overview provided by the previous figures, we used average rankings to compare the configurations of the best classifiers (KNN and SVM-G). In the Table 1.5 can be

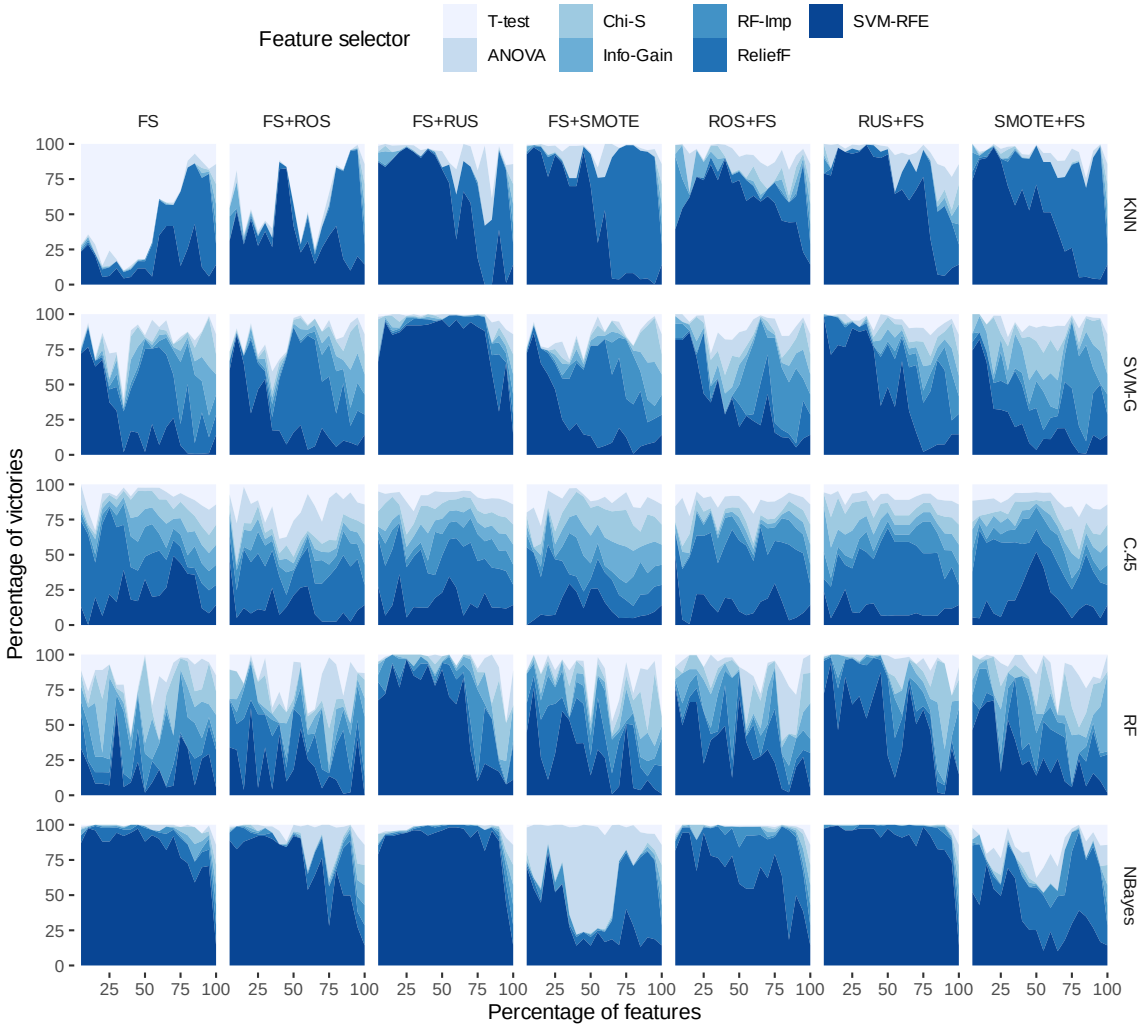


Fig. 1.4 Comparison of feature selector methods using the percentage of victories. The results are organized by balancing methods (columns) and classifiers (rows), while showing the percentage of features selected on the x axis.

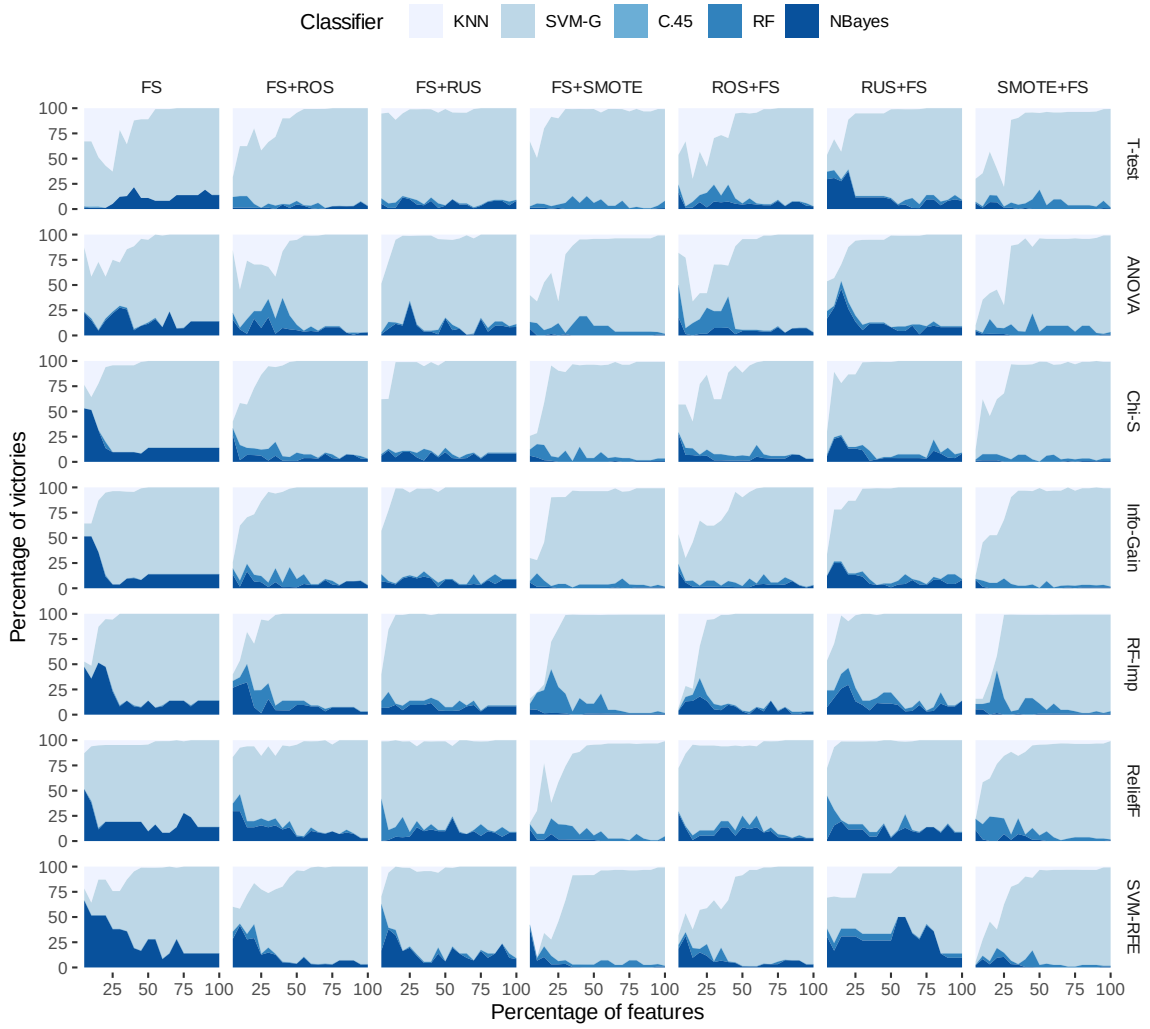


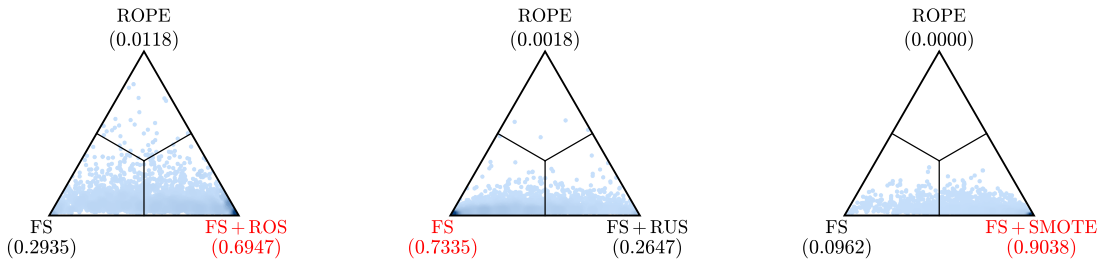
Fig. 1.5 Comparison of classifier performance using the percentage of victories. The results are divided by balancing methods (columns) and feature selector used (rows), while showing the percentage of features selected on the *x* axis.

Table 1.5 Average ranks of the most promising configurations previously identified.

Classifier	FS	Balancing	Avg. rank
SVM-G	SVM-RFE	FS+RUS	9.43
KNN	SVM-RFE	SMOTE+FS	12.93
KNN	SVM-RFE	FS+SMOTE	13.79
SVM-G	SVM-RFE	ROS+FS	14.00
SVM-G	SVM-RFE	SMOTE+FS	17.57
KNN	SVM-RFE	ROS+FS	20.21
SVM-G	SVM-RFE	FS+ROS	20.93
SVM-G	SVM-RFE	FS	22.00
SVM-G	SVM-RFE	FS+SMOTE	22.00
KNN	SVM-RFE	FS+RUS	22.86
SVM-G	SVM-RFE	RUS+FS	23.36
KNN	T-test	SMOTE+FS	25.14
KNN	ANOVA	SMOTE+FS	26.43
KNN	ANOVA	FS+SMOTE	27.64
SVM-G	ANOVA	ROS+FS	28.00
SVM-G	ANOVA	FS+RUS	28.57
SVM-G	T-test	FS	29.07
SVM-G	T-test	FS+RUS	29.14
SVM-G	T-test	ROS+FS	29.36
KNN	SVM-RFE	RUS+FS	29.64
KNN	T-test	FS	30.43
SVM-G	T-test	FS+ROS	30.64
KNN	T-test	FS+ROS	30.79
SVM-G	T-test	FS+SMOTE	31.00
KNN	SVM-RFE	FS+ROS	32.00
SVM-G	ANOVA	SMOTE+FS	32.21
KNN	ANOVA	ROS+FS	32.64
KNN	T-test	ROS+FS	32.93
SVM-G	T-test	SMOTE+FS	34.93
SVM-G	ANOVA	FS+ROS	35.07
SVM-G	ANOVA	FS+SMOTE	35.64
KNN	SVM-RFE	FS	35.93
KNN	T-test	FS+SMOTE	35.93
SVM-G	ANOVA	RUS+FS	35.93
SVM-G	ANOVA	FS	36.21
SVM-G	T-test	RUS+FS	36.64
KNN	ANOVA	FS+RUS	39.29
KNN	ANOVA	RUS+FS	42.57
KNN	ANOVA	FS+ROS	43.71
KNN	T-test	RUS+FS	45.00
KNN	ANOVA	FS	45.50
KNN	T-test	FS+RUS	46.21

seen the most promising combinations of classifiers, FS algorithms and balancing methods from the previous figures. It shows some interesting patterns. Combinations that use SVM-G as a classifier appear to perform better than those that use KNN. Regarding the FS algorithm, the dominance of the combinations that use SVM-RFE appears clear, as these combinations occupy the top positions. Finally, it appears that the balancing strategies have little influence on the positions that the combinations occupy; not only there is no clear pattern, but both the best and worst combinations use the same balancing strategy FS+RUS.

In an attempt to assess the effect of balancing the data and to complete the analyses performed so far, in what follows, we use the Bayesian test [18] to compare the FS configura-



(**a**) FS+ROS (**b**) FS+RUS (**c**) FS+SMOTE

Fig. 1.6 Example of three Bayesian tests for the classifier KNN and the feature selection ReliefF, where no balancing strategy is compared against using ROS, RUS, and SMOTE.

tions, which are the ones that do not balance the data, with the other configurations, which use different balancing strategies.

The Bayesian hypothesis testing is a relatively recent approach to the analysis of experimental results that tries to overcome the limits and problems which characterize null-hypothesis significance tests. It can be used to compare two different methods, obtaining the probabilities that one is better than the other, or that both have a performance that is practically equivalent. In Bayesian tests, this "*practical equivalence*" is given by the value of a parameter called the Region Of Practical Equivalence (ROPE), which was set at 0.001 for this work. In addition, only the performance values for the best balancing strategies in previous experiments were used, applying only the best 10% of selected characteristics, to reduce the number of results and to facilitate the analysis.

A Bayesian test constructs a probability distribution whose parameters are obtained from the differences in the experimental results observed when comparing two methods, using certain weights that are assumed to follow a Dirichlet distribution [137]. Given specific weights, the distribution can be used to calculate three probabilities: *i*) the probability that the first method is better than the second; *ii*) the probability that the second is better than the first; *iii*) the probability that both methods are practically equivalent. Through a Monte Carlo process [96], the weights can be sampled, obtaining different probability triplets. The three values of these triplets can be considered as barycentric coordinates that can be drawn as points in a simplex of coordinates $\{(1,0,0),(0,1,0),(0,0,1)\}$ [17].

As shown in Figure 1.6, where the classifier KNN and the feature selector ReliefF (with the 10% of the best features) without balancing is compared to the strategies FS+ROS, FS+RUS, and FS+SMOTE. Each triangle represents one test where the left corner shows the probability of not using any balancing, the right corner represents one of the different resampling techniques (ROS, RUS, and SMOTE), and the top corner represents the probability of no significant differences between them.

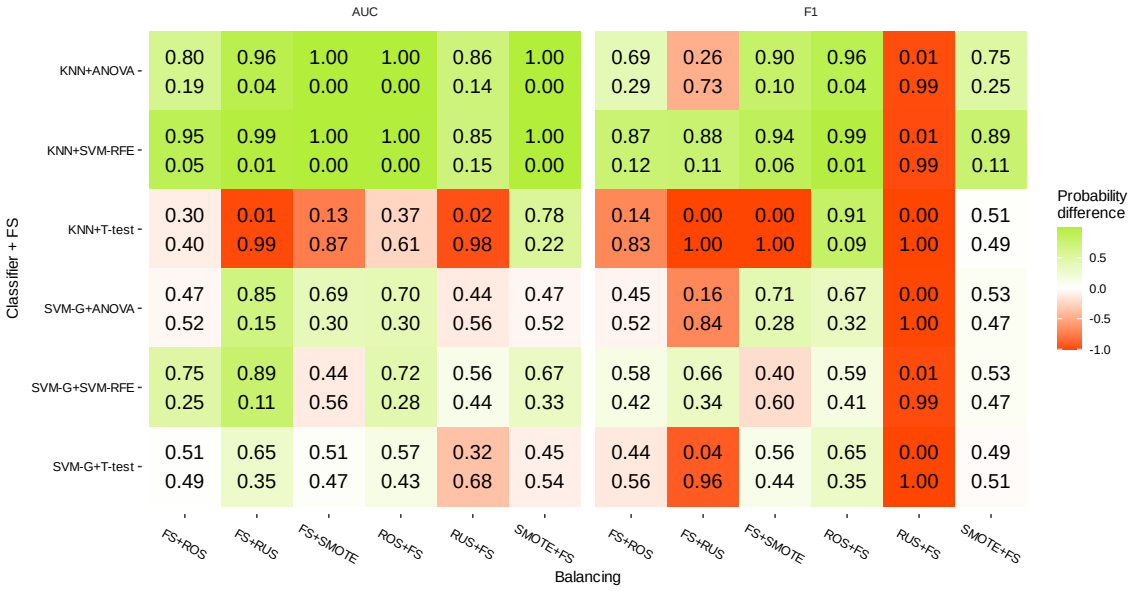


Fig. 1.7 Bayesian test results comparing each resampling strategy with not use any resampling.

Rather than using triangles, we substituted each triangle for colored tiles in the heatmaps of Figure 1.7, due to the high number of tests that were performed, to facilitate the presentation of the results. Two probability numbers are displayed in each tile: the top one is the probability that the best option is to use the balancing strategy shown at the bottom of each column, the lower one is the probability that no balancing is the best option. The color of the tile is obtained from the difference between these two values, with a scale that goes from green, when the difference is more in favor of using the balancing strategy, to red, when the difference is more in favor of not balancing. The white color represents the cases in which there is no clear winner between the two approaches. Each row is for a different combination of classifier and FS algorithm (shown on the left).

There are two heatmaps on the figure, the left one shows the results when the AUC metric is used, and the right side when the F_1-Score is used (we also computed the results for the G-Mean, but they have been left as additional material[3], since they are similar to those obtained with AUC). Note that the first three tiles on the top left of the right matrix (F_1-Score) display the values corresponding to the triangles explained in Figure 1.6.

Finally, we also applied Bayesian analysis to answer the question of whether it is better to balance before or after FS. The results of this comparison were very interesting (see Figure 1.8). The figure is divided into three blocks according to the balancing method used (RUS, ROS, and SMOTE). The results for the different combinations of classifier and selector are grouped by rows (only those combinations that gave the best results in previous

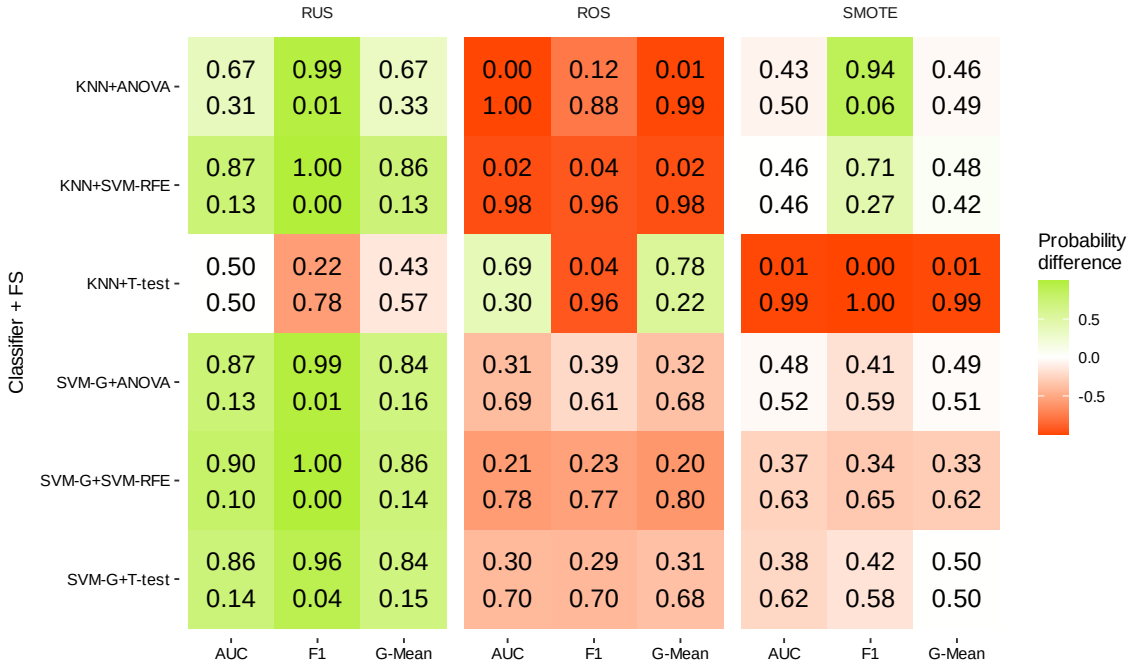


Fig. 1.8 Summary of the Bayesian tests to compare at which point it is better to apply the balancing techniques (top value corresponds to the application of balancing before FS, the lower value to the application afterwards).

experiments have been considered). The results obtained for the different performance measures are grouped in each column.

In each tile, the upper number is the probability that resampling before the FS stage is better, and the lower number the probability that resampling after the FS stage is better. The color of the tile is given by the difference between both values; if the difference is in favor of resampling before FS, the greener its color will be; if the difference is in favor of resampling after the FS stage, the color will be redder. When both strategies give similar results, the color will be close to white.

1.6 Discussion

As can be seen in Figure 1.7, in most cases the balancing strategies (top probability) are the ones that offered the best results, especially for KNN+ANOVA and KNN+SVM-RFE. The exception is when considering the RUS+FS balancing strategy, which seems to be clearly counterproductive for most combinations of classifiers and FS algorithms. The results of the combination FS+RUS are equally discouraging.

Also of interest is the behavior of the KNN+T-test combination, which only seems to benefit from the balance obtained with the SMOTE+FS strategy, if considering the AUC, or with the SMOTE+FS and ROS+FS strategies, if considering the F_1-Score.

When analyzing the order in which the balancing and feature selection methods should be applied, Figure 1.8 clearly shows that, in the case of RUS, it is better to apply balancing before feature selection. This result is confirmed by all the performance measures considered. The only exception to this general trend seems to be when using the KNN+T-test, where applying the balancing before the FS does not seem to offer many advantages, being in fact worse, if we look at the F_1 measurement. Interestingly, for KNN+T-test, this deviation from the general trend also appears for the other two balancing methods.

On the contrary, the results suggest that, in general, SMOTE and ROS perform better if applied after feature selection. This order of application is especially beneficial when the balancing method is ROS and the classifier is KNN. Although KNN+T-test is again an exception if we look at the values of the AUC and G-Mean measurements. The results observed with the combination of SMOTE with KNN+T-test are also very remarkable, suggesting that with this combination, balancing necessarily has to be done after feature selection to obtain the best results.

It is also interesting to note that the results with SMOTE appear to be halfway between using RUS and ROS.

These results extend the findings of previous studies [145, 225], which only gave as a general rule that to improve the final results balancing had to be done before the feature selection stage. According to our results, to choose the most appropriate order, one must also take into account the particular balancing method that will be used and, in some cases, even the classifier and the feature selection technique.

1.7 Conclusions

The objective of our study has been to test whether balancing improves the classification performance when used alongside a FS on unbalanced wide data.

The main novelty of this study is its much broader view than other previous studies, both in terms of the number of methods that were tested, and the number of datasets. Furthermore, when evaluating the methods, another notable novelty is that we have considered the results of various sizes of selected features, rather than restricting ourselves to a single size as in previous studies.

The conclusions we have reached, following thorough experimentation, have confirmed some of the results of previous studies. According to the Bayesian test using the 10%

best features selected, we can state that using a strategy that includes balancing generally outperforms the use of no balancing.

However, not all balancing strategies work in the same way and their performance is highly dependent on the classifier and the FS that is used. In so far as one balancing can improve the classifier or can be counterproductive. The same happens at the time of resampling (before or after), since resampling before the FS stage is generally better with RUS while resampling after the FS stage generally works better with ROS and SMOTE.

We can conclude that the best classifiers (among those used in this study) for wide data were KNN and SVM-G, while ReliefF, T-test, and SVM-RFE were the best FS algorithms. Furthermore, the best configuration was the SVM-RFE feature selector used before RUS for the SVM-G classifier.The percentage of chosen features among the best selected slightly affected the results, but not as much as using a balance method more suitable for the classifier.

Finally, the best results are obtained using RUS as the balancing method, SVM-RFE as the feature selector (applied before RUS) and SVM-G as the classifier. So this is a good combination with which to initially process wide data. If it is necessary to use any of the other classifiers included in our study, Table 1.4 summarizes the best balancing and feature selector combinations for each of them.

Based on the results of this study, we plan to use more advanced algorithms for this type of problem in future works, such as ensembles and hybrid algorithms for feature selection and other balancing algorithms. Given that the final performance of the combination of selector and balancing method (and the moment at which they are applied) may also depend on the characteristics of the dataset to which it is applied, in the future we will consider using meta-learning to analyze whether relationships can be established between the characteristics of the data and the best combination.

2

An Extensive Performance Comparison between Feature Reduction and Feature Selection Preprocessing Algorithms on Imbalanced Wide Data

This Journal paper compares the performance of feature reduction and resampling techniques in preprocesing wide data. It includes a large variety of feature reduction techniques according to linearity and supervision.

Authors: Ismael Ramos-Pérez, José Antonio Barbero-Aparicio, Antonio Canepa-Oneto, Álvar Arnaiz-González, and Jesús Maudes-Raedo.

Type: Journal

Published in: Information (JCI: Q2, SJR: Q2).

Year: 2024

Keywords: Feature Selection; Feature Reduction; Wide Data; High dimensional data; Imbalanced Data; Machine Learning.

Reference: [149]

Abstract

The most common preprocessing techniques used to deal with datasets having high dimensionality and a low number of instances—or wide data—are feature reduction (FR), feature selection (FS), and resampling. This study explores the use of FR and resampling techniques, expanding the limited comparisons between FR and filter FS methods in the existing literature, especially in the context of wide data. We compare the optimal outcomes from a previous comprehensive study of FS against new experiments conducted using FR methods. Two specific challenges associated with the use of FR are outlined in detail: finding FR methods that are compatible with wide data and the need for a reduction estimator of nonlinear approaches to process out-of-sample data. The experimental study compares 17 techniques, including supervised, unsupervised, linear, and nonlinear approaches, using 7 resampling strategies and 5 classifiers. The results demonstrate which configurations are optimal, according to their performance and computation time. Moreover, the best configuration—namely, k Nearest Neighbor (KNN) + the Maximal Margin Criterion (MMC) feature reducer with no resampling—is shown to outperform state-of-the-art algorithms.

2.1 Introduction

Within the machine learning field, the term "wide data" [104] refers to datasets containing a much greater number of features than instances. This type of data is common in bioinformatics [67, 160], and usually presents two main problems that affect the performance of learning algorithms: the curse of dimensionality and data imbalance.

The curse of dimensionality [88] refers to the difficulty of accurately generalizing problems with high-dimensional datasets when using machine learning algorithms. This increases both the processing time and required space, as well as the risk of overfitting, as it makes it difficult to distinguish meaningful patterns from noise.

Considering the low number of instances, wide data are prone to imbalance caused by the large difference in the number of instances per class [69]. The algorithms trained with this data may be biased towards the majority class, making it difficult to accurately classify data belonging to the minority classes.

One of the solutions that may mitigate these problems is the use of preprocessing techniques. In particular, the curse of dimensionality can be addressed with feature selection (FS) [157] and feature reduction (FR) [10] methods. FS methods identify and select the most informative and relevant features from a given dataset, discarding the noisy or redundant

data. In contrast, FR methods transform the original feature space into a lower-dimensional one using the information present in the original features.

Resampling methods [133] solve the imbalanced data problem through removing instances from the majority class or creating new ones for the minority class.

There are many application examples of these methods in different areas. For example, in medicine, FR improves the accuracy of epilepsy diagnosis through analyzing electroencephalography signals, avoiding invasive techniques [201]. FS has been also used for breast cancer detection [156] analyzing microarray data. Regarding engineering, the authors of [141] used Principal Component Analysis (PCA) in several predictors to remove noise from building energy consumption datasets. In another example, in fault diagnosis, FS was applied to select the best features extracted from magnet DC motors [194] or rotating machinery [232]. Furthermore, FR techniques are valuable in text mining tasks, such as document classification, e.g., in [9] PCA and Latent Semantic Indexing (LSI) were used to extract useful features for an SVM classifier.

This study aims to find the best strategies to process wide datasets, composed of combinations of FS or FR, resampling, and classifiers, through evaluating their performance and computation time. In the literature, studies comparing dimensionality reduction techniques with resampling methods have been limited to the use of FS [146]. In this case, new FR experiments are compared to the best results from [150], which extensively compared various FS, resampling methods, and classifiers on a wide dataset.

As mentioned in Section 2.2, the use of FR techniques with wide datasets requires thorough research to identify compatible approaches. For example, applying nonlinear transformations requires estimation to handle out-of-sample instances.

The scope of this study excludes the use of wrapper FS methods. As detailed in Section 2.3, their high computational cost is a crucial factor when processing wide data, as it presents a large number of features. Both the FS and FR algorithms used require the dimensionality to be set, which simplifies the comparison and visibility of the results, as the dimensionality can be set to be the same. The obtained results are analyzed to address the following objectives of this study:

1. To find an FR method that is compatible with wide data and provides a means to perform nonlinear transformations over out-of-data instances.

2. To compare the two previously mentioned types of preprocessing techniques (FR and FS) and determine which is more suitable to use on wide datasets.

3. To determine whether balancing is important while using FR methods and, if so, whether it is more convenient to use it before or after the FR step.

4. To determine the best FR method for each classifier.

While previous studies have included some comparisons of FS and FR algorithms over the same datasets [130], these evaluations are not particularly exhaustive. They predominantly focus on basic and widely used FR techniques, avoiding the use of nonlinear approaches. Furthermore, the lack of use of wide datasets makes it impossible to discern which techniques are optimal in such cases, as not all algorithms are compatible with such data.

Performing a large number of experiments to compare FS and FR approaches for wide datasets is one of the main novelties of this study. Due to the high presence of the imbalance problem in wide data, this study also focuses on resampling techniques in combination with FR and FS preprocessing methods. The code for all of the algorithms, allowing for their standardized use, can be found on *GitHub*[1].

The remainder of this paper is organized as follows. First, in Sections 2.2–2.4 the background for all the preprocessing methods used (i.e., FR, FS, and resampling) is provided. In Section 2.5, the experimental setup is detailed, while the results are shown in Section 2.6. Finally, the conclusions, limitations, and future work are presented in Sections 2.7–2.9, respectively.Finally, the conclusions and future work are presented in Section 2.7.

2.2 Feature Reduction

Feature reduction (FR) or manifold learning [10] methods are preprocessing techniques used to reduce the number of dimensions of high-dimensional datasets. This is useful for improving the performance of learning algorithms, enhancing data visualization, and facilitating feature extraction from images. The present study focuses on FR for wide data classification.

As previously explained, high dimensionality poses a significant challenge to classification algorithms, as it complicates the distinction between useful and noisy features. FR methods attempt to solve this problem through creating a new dataset with the desired dimensionality, combining all the original features. A good FR method should be able to determine the structure of the original dataset (manifold) and preserve it in a lower dimensional representation. This structure is divided into local and global structures. Preserving the local structure refers to preserving the distance of all individual points to their nearest neighbors, whereas the global structure refers to the rest of the further points. Preserving both structures simultaneously is difficult [135], and in FR methods, usually, only one is well retained.

There are some taxonomies that can be used to discriminate FR algorithms according to their behavior, and some of them can be very extensive, such as the one presented in

[1] https://github.com/Ismael-rp/feature_reduction_feature_selection_wide_data_comparison

the study [10]. The present manuscript divides FR methods according to two properties: supervised/unsupervised and linear/nonlinear. Unsupervised methods ignore the data labels when creating the new dataset, which makes them useful for clustering problems. On the other hand, supervised methods utilize data labels, allowing classes to be separated more effectively and, therefore, making these methods more suitable for classification problems.

Linear FR methods transform the data using a linear transformation which minimizes or maximizes some criteria and, at the same time, reduces the dimensionality as desired. As shown in Equation (2.1), matrix A with dimensions of $(r \times c)$ is reduced to a B matrix with k dimensions using a kernel or linear transformation K.

$$\overset{c}{\underset{r}{\left\{\left(\quad A \quad\right)\right.}} * \overset{k}{\underset{c}{\left\{\left(\quad K \quad\right)\right.}} = \overset{k}{\underset{r}{\left\{\left(\quad B \quad\right)\right.}} \tag{2.1}$$

Non-linear transformations are required to uncover the hidden manifold in nonlinear data. As most of these algorithms are unsupervised, all of the methods used in this study are unsupervised. Unlike their linear alternatives, due to their intrinsic behavior, nonlinear methods do not provide a way to reproduce the transformation on out-of-sample data; however, the authors of [215] presented a generalized and accurate approximation to solve this issue.

Based on the fact that every point in the space is linearly relocated to a new position in the lower-dimensional space under a nonlinear transformation, this linear transformation can be approximated through the following three steps. (1) Retrieve the K out-of-sample nearest neighbor instances from the training dataset. As the authors recommend, the K value was set to 5. (2) Reduce this neighbor sub-dataset to the desired dimensionality using Principal Component Analysis (PCA). (3) Using linear regression, obtain the linear projection to transpose the neighbor sub-dataset into the final positions obtained with the FR method.

The PCA and linear regression models are applied to the out-of-sample instance to be transformed. This process is repeated for each out-of-sample instance.

Unlike traditional datasets, wide data has a much greater number of columns than rows ($r << c$), preventing some of the most popular linear and nonlinear FR algorithms from being able to calculate the projection.

The FR methods used in this study are listed below, following the taxonomy mentioned above:

- Linear
  - Unsupervised
    - **Principal Component Analysis (PCA)** [142] is the most popular FR method, which reduces the feature dimensionality while maintaining the maximum data variance.
    - **Locality Pursuit Embedding (LPE)** [131] respects the local structure through maximizing the variance of each local patch according to Euclidean distances (unlike PCA, which preserves the global structure).
    - **Parameter-Free Locality Preserving Projection (PFLPP)** [52] is a parameter-free version of the Locality Preserving Projection (LPP) algorithm [72], which is a linear version of the nonlinear graph-based Laplacian Eigenmaps method [12].
    - **Random Projection (RNDPROJ)** [2] projects the data into a new random spherical hyperplane that is randomly selected using the origin. It is not a trivial computation problem.
  - Supervised
    - **Fisher Score (FSCORE)** [56] finds the projection that maximizes the ratio between each feature mean and the standard deviation of each class.
    - **Locality Sensitive Laplacian Score (LSLS)** [114] is based on the Laplacian score FS method [70]. It adjusts the Laplacian graph using the class label to simultaneously minimize the local within-class information and maximize the local between-class information.
    - **Local Fisher Discriminant Analysis (LFDA)** [183] is an improved version of the FDA-supervised FR method, which is suitable for reducing datasets in which individual classes are separated into several clusters.
    - **Maximum Margin Criterion (MMC)** [108] projects the data while maximizing the average margin between classes.
    - **Sliced Average Variance Estimation (SAVE)** [48] calculates the projection matrix by averaging the covariance of the data of each slice in which the whole dataset has been divided.
    - **Supervised Locality Pursuit Embedding (SLPE)** [234] is a supervised version of the LPE algorithm, which enhances the model using label data.

- Non-linear
  - **Classical Multidimensional Scaling (MDS)** [98] computes the dissimilarities between pairs of objects (assuming Euclidean distance). This matrix serves as the input for the algorithm that outputs a coordinate that minimizes a loss function called *strain*.
  - **Metric Multidimensional Scaling (MMDS)** [26] is a superset of the previous method. It iteratively updates the weights given by the MDS using the SMACOF algorithm, in order to minimize a stress function such as the residual sum of squares.
  - **Locally Linear Embedding (LLE)** [154] bases its performance on producing low-dimensional vectors that best reconstruct the original objects through computing the *k*NN and using this information to weight them.
  - **Neighborhood Preserving Embedding (NPE)** [71] first identifies the structure of the data neighborhood in the original space, then determines a linear subspace minimizing the reconstruction error of the local neighborhood structure [216].
  - **Locally Embedded Analysis (LEA)** [26] aims to preserve the local structure of the original data in the computed embedding space.
  - **Stochastic Neighbor Embedding (SNE)** [74] is a probabilistic approach that places the data in a low-dimensional space that optimally preserves the neighborhood of the original space.
  - An **Autoencoder** [155, 75] is a kind of artificial neural network that is trained in an unsupervised manner. The aim of the autoencoder is to capture the hidden information in the high-dimensional input space of the dataset. Autoencoders have the same number of artificial neurons in their first (input) and last (output) layers, while having less in their center layers (see Figure 2.1). During training, Autoencoders attempt to generate the same information in the output layer that is presented in the input layer. Therefore, the center layer aims to capture the intrinsic information of the dataset and, thus, can be used for feature reduction.

2.3 Feature Selection

Feature selection [157] is an alternative preprocessing approach to FR, which aims to solve the curse of dimensionality. Instead of combining all features into a completely new

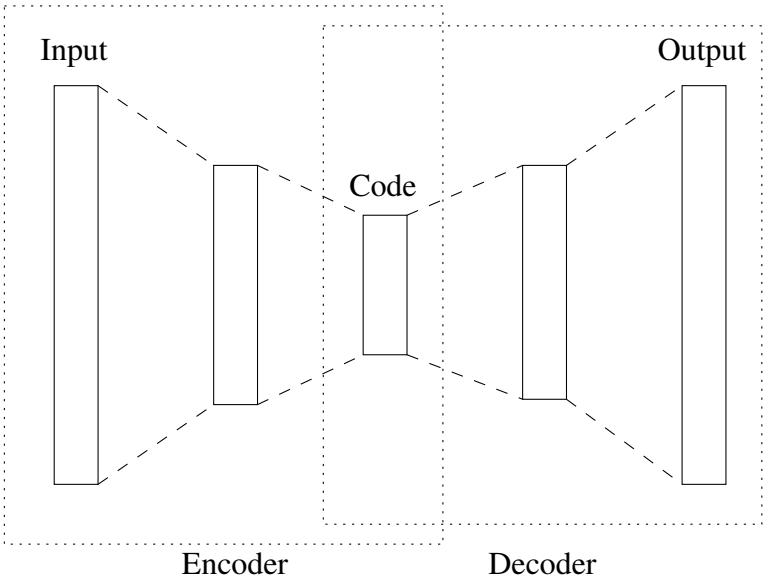


Fig. 2.1 Schematic representation of an Autoencoder.

low-dimensional dataset, FS methods identify which features are the most suitable for the classification step. These methods attempt to discard the noisy, irrelevant, and redundant features that limit the performance of the classifier.

Machine learning models that only use a few features are more interpretable than those that combine all original features into a new dataset. Generally speaking, the three types of feature selectors are filters, wrappers, and embedded:

- **Filter** methods [25] are mainly based on statistical measures. They analyze the features and rank them in an ordinal or numerical way according to their importance. Although these methods do not usually achieve the best performance for any classifier, they evade overfitting.

- **Wrapper** methods [92] perform any search algorithm to find the best feature subset for a specific classifier, according to a certain metric. Some of the most common methods are the recursive feature elimination (RFE) and genetic implementation methods. These methods obtain better performance than others; however, they tend to overfit and their computational cost is usually too high.

- **Embedded** methods [105] take advantage of the properties of classifiers such as support vector machines or decision trees to determine the importance of a feature subset. Although the selected subset can be used to train any model, it may perform better on the base classifier used to obtain it.

The method used for comparison with the best FR configuration is the SVM-RFE [63], the superior performance of which on wide data has been proven in a previous study [150] when compared with six of the most popular filters and embedded approaches (i.e., Ttest, Chi-squared, Random forest importance, ANOVA, Information gain, and ReliefF). SVM-RFE is an embedded FR method that recursively eliminates features according to their performance contribution, removing the feature whose associated weight in the current iteration is the minimum.

2.4 Imbalanced Data

There is a great chance that wide data suffer from imbalance due to the associated low number of instances. As previously stated, having a great difference in the number of instances between classes often causes a problem for the classifier [125], as its output may be biased with respect to the most-represented class. This problem can be solved through the use of any of the three types of resampling methods: removing instances from the majority class (undersampling methods), creating new instances for the minority class (oversampling methods), or combining both methods (hybrid methods).

The methods used in this study, which are the most popular ones, are described as follows:

- **Random Undersampling** (RUS) [79] removes instances randomly selected from the majority class.
- **Random Oversampling** (ROS) [79] duplicates instances randomly selected from the minority class.
- **Synthetic Minority Over-sampling Technique** (SMOTE) [32] creates synthetic instances of the minority class. For the creation of new instances, SMOTE randomly selects instances from the minority class. The feature values of the new instances are computed through interpolating the features of two instances randomly selected from the k nearest neighbors of the original instance (k being a parameter of the algorithm).

2.5 Experimental Setup

In this section, the experimental setup is explained. Some of the decisions made, such as datasets, classifiers, or balancing strategies applied, were the same as in our previous study [150].

2.5.1 Cross-Validation

For this study, 5×2-fold cross-validation was performed, where the original dataset was randomly split into 2 parts 5 times, and each part was used for training and testing; thus, every instance was used for both training and testing 5 times.

This type of cross-validation is particularly appropriate for imbalanced data as, considering the low number of instances for some classes, performing the usual 10-fold cross-validation would leave a small number of the minority class instances on the training set [49].

2.5.2 Data Sets

A total of 14 wide datasets were used to compare the methods, the main characteristics of which are listed in Table 2.1. In addition to information commonly used in the field, we include the ratio between features and instances, due to their wide nature. All of them contained two classes and their features were numeric. Every fold from the cross-validation was standardized, setting its mean to zero and standard deviation to one. The test folds were standardized using the mean and standard deviation obtained from the training folds. The imbalance ratio [61, 138] in the table was computed as the number of instances in the majority class divided by the number of instances in the minority class.

2.5.3 Dimensionality and Number of Features

For fairness when comparing FR and FS methods, it is appropriate to configure them to obtain datasets with the same dimensionality as the output. However, the number of features is limited in some of the nonlinear FR methods (SNE, MDS, MMDS, and LLE). Therefore, in such cases, the maximum dimensionality was set to the number of instances belonging to each fold.

2.5.4 Resampling Strategies

Each one of the three resampling methods detailed in Section 2.4 was used in two ways: (1) balancing before performing dimensionality reduction or (2) balancing after dimensionality reduction. Therefore, there were a total of seven strategies, including the option of not performing resampling.

Table 2.1 Data sets used in the experimental study. Datasets 1–9 were used in [237], while 10–14 were used in [109]. The column names refer to dataset name, number of examples, number of features, ratio features/examples, minority and majority class labels, min and max percentage of instances for the minority and majority classes, and imbalance ratio.

	Data Set	#Ex.	#Feat.	$\frac{\#\text{Feat.}}{\#\text{Ex.}}$	Class (min.; maj.)	%min.; %maj.	IR
1	Colon[1]	62	2 000	32.26	(Normal; Tumor)	0.35; 0.65	1.86
2	MLL_ALL[1]	72	12 582	174.75	(ALL; rem)	0.33; 0.67	2.03
3	MLL_AML[1]	72	12 582	174.75	(AML; rem)	0.39; 0.61	1.56
4	MLL_MLL[1]	72	12 582	174.75	(MLL; rem)	0.28; 0.72	2.57
5	SRBCT_1[1]	83	2 308	27.81	(1; rem)	0.35; 0.65	1.86
6	SRBCT_4[1]	83	2 308	27.81	(4; rem)	0.30; 0.70	2.33
7	Lung_1[1]	203	12 600	62.07	(rem; 1)	0.32; 0.68	2.12
8	Lung_4[1]	203	12 600	62.07	(rem; 4)	0.10; 0.90	9.00
9	Lung_5[1]	203	12 600	62.07	(rem; 5)	0.10; 0.90	9.00
10	Leukemia_BM[2]	72	7 130	99.03	(BM; rem)	0.29; 0.71	2.45
11	TOX_171_1[2]	171	5 748	33.61	(1; rem)	0.26; 0.74	2.85
12	TOX_171_2[2]	171	5 748	33.61	(2; rem)	0.26; 0.74	2.85
13	TOX_171_3[2]	171	5 748	33.61	(3; rem)	0.23; 0.77	3.35
14	TOX_171_4[2]	171	5 748	33.61	(4; rem)	0.25; 0.75	3.00

[1] https://jundongl.github.io/scikit-feature/datasets.html
[2] http://csse.szu.edu.cn/staff/zhuzx/Datasets.html

2.5.5 Classifiers

According to [23], the most popular algorithms for high-dimensional data are as follows: k-nearest neighbors (KNN), SVM-Gaussian, C4.5 trees, Random Forest, and Naive Bayes. For this reason, these five classifiers were used in this study.

2.5.6 Parameters

After preliminary experiments involving tuning the algorithm parameters, the parameters of the algorithms were set as stated in Table 2.2. This table lists all the algorithms used, as well as their corresponding parameters and implementation packages. Most parameters were set as defaults while others, such as the SVM-G classifier and SMOTE, were optimized as detailed in [150].

The SVM-G classifier was optimized using a grid parameter search, with $c = 10^9$ and $\gamma = 10^7$, resulting in optimal performance on all datasets. For SMOTE, values of k ranging from 1 to 20 were tested, and the performance was slightly better when the recommended value of 5 was used. Regarding the balancing algorithms, the balancing ratio was set to 1 for all datasets, such that the number of instances for both the majority and minority classes was the same.

As explained in Section 2.2, an algorithm to approximate the transformation on out-of-sample instances on nonlinear FR is needed; this is denoted "transformation approximation" in the table and, as the authors recommended, the parameter k was set to 5. Finally, autoencoders had a single inner layer with as many neurons as the desired output dimensionality size. Several well-known functions (linear, rectilinear uniform, sigmoidal, and tangential) were considered. The tangential function learned the fastest, requiring only 10 epochs to reach the best performance on the training fold; hence, it was selected as the activation function.

2.5.7 Metrics

Multiple metrics are commonly used to assess the performance of machine learning classifiers. In order to provide an unbiased set of performance metrics, five metrics are used in this study: Area Under the ROC Curve (AUC), F_1-Score, G-Mean, Matthews correlation coefficient, and Cohen's kappa

Some of these metrics use the confusion matrix, which consists of a 2×2 matrix (in binary classification) that summarizes the hits and misses of the classifier regarding a classification problem (see Table 2.3). The class of interest (usually the less-represented one) is called

Table 2.2 All the algorithms used in the study are grouped according to their type, including their parameters (when applicable) and their corresponding R packages, all of then have been accessed on June 2023. * The asterisk indicates that, in the method, the parameter K was set to 5 in the transformation estimator needed for the nonlinear feature reducers explained in Section 2.2.

	Algorithms	**Parameters**	**Package**
Classifier	KNN	$k = 1$	class [1]
	SVM-G	$c = 10^9$, $\gamma = 10^7$	e1071 [2]
	C4.5	Default	RWeka [3]
	Random Forest	Default	randomForest [4]
	Naive Bayes	Default	naivebayes [5]
Feature reduction Linear—Unsupervised	PCA	-	Rdimtools [6]
	LPE	Default	Rdimtools [6]
	PFLPP	-	Rdimtools [6]
	RNDPROJ	Default	Rdimtools [6]
Feature reduction Linear—Supervised	FSCORE	-	Rdimtools [6]
	LSLS	Default	Rdimtools [6]
	LFDA	Default	Rdimtools [6]
	MMC	-	Rdimtools [6]
	SAVE	Default	Rdimtools [6]
	SLPE	-	Rdimtools [6]
Feature reduction Non-linear *	MDS	-	Rdimtools [6]
	MMDS	-	Rdimtools [6]
	LLE	Default	Rdimtools [6]
	NPE	Default	Rdimtools [6]
	LEA	Default	Rdimtools [6]
	SNE	Default	Rdimtools [6]
	AUTOENCODER	epoch = 10, activation = "Tanh"	h2o
Feature selection	SVM-RFE		sigFeature [7]
Balancing	ROS	Ratio 1:1	Own impl.
	RUS	Ratio 1:1	Own impl.
	SMOTE	Ratio 1:1, $k = 5$	unbalanced [8]

[1] https://cran.r-project.org/web/packages/class/index.html
[2] https://cran.r-project.org/web/packages/e1071/index.html
[3] https://cran.r-project.org/web/packages/RWeka/index.html
[4] https://cran.r-project.org/web/packages/randomForest/randomForest.html
[5] https://cran.r-project.org/web/packages/naivebayes/index.html
[6] https://cran.r-project.org/web/packages/Rdimtools/index.html
[7] https://www.bioconductor.org/packages/release/bioc/html/sigFeature.html
[8] https://cran.r-project.org/web/packages/unbalanced/index.html

Table 2.3 Confusion matrix: true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

		Actual value	
		Positive	Negative
Pred.	Positive	TP	FP
	Negative	FN	TN

the positive class, while the other is called the negative class. The diagonal captures the hits of the classifier, while the other two cells contain the misses. These can be either a false positive (FP), when the classifier predicts positive but the actual label is negative, or a false negative (FN) in the opposite case.

To compute most of the aforementioned measures, some intermediate metrics that rely on the confusion matrix are needed:

- **Recall**, or the true positive rate, is the probability of classifying a positive instance as positive.

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

- **Specificity**, as opposed to recall, is the probability of considering a negative instance as negative.

$$specificity = \frac{TN}{TN + FP} \tag{2.3}$$

- **Fall-out**, or the false positive rate, is the probability of the probability of a false alarm occurring.

$$fall\text{-}out = \frac{FP}{TN + FP} \tag{2.4}$$

- Precision is the probability that an instance is classified as positive.

$$precision = \frac{TP}{TP + TN + FP + FN} \tag{2.5}$$

Finally, the five metrics used for the experiment are defined as follows:

- The Area Under the ROC Curve can be calculated in different ways. Although ROC can also be used to evaluate multiple possible classifier thresholds, in this study, only

one per fold is evaluated using the formula based on the true positive rate (recall) and the false positive rate (fall-out) from [57].

$$\text{AUC} = \frac{1 + recall - fall\text{-}out}{2} \tag{2.6}$$

- The F_1-Score is the harmonic mean between precision and recall.

$$F_1\text{-Score} = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.7}$$

- The G-Mean, which is widely used for imbalanced problems, is the geometric mean between recall and specificity.

$$\text{G-Mean} = \sqrt{recall \times specificity} \tag{2.8}$$

- The Matthews correlation coefficient (MCC – Do not confuse with the feature reduction method called Maximum Margin Criterion (MMC).) was originally presented by Matthews [129] and introduced to the Machine Learning community in [11]. The MCC has become a well-known performance measure of binary classification not affected by imbalanced datasets, and the authors of [41, 42] have recommended this metric over AUC and F_1-Score.

$$\text{MCC} = \frac{TN \times TP - FN \times FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.9}$$

- The Cohen's kappa measure compensates for the random hits that are usually observed in classification problems [45].

$$\text{K} = \frac{P_0 - P_e}{1 - P_e} \tag{2.10}$$

where P_0 is the ratio of success of the classifier and P_e is the ratio of success expected by chance.

Finally, average rankings [47, 59] were calculated to compare the performances of the different algorithm combination strategies. To compute the rankings, the strategy that achieved the best results on a specific dataset received a score of one, the strategy that achieved the second-best results received a score of two, and so on. In the case of a tie, the rankings of the tied methods were averaged. Average rankings were then computed through taking the average of the rankings computed on all datasets.

2.6 Results

This section attempts to answer the questions presented in Section 2.1. The performance of all feature reduction methods was compared, indicating the difference when any balancing method was used. Finally, the best algorithm configuration (i.e., FR or FS and classifier) was identified, including the feature selectors.

In order to reduce the number of figures and tables in this section, only the results for the MCC metric are shown for the advantages explained in Section 2.5.7. The Supplementary Materials[2] contain the information for all other metrics, from which it can be assessed that the results obtained for them were similar.

2.6.1 Best Feature Reducers

Table 2.4 compares the performance (average ranks) of all 90 possible configurations, obtained when combining the 5 classifiers with the 17 feature reducers plus the classifiers themselves without any preprocessing method (i.e., 18 configurations). The option without preprocessing, where the classifier was trained with all the features, was used as a baseline.

As can be seen from the table, the best configuration combined the supervised FR algorithm MMC and the KNN classifier. The second-best configuration was SVM-G using no preprocessing method. As in our previous study [150], the best results were obtained with the KNN and SVM-G classifiers.

Not all classifiers performed in the same way with all FR methods. The most suitable FR algorithm for each classifier is presented in Table 2.5 (the average ranks were computed independently for each classifier). Although the MMC and the KNN were positioned at the top of the ranking. MMC only outperformed the rest when coupled with KNN. For SVM-G, the best option was not using FR, whereas for the rest of the classifiers, FSCORE performed the best.

2.6.2 Best Preprocessing Algorithm

Having identified the best FR and classifier combination (i.e., MMC with KNN), before comparing it with the best FS method (SVM-RFE with SVM-G), we determined which of the seven balancing techniques was the best for each combination, in order to conduct a fairer comparison.

One of the objectives of this study is to compare FR and FS methods on wide datasets. In the previous section, the best FR and classifier combination (i.e., MMC with KNN)

[2]Supplementary Materials: https://www.mdpi.com/2078-2489/15/4/223

Table 2.4 Comparison of average ranks using the MCC metric, the 90 possible configurations when mixing our 5 classifiers and the 18 FR preprocessing methods, including as baseline the non-preprocessing option. The color code indicates the type of algorithm, linear unsupervised, linear supervised or non-linear unsupervised.

Classifier	FR algorithm	Avg. Rank
KNN	MMC	1.96
SVM-G	No	2.54
SVM-G	FSCORE	9.21
KNN	FSCORE	9.29
KNN	No	10.71
SVM-G	LLE	10.71
SVM-G	MDS	10.71
SVM-G	MMDS	10.71
RF	FSCORE	13.39
KNN	LLE	13.57
NBayes	FSCORE	14.79
NBayes	No	15.36
KNN	MDS	15.43
KNN	MMDS	15.43
SVM-G	NPE	15.43
KNN	PCA	16.64
SVM-G	SNE	19.36
RF	No	19.39
KNN	NPE	21.00
NBayes	LLE	21.64
SVM-G	Autoencoder	21.86
KNN	SNE	22.07
KNN	LPE	22.79
NBayes	MDS	26.50
NBayes	MMDS	26.50
C4.5	FSCORE	28.11
NBayes	NPE	29.57
NBayes	SNE	29.93
KNN	Autoencoder	30.36
C4.5	No	30.68
C4.5	NPE	30.86
NBayes	PCA	31.36
C4.5	MDS	34.14
SVM-G	LSLS	34.21
KNN	SAVE	34.36
RF	NPE	34.43
NBayes	Autoencoder	34.64
C4.5	PCA	35.21
C4.5	LLE	35.57
C4.5	MMDS	37.00
RF	Autoencoder	38.71
SVM-G	LPE	38.79
C4.5	LPE	40.07
KNN	LSLS	40.50
NBayes	LSLS	42.86
RF	LSLS	43.93
C4.5	Autoencoder	46.00
C4.5	LSLS	49.64
C4.5	MMC	49.93
RF	LPE	50.43
SVM-G	SAVE	52.14
RF	SAVE	52.36
SVM-G	LEA	53.43
RF	MMDS	56.64
SVM-G	RNDPROJ	56.64
C4.5	SNE	57.21
RF	MDS	57.64
RF	PCA	58.50
NBayes	LPE	59.43
RF	LLE	59.50
NBayes	LEA	61.29
NBayes	SAVE	62.14
RF	LEA	62.21
RF	MMC	62.50
KNN	LEA	63.21
C4.5	SAVE	65.00
KNN	RNDPROJ	67.43
NBayes	RNDPROJ	67.50
NBayes	MMC	67.93
RF	RNDPROJ	70.50
KNN	LFDA	71.93
RF	SNE	73.89
C4.5	RNDPROJ	76.07
KNN	SLPE	77.00
C4.5	SLPE	78.11
RF	LFDA	78.14
RF	SLPE	78.21
KNN	PFLPP	78.43
RF	PFLPP	78.64
SVM-G	SLPE	78.75
C4.5	LEA	79.29
C4.5	PFLPP	79.29
NBayes	PFLPP	79.29
SVM-G	LFDA	79.29
SVM-G	MMC	79.29
SVM-G	PCA	79.29
SVM-G	PFLPP	79.29
NBayes	SLPE	79.43
C4.5	LFDA	79.79
NBayes	LFDA	80.11

Table 2.5 Comparison of average ranks using the MCC metric, the 18 FR preprocessing methods, including as baseline the non preprocessing option. Each ranking is performed by a different classifier in order to detect what is more suitable. The color code indicates the type of algorithm, linear unsupervised, linear supervised or non-linear unsupervised.

a KNN

Feature reducer	Avg. Rank
MMC	1.07
FSCORE	3.86
No	4.57
LLE	5.29
MDS	5.93
MMDS	5.93
PCA	6.29
SNE	7.50
LPE	7.86
NPE	8.07
Autoencoder	10.79
SAVE	11.29
LSLS	12.57
LEA	14.57
RNDPROJ	15.43
LFDA	16.14
SLPE	16.79
PFLPP	17.07

b SVM-G

Feature reducer	Avg. Rank
No	1.07
FSCORE	3.86
LLE	4.21
MDS	4.21
MMDS	4.21
NPE	5.93
Autoencoder	6.64
SNE	7.21
LSLS	8.79
LPE	9.71
SAVE	11.57
LEA	11.86
RNDPROJ	12.43
SLPE	15.71
LFDA	15.89
MMC	15.89
PCA	15.89
PFLPP	15.89

c C4.5

Feature reducer	Avg. Rank
FSCORE	3.46
NPE	3.57
MDS	4.43
No	4.54
LLE	4.71
PCA	5.36
MMDS	5.64
LPE	6.14
Autoencoder	8.79
LSLS	9.64
MMC	9.93
SNE	11.93
SAVE	13.36
RNDPROJ	15.25
SLPE	15.82
LEA	16.04
PFLPP	16.04
LFDA	16.36

d RF

Feature reducer	Avg. Rank
FSCORE	1.29
No	2.07
NPE	3.50
Autoencoder	4.50
LSLS	6.14
LPE	7.43
SAVE	8.36
MMDS	9.36
MDS	9.64
PCA	10.14
LEA	11.00
LLE	11.00
MMC	11.36
RNDPROJ	12.93
SNE	14.93
LFDA	15.64
SLPE	15.64
PFLPP	16.07

e NBayes

Feature reducer	Avg. Rank
FSCORE	2.86
No	3.14
LLE	3.79
MDS	5.71
MMDS	5.71
NPE	6.14
SNE	6.21
PCA	6.29
Autoencoder	7.00
LSLS	9.43
LPE	12.50
SAVE	12.71
LEA	13.00
MMC	13.86
RNDPROJ	14.00
PFLPP	16.14
SLPE	16.21
LFDA	16.29

Table 2.6 Average ranks using the MCC metric of the balancing strategies for (a) the best configuration that uses an FS method and (b) the best configuration that uses an FR method.

a SVM-RFE + SVM-G

Balacing		Avg. Rank
Prior	**Posterior**	
ROS	No	3.11
No	ROS	3.11
No	SMOTE	3.46
No	No	3.57
SMOTE	No	3.86
No	RUS	4.25
RUS	No	6.64

b MMC + KNN

Balacing		Avg. Rank
Prior	**Posterior**	
No	No	3.50
No	ROS	3.50
No	SMOTE	3.50
No	RUS	3.68
SMOTE	No	4.00
ROS	No	4.29
RUS	No	5.54

was identified, whereas the best FS and classifier combination in our previous study [150] was SVM-RFE with SVM-G. In order to carry out a fair comparison between these two configurations, it was necessary to determine the most suitable balancing technique for each of them. The seven possible balancing strategies were described in Section 2.4.

The resampling technique was chosen last as, as stated in [150], it is the least influential preprocessing step, which is highly dependent on the number of selected features.

In Table 2.6, average ranks for both configurations (including the no-balancing approach) are shown. For the FR method, there was a tie between not using any balancing at all and using resampling (ROS or SMOTE) as a post-balancing method. For the sake of simplicity, the option of not balancing was chosen.

For the FS method, there was again a tie using ROS balancing either before or after preprocessing. The selection of either of the two options was arbitrary and, so, the initial balancing option was used.

With the aim of assessing whether or not the differences between these two configurations are significant, Bayesian tests [18] were used to compare them one vs. one.

This hypothesis test is conducted to compare two different methods, obtaining the probability that one is better than the other, or that both have practically equivalent performance. In this test, this equivalence is represented by the so-called region of practical equivalence (ROPE). A parameter for the ROPE is needed, in order to declare the size of this region. If the difference between two parameters is in the ROPE, it is considered that there is no significant difference between them. If this area is too big, the test indicates that there is no statistical difference between the methods.

The results of performing the Bayesian tests when setting the ROPE to 0.01 for comparison of the best configurations under each of the five metrics are provided in Figure 2.2.

The left side of the triangles corresponds to the use of an FS configuration (balancing using ROS before the FS with SVM-RFE and using SVM-G as a classifier), whereas the right side corresponds to the FR alternative (using the FR method MMC with the KNN classifier). For three out of the five metrics (F_1-Score, MCC, and Kappa), the FR combination performed significantly better than the FS combination; meanwhile, the other two (AUC and G-Mean) did not show any significant differences. As none of the tests supported the left side (SVM-RFE) and most of the tests suggested that the right side (MMC) performed better, it can be determined that, for these datasets, the FR option was the best one.

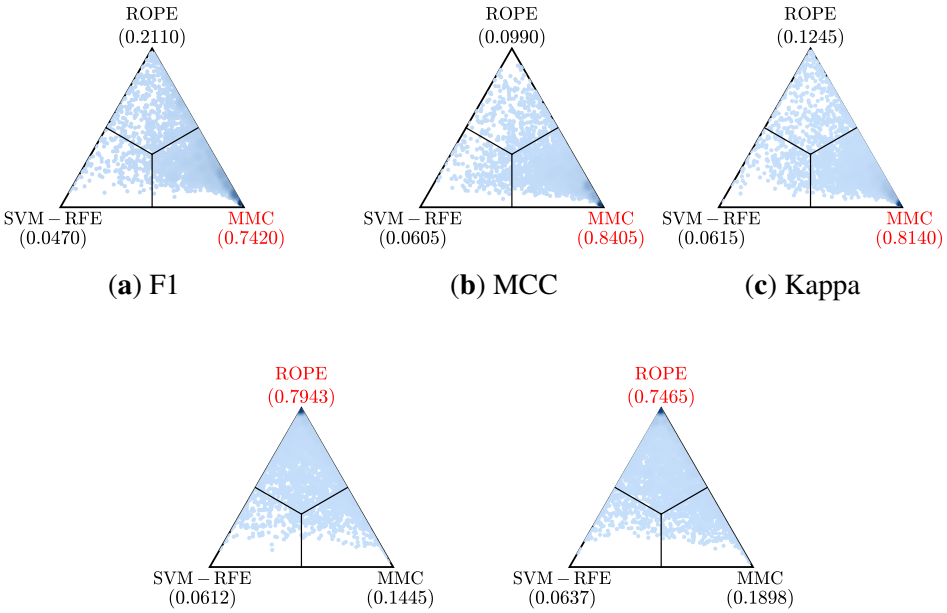
The execution times of the 18 preprocessing methods (i.e., 17 FR methods and the best FS method) are shown in Figure 2.3, sorted according to the average time needed to process all of the dataset folds. The best FS method (SVM-RFE) and the preprocessing methods that performed the best for at least one classifier are highlighted in blue. Their averages are listed in Table 2.7. The high variance between execution times obtained for the same preprocessing method was due to the varying size of the datasets. The most accurate methods (MMC and SVM-RFE) were also among the fastest, being considerably faster than the worst method; however, MMC was generally slower than SVM-RFE. Finally, the FSCORE method, which showed promising performance relative to the baseline, completed processing within a few seconds.

2.7 Discussion and Conclusions

In this research paper, the question of whether FR or FS performs better when considering wide data was answered. It was empirically proven that the best configuration of an FR algorithm determined in this study (MMC + KNN) outperformed the best FS algorithm (SVM-RFE + ROS + SVM-G) in the previous literature [150].

During the search for the best FR algorithm for use on wide data, it was found that not all the FR methods perform in the same way for all the classifiers. In this study, the best FR method was the MMC for KNN and FSCORE for the other classifiers, except SVM-G, which ranked second after the no-preprocessing stage.

As a general recommendation, our suggestions for practitioners dealing with wide data problems can be summarized as follows: (1) Start with the FSCORE method, as it is very fast and shows good performance. (2) When higher classification performance is desired, the MMC + KNN or ROS + SVM-RFE + SVM-G configuration can be used, with MMC providing better performance without the need for resampling and evading the time required for parameter tuning in the SVM-G classifier. Nevertheless, it must be noted that MMC has a higher processing time than SVM-RFE.



(**a**) F1 (**b**) MCC (**c**) Kappa

(**d**) AUC (**e**) G-Mean

Fig. 2.2 Results of performing Bayesian tests for each of the five metrics comparing the best FS and FR configurations. The best FS configuration is represented on the left side (balancing using ROS before selecting the features with SVM-RFE and using SVM-G as classifier), whereas the best FR configuration is shown on the right side (reducing dimensionality with MMC and KNN as classifier).

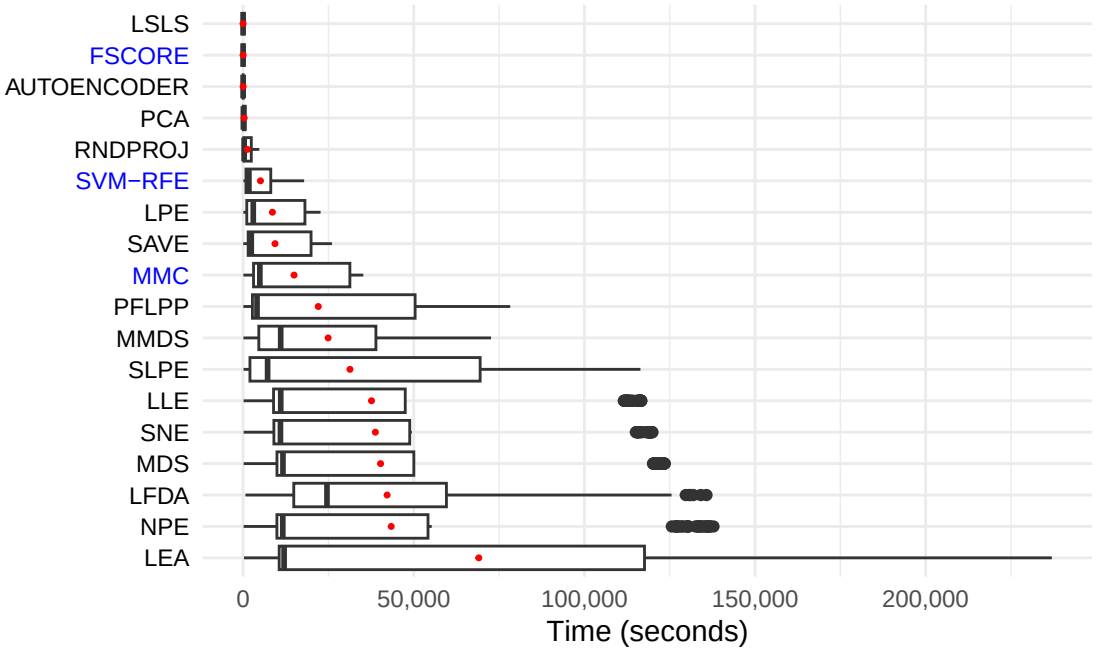


Fig. 2.3 Box plots in the *y*-axis with the 18 preprocessing methods shown alongside the time taken to process each fold (in seconds). The methods are sorted according to their average execution time (shown as a central red dot). The preprocessing methods with highest-ranking performances for at least one classifier are highlighted in blue, and the best FS method (SVM-RFE) is also highlighted.

Table 2.7 Average execution time (hours, minutes, and seconds) for each preprocessing method to compute every fold, sorted in ascending order.

Preprocessing	**Time**		
	H.	**M.**	**S.**
LSLS			3
FSCORE			3
AUTOENCODER			15
PCA		5	16
RNDPROJ		21	17
SVM-RFE	1	24	56
LPE	2	23	15
SAVE	2	36	4
MMC	4	8	58
PFLPP	6	7	9
MMDS	6	55	17
SLPE	8	42	3
LLE	10	27	7
SNE	10	46	2
MDS	11	11	34
LFDA	11	43	20
NPE	12	3	17
LEA	19	11	8

Different classifiers may benefit from different preprocessing methods. This study also provides performance results for FR methods, which are some of the most popular algorithms for high-dimensional data. If any of the classifiers used in this study are more suited to a specific problem, it is suggested that the results are checked to determine the best preprocessing method for that particular classifier.

2.8 Limitations

As with the majority of studies, the design of the current research is subject to limitations. When assessing machine learning strategies on extensive datasets, the limited number of instances for testing raises the risk of selecting a sub-optimal model as the top performer [101]. We have attempted to address this problem by using a relatively large number of datasets and applying 5×2-fold cross-validation (*i.e.* splitting the dataset into 2 folds and repeating the process 5 times).

Although we have found a very effective model configuration, the algorithms used during both studies are a relatively small representative collection of each of the applied domains (FS, FR, resampling, and classification). Therefore, there could be other configurations equally even more suitable for these and other broad data problems.

Finally, it is important to remember that the data used in the experimentation are exclusively from microarrays. Therefore, their applicability in other contexts may produce different results. Nonetheless, we consider that these results can serve as a reference.

2.9 Future Work

Different approaches to reduce the data dimensionality of wide data, such as wrapper FS methods or combinations of FR and FS, can be explored in future work.

A future research direction is the use of FR or FS in other areas, for example, in meta-learning, where data characteristics are studied in order to determine the most suitable preprocessing and classifier algorithms.

Another unexplored area is the analysis of feature reduction and feature selection methods in semi-supervised contexts, where there are only a few labeled instances and usually a large number of unlabeled instances [189]. These preprocessing algorithms may take advantage of the manifold assumption that the data lie on or near a low-dimensional manifold within the high-dimensional input space.

3

Systematic review of semi-supervised feature selection techniques

This chapter presents a systematic review of Semi-Supervised Feature Selection algorithms based on 101 studies. The review provides a comprehensive overview of the field and proposes a new taxonomy.

Authors: Ismael Ramos-Pérez, Álvar Arnaiz-González, Jesús Maudes-Raedo and Juan J. Rodríguez

Type: Journal

Published in: Under revision

Keywords: Semi-Supervised Learning, Feature Selection, Machine Learning, Systematic review.

Abstract

Semi-Supervised Feature Selection (SSFS) techniques combine the advantages of Feature Selection (FS) and Semi-Supervised Learning (SSL), thereby reducing the problem of high-dimensional data in SSL contexts. The aim of this review is to provide a comprehensive overview of work within that field, helping researchers to understand the current state of the

art and to identify future research directions. An updated taxonomy for SSFS is presented, based on 101 studies, published up until the end of 2023. The new FS taxonomy expands upon the existing one, by dividing wrappers into two types: classical search wrappers, which search for the best subset of features, and pseudo-labelling wrappers, which pseudo-label instances to increase the available data. The information extracted from those studies includes descriptions of the SSFS algorithms, statistics on the papers that were reviewed, and decisions on experimental design.

3.1 Introduction

Semi-supervised Feature Selection (SSFS) algorithms are machine-learning techniques used to solve the so-called "curse of dimensionality" problem in Semi-Supervised Learning (SSL) problems.

The curse of dimensionality refers to a problematic situation where the input data have a very large number of dimensions. Within that large number of variables, many may be redundant or irrelevant to the solution of a specific problem. That issue can decrease the performance and generalisation capabilities of the models. It is addressed through the use of Feature Selection (FS) algorithms that can select the most relevant features of the problem, either by searching for the best subset of features, or by ranking the features according to their relevance.

Moreover, only a small number of instances are labelled in SSL, unlike in supervised learning where all the instances are labelled. A limitation that constrains supervised models to the use of labelled instances, reducing their potential performance. SSL algorithms are designed to take advantage of the unlabelled instances and their associated information, to improve the performance of the final models.

Semi-Supervised Feature Selection (SSFS) has been effectively applied for the enhancement of model performance across several problem domains. Examples from the literature include its use in video processing, as seen in [123] for face recognition tasks, and image and multimedia annotation tasks as demonstrated in [77], and in [221], respectively. It has applications within such fields as quantitative/qualitative structure activity relationships data analysis (chemistry) [68], ribonucleic analysis (biology) [80], and tomato maturity analysis (agriculture) [81]. Furthermore, its potential extends to physics applications, notably for the analysis of very high-resolution satellite images, as presented in [38].

There have been two reviews of SSFS in the past: the first review [8], in 2016, was focused on FS for gene selection and included 10 SSL-related algorithms. The second one [169], published in 2017, was focused exclusively on SSFS and included 28 algorithms.

The current systematic review of 101 studies includes 103 new SSFS algorithms from the start of this topic until the end of 2023. A number of studies that is significantly higher than in the past two reviews, and the studies themselves are more recent, thereby providing an updated overview of this field of study.

Moreover, the SSFS taxonomy proposed in the previous review [169] is updated in this systematic review. The information extracted from 99 journal and conference papers includes details of the SSFS algorithms, statistics on the papers, and common practices regarding their experimental designs.

A comprehensive overview of the field is provided in this review, so that researchers can gain a better understanding of the current state of the art. Its results will help them to select algorithms for particular problems, to compare results, and to identify future research directions.

The structure of the review is as follows: the review methodology is detailed in Section 3.2, general concepts related to SSFS are introduced in Section 3.3, and the new taxonomy that is proposed and a classification of the 103 SSFS algorithms is presented in Section 3.4. A summary of the information extracted from the sample of papers is provided in Section 3.5, and the conclusions are presented in Section 3.6, together with future lines of research.

3.2 Review methodology

The process used for searching and filtering relevant papers on the topic of SSFS is described in this section. A summary of the process can be found in Figure 3.1.

First of all, the search for articles was restricted to the *Scopus* search engine. Given the large volume of items provided, no further results from other search engines were included. This search was performed selecting elements up to 31st of December of 2023 using the following query:

```
TITLE(("feature selection" OR "feature ranking") AND ("semi-supervised"
OR "semisupervised")) OR KEY(("feature selection" OR "feature ranking") AND
("semi-supervised" OR "semisupervised")).
```

This query found articles containing terms related to FS and SSL in the title or keywords. From this initial search, 455 items were obtained, which were divided into two groups. The first group contained 257 items, excluding conference proceedings, which belonged to a second group of 198 elements. Different filters were applied to each group.

Several filters were applied. First, all papers not written in English were filtered out, resulting in the removal of 28 papers.

The next filter implied a close study of all the abstracts, so as to remove any that were not completely focused on SSFS. Those that presented new strategies and that contained more than one new FS, such as a classifier or other pre-processing algorithms, were also removed. For clarity, the studies that were included had to detail a new SSFS method that had been evaluated in specific experiments. With this filter, a total of 125 articles were removed.

After analysing the full text, a final filter was applied to weed out articles of insufficient quality. The most common reasons were not comparing the proposed methods with any others, not explaining the algorithms sufficiently well, or not using sufficient experimental data. In that step, 11 articles were removed.

Conference papers on aspects not cited in any of the previously selected papers were removed (190). So, only 8 conference papers were included in the sample.

Finally, the sample for this review included 101 papers in total, of which 91 were journal articles proposing new algorithms, 8 were conference papers, and 2 were previous reviews.

The information extracted from those articles, including details on the SSFS models, is presented in Section 3.4, and information on the experiments is discussed in Section 3.5.

3.3 General concepts

The general concepts of FS, SSL, and the combination of both fields are introduced in this section. The main mathematical notions are also discussed towards the end of the paper.

3.3.1 Feature selection

FS [157] is a pre-processing task that identifies a subset of features within a dataset that provides the best performance. The data are analysed to find the most informative features, while avoiding irrelevant, redundant, and noisy ones. Feature elimination reduces the dimensionality of the dataset, while aiming to increase the accuracy of the model, to decrease its training time, and to enhance the ease with which the results may be interpreted.

The classic taxonomy [24] for FS in supervised learning is divided into three main categories:

- **Filter** methods [25] are used to analyse features, usually with statistical measures, regardless of the final learning model that is used. Popular examples are the Pearson correlation, the Chi-square test or the analysis of the variance (ANOVA). These methods rank the features by their criteria, sometimes assigning them a number. As they are the fastest type of FS, they are more suitable for big data [144]. However, they normally analyse the feature importance singularly alone and not as a group, thereby

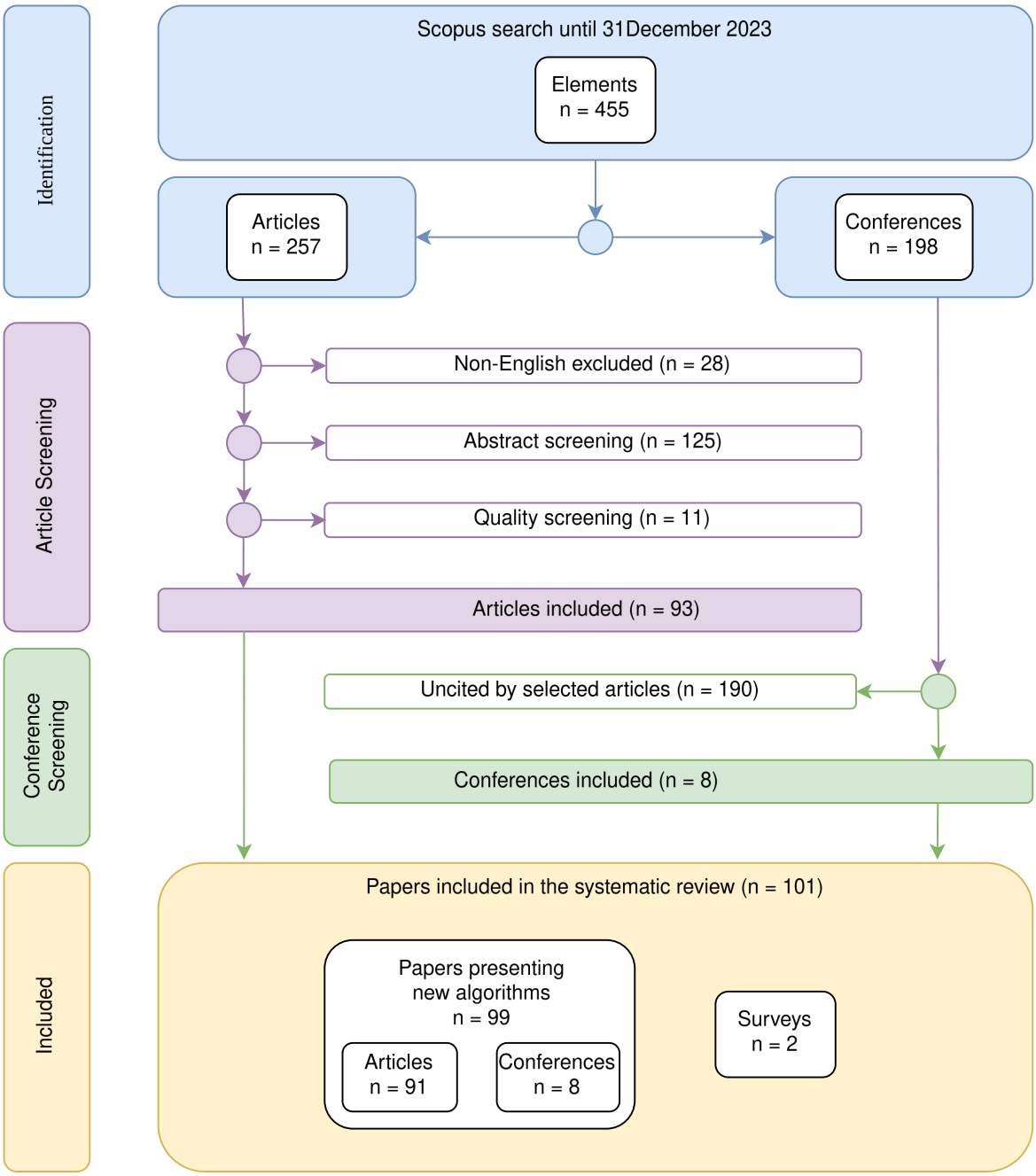


Fig. 3.1 Diagram with the process used for searching and filtering relevant papers on the topic of SSFS.

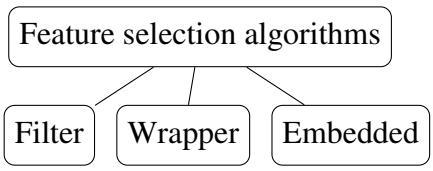


Fig. 3.2 Taxonomy of feature selection algorithms [157].

avoiding multi-correlational interaction. However, as they provide a global solution that is not optimised for any specific model, they cannot guarantee an optimal result.

- **Wrappers** methods [93], through a search strategy, explore the space of feature subsets to find the subset that offers the best performance for a particular learning algorithm. While wrappers are slower than any other FS types, as the learning algorithm has to be trained before it can explore the subset and evaluate its performance, their performance is generally superior. However, there is a higher risk of overfitting, due to the exhaustive search process. Some of the most popular include forward and backward search, and Genetic Algorithms (GA).

- **Embedded** methods [117] use part of the training process of a model to obtain the most relevant features. Some of those models are Random Forest Importance (RFI) and Support Vector Machine-Recursive Feature Elimination. In contrast to wrapper methods, the model used for FS is not necessarily the same as the final model. These techniques present reduced risks of overfitting and show faster processing times compared to wrappers. However, they are slower than filters.

3.3.2 Supervision types

One of the best-known taxonomies in machine learning divides the problems according to the type of supervision. Supervision can be understood as the availability of labels (a feature of interest) in the training data. When all data instances contain labels, in the case of classification problems, they are used to learn differences between classes using supervised algorithms such as decision trees and Support Vector Machines (SVM). The metrics used to evaluate those models are based on the confusion matrix, which shows the number of correct and incorrect predictions for each class. Some of the most common metrics are accuracy, F_1-Score, and the Matthews Correlation Coefficient (MCC).

On the other hand, if the data have no labels, the problem is described as unsupervised [78]. In this case, clustering algorithms such as *k*-means and DBSCAN are used to assign each instance to a group, with no need for pre-existing labels. This approach is useful when the underlying structure of the data must be explored. Metrics such as the Adjusted Rand Index or Silhouette Coefficient can be used to evaluate the quality of the clusters that are obtained.

Finally, the dataset in SSL [189] is composed of instances with and without labels. Usually, the number of labelled instances is much smaller than the number of unlabelled ones. Algorithms such as co-training [208] and S3VM [19] use both labelled and unlabelled data to optimise model accuracy. SSL is useful when generating labels for all data might be

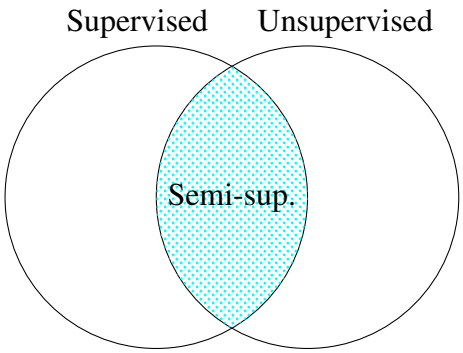


Fig. 3.3 Supervision categories in machine learning.

costly or unfeasible. In those sorts of problems, the metrics used in supervised learning can also be used to evaluate the models.

3.3.3 Semi-supervised learning

As previously noted, SSL approaches seek to extract information from labelled and unlabelled data. For this purpose, the following assumptions [189] are thought to be satisfied:

- **Smoothness assumption**: Nearby instances within the same space share the same label.
- **Low-Density assumption**: Decision boundaries are in regions of low sample density.
- **Manifold assumption**: Instances belonging to the same manifold belong to the same class.
- **Cluster assumption**: There are multiple hidden manifolds in the data, and each one describes the patterns of a class.

A popular taxonomy of the SSL classification algorithms was suggested by [189]. Despite the fact that it has many subdivisions, only the top levels are explained in the next section.

Inductive and transductive methods

The first split of the taxonomy differentiates between inductive and transductive methods. On the one hand, inductive methods train a model using unlabelled and labelled data. Once the model is trained, it can be used to predict the labels of new instances that have never before been seen. On the other hand, transductive methods do not seek to return a trained model to be used to predict new instance labels. Instead, they predict the label of the unlabelled

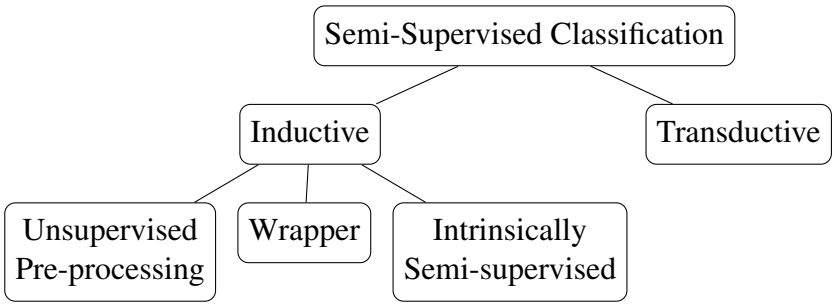


Fig. 3.4 Taxonomy of Semi-supervised classification [189].

instances available during training, after processing a global overview of all the data. If new instances need to be labelled, they must be included in the training data set, and the algorithm must be retrained from scratch. Graph-based models are the norm [181].

Unsupervised pre-processing

These methods extract some information from the data before the training step to facilitate the learning model task: Feature extraction, which extracts useful information for the classifier. Cluster-then-label, which pre-clusters the data before labelling. And pre-training, which initialises the learning model parameters.

Wrapper

Wrapper methods train learning models using labelled instances to pseudo-label the unlabelled ones. They usually have several steps where instances are labelled according to the confidence of the predictions. However, if an error (*i.e.*, a mislabelled instance) is made in one of the initial steps, this error could incrementally affect the following steps (*i.e.*, adding noise to the model). There are several groups in which wrapper methods fit, depending on the number and type of classifiers and the labelling strategy, such as single-view, multi-view, and boosting-based. Some of the most popular are self-training [152] and tri-training [236].

Intrinsically Semi-Supervised

Unlike the two previous methods, intrinsically semi-supervised methods process the labelled and unlabelled data by themselves. They usually optimise a function, as with neural networks, rather than adapting the data to a common classifier by applying pseudo-labelling or pre-processing steps.

3.3.4 SSFS filter concepts

Many SSFS filters are based on the Spectral graph theory [182], a branch of mathematics focused on understanding the properties of graphs through their associated matrices. The Laplacian matrix is one of the most common matrices studied in this field.

Laplacian matrix

The Laplacian matrix is constructed by subtracting the adjacency matrix, which represents the connections between nodes or instances (with 1 indicating a connection), from the degree matrix, which contains the (number of connections) degree of each node. Its formula is represented as $L = D - A$, where L is the Laplacian matrix, D is the degree matrix, and A is the adjacency matrix. The formula is shown in Equation 3.1 and an illustrative example is provided in Equation 3.2.

$$L_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\underset{D}{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}} - \underset{A}{\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}} = \underset{L}{\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix}} \tag{3.2}$$

The use of the adjacency matrix varies, depending on the specific objectives of each algorithm. For example, the matrix is adapted in SSFS to calculate the distances of labelled and unlabelled instances in different ways. The most popular FS metric which takes advantage of the Laplacian matrix properties is the Laplacian Score.

Laplacian score

The Laplacian Score [70] (LS) is an unsupervised score used for FS that reflects the locality-preserving power of each feature. It is based on the premise that if two data points are close, then they are likely to be related (Smoothness assumption).

This score is based on the Laplacian matrix. However, instead of considering the connections between nodes within the adjacency matrix as simply 1, Equation 3.3, a gradual

equation, is used where t is a suitable constant. The full formula is expressed in Equation 3.4 where $\widetilde{f_r}$ is defined by Equation 3.5 and f_r represents the r-th feature whose score is calculated.

$$A_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}} \tag{3.3}$$

$$L_r = \frac{\widetilde{f_r}^T L \widetilde{f_r}}{\widetilde{f_r}^T D \widetilde{f_r}} \tag{3.4}$$

$$\widetilde{f_r} = f_r - \frac{f_r D 1}{1^T D 1} \tag{3.5}$$

Pairwise constraints

Some of these SSFS algorithms are based on pairwise constraints, which specify whether a pair of data samples belong to either the same class, known as must-link constraints (ML), or to different classes, referred to as cannot-link constraints (CL). These constraints are defined as:

$ML = \{(x_i, x_j) | x_i \text{ and } x_j \text{ belongs to the same class}\}$

$CL = \{(x_i, x_j) | x_i \text{ and } x_j \text{ belongs to a different class}\}$

The classic algorithm constraint score 1 (C1) and constraint score 2 (C2) [228] are used in supervised learning. These algorithms use ML and CL constraints to rank the features, assuming that the most informative features keep pairs of objects of the same class relatively close, while keeping those of different classes far apart.

Both approaches calculate the score using Equation 3.6 and Equation 3.7. The second option provides a lambda parameter that balances the weight of both constraints (1 by default).

$$C_r^1 = \frac{\sum_{(x_i,x_j)\in M}(f_{ri} - f_{rj})^2}{\sum_{(x_i,x_j)\in C}(f_{ri} - f_{rj})^2} \tag{3.6}$$

$$C_r^2 = \sum_{(x_i,x_j)\in M}(f_{ri} - f_{rj})^2 - \lambda \sum_{(x_i,x_j)\in C}(f_{ri} - f_{rj})^2 \tag{3.7}$$

The formulations can be optimised using the spectral graph, as shown in Equation 3.8 and Equation 3.9, where the adjacency matrices (L^M and L^C) are described in Equations 3.10 and 3.11.

$$C_r^1 = \frac{f_r^T L^M f_r}{f_r^T L^C f_r} \tag{3.8}$$

$$C_r^2 = f_r^T L^M f_r - \lambda f_r^T L^C f_r \tag{3.9}$$

$$A_{i,j}^M L = \begin{cases} 1 \text{ if } (x_i, x_j) \in M \text{ or } (x_j, x_i) \in M, \\ 0 \text{ otherwise} \end{cases} \tag{3.10}$$

$$A_{i,j}^C L = \begin{cases} 1 \text{ if } (x_i, x_j) \in C \text{ or } (x_j, x_i) \in C, \\ 0 \text{ otherwise} \end{cases} \tag{3.11}$$

This theory will serve as a basis for other algorithms in SSL, such as constraint score 3 (C3) and constraint score 4 (C4) [85].

Sparse models

The use of the sparse model is an effective strategy to improve the efficiency of some SSFS [110]. These models apply the p-norm as a constraint, which forces the model weights to be as close to zero as possible. A feature that makes them simpler and facilitates interpretation and generalisation, similar to the behaviour of the Lasso model.

Equation 3.12 applies the L_p norm to a vector where the higher the vector values, the larger the number that is returned.

$$\|x\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}} \tag{3.12}$$

The higher the p parameter value, the higher the penalty of the components, the absolute value of which will be larger. The most common values for p are 1, which gives rise to the well-known Manhattan norm or L1 norm, and 2, which corresponds to the Euclidean or L2 norm. However, it was observed in this review that authors often used the value $1/2$. That normalisation can be applied to a two-dimensional A matrix, using the $L_{p,q}$ Equation 3.13, where $a_{i,j}$ represents the matrix element found in the position marked by the indices i and j.

$$\|A\|_{p,q} = \left(\sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^p \right)^{\frac{p}{q}} \right)^{\frac{1}{q}} \tag{3.13}$$

3.4 Taxonomy for SSFS algorithms

As mentioned above, SSFS addresses the high dimensionality challenge in semi-supervised contexts. Those strategies combine approaches linked to both FS and SSL problems.

A new taxonomy for SSFS is presented in this section, alongside a short description and categorisation of all the algorithms covered in this study. The categorisation is based on the taxonomy established in Section 3.4.1, classifying the data primarily into filters, wrappers, and embedded methods.

Algorithm classification is complex, due to the frequent conceptual overlaps between them. Many could be classified into multiple categories, particularly filters, most of which use the Laplacian matrix or are sparse. Therefore, according to our assessment or the emphasis provided by various authors, each method has been assigned to the group that highlights the most significant contribution. However, when the approach is consistent, the methods belonging to the same author or group of authors are grouped together. This taxonomy may need future updates, due to the publication of new and original types of algorithms.

It is important to note that some methods do not have a specific name, as the authors never provided one.

In addition, in Table 3.1, a summary of all 103 SSFS algorithms is presented in the 99 journal and conference papers included in this literature review. A list containing a brief description of all the algorithms is provided in the supplementary material (Appendix A).

3.4.1 Taxonomy

A taxonomy was developed in the previous review [169], based on the types of algorithms found in 28 references up until early 2016. That taxonomy was divided into two perspectives: the FS view, in which the traditional taxonomic structure was used in the top layer, and the second layer, in which the types were classified according to the SSFS approaches that were found.

In the present review, as explained in Section 3.2, all items up to December 31, 2023, were included. According to the initial search described in the same section, the total number of published items was 455.

After reviewing the papers, it was decided to simplify the previous taxonomy, considering using only the FS view more suitable for this field and showing the other aspects that are not exclusive as sub-levels. These secondary aspects are more likely to be mixed than the top levels.

The proposed taxonomy is shown in Figure 3.5. In that taxonomy, there are two wrapper types: search wrappers, the common FS wrappers based on search techniques; and, pseudo-labelling wrappers, which use the SSL taxonomy wrappers like self-training and co-training methods to pseudo-label instances to increase the amount of data that could be used by other FS techniques. By their nature, those techniques are not used alone.

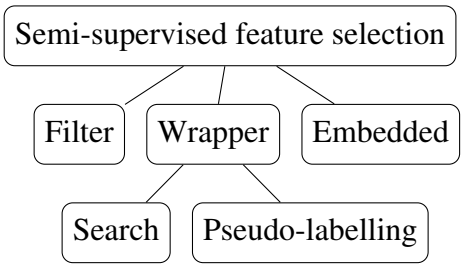


Fig. 3.5 Proposed taxonomy for SSFS, the new class "Pseudo-labelling" is included.

3.4.2 Classification of algorithms

In Table 3.1, a summary of all the SSFS algorithms is presented, including their classification in the SSFS topology, year of publication, task (*i.e.*, classification, regression), and the links to the available code.

In addition to the primary sub-types of filter algorithms, based on techniques such as Laplacian score, pairwise constraints, and sparse models, as detailed in Section 3.3.4, those algorithms also include other sub-types, such as adaptive graph-based models, where the graph structure can dynamically evolve to capture the underlying data structure more efficiently. Methods reliant on regression techniques and clustering incorporate various metrics such as the Fisher score, the Hessian matrix, Bayesian methodologies, Minimum Redundancy Maximum Relevance (MRMR), FR, Relief, and fuzzy logic principles. Some algorithms integrate the concept of universum data [200], which represents instances not belonging to any class of interest. Moreover, SSFS tagged as "other" are algorithms that do not fit into any previously defined categories.

Wrappers are sub-classified into the aforementioned search and pseudo-labelling, whereas embedded methods are classified according to the inner classifier, such as trees or SVM.

In Figure 3.6, the number of algorithms according to the task or label is presented. It can be seen that most of the algorithms are designed for classification rather than regression problems, with significantly fewer for multi-label and multi-target problems. It may be noted that only one algorithm supports both classification and regression.

Table 3.1 Summary of the SSFS algorithms presented in the sample of papers. Types are selected according to the new taxonomy, and sub-types, according to the properties of the secondary algorithm.

Reference	SSFS Name	Year	Type	Sub-type	Output
[231]	LSDF[1]	2008	Filter	Laplacian score based	Classification
[36]	ALDS	2010	Filter	Laplacian score based	Classification
[184]	SSMCFS	2015	Filter	Laplacian score based	Classification
[139]	RFR	2021	Filter	Laplacian score based	Classification
[35]	SSML	2022	Filter	Laplacian score based	Classification
[164]	SSFSM-DTI[2]	2022	Filter	Laplacian score based	Classification

Continued on next page

Table 3.1 – continued from previous page

Reference	SSFS Name	Year	Type	Sub-type	Output
[51]	SSLS	2013	Filter	Laplacian score based	Regression
[168]	S2FSGL	2017	Filter	Laplacian score based	Regression
[171]	S3FSGL	2018	Filter	Laplacian score based	Regression
[95]	-	2019	Filter	Laplacian score based	Regression
[140]	SSNDI	2020	Filter	Laplacian score based	Regression
[81]	SIDLS	2021	Filter	Laplacian score based	Regression
[85]	C4	2011	Filter	Pairwise constraints	Classification
[73]	CSFS	2011	Filter	Pairwise constraints	Classification
[33]	PCDLRD	2023	Filter	Pairwise constraints	Classification
[167]	SSCS	2017	Filter	Pairwise constraints	Regression
[111]	SFS	2022	Filter	Pairwise constraints	Regression
[4]	3-3FS	2021	Filter	Pairwise constraints	Multi-label
[15]	CLS	2011	Filter	Laplacian score + Pairwise constraints	Classification
[16]	CSFSR	2014	Filter	Laplacian score + Pairwise constraints	Classification
[214]	CCLS	2016	Filter	Laplacian score + Pairwise constraints	Classification
[5]	S-CLS	2016	Filter	Laplacian score + Pairwise constraints	Multi-label
[124]	ISR	2018	Filter	Sparse model	Classification
[77]	-	2021	Filter	Sparse model	Regression
[170]	GS3FS	2020	Filter	Sparse model	Classif. or Regres.
[175]	SFS-BLL[3]	2023	Filter	Sparse model	Multi-label
[123]	OGE-SFS	2018	Filter	Adaptive graph model	Classification
[222]	ALFS	2019	Filter	Adaptive graph model	Classification
[235]	SADA	2021	Filter	Adaptive graph model	Classification
[53]	Sr-SemiDFS[4]	2022	Filter	Adaptive graph model	Classification
[34]	RDMRS2FS	2022	Filter	Adaptive graph model	Classification
[103]	ASLCGLFS	2022	Filter	Adaptive graph model	Classification
[173]	MASFS	2020	Filter	Adaptive graph model	Regression
[203]	SFS-LARLRM	2021	Filter	Adaptive graph model	Regression
[30]	CSFS	2014	Filter	Adaptive graph model	Multi-label
[126]	SFAM	2021	Filter	Adaptive graph model	Multi-label
[102]	AGLRM	2022	Filter	Adaptive graph model	Multi-label
[226]	EMSFS	2023	Filter	Adaptive graph model	Multi-label
[65]	S2FS2R	2015	Filter	Regression based	Classification
[180]	FSHR	2016	Filter	Regression based	Classification
[233]	sSelect	2007	Filter	Clustering	Classification
[44]	SSFC	2019	Filter	Clustering	Classification
[153]	PCFS	2020	Filter	Clustering	Classification
[80]	LPFS[5]	2021	Filter	Clustering	Classification
[68]	SSFLS	2018	Filter	Fisher Score	Regression
[76]	LGDF	2013	Filter	Fisher Score	Multi-label
[174]	HFSL	2015	Filter	Hessian	Classification
[172]	SMHFS	2019	Filter	Hessian	Classification
[166]	HSFSGU[6]	2023	Filter	Hessian	Classification
[166]	HLSFSGU	2023	Filter	Hessian	Classification
[28]	BASSUM	2011	Filter	Bayesian	Classification
[199]	SRFS	2017	Filter	Bayesian	Classification
[161]	Semi-IAMB[7]	2018	Filter	Bayesian	Classification
[163]	SSHIBA[8]	2021	Filter	Bayesian	Classification
[209]	RRPC	2017	Filter	MRMR	Classification
[213]	SSMRMR	2018	Filter	MRMR	Classification
[40]	HM-ICS	2023	Filter	Relief	Classification
[186]	LPLIR	2020	Filter	Relief	Multi-target
[227]	SFS-SLL	2022	Filter	Fuzzy	Classification
[118]	SemiFREE	2023	Filter	Fuzzy	Classification
[147]	UVS	2016	Filter	Universum data	Classification
[147]	ULS	2016	Filter	Universum data	Classification
[147]	USS	2016	Filter	Universum data	Classification
[122]	-	2013	Filter	Other	Classification
[120]	GLSPFS	2014	Filter	Other	Classification
[223]	UFSSI	2016	Filter	Other	Classification
[38]	ASFS	2016	Filter	Other	Classification
[196]	-	2016	Filter	Other	Classification
[197]	-	2018	Filter	Other	Classification
[161]	Semi-JMI[0]	2018	Filter	Other	Classification
[119]	RSES	2019	Filter	Other	Classification
[107]	NGAR	2019	Filter	Other	Classification
[113]	S2LFS	2021	Filter	Other	Classification
[229]	UDM-SFS	2022	Filter	Other	Classification
[212]	LRF	2022	Filter	Other	Classification
[230]	IMP4ARA	2023	Filter	Other	Classification
[87]	SemiACO	2023	Filter	Other	Classification

Continued on next page

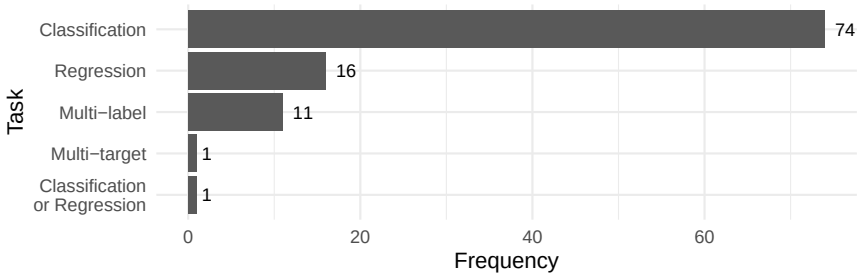


Fig. 3.6 Number of algorithms for each task.

Table 3.1 – continued from previous page

Reference	SSFS Name	Year	Type	Sub-type	Output
[198]	N-Semi-IG	2023	Filter	Other	Classification
[31]	SFMC	2016	Filter	Other	Regression
[195]	-	2017	Filter	Other	Multi-label
[221]	-	2017	Filter	Other	Multi-label
[210]	SCFS	2018	Filter	Other	Multi-label
[220]	GA-TSVM	2018	Wrapper	Search	Classification
[185]	-	2021	Wrapper	Search	Multi-target
[151]	FW-SemiFS	2008	Wrapper	Pseudo-labelling	Classification
[14]	EnsCLS[9]	2016	Wrapper	Pseudo-labelling	Classification
[205]	GMDH-SSFS	2017	Wrapper	Pseudo-labelling	Classification
[54]	TSLA-FSGA[10]	2022	Wrapper	Pseudo-labelling	Classification
[177]	FDG	2023	Wrapper	Pseudo-labelling	Classification
[176]	ISFS	2023	Wrapper	Pseudo-labelling	Classification
[178]	SFM	2023	Wrapper	Pseudo-labelling	Classification
[13]	SEFR	2012	Embedded	Tree	Classification
[162]	OFFS	2017	Embedded	Tree	Classification
[3]	SSS[11]	2021	Embedded	Tree	Multi-target
[224]	FS-Manifold	2010	Embedded	SVM	Classification
[46]	SENFS	2013	Embedded	SVM	Classification
[37]	RLSR	2017	Embedded	Linear regression	Classification
[192]	SDSSFS	2021	Embedded	Linear regression	Classification
[218]	DSSFS	2018	Embedded	Linear regression	Regression
[39]	SRLSR	2020	Embedded	Linear regression	Regression
[148]	A-SFS	2022	Embedded	Autoencoder	Classification
[7]	KNN-FRS-SSFS	2023	Embedded	KNN-Fuzzy	Classification
[165]	SRS3FS	2023	Embedded	Spline	Classification

[1] https://cran.r-project.org/web/packages/Rdimtools/
[2] https://github.com/LBDSoft/BRNS
[3] https://github.com/shidan0122/SFS-BLL.git
[4] https://github.com/46551972/SrDFS
[5] https://github.com/Jiang1Xue/LPFS
[6] https://github.com/rsheikhpour/HSFSGU
[7] https://github.com/sechidis/2018-MLJ-Semi-supervised-feature-selection
[8] https://github.com/sevisal/SSHIBA
[9] http://perso.univ-lyon1.fr/haytham.elghazel/EnsCLS/EnsCLS.zip
[10] https://github.com/vfeofanov/TSLA-FSGA
[11] https://github.com/eadiyeke/frfiles

3.5 Relevant information analysis

In this section, the most relevant information from the 99 journal and conference papers is organised and divided into two subsections: papers and results evaluation.

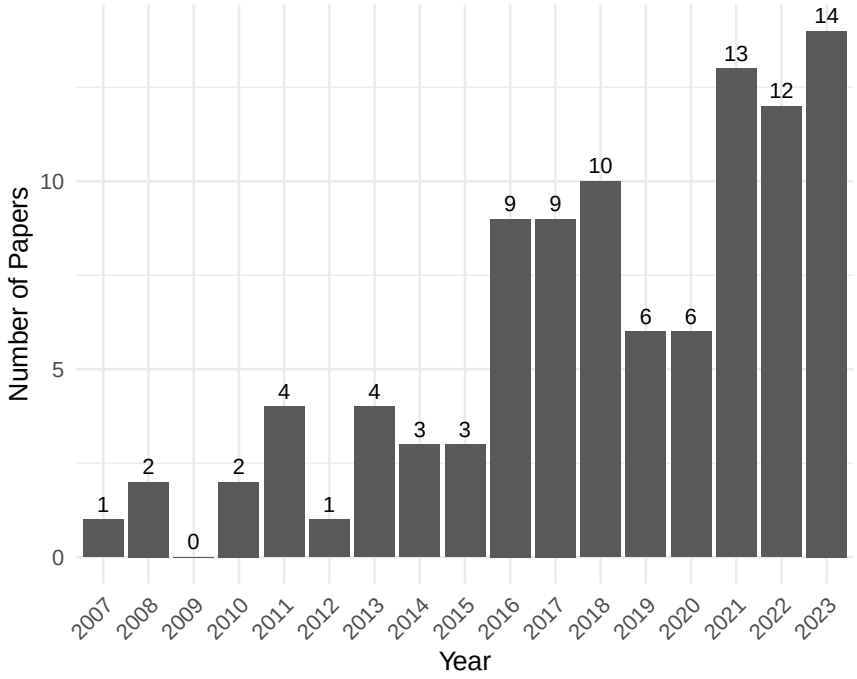


Fig. 3.7 Box plot showing the number of papers for the SSFS analysed on a yearly basis.

3.5.1 Papers

The information extracted from the papers contains data on the annual number of papers, the most common journals on this topic, and the authors who have published more articles.

Papers per year

The box plot presented in Figure 3.7 illustrates the number of semi-supervised feature selection articles analysed for this systematic review, covering the period from 2007 to 2023. A progressive increase in the number of papers can be observed, starting from 1 in 2007 and reaching 14 in 2023. An increasing trend that was applicable to several research topics. Note that, due to the filters applied in Section 3.2, the number of papers corresponding to 2009 is zero.

Journals

Table 3.2 presents a ranking of the journals that contain at least 3 of the revised papers (literature reviews included), amounting to a total of 15 journals that reflect a Pareto-type distribution, as is common in this type of analysis.

Table 3.2 Top journals with at least 3 appearances in the analysed papers.

Journal	Freq.
Knowledge-Based Systems	10
Neurocomputing	8
Applied Intelligence	6
IEEE Transactions on Neural Networks and Learning Systems	6
IEEE Transactions on Knowledge and Data Engineering	5
Information Sciences	5
Pattern Recognition	4
IEEE Transactions on Cybernetics	3
Knowledge and Information Systems	3
Pattern Recognition Letters	3

Table 3.3 Authors who have coauthored at least 5 documents analysed in this review.

Author	Papers
Nie, Feiping	10
Sheikhpour, Razieh	8
Chen, Hongmei	6
Chen, Xiaojun	6
Li, Tianrui	6
Sarram, Mehdi Agha	6
Wang, Xiao-dong	6
Benabdeslem, Khalid	5
Gharaghani, Sajjad	5

Authors

A list of authors who participated in the writing of at least 5 documents can be found in Table 3.3.

3.5.2 Results evaluation

In this section, the most common evaluation methodologies, the experimental setups, the classifiers and the normalisation methods, the results validation techniques, and the statistical tests are all presented. The approaches to determine the number of selected features and labelled instances are explored. And finally, the most common datasets used for experimentation in this field are compared.

Experimental setup

The range of experimental setups applied in the studies was very similar. Firstly, the initial data set, most of which fully supervised, was divided into two parts: training and testing. The training part was divided into supervised and unsupervised subsets by removing the labels of the latter subset. The proportions were arbitrarily determined by the researchers. Subsequently, in some cases, the data were normalised to mitigate potential adverse effects on the algorithms.

The following step was feature selection where a fixed number of features were chosen, either automatically by the algorithm or at the discretion of the researcher. After reducing the dataset, a supervised algorithm was trained. Although an SSL classifier could be used, a supervised one was preferred for simplicity and for a better understanding of the FS-algorithm error types. Finally, the process was repeated to apply train/test using different splits, usually based on 10-fold cross-validation.

It is usual to include several experiments with different learning models, different numbers of selected features and different trade-offs between the supervised and unsupervised parts. That practice might include either not using FS, to demonstrate the effectiveness of features in the problem, or employing supervised FS algorithms, to demonstrate the superiority of SSL approaches.

Comparison with other supervision types

Figure 3.8 represents an area plot displaying the number of SSFS algorithms against which the method proposed in the paper was compared for a performance assessment. It shows the SSFS used by year and type of supervision (semi-supervised, supervised, unsupervised). A gradual increase in the number of algorithms compared over the years may be observed. Note that in the first years, when SSFS algorithms emerged, only a few algorithms of the same type were compared. The distribution of papers *per* year can be uneven, as shown in Figure 3.7. Some years have little or no data representation, such as 2009, for which no papers were collected. In addition, it is worth mentioning that the new algorithm improved supervised and unsupervised approaches. Similarly, comparing the new algorithm with the option of selecting no features at all was only addressed in 32.3% of the sample.

Classifiers

Supervised learning models are the most frequent employed in these studies. The distribution of the most common models is presented in Figure 3.9 where SVM and KNN (for classification) and regression based models (for regression) are the most prevalent. It should be

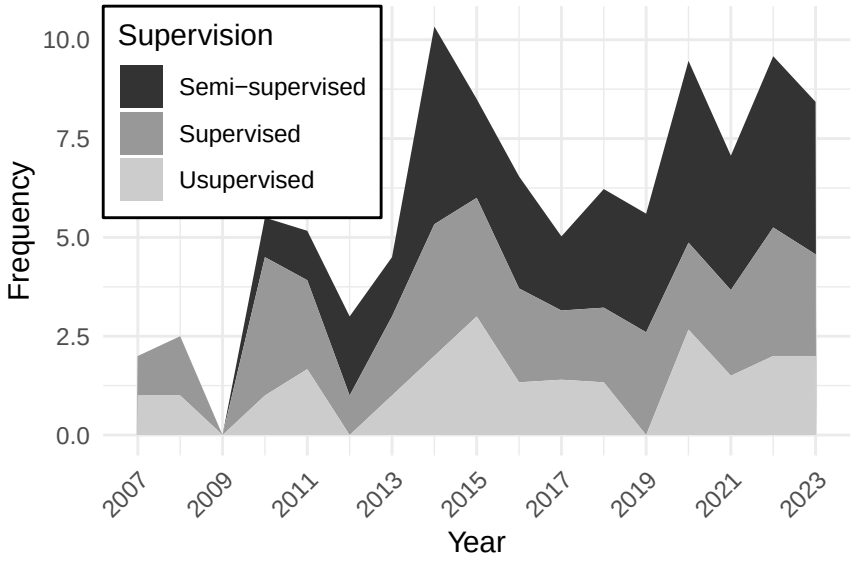


Fig. 3.8 Area plot showing the average number of feature selection algorithms against which papers are compared, grouped by year and type of supervision (semi-supervised, supervised, unsupervised).

noted that more than one learning model was typically trained in each article, although which model was never specified in some studies.

Normalisation step

Data normalisation is used to set all features into the same intervals to mitigate problems associated with the distribution of the data. Only 22.2% of the sample reported any data normalisation. The selected normalisation methods were Min-Max (18 studies), which sets the intervals between 0 and 1, and the Z-Score (4 studies), which standardises the mean to 0 and the standard deviation to 1 in every feature.

Validation type

Researchers use validation techniques, such as cross-validation, to evaluate the performance and generalisation capability of machine learning models and to avoid overfitting. Cross-validation divides a data set into k folds, training the model with $k-1$ folds and reserving the other fold for testing. This process is repeated k times, using each fold once as a test set, which ensures that all parts of the data set are used for testing at some point.

There are several ways to perform a k-fold cross-validation, though they are all based on the same principle. A summary of the validation types performed and the type in the reviewed papers is shown in Figure 3.10. The most common practice is cross-validation (48),

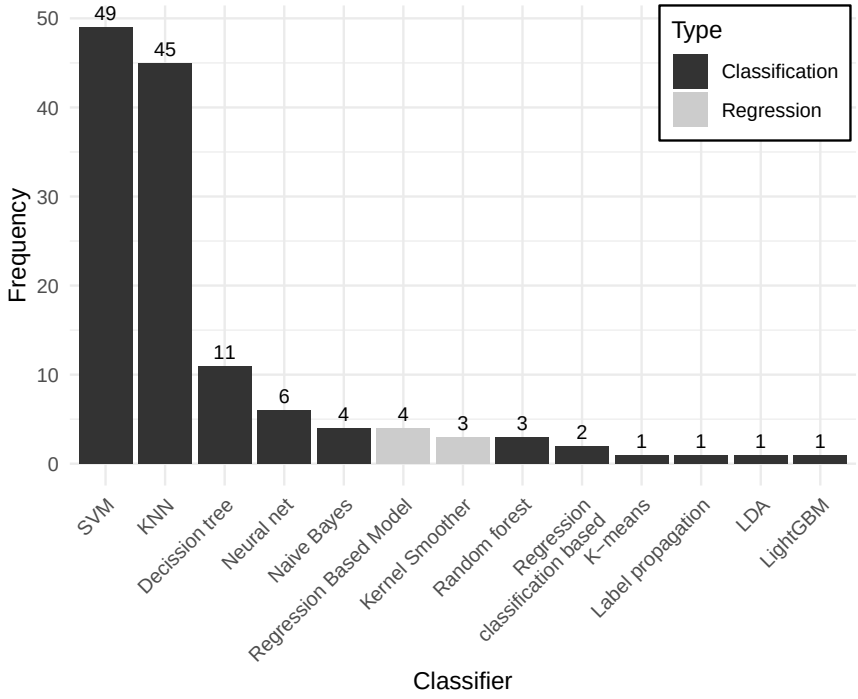


Fig. 3.9 Most frequent learning algorithm groups.

especially 5-fold cross-validation (18) and 10-fold cross-validation (20), some studies to save computational and programming time, only employ cross-validation to tune the classifiers (8). Others run the experiments multiple times (22), randomly re-selecting the training and test instances every time, in the hope that all instances will appear in both the training and testing sets. The techniques used in the remaining studies under review were never specified.

Evaluation metrics

The most popular metrics for performance evaluations of the strategies are detailed in Table 3.4. Despite the presence of metrics designed to address class imbalance, such as

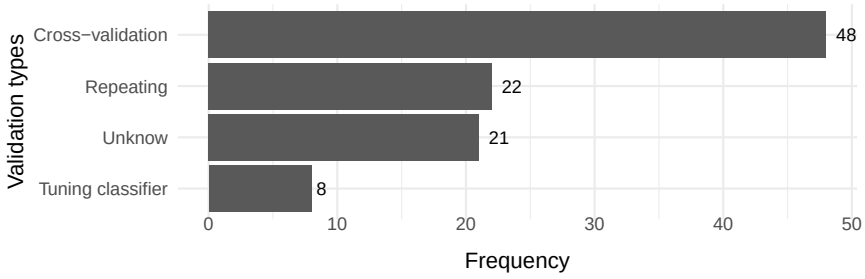


Fig. 3.10 Types of validation used in the reviewed papers.

Table 3.4 Top most popular metrics with at least 5 occurrences in the analysed papers.

Metric	Task	Freq.
Accuracy	Classification	68
Mean Average Precision	Classification	10
Area Under the ROC Curve (AUC)	Classif. / Multi-label	8
Root Mean Squared Error	Regression	7
F_1-Score (F1)	Classification	6
Hamming loss	Multi-label	5
One error	Multi-label	5
Coefficient of determination	Regression	5

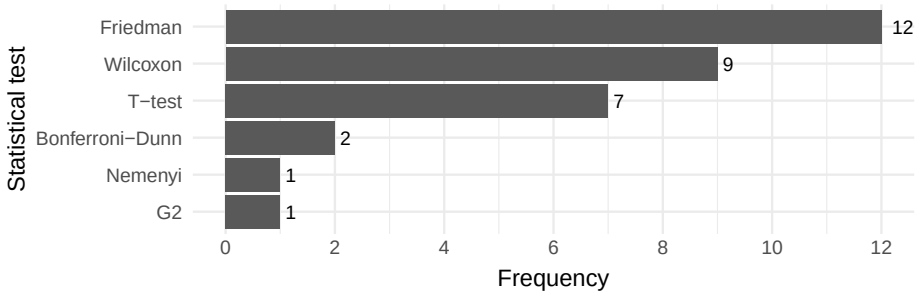


Fig. 3.11 Statistical tests employed for the validation of results.

F_1-Score or AUC, it may be noted that accuracy remains the most commonly used metric in the studies. It suggests a need to consider more appropriate metrics for unbalanced scenarios in future research.

Statistical tests

It is a common recommendation to use statistical tests when comparing experimental results. Researchers can draw conclusions based on observed data on the basis of those test results. The results confirm whether the differences between some strategies are statistically significant.

Only 32.3% of the studies used statistical tests. Figure 3.11 shows the frequency of use of each of these tests. The widely recognised tests were also the most widely used: the Friedman, the Wilcoxon Signed Rank, and the Student *t*-tests. While the Likelihood Ratio (G2), the Bonferroni-Dunn, and the Nemenyi tests were used almost anecdotally.

Parameters

In machine learning, the choice of parameters is one of the main challenges. In the field of FS, this decision corresponds to the choice of the number of attributes. The parameter has to be explored for filters and embedded approaches, both for researching new algorithms and for finding ways to obtain optimal results. Similarly, in the SSL context, the number of labelled instances is a critical experimentation parameter, which used to be explored when comparing various algorithms. The selection of these parameters is complicated by their dependence on a specific data set, the classifier, and other pre-processing steps that may be applied [150].

Those values can be selected as a percentage, relative to the dataset dimensions, or as an absolute number. Since the number of instances is compared in most studies in percentile terms and the number of features with an absolute number, their data will be presented in those terms.

The values of those parameters may vary in each paper, as it is common to perform experiments with different configurations. Both parameters have therefore been divided into three aspects: the maximum number, the minimum number, and the number of steps tested for each parameter. These data are represented in box plots in Figure 3.12.

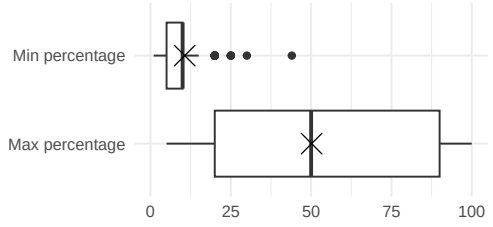
Datasets

As mentioned in Section 3.1, SSFS algorithms have been used in a wide variety of contexts. The 40 most popular datasets are summarised alongside their characteristics in Table 3.5, and the corresponding links are listed in the supplementary material(Appendix A). As can be seen, the most common datasets are related to image processing, medicine, and handwritten text recognition.

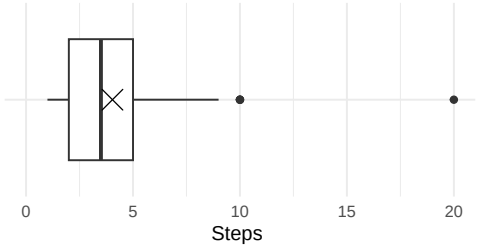
3.6 Conclusions and future lines of research

The field of SSFS has gained popularity over recent years. However, algorithm explanations and comparisons have only been presented so far in two reviews, in 2016 and in 2017. Neither study has been updated and the number of SSFS presented in each one is limited.

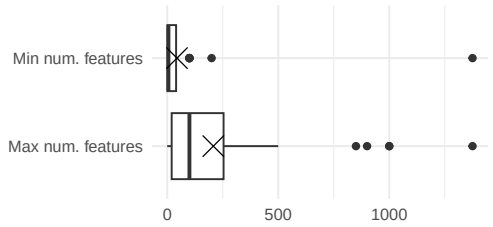
In all, 101 contributions, since research began in this field in 2007 up until the end of 2023, have been analysed in this systematic review. A new taxonomy based on classical FS has also been proposed in this paper and used to classify 103 of the most relevant SSFS algorithms.



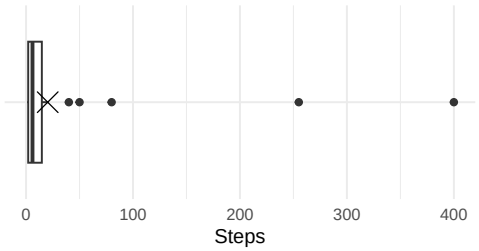
(**a**) Maximum and minimum percentages for the labelled data.



(**b**) Steps of percentages of labelled data.



(**c**) Minimum and maximum number of selected features.



(**d**) Steps of selected features.

Fig. 3.12 Statistics on the decisions taken when choosing the percentage of labelled data (a and b) and the number of features (c and d) in the experiments when comparing algorithm performance. (a) and (c) show the common maximum and minimum values of these parameters, whereas (b) and (d) show the number of steps taken. The means are marked by a cross ($\times$).

Table 3.5 Top 40 most commonly used datasets in the reviewed studies sorted by the number of times have been used. The table includes the data domain, the number of instances, features and classes. For picture datasets the features have been represented as image dimensions.

# Papers	Data	Type	Instances	Features	Classes
18	Coil20	Images	1440	128×128	20
17	Ionosphere	Radar	351	34	2
17	Sonar	Sonar	208	60	2
16	Image Segmentation	Images	2310	19	7
16	ORL	Face recognition	1440	1024	20
15	USPS	Handwriting	9298	256	10
13	WBDC	Images	569	30	2
12	NUS-WIDE	Images	269648	128	81
12	Wine	Other	178	13	3
12	Colon	Microarray	62	2000	2
11	Binary alphabet	Images	1404	74×86	34
11	Dermatology	Medicine	366	34	6
11	glass	Physics and Chemistry	214	9	6
11	Isolet	Voice recognition	7797	617	26
10	Yeast	Medicine	2417	103	14
9	Statlog German	Social Science	1000	24	2
9	Heart disease	Medicine	303	13	4
9	WPBC	Medicine	198	33	2
9	Statlog vehicle	Images	946	18	4
9	Pie10P	Face recognition	210	2420	10
8	Scene Image	Images	2407	294	6
7	Breast	Medicine	699	9	2
7	Madelon	Artificial data	4400	500	2
7	Musk	Physics and Chemistry	476	166	2
7	Yale	Face recognition	165	1024	15
7	Waveform	Physics and Chemistry	5000	40	3
7	Leukemia	Medicine	72	7070	2
6	hepatitis	Health and Medicine	155	19	2
6	Libras Movement	Other	360	90	15
6	PcMac	Text	1943	3289	2
6	Prostate Tumor	Medicine	102	10509	2
6	Pima	Medicine	768	8	2
6	HumanEVA	Images	10000	168	5
5	Semeion	Handwriting	1593	256	10
5	CNAE-9	Business	1080	856	9
5	Ecoli	Biology	336	343	8
5	MNIST	Handwriting	90000	784	10
5	YaleB	Images	2452	168×192	38
5	MIML	Images	2000	15	5
4	Parkinsons	Medicine	197	22	2

After analysing the literature, several interesting conclusions have been drawn. When comparing SSFS algorithms, researchers have often explored two critical parameters that significantly influenced algorithm performance: the number of labelled instances and the number of selected features. Determining the values for those parameters is challenging for different reasons and often relies on arbitrary choices. Consequently, researchers have frequently experimented with different configurations, to ensure a fair comparison.

The predominant choice of learning models in SSFS has tended to be supervised, with SVM and KNN as the most popular options. Moreover, the absence of any normalisation step is remarkable.

Accuracy is used in most studies (68.3%) to analyse the different strategies and their performance. However, proper cross-validation techniques to validate the results were only employed in one-third of the studies. Additionally, statistical tests to evaluate the performance of the strategies, such as the Friedman, the Wilcoxon Signed Rank, and the Student t-tests were used in only 32.3% of the studies .

Following the production of this systematic review, some recommendations can be presented based on the observed weaknesses. Consider data normalisation as a preliminary step before applying algorithms. Use more appropriate metrics, such as F1-Score or AUC, to address problems inherent to class imbalance. Use statistical tests such as the Friedman, the Wilcoxon, and Bayesian tests to improve the robustness of the studies.

Regarding the transparency and reproducibility of the studies, the importance may be emphasised of publishing the code of the algorithms. The codes of only 10% of the algorithms are currently available, which hinders any replicability of the results and prolongs the time needed to research new studies. This practice may discourage researchers and limit the comparison of algorithms, negatively affecting the quality of future studies.

As a future research line, it could be interesting to conduct a comprehensive comparison between SSFS methods and SSL classifiers. Since most classifiers are supervised, such a comparison could provide valuable insights into specific contexts. Finally, it might also be worth exploring which learning models and which SSFS pairs are better than others, given that it has been observed that FS performance tends to depend on the model that is used.

References

[1] Abdi, L. and Hashemi, S. (2016). To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):238–251.

[2] Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671–687. Random projections.

[3] Adıyeke, E. and Baydoğan, M. G. (2021). An ensemble-based semi-supervised feature ranking for multi-target regression problems. *Pattern Recognition Letters*, 148:36–42.

[4] Alalga, A., Benabdeslem, K., and Mansouri, D. E. K. (2021). 3-3fs: ensemble method for semi-supervised multi-label feature selection. *Knowledge and Information Systems*, 63(11):2969–2999.

[5] Alalga, A., Benabdeslem, K., and Taleb, N. (2016). Soft-constrained laplacian score for semi-supervised multi-label feature selection. *Knowledge and Information Systems*, 47(1):75–98.

[6] Alshorman, O., Irfan, M., Saad, N., Zhen, D., Haider, N., Glowacz, A., and Alshorman, A. (2020). A Review of Artificial Intelligence Methods for Condition Monitoring and Fault Diagnosis of Rolling Element Bearings for Induction Motor.

[7] An, S., Zhang, M., Wang, C., and Ding, W. (2023). Robust fuzzy rough approximations with kNN granules for semi-supervised feature selection. *Fuzzy Sets and Systems*, 461.

[8] Ang, J. C., Mirzal, A., Haron, H., and Hamed, H. N. A. (2016). Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5):971–989.

[9] Ayadi, R., Maraoui, M., and Zrigui, M. (2015). Lda and lsi as a dimensionality reduction method in arabic document classification. *Communications in Computer and Information Science*, 538:491 – 502. Cited by: 14.

[10] Ayesha, S., Hanif, M. K., and Talib, R. (2020). Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58.

[11] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A. F., and Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview . *Bioinformatics*, 16(5):412–424.

[12] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396. Laplacian Eigenmaps.

[13] Bellal, F., Elghazel, H., and Aussem, A. (2012). A semi-supervised feature ranking method with ensemble learning. *Pattern Recognition Letters*, 33(10):1426–1433.

[14] Benabdeslem, K., Elghazel, H., and Hindawi, M. (2016). Ensemble constrained laplacian score for efficient and robust semi-supervised feature selection. *Knowledge and Information Systems*, 49(3):1161–1185.

[15] Benabdeslem, K. and Hindawi, M. (2011). Constrained laplacian score for semi-supervised feature selection. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I 11*, pages 204–218. Springer.

[16] Benabdeslem, K. and Hindawi, M. (2014). Efficient semi-supervised feature selection: Constraint, relevance, and redundancy. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1131–1143.

[17] Benavoli, A., Corani, G., Demšar, J., and Zaffalon, M. (2017). Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688.

[18] Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., and Ruggeri, F. (2014). A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In *International conference on machine learning*, pages 1026–1034. PMLR.

[19] Bennett, K. and Demiriz, A. (1998). Semi-supervised support vector machines. *Advances in Neural Information processing systems*, 11.

[20] Bernardini, M., Romeo, L., Misericordia, P., and Frontoni, E. (2020). Discovering the Type 2 Diabetes in Electronic Health Records Using the Sparse Balanced Support Vector Machine. *IEEE Journal of Biomedical and Health Informatics*, 24(1):235–246.

[21] Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128.

[22] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

[23] Bolón-Canedo, V. and Alonso-Betanzos, A. (2018). *Recent advances in ensembles for feature selection*, volume 147. Springer.

[24] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2016). Feature selection for high-dimensional data. *Progress in Artificial Intelligence*, 5:65–75.

[25] Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., and Lang, M. (2020). Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis*, 143:106839.

[26] Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

[27] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[28] Cai, R., Zhang, Z., and Hao, Z. (2011). BASSUM: A bayesian semi-supervised method for classification feature selection. *Pattern Recognition*, 44(4):811–820.

[29] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215.

[30] Chang, X., Nie, F., Yang, Y., and Huang, H. (2014). A convex formulation for semi-supervised multi-label feature selection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1). Number: 1.

[31] Chang, X. and Yang, Y. (2016). Semisupervised feature analysis by mining correlations among multiple tasks. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2294–2305.

[32] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[33] Chen, H., Chen, H., Li, W., and Li, T. (2023a). Semi-supervised feature selection based on pairwise constraint-guided dual space latent representation learning and double sparse graphs discriminant. *Applied Intelligence*, 53(10):12288–12307.

[34] Chen, H., Chen, H., Li, W., Li, T., Luo, C., and Wan, J. (2022a). Robust dual-graph regularized and minimum redundancy based on self-representation for semi-supervised feature selection. *Neurocomputing*, 490:104–123.

[35] Chen, X., Chen, R., Wu, Q., Nie, F., Yang, M., and Mao, R. (2022b). Semisupervised feature selection via structured manifold learning. *IEEE Transactions on Cybernetics*, 52(7):5756–5766.

[36] Chen, X., Fang, T., Huo, H., and Li, D. (2010). Semisupervised feature selection for unbalanced sample sets of VHR images. *IEEE Geoscience and Remote Sensing Letters*, 7(4):781–785.

[37] Chen, X., Nie, F., Yuan, G., and Huang, J. Z. (2017). Semi-supervised feature selection via rescaled linear regression. *IJCAI International Joint Conference on Artificial Intelligence*, 0:1525 – 1531. Cited by: 90; All Open Access, Bronze Open Access.

[38] Chen, X., Qi, J., Chen, Y., Hua, L., and Shao, G. (2016). Adaptive semisupervised feature selection without graph construction for very-high-resolution remote sensing images. *Journal of Applied Remote Sensing*, 10(2):025002.

[39] Chen, X., Yuan, G., Nie, F., and Ming, Z. (2020). Semi-supervised feature selection via sparse rescaled linear square regression. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):165–176.

[40] Chen, X., Zhang, L., and Zhao, L. (2023b). Iterative constraint score based on hypothesis margin for semi-supervised feature selection. *Knowledge-Based Systems*, 271.

[41] Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13.

[42] Chicco, D. and Jurman, G. (2023). The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification. *BioData Mining*, 16(1):1–23.

[43] Chung, F. R. (1997). *Spectral graph theory*, volume 92. American Mathematical Soc.

[44] Coelho, F., Castro, C., Braga, A. P., and Verleysen, M. (2019). Semi-supervised relevance index for feature selection. *Neural Computing and Applications*, 31:989–997.

[45] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

[46] Dai, K., Yu, H.-Y., and Li, Q. (2013). A semisupervised feature selection with support vector machine. *Journal of Applied Mathematics*, 2013:e416320. Publisher: Hindawi.

[47] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30.

[48] Dennis Cook, R. (2000). Save: a method for dimension reduction and graphics in regression. *Communications in statistics-Theory and methods*, 29(9-10):2109–2121.

[49] Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*.

[50] Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C. I., and Kuncheva, L. I. (2015). Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, 325:98–117.

[51] Doquire, G. and Verleysen, M. (2013). A graph laplacian based approach to semi-supervised feature selection for regression problems. *Neurocomputing*, 121:5–13.

[52] Dornaika, F. and Assoum, A. (2013). Enhanced and parameterless locality preserving projections for face recognition. *Neurocomputing*, 99:448–457.

[53] Fan, M., Zhang, X., Hu, J., Gu, N., and Tao, D. (2022). Adaptive data structure regularized multiclass discriminative feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5859–5872. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[54] Feofanov, V., Devijver, E., and Amini, M.-R. (2022). Wrapper feature selection with partially labeled data. *Applied Intelligence*, 52(11):12316–12329.

[55] Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., and Herrera, F. (2018). *Learning from imbalanced data sets*, volume 10. Springer.

[56] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

[57] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[58] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[59] Garcia, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12).

[60] García, S., Luengo, J., and Herrera, F. (2015). *Data preprocessing in data mining*, volume 72. Springer.

[61] García, V., Sánchez, J. S., and Mollineda, R. A. (2012). On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21.

[62] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A., editors (2006). *Feature extraction: foundations and applications*, volume 207. Springer.

[63] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

[64] Hamed, T., Dara, R., and Kremer, S. C. (2014). An accurate, fast embedded feature selection for svms. In *2014 13th International Conference on Machine Learning and Applications*, pages 135–140.

[65] Han, Y., Yang, Y., Yan, Y., Ma, Z., Sebe, N., and Zhou, X. (2015). Semisupervised feature selection via spline regression for video semantic recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 26(2):252–264.

[66] Hang, Q., Yang, J., and Xing, L. (2019). Diagnosis of rolling bearing based on classification for high dimensional unbalanced data. *IEEE Access*, 7:79159–79172.

[67] Hao, Z., Lv, D., Ge, Y., Shi, J., Weijers, D., Yu, G., and Chen, J. (2020). Rideogram: drawing svg graphics to visualize and map genome-wide data on the idiograms. *PeerJ Computer Science*, 6:e251.

[68] Hasanloei, M. A. V., Sheikhpour, R., Sarram, M. A., Sheikhpour, E., and Sharifi, H. (2018). A combined fisher and laplacian score for feature selection in QSAR based drug design using compounds with known and unknown activities. *Journal of Computer-Aided Molecular Design*, 32(2):375–384.

[69] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

[70] He, X., Cai, D., and Niyogi, P. (2005a). Laplacian score for feature selection. *Advances in neural information processing systems*, 18.

[71] He, X., Cai, D., Yan, S., and Zhang, H. J. (2005b). Neighborhood preserving embedding. *Proceedings of the IEEE International Conference on Computer Vision*, II:1208–1213. NPE - Neighborhood preserving embedding.

[72] He, X. and Niyogi, P. (2003). Locality preserving projections. *Advances in Neural Information Processing Systems*, 16. LPP.

[73] Hindawi, M., Allab, K., and Benabdeslem, K. (2011). Constraint selection-based semi-supervised feature selection. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1080–1085.

[74] Hinton, G. E. and Roweis, S. (2002). Stochastic neighbor embedding. *Advances in neural information processing systems*, 15.

[75] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313:504–507. Autoencoders.

[76] Huang, H. and Feng, H. (2013). LOCALITY AND GLOBALITY DISCRIMINANT FEATURE AND ITS APPLICATION IN HYPERSPECTRAL IMAGE CLASSIFICATION. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(4):1350010.

[77] Hui, L. and Jiqing, H. (2021). Semi-supervised robust feature selection with lq-norm graph for multiclass classification. *Chinese Journal of Electronics*, 30(4):611–622. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/cje.2021.05.003.

[78] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

[79] Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI*, pages 111–117.

[80] Jiang, X., Chen, M., Song, W., and Lin, G. N. (2021a). Label propagation-based semi-supervised feature selection on decoding clinical phenotypes with RNA-seq data. *BMC Medical Genomics*, 14(1):141.

[81] Jiang, Y., Chen, S., Bian, B., Li, Y., Sun, Y., and Wang, X. (2021b). Discrimination of tomato maturity using hyperspectral imaging combined with graph-based semi-supervised method considering class probability information. *Food Analytical Methods*, 14(5):968–983.

[82] Johnson, K. J. and Synovec, R. E. (2002). Pattern recognition of jet fuels: comprehensive gc×gc with anova-based feature selection and principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 60(1):225–237. Fourth International Conference on Environ metrics and Chemometrics held in Las Vegas, NV, USA, 18-20 September 2000.

[83] Juez-Gil, M., Arnaiz-González, A., Rodríguez, J. J., and García-Osorio, C. (2021). Experimental evaluation of ensemble classifiers for imbalance in big data. *Applied Soft Computing*, 108:107447.

[84] Juez-Gil, M., Saucedo-Dorantes, J. J., Arnaiz-González, Á., López-Nozal, C., García-Osorio, C., and Lowe, D. (2020). Early and extremely early multi-label fault diagnosis in induction motors. *ISA transactions*, 106:367–381.

[85] Kalakech, M., Biela, P., Macaire, L., and Hamad, D. (2011). Constraint scores for semi-supervised feature selection: A comparative study. *Pattern Recognition Letters*, 32(5):656–665.

[86] Karasu, S. k. and Altan, A. (2019). Recognition model for solar radiation time series based on random forest with feature selection approach. In *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 8–11.

[87] Karimi, F., Dowlatshahi, M. B., and Hashemi, A. (2023). SemiACO: A semi-supervised feature selection based on ant colony optimization. *Expert Systems with Applications*, 214:119130.

[88] Keogh, E. J. and Mueen, A. (2017). Curse of dimensionality. *Encyclopedia of machine learning and data mining*, 2017:314–315.

[89] Kerber, R. (1992). Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 123–128. AAAI Press.

[90] Kim, T. K. (2015). T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540.

[91] Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In Sleeman, D. and Edwards, P., editors, *Machine Learning Proceedings 1992*, pages 249–256. Morgan Kaufmann, San Francisco (CA).

[92] Kohavi, R. and John, G. H. (1997a). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324.

[93] Kohavi, R. and John, G. H. (1997b). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324. Relevance.

[94] Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 784 LNCS, pages 171–182. Springer Verlag.

[95] Krishnasamy, G. and Paramesran, R. (2019). Multiview laplacian semisupervised feature selection by leveraging shared knowledge among multiple tasks. *Signal Processing: Image Communication*, 70:68–78.

[96] Kroese, D. P., Brereton, T., Taimre, T., and Botev, Z. I. (2014). Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392.

[97] Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

[98] Kruskal, J. B. (1964b). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27. MDS - Multi Dimensional Scaling.

[99] Kuncheva, L. I., Garrido-Labrador, J. L., Ramos-Pérez, I., Hennessey, S. L., and Rodríguez, J. J. (2023). An experiment on animal re-identification from video. *Ecological Informatics*, 74:101994.

[100] Kuncheva, L. I., Garrido-Labrador, J. L., Ramos-Perez, I., Hennessey, S. L., and Rodríguez, J. J. (2024). Semi-supervised classification with pairwise constraints: A case study on animal identification from video. *Information Fusion*, 104:102188.

[101] Kuncheva, L. I., Matthews, C. E., Arnaiz-González, Á., and Rodríguez, J. J. (2020). Feature selection from high-dimensional data with very low sample size: A cautionary tale.

[102] Lai, J., Chen, H., Li, T., and Yang, X. (2022a). Adaptive graph learning for semi-supervised feature selection with redundancy minimization. *Information Sciences*, 609:465–488.

[103] Lai, J., Chen, H., Li, W., Li, T., and Wan, J. (2022b). Semi-supervised feature selection via adaptive structure learning and constrained graph learning. *Knowledge-Based Systems*, 251:109243.

[104] Lai, K., Twine, N., O'brien, A., Guo, Y., and Bauer, D. (2018). Artificial intelligence and machine learning in bioinformatics. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 1:272–286.

[105] Lal, T. N., Chapelle, O., Weston, J., and Elisseeff, A. (2006). Embedded methods. *Feature Extraction: Foundations and Applications*, pages 137–165.

[106] Lazebnik, T. and Rosenfeld, A. (2023). Fspl: A meta-learning approach for a filter and embedded feature selection pipeline. *International Journal of Applied Mathematics and Computer Science*, 33(1).

[107] Li, B., Xiao, J., and Wang, X. (2019). Feature selection for partially labeled data based on neighborhood granulation measures. *IEEE Access*, 7:37238–37250. Conference Name: IEEE Access.

[108] Li, H., Jiang, T., and Zhang, K. (2006). Efficient and robust feature extraction by maximum margin criterion. *IEEE Transactions on Neural Networks*, 17:157–165. MMC - Maximun Margin Criterion.

[109] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2018). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45.

[110] Li, X., Wang, Y., and Ruiz, R. (2020). A survey on sparse learning models for feature selection. *IEEE transactions on cybernetics*, 52(3):1642–1660.

[111] Li, X., Zhang, Y., and Zhang, R. (2022). Semisupervised feature selection via generalized uncorrelated constraint and manifold embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5070–5079.

[112] Li, Y.-F. and Zhou, Z.-H. (2010). S4vm: Safe semi-supervised support vector machine. *Computing Research Repository*.

[113] Li, Z. and Tang, J. (2021). Semi-supervised local feature selection for data classification. *Science China Information Sciences*, 64(9).

[114] Liao, B., Jiang, Y., Liang, W., Zhu, W., Cai, L., and Cao, Z. (2014). Gene selection using locality sensitive laplacian score. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11:1146–1156. LSLS - Locality Sensitive Laplacian Score.

[115] Lin, Y., Hu, Q., Liu, J., Li, J., and Wu, X. (2017). Streaming feature selection for multilabel learning based on fuzzy mutual information. *IEEE Transactions on Fuzzy Systems*, 25(6):1491–1507.

[116] Liu, H. and Setiono, R. (1995). Chi2: feature selection and discretization of numeric attributes. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 388–391. IEEE.

[117] Liu, H., Zhou, M., and Liu, Q. (2019a). An embedded feature selection method for imbalanced data classification. *IEEE/CAA Journal of Automatica Sinica*, 6(3):703–715.

[118] Liu, K., Li, T., Yang, X., Chen, H., Wang, J., and Deng, Z. (2023). SemiFREE: Semi-supervised feature selection with fuzzy relevance and redundancy. *IEEE Transactions on Fuzzy Systems*, pages 1–13. Conference Name: IEEE Transactions on Fuzzy Systems.

[119] Liu, K., Yang, X., Yu, H., Mi, J., Wang, P., and Chen, X. (2019b). Rough set based semi-supervised feature selection via ensemble selector. *Knowledge-Based Systems*, 165:282–296.

[120] Liu, X., Wang, L., Zhang, J., Yin, J., and Liu, H. (2014). Global and local structure preservation for feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1083–1095. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[121] Liu, Y., Jiang, B., Feng, J., Hu, J., and Zhang, H. (2020). Classification of EEG signals for epileptic seizures using feature dimension reduction algorithm based on LPP. *Multimedia Tools and Applications*.

[122] Liu, Y., Nie, F., Wu, J., and Chen, L. (2013). Efficient semi-supervised feature selection with noise insensitive trace ratio criterion. *Neurocomputing*, 105:12–18.

[123] Luo, M., Chang, X., Nie, L., Yang, Y., Hauptmann, A. G., and Zheng, Q. (2018a). An adaptive semisupervised feature analysis for video semantic recognition. *IEEE Transactions on Cybernetics*, 48(2):648–660.

[124] Luo, T., Hou, C., Nie, F., Tao, H., and Yi, D. (2018b). Semi-supervised feature selection via insensitive sparse regression with application to video semantic recognition. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1943–1956.

[125] Luque, A., Carrasco, A., Martín, A., and de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231.

[126] Lv, S., Shi, S., Wang, H., and Li, F. (2021). Semi-supervised multi-label feature selection with adaptive structure learning and manifold learning. *Knowledge-Based Systems*, 214:106757.

[127] Mackutė-Varoneckienė, A. and Krilavičius, T. (2014). Empirical study on unsupervised feature selection for document clustering. In *Human Language Technologies–The Baltic Perspective*, pages 107–110. IOS Press.

[128] Maldonado, S., Weber, R., and Famili, F. (2014). Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information Sciences*, 286:228–246.

[129] Matthews, B. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

[130] Mendes Junior, J. J. A., Freitas, M. L., Siqueira, H. V., Lazzaretti, A. E., Pichorim, S. F., and Stevan, S. L. (2020). Feature selection and dimensionality reduction: An extensive comparison in hand gesture classification by semg in eight channels armband approach. *Biomedical Signal Processing and Control*, 59:101920.

[131] Min, W., Lu, K., and He, X. (2004). Locality pursuit embedding. *Pattern Recognition*, 37:781–788. LPE - Locality pursuit embedding.

[132] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill.

[133] Mohammed, R., Rawashdeh, J., and Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pages 243–248. IEEE.

[134] Mordohai, P. and Medioni, G. (2010). Dimensionality estimation, manifold learning and function approximation using tensor voting. *Journal of Machine Learning Research*, 11(1).

[135] Muntasa, A., Sirajudin, I. A., and Purnomo, M. H. (2011). Appearance global and local structure fusion for face image recognition. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 9(1):125–132.

[136] Ng, W. W. Y., Hu, J., Yeung, D. S., Yin, S., and Roli, F. (2015). Diversified sensitivity-based undersampling for imbalance classification problems. *IEEE Transactions on Cybernetics*, 45(11):2402–2412.

[137] Ongaro, A. and Migliorati, S. (2013). A generalization of the Dirichlet distribution. *Journal of Multivariate Analysis*, 114:412–426.

[138] Orriols-Puig, A. and Bernadó-Mansilla, E. (2009). Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, 13:213–225.

[139] Pang, Q. and Zhang, L. (2021). A recursive feature retention method for semi-supervised feature selection. *International Journal of Machine Learning and Cybernetics*, 12(9):2639–2657.

[140] Pang, Q.-Q. and Zhang, L. (2020). Semi-supervised neighborhood discrimination index for feature selection. *Knowledge-Based Systems*, 204:106224.

[141] Parhizkar, T., Rafieipour, E., and Parhizkar, A. (2021). Evaluation and improvement of energy consumption prediction models using principal component analysis based feature reduction. *Journal of Cleaner Production*, 279:123866.

[142] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. PCA.

[143] Peck, R. and Devore, J. L. (2011). *Statistics: The exploration & analysis of data*. Cengage Learning.

[144] Peralta, D., Del Río, S., Ramírez-Gallego, S., Triguero, I., Benitez, J. M., and Herrera, F. (2015). Evolutionary feature selection for big data classification: A MapReduce approach. *Mathematical Problems in Engineering*.

[145] Pes, B. (2020). Learning from high-dimensional biomedical datasets: The issue of class imbalance. *IEEE Access*, 8:13527–13540.

[146] Pes, B. (2021). Learning from high-dimensional and class-imbalanced datasets using random forests. *Information*, 12(8).

[147] Qiu, J. and Pan, Z. (2016). Semisupervised feature selection with universum. *Mathematical Problems in Engineering*, 2016:e5874161. Publisher: Hindawi.

[148] Qiu, Z., Zeng, W., Liao, D., and Gui, N. (2022). A-SFS: Semi-supervised feature selection based on multi-task self-supervision. *Knowledge-Based Systems*, 252:109449.

[149] Ramos-Pérez, I., Barbero-Aparicio, J. A., Canepa-Oneto, A., Arnaiz-González, Á., and Maudes-Raedo, J. (2024). An extensive performance comparison between feature reduction and feature selection preprocessing algorithms on imbalanced wide data. *Information*, 15(4):223.

[150] Ramos-Pérez, I., Álvar Arnaiz-González, Rodríguez, J. J., and García-Osorio, C. (2022). When is resampling beneficial for feature selection with imbalanced wide data? *Expert Systems with Applications*, 188:116015.

[151] Ren, J., Qiu, Z., Fan, W., Cheng, H., and Yu, P. S. (2008). Forward semi-supervised feature selection. In *Advances in Knowledge Discovery and Data Mining: 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20-23, 2008 Proceedings 12*, pages 970–976. Springer.

[152] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. *Proceedings - Seventh IEEE Workshop on Applications of Computer Vision, WACV 2005*.

[153] Rostami, M., Berahmand, K., and Forouzandeh, S. (2020). A novel method of constrained feature selection by the measurement of pairwise constraints uncertainty. *Journal of Big Data*, 7(1):83.

[154] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

[155] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[156] Sachdeva, R. K., Bathla, P., Rani, P., Kukreja, V., and Ahuja, R. (2022). A systematic method for breast cancer classification using rfe feature selection. *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering, ICACITE 2022*, pages 1673–1676.

[157] Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

[158] Sahu, B., Dehuri, S., and Jagadev, A. (2018). A Study on the Relevance of Feature Selection Methods in Microarray Data. *The Open Bioinformatics Journal*, 11(1).

[159] Sáiz-Manzanares, M. C., Pérez, I. R., Rodríguez, A. A., Arribas, S. R., Almeida, L., and Martin, C. F. (2021). Analysis of the learning process through eye tracking technology and feature selection techniques. *Applied Sciences*, 11(13):6157.

[160] Salesi, S., Cosma, G., and Mavrovouniotis, M. (2021). TAGA: Tabu asexual genetic algorithm embedded in a filter/filter feature selection approach for high-dimensional data. *Information Sciences*, 565:105–127.

[161] Sechidis, K. and Brown, G. (2018). Simple strategies for semi-supervised feature selection. *Machine Learning*, 107(2):357–395.

[162] Settouti, N., Chikh, M. A., and Barra, V. (2017). A new feature selection approach based on ensemble methods in semi-supervised classification. *Pattern Analysis and Applications*, 20(3):673–686.

[163] Sevilla-Salcedo, C., Gómez-Verdejo, V., and Olmos, P. M. (2021). Sparse semi-supervised heterogeneous interbattery bayesian analysis. *Pattern Recognition*, 120:108141.

[164] Sharifabad, M. M., Sheikhpour, R., and Gharaghani, S. (2022). BRNS + SSFSM-DTI: A hybrid method for drug-target interaction prediction based on balanced reliable negative samples and semi-supervised feature selection. *Chemometrics and Intelligent Laboratory Systems*, 220:104462.

[165] Sheikhpour, R. (2023). A local spline regression-based framework for semi-supervised sparse feature selection. *Knowledge-Based Systems*, 262:110265.

[166] Sheikhpour, R., Berahmand, K., and Forouzandeh, S. (2023). Hessian-based semi-supervised feature selection using generalized uncorrelated constraint. *Knowledge-Based Systems*, 269:110521.

[167] Sheikhpour, R., Sarram, M. A., and Gharaghani, S. (2017a). Constraint score for semi-supervised feature selection in ligand-and receptor-based QSAR on serine/threonine-protein kinase PLK3 inhibitors. *Chemometrics and Intelligent Laboratory Systems*, 163:31–40.

[168] Sheikhpour, R., Sarram, M. A., Gharaghani, S., and Chahooki, M. A. Z. (2017b). Feature selection based on graph laplacian by using compounds with known and unknown activities. *Journal of Chemometrics*, 31(8):e2899. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cem.2899.

[169] Sheikhpour, R., Sarram, M. A., Gharaghani, S., and Chahooki, M. A. Z. (2017c). A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141–158.

[170] Sheikhpour, R., Sarram, M. A., Gharaghani, S., and Chahooki, M. A. Z. (2020-08-01). A robust graph-based semi-supervised sparse feature selection method. *Information Sciences*, 531:13–30.

[171] Sheikhpour, R., Sarram, M. A., and Sheikhpour, E. (2018). Semi-supervised sparse feature selection via graph laplacian based scatter matrix for regression problems. *Information Sciences*, 468:14–28.

[172] Shi, C., Duan, C., Gu, Z., Tian, Q., An, G., and Zhao, R. (2019). Semi-supervised feature selection analysis with structured multi-view sparse regularization. *Neurocomputing*, 330:412–424.

[173] Shi, C., Gu, Z., Duan, C., and Tian, Q. (2020). Multi-view adaptive semi-supervised feature selection with the self-paced learning. *Signal Processing*, 168:107332.

[174] Shi, C., Ruan, Q., An, G., and Zhao, R. (2015). Hessian semi-supervised sparse feature selection. *IEEE Transactions on Multimedia*, 17(1):16–28.

[175] Shi, D., Zhu, L., Li, J., Cheng, Z., and Liu, Z. (2023). Binary label learning for semi-supervised feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2299–2312.

[176] Shu, W., Yan, Z., Yu, J., and Qian, W. (2023a). Information gain-based semi-supervised feature selection for hybrid data. *Applied Intelligence*, 53(6):7310–7325.

[177] Shu, W., Yu, J., Chen, T., and Qian, W. (2023b). Neighbourhood discernibility degree-based semisupervised feature selection for partially labelled mixed-type data with granular ball. *Applied Intelligence*.

[178] Shu, W., Yu, J., Yan, Z., and Qian, W. (2023c). Semi-supervised feature selection for partially labeled mixed-type data based on multi-criteria measure approach. *International Journal of Approximate Reasoning*, 153:258–279.

[179] Silva, V. and Tenenbaum, J. (2002). Global versus local methods in nonlinear dimensionality reduction. *Advances in neural information processing systems*, 15.

[180] Song, X., Zhang, J., Han, Y., and Jiang, J. (2016). Semi-supervised feature selection via hierarchical regression for web image classification. *Multimedia Systems*, 22(1):41–49.

[181] Song, Z., Yang, X., Xu, Z., and King, I. (2022). Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*.

[182] Spielman, D. A. (2007). Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 29–38. IEEE.

[183] Sugiyama, M. (2006). Local fisher discriminant analysis for supervised dimensionality reduction. *ACM International Conference Proceeding Series*, 148:905–912. LFDA - Local Fisher Discriminant Analysis.

[184] Sun, Y. and Wen, G. (2015). Emotion recognition using semi-supervised feature selection with speaker normalization. *International Journal of Speech Technology*, 18(3):317–331.

[185] Syed, F. H., Tahir, M. A., Rafi, M., and Shahab, M. D. (2021). Feature selection for semi-supervised multi-target regression using genetic algorithm. *Applied Intelligence*, 51(12):8961–8984.

[186] Tang, B. and Zhang, L. (2020). Local preserving logistic i-relief for semi-supervised feature selection. *Neurocomputing*, 399:48–64.

[187] Tomek, I. (1976). Two modifications of cnn. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(11):769 – 772. Cited by: 1059.

[188] Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., and Moore, J. H. (2018). Relief-based feature selection: Introduction and review. *Journal of Biomedical Informatics*, 85:189–203.

[189] Van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, 109(2):373–440.

[190] Velliangiri, S., Alagumuthukrishnan, S., et al. (2019). A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111.

[191] Vidya, A., Shanthi, D., Gokulakrishnan, P., and Manivannan, K. (2019). Lehality prediction of highly disproportionate data of icu deceased using extreme learning machine. *International Journal of Innovative Technology and Exploring Engineering*.

[192] Wang, C., Chen, X., Yuan, G., Nie, F., and Yang, M. (2021a). Semisupervised feature selection with sparse discriminative least squares regression. *IEEE Transactions on Cybernetics*, 52(8):8413–8424.

[193] Wang, S., Celebi, M. E., Zhang, Y.-D., Yu, X., Lu, S., Yao, X., Zhou, Q., Miguel, M.-G., Tian, Y., Gorriz, J. M., et al. (2021b). Advances in data preprocessing for biomedical data fusion: An overview of the methods, challenges, and prospects. *Information Fusion*, 76:376–421.

[194] Wang, W., Lu, L., and Wei, W. (2022). A novel supervised filter feature selection method based on gaussian probability density for fault diagnosis of permanent magnet dc motors. *Sensors*, 22(19):7121.

[195] Wang, X.-d., Chen, R.-C., Hong, C.-q., Zeng, Z.-q., and Zhou, Z.-l. (2017a). Semi-supervised multi-label feature selection via label correlation analysis with l1-norm graph embedding. *Image and Vision Computing*, 63:10–23.

[196] Wang, X.-d., Chen, R.-C., Yan, F., and Zeng, Z.-q. (2016). Semi-supervised feature selection with exploiting shared information among multiple tasks. *Journal of Visual Communication and Image Representation*, 41:272–280.

[197] Wang, X.-D., Chen, R.-C., Yan, F., Zeng, Z.-Q., and Hong, C.-Q. (2018). Semi-supervised adaptive feature analysis and its application for multimedia understanding. *Multimedia Tools and Applications*, 77(3):3083–3104.

[198] Wang, Y. and Wang, J. (2023). Neurodynamics-driven holistic approaches to semi-supervised feature selection. *Neural Networks*, 157:377–386.

[199] Wang, Y., Wang, J., Liao, H., and Chen, H. (2017b). An efficient semi-supervised representatives feature selection algorithm based on information theory. *Pattern Recognition*, 61:511–523.

[200] Weston, J., Collobert, R., Sinz, F., Bottou, L., and Vapnik, V. (2006). Inference with the universum. In *Proceedings of the 23rd international conference on Machine learning*, pages 1009–1016.

[201] Wijayanto, I., Humairani, A., Hadiyoso, S., Rizal, A., Prasanna, D. L., and Tripathi, S. L. (2023). Epileptic seizure detection on a compressed eeg signal using energy measurement. *Biomedical Signal Processing and Control*, 85:104872.

[202] Wu, Q., Zhang, H., Jing, R., and Li, Y. (2019). Feature selection based on twin support vector regression. In *2019 IEEE symposium series on computational intelligence (SSCI)*, pages 2903–2907. IEEE.

[203] Wu, X., Chen, H., Li, T., and Wan, J. (2021). Semi-supervised feature selection with minimal redundancy based on local adaptive. *Applied Intelligence*, 51(11):8542–8563.

[204] Xiang, S., Nie, F., Meng, G., Pan, C., and Zhang, C. (2012). Discriminative least squares regression for multiclass classification and feature selection. *IEEE transactions on neural networks and learning systems*, 23(11):1738–1754.

[205] Xiao, J., Cao, H., Jiang, X., Gu, X., and Xie, L. (2017). GMDH-based semi-supervised feature selection for customer classification. *Knowledge-Based Systems*, 132:236–248.

[206] Xiao, Y., Wu, J., Lin, Z., and Zhao, X. (2018). A semi-supervised deep learning method based on stacked sparse auto-encoder for cancer prediction using rna-seq data. *Computer methods and programs in biomedicine*, 166:99–105.

[207] Xiao, Z., Dellandrea, E., Dou, W., and Chen, L. (2008). *ESFS: A new embedded feature selection method based on SFS*. PhD thesis, Ecole Centrale Lyon; Université de Lyon; LIRIS UMR 5205 CNRS/INSA de Lyon ....

[208] Xu, C., Tao, D., and Xu, C. (2013). A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

[209] Xu, J., Tang, B., He, H., and Man, H. (2017). Semisupervised feature selection based on relevance and redundancy criteria. *IEEE Transactions on Neural Networks and Learning Systems*, 28(9):1974–1984.

[210] Xu, Y., Wang, J., An, S., Wei, J., and Ruan, J. (2018). Semi-supervised multi-label feature selection by preserving feature-label space consistency. *International Conference on Information and Knowledge Management, Proceedings*, page 783 – 792. Cited by: 24.

[211] Yang, J., Li, T., Liang, G., He, W., and Zhao, Y. (2019). A simple recurrent unit model based intrusion detection system with DCGAN. *IEEE Access*, 7:83286–83296.

[212] Yang, T., Deng, Y., Yu, B., Qian, Y., and Dai, J. (2022). Local feature selection for large-scale data sets limited labels. *IEEE Transactions on Knowledge and Data Engineering*.

[213] Yang, X.-K., He, L., Qu, D., and Zhang, W.-Q. (2018). Semi-supervised minimum redundancy maximum relevance feature selection for audio classification. *Multimedia Tools and Applications*, 77(1):713–739.

[214] Yang, X.-K., He, L., Qu, D., Zhang, W.-Q., and Johnson, M. T. (2016). Semi-supervised feature selection for audio classification based on constraint compensated laplacian score. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016(1):9.

[215] Yang, Y., Nie, F., Xiang, S., Zhuang, Y., and Wang, W. (2010). Local and global regressive mapping for manifold learning with out-of-sample extrapolation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 649–654.

[216] Yao, C. and Guo, Z. (2023). Revisit neighborhood preserving embedding: A new criterion for measuring the manifold similarity in dimension reduction. *Available at SSRN 4349051*.

[217] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

[218] Yuan, G., Chen, X., Wang, C., Nie, F., and Jing, L. (2018a). Discriminative semi-supervised feature selection via rescaled least squares regression-supplement. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). Number: 1.

[219] Yuan, H., Zheng, J., Lai, L. L., and Tang, Y. Y. (2018b). Sparse structural feature selection for multitarget regression. *Knowledge-Based Systems*, 160:200–209.

[220] Zemmal, N., Azizi, N., Sellami, M., Zenakhra, D., Cheriguene, S., Dey, N., and Ashour, A. S. (2018). Robust feature selection algorithm based on transductive svm wrapper and genetic algorithm: application on computer-aided glaucoma classification. *International Journal of Intelligent Systems Technologies and Applications*, 17(3):310–346.

[221] Zeng, Z., Wang, X., and Chen, Y. (2017). Multimedia annotation via semi-supervised shared-subspace feature selection. *Journal of Visual Communication and Image Representation*, 48:386–395.

[222] Zeng, Z., Wang, X., Yan, F., and Chen, Y. (2019). Local adaptive learning for semi-supervised feature selection with group sparsity. *Knowledge-Based Systems*, 181:104787.

[223] Zeng, Z., Wang, X., Zhang, J., and Wu, Q. (2016). Semi-supervised feature selection based on local discriminative information. *Neurocomputing*, 173:102–109.

[224] Zenglin Xu, King, I., Lyu, M. R.-T., and Rong Jin (2010). Discriminative semi-supervised feature selection via manifold regularization. *IEEE Transactions on Neural Networks*, 21(7):1033–1047.

[225] Zhang, C., Bi, J., and Soda, P. (2017). Feature selection and resampling in class imbalance learning: Which comes first? an empirical study in the biological domain. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 933–938.

[226] Zhang, C., Jiang, B., Wang, Z., Yang, J., Lu, Y., Wu, X., and Sheng, W. (2023a). Efficient multi-view semi-supervised feature selection. *Information Sciences*, 649:119675.

[227] Zhang, C., Zhu, L., Shi, D., Zheng, J., Chen, H., and Yu, B. (2022a). Semi-supervised feature selection with soft label learning. *IEEE/CAA Journal of Automatica Sinica*.

[228] Zhang, D., Chen, S., and Zhou, Z.-H. (2008). Constraint score: A new filter method for feature selection with pairwise constraints. *Pattern Recognition*, 41(5):1440–1451.

[229] Zhang, H., Gong, M., Nie, F., and Li, X. (2022b). Unified dual-label semi-supervised learning with top-k feature selection. *Neurocomputing*, 501:875–888.

[230] Zhang, Q., Zhao, Z., Liu, F., and Li, Z. (2023b). Uncertainty measurement for single cell RNA-seq data based on class-consistent technology with application to semi-supervised gene selection. *Applied Soft Computing*, 146:110645.

[231] Zhao, J., Lu, K., and He, X. (2008). Locality sensitive semi-supervised feature selection. *Neurocomputing*, 71(10):1842–1849.

[232] Zhao, X. and Jia, M. (2018). Fault diagnosis of rolling bearing based on feature reduction with global-local margin fisher analysis. *Neurocomputing*, 315:447–464.

[233] Zhao, Z. and Liu, H. (2007). Semi-supervised feature selection via spectral analysis. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 641–646. SIAM.

[234] Zheng, Z., Yang, F., Tan, W., Jia, J., and Yang, J. (2007). Gabor feature-based face recognition using supervised locality preserving projection. *Signal Processing*, 87:2473–2483. SLPE - Supervised Locality Pursuit Embedding.

[235] Zhong, W., Chen, X., Nie, F., and Huang, J. Z. (2021). Adaptive discriminant analysis for semi-supervised feature selection. *Information Sciences*, 566:178–194.

[236] Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

[237] Zhu, Z., Ong, Y.-S., and Dash, M. (2007). Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition*, 40(11):3236–3248.

A

Supplementary Materials: Systematic review semi-supervised feature selection

This document presents the supplementary materials of the paper titled “Systematic review of semi-supervised feature selection methods”. The material includes a list of the 103 semi-supervised feature selection (SSFS) methods reviewed in the main document with a brief description. Additionally, in Table A.1, the links to the top 40 most commonly used datasets already mentioned in the main document are provided.

A.1 Algorithms description list

- **Filter - Laplacian score - Classification**
  - **LSDF** [231] uses labelled points to maximise the margin between classes. In contrast, unlabelled points are used to discover the geometric structure of the data space.
  - **ALDS** [36] incorporates asymmetric class miss classification costs in the weight matrices.

  - **SSMCFS** [184] extends the MCFS (multi-cluster feature selection) model based on the Laplacian matrix eigendecomposition for SSL.
  - **RFR** [139] is based on the Neighbourhood Discriminant Index (NDI) FS and uses a forward iterative approach to the proposed Laplacian score (LS).
  - **SSML** [35] used for multi-modality data, where samples in some classes are in several clusters.
  - **SSFSM-DTI** [164] is an optimised version of LSDF for drug-target interaction datasets.

- **Filter - Laplacian score - Regression**
  - **SSLS** [51] is a SSL version of the LS, presented alongside a supervised version in the same article.
  - **S2FSGL** [168] focuses on quantitative structure-activity relationship (QSAR) models.
  - **S3FSGL** [171] is a sparse version of the previous method called S2FSGL.
  - [95] is a multi-view and multi-objective algorithm based on the Laplacian matrix.
  - **SSNDI** [140] is a version of the supervised FS NDI, based on the neighbourhood discrimination index combined with the LS to create.
  - **SIDLS** [81] uses the LS with the divergence of spectral information, which uses class probability information to construct graphs.

- **Filter - Pairwise constraints - Classification**
  - **C4** [85]
  - **CSFS** [73]
  - **PCDLRD** [33] is a sparse model using the $L_{2,1}$ norm.

- **Filter - Pairwise constraints - Regression**
  - **SSCS** [167] is oriented towards QSAR data.
  - **SFS** [111] integrates ridge regression with pairwise constraints to improve accuracy in regression problems.

- **Filter - Pairwise constraints - Multi-label**

  - **3-3FS** [4] is an ensemble that relies on data resampling through three different methods: bagging, random sub-space method and a new resampling algorithm called random sub-labelling strategy, centred on the label ensemble.

- **Filter - Laplacian score + pairwise constraints - Classification**

  - **CLS** [15] uses C1 to constrain the effectiveness of LS for efficient semi-supervised feature selection, knowing that the LS assumption that data points from the same class are inherently close to each other may not always hold true.
  - **CSFSR** [16] first focuses on reducing redundancy between selected features by selecting pairwise constraints before applying a filter based on CLS.
  - **CCLS** [214] is an algorithm optimised for audio classification.

- **Filter - Laplacian score + pairwise constraints - Classification**

  - **S-CLS** [5] utilises CLS with an ensemble technique incorporating data resampling (bagging) and a random subspace strategy to mitigate noise on multi-label problems.

- **Filter - Sparse models - Classification**

  - **ISR** [124] extends label information through label propagation and utilises a proposed Insensitive Regression Model (IRM) with a $L_{2,p}$ norm.

- **Filter - Sparse models - Regression**

  - [77] propose an SSFS method based on Flexible Manifold Embedding (FME) FS, applying sparse regularisation techniques to enhance model smoothness and reduce sensitivity to noise.

- **Filter - Sparse models - Classification or Regression**

  - **GS3FS** [170] utilises a mixed convex and non-convex $L_{2,p}$ norm ($0 < p \leq 1$) minimisation approach for both classification and regression, focusing on regularisation and loss function optimisation.

- **Filter - Sparse models - Multi-label**

  - **SFS-BLL** [175] presents the Binary Hash Learning, which increases the number of labels by binary hash constraints.

- **Filter - Adaptive graph models - Classification**

  - **OGE-SFS** [123] employs adaptive neighbour assignment and an adaptive loss function to shape the model.
  - **ALFS** [222] integrates a local adaptive loss function with a global sparsity constraint to enhance adaptability across various data distributions.
  - **SADA** [235] iteratively learns an adaptive similarity matrix by a projection matrix and controlling sparsity through the $L_{2,p}$ norm.
  - **Sr-SemiDFS** [53] formulates its approach assuming two data structures: soft data structure for pairwise weights and hard data structure for estimated labels obtained from clustering or semi-supervised classification.
  - **RDMRS2FS** [34] utilises an adaptive redundancy regularisation term based on self-representation to minimise redundancy between features.
  - **ASLCGLFS** [103] aims to enhance the quality of the similarity matrix by leveraging label information to guide graph learning, addressing that this crucial information is ignored in most adaptive graph learning methods.

- **Filter - Adaptive graph models - Regression**
  - **MASFS** [173] algorithm introduces self-paced learning (SPL) to the SSFS, enabling the Laplacian weight graph to adapt according to label information.
  - **SFS-LARLRM** [203] incorporates local adaptive and minimal redundancy, employing a similarity penalty to promote discrimination and reduce redundancy within the selected feature subset.

- **Filter - Adaptive graph models - Multi-label**
  - **CSFS** [30] criticises the reliance of multi-label SSFS methods on the manifold assumption for exploring label correlations, proposing an optimisation strategy that combines adaptive global structure learning with manifold learning.
  - **SFAM** [126] note that sparse feature selection often leads to the selection of redundant features due to similar weights assignation; therefore, it introduces redundancy minimisation regularisation to enhance the quality of the similarity matrix and mitigate the negative impact of redundant features.
  - **AGLRM** [102] deals with the computational load by large datasets relying on eigen-decomposition.
  - **EMSFS** [226] is a multi-view algorithm that combines graph learning with label propagation to construct a bipartite graph between training samples and generated anchors.

- **Filter - Regression-based models - Classification**
  - **S2FS2R** [65] employs a sparse framework based on splines.
  - **FSHR** [180] adopts a hierarchical regression approach.
- **Filter - Clustering - Classification**
  - **sSelect** [233] based on spectral analysis and operating under cluster assumptions, introduces an index for clustering purposes.
  - **SSFC** [44] introduces a novel mutual information measure alongside hierarchical feature clustering to eliminate redundant features.
  - **PCFS** [153] combines pairwise constraints while optimising a clustering measure known as Dim-reduce.
  - **LPFS** [80] is a clustering model based on label propagation.
- **Filter - Fisher Score based - Regression**
  - **SSFLS** [68] is an algorithm optimised for QSAR datasets.
- **Filter - Fisher Score based - Multi-label**
  - **LGDF** [76] combines Fisher score with label propagation.
- **Filter - Hessian - Classification**
  - **HFSL** [174] combines Hessian matrix with a $L_{2,1/2}$ norm.
  - **SMHFS** [172] employs multi-view sparse regularisation followed by a multi-view Hessian approach.
  - **HSFSGU** [166], uses a Hessian regularization matrix and a $L_{2,p}$-norm regularization to maintain the topological structure of data.
  - **HLSFSGU** [166] is an alternative version of HSFSGU that combines Hessian and Graph Laplacian matrices.
- **Filter - Bayesian - Classification**
  - **BASSUM** [28]
  - **SRFS** [199] applies information theory to identify and eliminate irrelevant and redundant features.
  - **Semi-IAMB** [161] based on Joint Mutual Information (JMI).

  - **SSHIBA** [163] is a multi-view multi-task approach based on the Bayesian Inter-Battery Factor Analysis (BIBFA) model.

- **Filter - Minimum Redundancy Maximum Relevance - Classification**
  - **RRPC** [209] combines Pearson correlation with MRMR,
  - **SSMRMR** [213] applies MRMR with the CCLS mentioned above to measure feature relevance effectively.

- **Filter - Relief - Classification**
  - **HM-ICS** [40] integrates FS Relief-SC and C1 using pairwise constraints.

- **Filter - Relief - Multi-target**
  - **LPLIR** [186] is based on the Logistic I-Relief, employs local preserving regularisation to maximise the margin of labelled data while preserving local structural information.

- **Filter - Fuzzy - Classification**
  - **SFS-SLL** [227] learns initial soft labels based on local distance clustering centres using fuzzy C-means clustering in conjunction with a sparse regression model.
  - **SemiFREE** [118] adopt relevance-maximisation and redundancy-minimisation principles.

- **Filter - Universum data - Classification**
  - **UVS** [147] based on Variance Score.
  - **ULS** [147] based on Laplacian Score.
  - **USS** [147] based on Sparsity Score.

- **Filter - Other - Classification**
  - [122] introduce a noise-insensitive trace ratio criterion to mitigate the impact of noisy data.
  - **GLSPFS** [120] employs feature reduction to retain global and local structures in feature selection.
  - **UFSSI** [223] exploits local discriminative information to construct a linear regression model under the $L_{2,1}$ norm.

- **ASFS** [38] algorithm evaluates each feature by linear objective functions based on loss functions and probability distribution matrices.
- [196] proposes a multi-task learning approach for large datasets in multimedia annotation, avoiding Laplacian graph construction.
- [197] integrates adaptive local manifold learning and FS, applying sparse constraints like the $L_{2,1}$ norm.
- [221] propose an efficient approach which avoids creating the Laplacian graph.
- **Semi-JMI** [161] is based on Incremental Association Markov Blanket (IAMB)
- **RSES** [119] introduces an ensemble based on rough sets, proposing the Local Neighbourhood Decision Error Rate (LNDER) to construct multiple fitness functions for feature significance evaluation.
- **NGAR** [107] utilises approximate sets and neighbourhood granulation measures.
- **S2LFS** [113] introduces a class-specific algorithm allowing the selection of different feature subsets for different classes.
- **UDM-SFS** [229] employs a soft label matrix to identify inaccessible samples near class boundaries while imposing an $L_{2,0}$ norm constraint on the projection matrix.
- **LRF** [212] presents an algorithm for large-scale datasets based on the locally related family framework.
- **IMP4ARA** [230] incorporates an uncertainty measurement based on class-consistent technology.
- **SemiACO** [87] is based on the ACO algorithm, which minimises redundancy and maximises relevance with a nonlinear heuristic function.
- **N-Semi-IG** [198] employs a one-layer projection neural network to optimise multi-information measures by minimising redundancy and maximising relevance.

- **Filter - Other - Regression**
  - **SFMC** [31] presents a multi-task learning algorithm for video recognition, aiming for versatility across different problem types while considering feature correlations in batch mode feature selection.

- **Filter - Other - Multi-label**

  - [195] presents a sparse algorithm which seeks a shared structure for assigning multiple labels to a single sample simultaneously.
  - [221] use an efficient approach which avoids creating the Laplacian graph.
  - **SCFS** [210] relies on probabilistic neighbourhood similarities and correlation information in label space for SSFS.

- **Wrapper Search - Classification**
  - **GA-TSVM** [220] uses a genetic algorithm alongside a transductive SVM.

- **Wrapper Search - Multi-target regression**
  - [185] uses as base classifier a transductive SVM.

- **Wrapper Pseudo-labelling - Classification**
  - **FW-SemiFS** [151] combines pseudo-labelling with the forward search wrapper, using the search by adding pseudo-labelled instances at each iteration and then selecting the most relevant ones.
  - **EnsCLS** [14] employs K-Nearest Neighbors (KNN) label propagation for pseudo-labelling, incorporating a feature selection method based on information entropy and granulation.
  - **GMDH-SSFS** [205] combines pseudo-labelling with neural network-based embedding using the group data manipulation (GMDH) method.
  - **TSLA-FSGA** [54] labels unlabelled instances with transductive self-training and searches for the best ones using a genetic approach.
  - **FDG** [177] is an ensemble which uses the proposed neighbourhood discernibility measure, compatible with mixed data and missing values.
  - **ISFS** [176] is used for hybrid data utilising information gain-based pseudo-labelling with an enlarged neighbourhood granule.
  - **SFM** [178] is a mixed input type-compatible algorithm, utilises a novel label propagation algorithm combining K-nearest neighbour with a new feature multi-criteria measure derived from dependency, information entropy, and information granulation.

- **Embedded - Tree based - Classification**

  - **SEFR** [13] employs the random forest out-of-the-bag permutation technique to select features.
  - **OFFS** [162] applies random forest out-of-the-bag permutation technique with co-forest.

- **Embedded - Tree based - Multi-target**
  - **SSS** [3] employs a set of random trees for multi-target regression.

- **Embedded - SVM - Classification**
  - **FS-Manifold** [224] combines the SVM with manifold regularization.
  - **SENFS** [46] applies the elastic net penalty to the SVM.

- **Embedded - Linear regression - Classification**
  - **RLSR** [37] applies an $L_{2,p}$ norm to improve model performance in classification problems.
  - **SDSSFS** [192] extends the RLSR using the e-dragging technique.

- **Embedded - Linear regression - Classification**
  - **SRLSR** [39] is an adaptation of RLSR to regression problems.
  - **DSSFS** [218] is another extension to RLSR which incorporates a technique called e-dragging [204].

- **Embedded - Autoencoder - Classification**
  - **A-SFS** [148] checks the weights of an autoencoder to assume the importance of the features.

- **Embedded - KNN - Classification**
  - **KNN-FRS-SSFS** [7] uses the KNN with fuzzy logic to effectively work on datasets with large density variations.

- **Embedded - Spline - Classification**
  - **SRS3FS** [165] is a sparse spline-based wrapper designed to work on regression problems.

A.2 Links to datasets

Table A.1 Links of the top most commonly used datasets

Data	URL
Coil20	https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
Ionosphere	https://archive.ics.uci.edu/dataset/52/ionosphere
Sonar	https://archive.ics.uci.edu/dataset/151/connectionist+bench+sonar+mines+vs+rocks
Image Segmentation	https://archive.ics.uci.edu/dataset/50/image+segmentation
ORL & Research Laboratory	https://jundongl.github.io/scikit-feature/datasets.html
USPS & States Postal Service data set	https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html
WBDC	http://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
NUS-WIDE	https://mulan.sourceforge.net/datasets-mlc.html
Wine	https://unifeat.github.io/datasets.html
Colon	https://jundongl.github.io/scikit-feature/datasets.html
Binary alphabet	http://www.escience.cn/system/file?fileId=82035
Dermatology	https://archive.ics.uci.edu/dataset/33/dermatology
glass	https://archive.ics.uci.edu/dataset/42/glass+identification
Isolet	https://archive.ics.uci.edu/dataset/54/isolet
Yeast	https://mulan.sourceforge.net/datasets-mlc.html
Statlog German	http://archive.ics.uci.edu/dataset/144/statlog+german+credit+data
Heart disease	https://archive.ics.uci.edu/dataset/45/heart+disease
WPBC	https://archive.ics.uci.edu/dataset/16/breast+cancer+wisconsin+prognostic
Statlog vehicle	https://archive.ics.uci.edu/dataset/149/statlog+vehicle+silhouettes
Pie10P	https://jundongl.github.io/scikit-feature/datasets.html
Scene Image	https://mulan.sourceforge.net/datasets-mlc.html
Breast	https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original
Madelon	https://unifeat.github.io/datasets.html
Musk	https://archive.ics.uci.edu/dataset/74/musk+version+1
Yale	https://jundongl.github.io/scikit-feature/datasets.html
Waveform	https://archive.ics.uci.edu/dataset/108/waveform+database+generator+version+2
Leukemia	https://jundongl.github.io/scikit-feature/datasets.html
hepatitis	https://archive.ics.uci.edu/dataset/46/hepatitis
Libras Movement	http://archive.ics.uci.edu/dataset/181/libras+movement
PcMac	https://jundongl.github.io/scikit-feature/datasets.html
Prostate Tumor	https://unifeat.github.io/datasets.html
Pima	https://sci2s.ugr.es/keel/category.php?cat=clas
HumanEVA	http://humaneva.is.tue.mpg.de/
Semeion	https://archive.ics.uci.edu/dataset/178/semeion+handwritten+digit
CNAE-9	https://archive.ics.uci.edu/dataset/233/cnae+9
Ecoli	https://archive.ics.uci.edu/dataset/39/ecoli
MNIST	https://archive.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits
YaleB	https://www.kaggle.com/datasets/tbourton/extyalebcroppedpng
MIML	https://www.lamda.nju.edu.cn/data_MIMLimage.ashx
Parkinsons	https://www.kaggle.com/datasets/tbourton/extyalebcroppedpng