TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Feature Importance in Imbalanced Binary Classification with Ensemble Methods

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Embedded Systems

by

Sasa Glamocak, BSc
Registration Number 11723498

to the Faculty of Electrical Engineering and Information Technology

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.-Ing. Tanja Zseby
Assistance: Senior Scientist Dr.techn. Félix Iglesias Vazquez

Vienna, 27th December, 2023

Erklärung zur Verfassung der Arbeit

Sasa Glamocak, BSc

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, 27. Dezember 2023

Acknowledgements

I would like to take this opportunity to thank everyone who supported me during this thesis.

First and foremost, I would like to express my gratitude to Senior Scientist Dr.techn. Félix Iglesias Vázquez for his patience and guidance throughout this thesis. Hopefully, I have met your expectations. Additionally, I would like to thank Univ.Prof. Dipl.-Ing. Dr.-Ing. Tanja Zseby for her support and the knowledge I acquired from her lessons. I wish you the best of luck and great success in your future work.

I want to sincerely thank my parents, Ranko and Vesna, who are the two people I love and respect the most in the world. Their encouragement and faith in me have helped me to grow in every aspect of my life. I am also grateful to my brothers, Danijel and Rale, as well as to their families. Lastly, I would like to thank my beloved girlfriend, Doris, who has been with me for many years and I hope it remains so. Thank you all, because any success is much greater if there is someone to share it with.

Kurzfassung

Die Merkmalsauswahl ist zu einem wichtigen Vorverarbeitungsschritt in der Welt des maschinellen Lernens und der Datenanalyse geworden, insbesondere wenn es um Daten geht, die eine erhebliche Menge an Attributen aufweisen. Dieser Prozess ist wichtig, da er (a) die Leistung von Prädiktoren verbessert, (b) den Rechenaufwand reduziert und (c) dabei hilft, den zugrunde liegenden Prozess hinter den Daten zu verstehen. Darüber hinaus sind viele aktuelle Anwendungen binäre Klassifizierungsprobleme mit unausgeglichenen Klassen in den Datensätzen; zum Beispiel Betrugserkennung, medizinische Diagnose und Cybersicherheit. Der Einfluss des Klassenungleichgewichts auf die Merkmalsauswahl wurde in der wissenschaftlichen Literatur jedoch bisher nicht ausreichend berücksichtigt.

In dieser Studie analysieren wir Methoden zur Merkmalsauswahl, die für das Problem der Merkmalsauswahl in unausgeglichenen Daten entwickelt wurden. Ziel ist es, nicht nur die zuverlässigste Methode, sondern auch die Bedeutung der Ergebnisse der Merkmalsauswahl und deren Interpretation offen zu legen. Unsere Studie umfasst Ensembles als Kernalgorithmen (Random Forest, XGBoost), Datenausgleichstechniken (Random Undersampling (RUS), Synthetic Minority Oversampling Technique (SMOTE), kostensensitives Lernen), Alternativen zur Merkmalsbewertung (Mean Decrease in Impurity (MDI), Permutation Feature Importance (PI), SHapley Additive exPlanations (SHAP)) und Korrekturen basierend auf Bewertungen der Multikollinearität mithilfe von Variance Inflation Factor (VIF). Experimente werden mit unausgeglichenen Datensätzen aus verschiedenen Bereichen durchgeführt.

Wir bewerten die Leistung der ausgewählten Attribute mit dem ROC AUC Wert. Die Versuchsergebnisse zeigen, dass mehrere Kombinationen durchweg hohe ROC AUC Werte aufweisen, insbesondere XGBoost in Kombination mit SMOTE und SHAP sowie Random Forest in Kombination mit RUS und SHAP. Insbesondere sticht PI als eine außergewöhnlich diskriminierende Technik zur Merkmalsbewertung hervor. Allerdings verbessert die VIF Korrektur die ROC AUC Leistung oder Zuverlässigkeit getesteter Kombinationen nicht durchgängig.

Diese Forschung bietet eine umfassende Analyse verschiedener Strategien, um das Problem der Merkmalsauswahl in unausgeglichenen Daten anzugehen. Durch die Identifizierung leistungsstarker, zuverlässiger und diskriminierender Merkmalsauswahlkombinationen bietet diese Studie wertvolle Erkenntnisse zur Verbesserung der Anomalieerkennung in kritischen Bereichen wie der Netzwerksicherheit. Unter der Annahme, dass die getesteten Kombinationen zeitlich realisierbar sind, können ein geringerer Rechen-

aufwand und die Auswahl von Merkmalen, die zu präzisen Vorhersagen beitragen, die Sicherheitsmaßnahmen in der Praxis erheblich verbessern. Zukünftige Forschungsarbeiten können auf unseren Ergebnissen aufbauen, um bessere Anomalieerkennungssysteme zu entwickeln.

Abstract

Feature selection has become an important pre-processing step in the world of machine learning and data analysis, particularly when dealing with data that has a substantial amount of attributes. This process is essential as it (a) improves the performance of predictors, (b) reduces computational requirements, and (c) helps understand the underlying process behind the data. Moreover, many current applications are binary classification problems with imbalanced classes in the datasets; for instance, fraud detection, medical diagnosis, and cybersecurity. However, the impact of class imbalance on the feature selection has not received proper attention in the scientific literature so far.

In this study, we analyze feature selection methods designed for the problem of feature selection in imbalanced data. The goal is to disclose not only the most reliable method, but also the meaning of the feature selection scores and how to interpret them. Our study covers ensembles as core algorithms (Random Forest, XGBoost), data balancing techniques (Random Undersampling (RUS), Synthetic Minority Oversampling Technique (SMOTE), cost-sensitive learning), feature scoring alternatives (Mean Decrease in Impurity (MDI), Permutation Feature Importance (PI), SHapley Additive exPlanations (SHAP)), and corrections based on evaluations of multicollinearity using Variance Inflation Factor (VIF). Experiments are performed on imbalanced datasets from various domains.

We assess the performance of the selected feature subsets with ROC AUC scores. Experiment results show that several combinations demonstrate consistently high ROC AUC scores, especially XGBoost combined with SMOTE and SHAP, as well as Random Forest united with RUS and SHAP. Notably, PI stands out as an exceptionally discriminative feature scoring technique. However, VIF correction does not consistently improve the ROC AUC performance or stability of tested combinations.

This research provides a comprehensive analysis of various methods to address the problem of feature selection in imbalanced data. By identifying high-performing, stable, and discriminative feature selection combinations, this study offers valuable insights for improving anomaly detection in critical domains like network security. Assuming the time viability of tested combinations, reduced computational demands and selection of the features that help generate accurate predictions can significantly improve security measures in practice. Future research can build upon our results to develop more sophisticated anomaly detection systems.

Contents

CHAPTER 1

Introduction

In this chapter, we provide the background of our research. We will discuss the motivation and need for this study and outline the targeted goals and methodology. Finally, we give a brief overview of the thesis structure.

1.1 Background

The rapid development in information technology and communication networks has immersed humanity in the world of data. The importance of data has been steadily growing with the rise of machine and deep learning, which have garnered attention in both research and industry. The data is now a valuable asset in various domains, such as healthcare, bioinformatics, computer vision, and network security. However, the nature of this data is high-dimensional, obscuring an analysis that would result in valuable insights and domain knowledge.

It is of great importance to find strategies to reduce the data dimensionality, without dropping any important information. Well-established approaches involve *feature selection* and *feature extraction*. Feature selection performs the selection of the most relevant features in a dataset. The main objective is to reduce the information redundancy and complexity within the chosen subset of features. This process introduces several benefits; (a) improved performance of predictors, (b) reduced computational requirements, and (c) simplified understanding of the underlying process behind the data [1]. The high dimensional data causes many problems to machine learning algorithms and one of them is known as the *curse of dimensionality*. This term was originally introduced by Bellman [2], who described that the number of data samples needed to estimate a function exponentially grows with the number of the input variables in that function. The dimensional sparsity has a detrimental impact on most machine learning algorithms, which are typically designed for lower-dimensional spaces. The model that has access to a large number of features, becomes capable of generating mathematically more complex

functions. As a result, this enlarged complexity can potentially lead to overfitting and cause performance deterioration on unseen data. High-dimensional data introduces further limitations such as increased memory storage requirements and computation costs. Therefore, it is crucial to find a strategy to reduce the data dimensionality. In addition to feature selection, feature extraction is a widely used approach. This process generates completely new features which are linear or non-linear combinations of the existing ones.

Feature selection is of utmost importance in numerous real-life scenarios and applications, especially those that exemplify imbalanced binary classification; for instance, anomaly detection, fraud detection, medical diagnosis, and others. However, how imbalanced classes affect the feature selection process and the dynamics of the scores have not been adequately studied despite their importance. Additionally, the classification of network traffic data with an imbalanced class distribution has been a great difficulty for many well-known classifiers, which mainly assume that the underlying data is balanced. Numerous researchers have shown the potential benefits of employing data sampling techniques to mitigate the model's bias towards the majority class and improve classification accuracy [3, 4, 5, 6]. Moreover, several researchers have attempted to answer whether it is more effective to apply the data sampling before or after feature selection in imbalanced wide datasets [7]. The authors in [8] also investigated the effect of the sampling ratio on classification accuracy and concluded that the exact ratio mainly depends on the dataset, sampling method, and used performance metric. An extensive review of methods for imbalanced data is provided in [9].

In addition to building a model with high predictive performance, it is important to comprehend the process of decision making. Artificial intelligence has experienced tremendous growth over the last decade, securing a firm place in the industry, where machine learning systems consistently outperform human capabilities. However, to improve the system's performance, it is necessary to increase its internal complexity. In the end, it is challenging to understand the operation of these models and therefore, they are labeled as "black boxes". The systems that are not interpretable are also difficult to be trusted, especially in critical areas such as healthcare or self-driving cars. This need for trustworthiness led to a new field in machine learning research - eXplainable Artificial Intelligence (XAI) [10], which became essential for system verification, regulatory compliance, elucidation of ethical concerns, trustworthiness, and system diagnostics [11]. There are many open-source libraries for machine learning interpretability, which can be used to better understand the inference of the models [12, 13, 14].

Therefore, in the era of deep learning and big data, *feature selection* is more important than ever, as it not only reduces the effort in computational and analytical terms, but – and this is its major purpose – it discloses the real value and contribution of each original feature for the aimed classification task, therefore helping us to understand how and why a problem is solved in addition to solving a problem.

1.2 Motivation

We currently live in an era dominated by computer technology and worldwide communication networks. Only a few decades ago, it was unimaginable to communicate seamlessly with a device or a person located on the opposite side of the world, but over time, almost instantaneous global communication has become a reality. With the development of communication networks, primarily the Internet, an immeasurable amount of network traffic is transmitted every day, significantly easing various spheres of our lives such as automation of tasks, remote controlling, digital storage of information, financial transactions, entertainment, education, and others. However, despite these technological advancements, many challenges emerged as well. Systems are increasingly burdened and vulnerable to sophisticated malicious attacks that have become harder to distinguish from normal traffic. Therefore, the security of networks and information along with the ability to detect such threats is becoming more and more important. Despite the prevalence of network traffic, harmful attacks are considerably less frequent compared to normal activities. Therefore, this issue is closely related to the problem of imbalanced data in binary classification, where attacks are in the minority class. This highlights the need to address this area and our attention has been attracted by the interesting approach employed in the previous work [15]. While this prior research provides valuable insights, it leaves space for further investigation and discussion.

This work builds upon the master thesis: *Comparison of Ensemble-Based Feature Selection Methods for Binary Classification of Imbalanced Data Sets* written by Erjola Zeraliu [15]. The author conducted a series of experiments, demonstrating that the combination of ensemble trees with stability selection [16] and Permutation Feature Importance (PI) [17] is a promising approach for feature selection. The remarkable results in performance were particularly obtained with Gradient Boosting [18]. However, our research expands upon this groundwork by incorporating a broader list of imbalanced datasets, diverse feature scoring methods, and the addressing of imbalanced class distributions. In addition, we aim to address the potential issue of having correlated features, a phenomenon known as *multicollinearity* [19]. It will be analyzed to what extent the process of feature selection using ensemble trees is impacted by multicollinearity. However, it is important to emphasize that multicollinearity does not affect the predictive performance of such models [20, 21].

To the best of our knowledge, there is no comprehensive research that offers a systematic and in-depth analysis of the ensemble trees in combination with sampling techniques and diverse feature selection methods in the context of imbalanced binary classification.

1.3 Goals

Assuming the imbalanced binary classification problem, this thesis tackles the following research questions:

- **R1: Accuracy**
  Which feature selection methods are best suited for generating feature sets that lead to a good classification performance?

- **R2: Stability**
  How reliable are approaches for feature selection in terms of correlation between the cumulative importance of the selected subset and its accuracy?

- **R3: Discriminant power**
  How discriminant are scores in feature selection approaches to clearly differentiate between relevant and irrelevant features?

- **R4: Particular impacts**
  What are the implications associated with particular elements of the feature selection pipeline, namely: classification algorithms, data-balancing techniques, multicollinearity removal, and scoring methods?

1.4 Methodology

To achieve our goals we did the following:

1. **Dataset preparation:** We prepared 17 distinct imbalanced datasets for binary classification. Each input dataset was divided into train and test sets with a split ratio of 70/30.

2. **Multicollinearity elimination:** We used the Variance Inflation Factor (VIF) to investigate the effects of multicollinearity measures on the feature selection process. Features with $VIF > 5$ were identified as multicollinear and removed from both the training and test datasets.

3. **Data balancing techniques:** We applied Random Undersampling (RUS) and Synthetic Minority Oversampling Technique (SMOTE) sampling techniques to balance the training data, achieving a ratio between minor and major classes of 1:1.

4. **Classifier training:** We trained and fine-tuned the core classifier that was the backbone of tested feature selection combinations. Employed classifiers were Random Forest (RF), XGBoost (XGB), and their cost-sensitive versions (Random Forest Balanced Cost-Sensitive Version (RF_bal), XGBoost Balanced Cost-Sensitive Version (XGB_bal)). The cost-sensitive versions were not combined with data sampling techniques from the previous step.

5. **Feature importance evaluation:** Core classifiers were combined with feature attribution methods (internal Mean Decrease in Impurity (MDI), PI, or SHapley Additive exPlanations (SHAP)) to compute feature importance scores based on the training data.

6. **Feature subsets selection:** Based on the feature scores, three distinct feature subsets were generated: a subset including the half of the most important features and subsets containing features that contribute to at least 50% or 80% of the total feature importance.

7. **Feature subsets performance:** The quality of the selected subsets was assessed using core classifiers of the corresponding feature selection combination. Receiver Operating Characteristic Area Under the Curve (ROC AUC) scores were recorded as performance metrics.

8. **Comparative results analysis:** Finally, we compared the tested feature selection combinations by examining:

    - Mean ROC AUC and ranking scores of feature selection combinations.
    - Wilcoxon signed-rank test, which showed if there was a significant performance difference between a pair of combinations.
    - Pearson and Spearman correlation analysis between ROC AUC test scores and cumulative feature importance of the feature subset under test (best half, 50% or 80% subset).
    - Mean Coefficient of Variation of feature importance scores to determine the discriminatory power of a feature selection combination.

1.5 Arrangement of the Thesis

The rest of the thesis is arranged in four chapters:

- In chapter 2, we introduce background knowledge on feature selection, data balance strategies and machine learning, with a special focus on aspects related to interpretability.
- In chapter 3, we present the work methodology and the conducted experiments. The concept of multicollinearity and its measurements is additionally introduced. We describe the feature selection combinations and datasets used for evaluation, as well as performance metrics.
- In chapter 4, we show and discuss the results of the conducted experiments. We compare employed feature selection combinations from the various aspects relevant to the stated goals.
- Finally, we conclude in chapter 5 and summarize the findings that emerged from the exploration of the topic and the careful revision of experiments. We also comment on possible future research lines that could continue our work.

CHAPTER 2

Background Knowledge

In this section, the background knowledge for this thesis is provided. Firstly, feature selection and its methods are explained, followed by a brief overview of machine learning focusing on ensemble learning techniques and algorithms. Lastly, data balance strategies and interpretability techniques are discussed.

Note that these topics cover the theoretical framework assumed in this thesis, namely: evaluating ensemble machine learning scores for feature selection in imbalanced binary classification.

2.1 Feature Selection

Feature selection is the process of selecting the informative features from a dataset. The selected subset should represent the initial data without substantial information loss. The discarded features are either irrelevant, having no influence on the output, or redundant, meaning that two or more features share the same information [22]. Feature selection provides many benefits such as improved prediction performance of predictors, increased computation efficiency, decreased memory storage, and better generalization of models [23].

Generally, the feature selection process involves 4 basic steps [24]:

1. Subset generation
2. Evaluation of the subset
3. Stopping criterion
4. Result validation

Subset generation is a search process where a subset of features represents the state to be evaluated in the search space. Two important factors are *successor generation* and *search organization*. Successor generation defines the direction of the search strategy at each state and it includes several options [24]:

- **Forward selection** begins with an empty subset of features and selects one feature at a time from those not yet included. An evaluation criterion determines which one of the unselected features should be added to the current subset. The feature causing the least error is included only if the newly created subset outperforms the current one. Otherwise, the process can terminate, as the inclusion of any unselected features does not improve the performance. Additionally, a predefined maximum number of features in the subset can be used as a stopping criterion. The great limitation of this approach is the neglect of feature interactions [24].

- The **backward selection** is the opposite of forward generation. Rather than starting with an empty feature subset, it begins with a subset containing all features and iteratively removes one feature at a time, selecting the one whose elimination causes the least error. This process results in a reduced subset with enhanced performance. Stopping criteria can be used in the same way as for the forward approach [24].

- Another approach is **compound generation**, which combines M consecutive forward steps with N backward steps, where $M \neq N$. Using evaluation criteria, either forward or backward steps are selected, and the process should conclude when $M = N$ [24].

- The **random** method incorporates all the previously described strategies, generating a random feature subset in a single step [24].

- Finally, the **weighting** approach assigns different weights to features in each step based on sampled data instances, indicating their importance [24].

On the other hand, search organization determines the process of feature selection. The task of generating subsequent candidate subsets is a great challenge since the number of possible subsets increases exponentially with the number of features, 2^n. Consequently, an exhaustive search is not a feasible option. Researchers proposed various strategies to address this issue, including [24]:

- **Sequential search:** The well-known sequential methods are sequential forward selection, sequential backward elimination, and bi-directional selection [25]. This process is iterative, involving the addition, removal, or both, of one or more features at a time. While it is a simple algorithm to implement, the optimal subset may not be found [24].

- **Exponential search:** The exhaustive search is computationally extremely intensive, and often, infeasible process. However, it is possible to employ heuristic functions to shrink the search space without jeopardizing the potential loss of the most optimal feature subset. Algorithms such as BRANCH AND BOUND [26] and Beam Search [27] utilize this approach.

- **Random search:** This search method starts with a random subset of features and continues to generate the subsequent subsets in a completely random manner (e.g., the Las Vegas method [28]). Another approach is to introduce randomness into the sequential approach when selecting the feature to be added or removed. The inclusion of randomness aims to decrease the likelihood of becoming stuck at local minima in the search space [24].

To identify the most optimal subset of features, it is necessary to assess the quality of the selected subset. If the currently chosen feature subset performs better than its best predecessor, it becomes the new best subset and serves as the starting point for subsequent subsets. Two distinct evaluation criteria, based on their dependency on the mining algorithm, can be defined for this purpose: *independent* and *dependent criteria* [24].

Independent criteria do not rely on the mining algorithm and evaluate the relevance of the features using only the intrinsic characteristics of the data. Examples of independent measures include [24]:

- **Distance or divergence measures [29]** calculate the probabilistic distance between the class-conditional probability distributions [24]. A feature is favored if it leads to greater dissimilarity between the observed class probability distributions. Some distance measures are Kullback-Liebler [30], Bhattacharya [31], and Matusita.

- **Information or uncertainty measures** are derived from the information gain of features that represent "the difference between the prior uncertainty and expected posterior uncertainty" [24]. Shannon entropy is a well-known information measure [32].

- **Dependency measures [33, 34]**, also known as correlation measures, identify features strongly correlated with the target as more important in feature selection process[24].

- **Consistency measures [25]** search for the minimum number of features that effectively split the classes as well as all features [24].

On the other hand, dependent criteria select the best features based on their performance with a mining algorithm. Typically, independent criteria are employed by filter methods, while wrapper models utilize dependent criteria.

The previous steps (subset generation and evaluation) are repeated until the predefined stopping criteria are met, indicating the conclusion of the entire process. The feature selection process usually ends when reaching the maximum number of iterations, the minimum or maximum number of features in the subset, the minimum classification error rate, or when the addition or removal of features no longer improves performance. Depending on the algorithms employed, the output can be an optimal subset of features (performing the best in terms of accuracy) or weighted features.

Finally, the quality of the selected feature subset is evaluated in the validation phase, often using new, unseen data.

Generally, all feature selection methods can be categorized into three classes [1, 35]:

- Filter methods
- Wrapper methods
- Embedded methods

A brief introduction to these methods is provided in the following sections. Additionally, an overview of feature selection methods is presented in Table 2.1.

			Search Strategies	
		Exponential	**Sequential**	**Random**
Evaluation Criteria	**Filter**	Distance	B&B[26], BFF[36], BOBRO[37], OBLIVIIN[38]	Relief[39], RelifS[40], SFS[41], Segen's[42], SBS[43]	
		Information	MDLM[44], CARDIE[45]	DTM[45], Koller's[46], FG[25], FCBF[47], BSE[48]	
		Dependency	Bobrowski's[44]	CFS[34], RRESET[49], POE+ACC[50], DVMM[51]	
		Consistency	FOCUS[52], ABB[53], MIFESI[54], Schlimmer's[55]	Set Cover[56], WINNOW[57]	LIV[25], QBB[25], LVF[58]
	Wrapper	Predictive accuracy	BS[27], AMB&B[59], FSLC[60], FSBC[61], CARDIE[45], OBLIVIIN[57]	SBS-SLASH[48], WSFG[62], WSBG[62], BDS[27], PQSS[27], RC[63], SS[64], Querios'[48], BSE[48], K2-AS[65], RACE[64], SBS-W[62]	SA[27], RGSS[27], LVW[66], RMHC-PF[67], GA[68], RVE[69]
	Embedded	Filter + Wrapper		BBHFS[70], Xing's[71]	

Table 2.1: Classification of feature selection methods according to search strategy and evaluation criteria [24]

2.1.1 Filter Methods

Filter methods represent the fastest and simplest approach for feature selection. They operate independently of the learning algorithm and are exclusively used as a pre-processing step. These methods use an independent measure and intrinsic characteristics of features to evaluate their importance. Importantly, there is no feedback from the classification algorithm as in wrapper or embedded methods [35].

The main drawback of filter methods arises when a measure of statistical significance evaluates the importance of features, causing a discrepancy between the objective of the filter methods (e.g., significance based on p-values) and the model's requirements (predictive performance) [72]. Consequentially, this inconsistency can have a detrimental impact on a feature subset quality and decrease the predictive performance of classification algorithms in later analysis [72].

Filter methods can be further classified into two groups [35]:

- Attribute evaluation methods
- Subset evaluation methods

Attribute evaluation methods evaluate the importance of each feature separately. The final feature subset is determined by feature ranking according to chosen criteria. In contrast, subset evaluation methods evaluate all features simultaneously, applying criteria such as information gain [35].

There are various attribute evaluation methods that can be summarized based on the type of features and outcome. When evaluating individual categorical features, the following methods can be used based on the type of outcome [72]:

- When the target is categorical, then odds ratio or chi-squared test can be used to measure the relationship with the target.
- On the other hand, when the target is numerical, the possible tests are basic t-Tests and Analysis of Variance (ANOVA). The same tests can be used if the target is categorical and features numerical.

When both features and target are numerical, a simple Pearson or Spearman pairwise correlation can be calculated. However, when the relationship is nonlinear, then the Maximal Information Coefficient (MIC) may be a better choice [72].

Other well-known filter methods are FOCUS [52], Automatic Branch and Bound (ABB) [53], and relief [39].

2.1.2 Wrapper Methods

In contrast to filters, wrapper methods incorporate the classifiers directly into the feature selection process. The performance of the classification algorithm (e.g., error rate) is used to evaluate the quality of the selected subset. Examples of this approach would be backward or stepwise selection as well as Genetic Algorithms (GA) [35, 72].

Wrapper methods follow an iterative search procedure that provides the model with the subset of features and uses its performance as a guiding factor for selecting the subsequent subset. The ultimate goal is to iteratively converge to a smaller, optimized feature subset that increases the classifier's performance. In general, there is a greedy approach where a subset of features is chosen, if it currently gives the largest performance improvement. However, the predictive performance can be trapped at local minima. On the other hand, a non-greedy approach re-evaluates previous feature combinations and iterates in the direction that at the moment maybe is not favorable, but in the subsequent iterations can be highly beneficial [72].

Backward selection, also known as Recursive Feature Elimination (RFE), is a greedy wrapper method [72]. This method begins by ranking the features based on a chosen importance measure, and the ranking serves as the basis for the RFE search direction. An initial model is created using all features and its performance is obtained. Then the least important feature is removed and the model is rebuilt on the new feature subset. This is repeated until a low number of features are in the model, or until a significant performance decrease. It is important to note, that this method is greedy because the feature importance is assessed only at the beginning. Therefore, it can have a negative impact, since some features may be more important in the presence of others [72].

Non-greedy wrapper methods, such as GA and Simulated Annealing (SA), offer an alternative approach to feature selection. The SA method does not make the feature selection process greedy due to the integration of some randomness. There is a higher probability that these methods find the globally best features subset. However, searching for an optimal subset can take a lot of time and this is its main limitation. Additionally, wrappers use training data to select the best features and therefore there is a risk of overfitting [72].

2.1.3 Embedded Methods

The potential limitations of the previous strategies are solved by embedded methods. These methods create feature subsets as an integral part of the classifier, bridging the gap between filter methods that lack classifier dependence and computationally expensive wrapper methods [35, 72].

The well-known embedded methods are [72]:

- **Tree- and rule-based methods:** These methods find the best features by splitting the points to maximize the homogeneity of target samples in resulting divisions.

If a feature is not used in any split, it can be seen as less important and can be removed from the dataset.

- **Regularization models:** The regularization methods introduce feature coefficients as a penalty term to the loss function. The lasso regression or L1 regularization may force some of the coefficients to zero, making them completely unimportant. Consequentially, these features can be also removed from the model.

Another notable embedded method is Multivariate Adaptive Regression Spline (MARS) model, which uses piecewise linear functions in aggregation to make predictions. When a feature is absent from all MARS features, it is not considered important and will be eliminated. This behavior is similar to trees [72].

Embedded methods offer distinct advantages over wrapper and filter methods. They are relatively fast and eliminate the need for an external feature selection method since it is integrated into the model fitting process. Additionally, the feature selection is directly associated with the objective function of a fitted model (e.g., impurity in tree-based methods). However, its model dependency represents a limitation [35, 72].

2.1.4 Feature Selection Challenges

Feature selection comes with several challenges that need to be addressed [73]:

- **Scalability:** The increasing size of datasets is a great difficulty for classification and feature selection algorithms. Modern datasets can contain thousands of features and instances, making it demanding for feature selection algorithms to precisely calculate the feature importance scores. These algorithms often need to consider all dataset instances to make accurate estimations. Therefore, it is essential to develop methods that can handle large datasets, ensuring their practical use in real-world applications [73].
- **Linked data:** Many feature selection and classification algorithms assume data independence and an equal data distribution. However, this is not a real-world scenario where data points may be interconnected. Some examples of connected data include protein networks in the bioinformatics field, text analysis, and social networks. It is a great challenge to use the relations between linked data for feature selection and some approaches were proposed in [74, 75, 76].
- **Stability:** Expanding upon the selection of a feature subset that yields superior classification performance, another important aspect of a feature selection method is its stability. Feature selection stability represents the sensitivity or robustness of a method to small perturbations in the training data. Ideally, a reliable feature selection method should consistently select the same or very similar subsets of important features across different subsets of data. However, the stability can be greatly affected by the dataset properties such as the number of features, number

of instances, and data distribution [73].
Stability selection was proposed by Meinshauen et al. [16] to improve feature selection stability. This approach can increase the quality and stability of any base feature selection method, where they used Lasso as an example. It involves dividing the training data into many smaller subsamples where the final importance of a feature is calculated based on the proportion of subsamples where the feature is assigned a high importance score. In this way, only the consistently important features across all subsamples will be included in the final feature subset. Nevertheless, stability selection may face difficulties when dealing with correlated features [77, 78].

2.1.5 Feature Selection versus Feature Extraction

Although beyond the scope of this work, it is important to highlight another approach for dimensionality reduction known as *feature extraction*. While the ultimate goal of both feature selection and feature extraction is partly similar (dimensionality reduction), these two processes differ significantly and should be used with caution.

Feature selection removes the irrelevant or redundant features that are unnecessary for later analysis and classification. On the other hand, the feature extraction process combines existing features into entirely new features and therefore transfers the data into new space with the reduced linear dependence between the features. This means that two or more features can be replaced with a single feature derived from their combination. Methods like Principal Component Analysis (PCA), Independent Component Analysis (ICA), or Singular Value Decomposition (SVD) can produce feature weights that could potentially be used to select the most important features. However, this is not advisable since all initial features are used to create the new space. Additionally, these methods capture only the linear relationships between features [79].

It is noteworthy that both approaches can be used in a supplementary way, where feature selection is used to remove the irrelevant features before the employment of feature extraction [79]. For further discussion on the feature extraction topic, please refer to [80, 81].

2.2 Machine Learning Fundamentals

Machine learning is a subfield of computer science that focuses on developing statistical models that use data to create an inference function and solve a practical problem. The type of learning depends on the input data and there are supervised, unsupervised, semi-supervised, and reinforcement learning [82].

2.2.1 Types of Learning

2.2.1.1 Supervised Learning

Supervised learning is a process in machine learning where a model is trained using a dataset consisting of labeled examples. Each example is represented as a high-dimensional feature vector, where each dimension corresponds to a single feature. The label can be an element from a finite set of classes, a real number, or have a more complex structure, such as a vector, or a matrix [82].

Depending on the nature of the label, there are two problems in supervised learning, *classification* and *regression*. The model's objective in classification is to separate data samples into different predefined categories, predicting strictly discrete values as labels. A special form of classification where the label can take only two possible values is called *binary classification*. Binary classification is often in scenarios where there are two distinct values representing a normal and abnormal state, such as {*spam*, *non-spam*} in email filtering. Otherwise, when there are more than two possible labels, it is defined as *multiclass classification*. An example of multiclass classification is a classification of different network attacks. On the other hand, a regression model learns a function that fits the data and predicts continuous values. For instance, regression might be used to predict a continuous variable like the price of a house [82].

There are many classification and regression algorithms including linear regression, logistic regression, decision trees, Naive Bayes, K-Nearest Neighbors, Suppor Vector Machine (SVM), or techniques based on neural networks such as diverse deep learning approaches [82].

2.2.1.2 Unsupervised Learning

In contrast to supervised learning, in unsupervised learning data samples are not labeled and there are only feature vectors. Labeling data is a resource-intensive and time-consuming process, which requires supervisors, that are often represented by humans. However, even in the absence of labels, data can still yield valuable insights relevant to problem-solving and pattern recognition. The most common form of unsupervised learning is clustering, where data points are grouped into *clusters*. Data points inside a cluster are somehow more similar to one another than to datapoints in other clusters. Additionally, unsupervised learning can be used for dimensionality reduction, where the output is a feature vector with reduced dimensionality. K-Means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) are famous clustering algorithms [82].

Anomaly or **outlier detection** is another important unsupervised learning technique that detects abnormal data in a dataset [83]. Hawkins defines an outlier as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [84]. This definition aligns with the traditional approach to outlier detection, where the determination of whether a data point is an

outlier is based on its relative distance to other data points that are close in both time and space [85]. This approach is entirely unsupervised, as the algorithm uses intrinsic data properties to calculate distances and densities, and thereby to determine what is a normal point and what is an outlier [86]. However, this strategy may be inappropriate in certain scenarios, such as the detection of collective anomalies [85]. In response, Iglesias et al. define an anomaly as "something that diverges from what has been previously defined as normal data points" [85]. It is noteworthy that such a definition renders this technique rather semi-supervised, and it is also often referred to as one-class classification. Furthermore, even though the semantic meaning of anomalies and outliers might not be the same [87], the terms anomaly detection, outlier detection, and one-class classification are often used interchangeably in the literature. This interchangeability comes from the fact that the methods used for their detection are essentially the same [85, 83]. Hence, the exact definition and approach to anomaly detection (unsupervised, semi-supervised) depends largely on the application [85].

The unsupervised anomaly detection algorithms can be categorized into the following groups [88]:

- **Nearest-neighbor-based:** These techniques assume that normal data points are located in dense areas, whereas anomalies can be found far from their closest neighbors [88]. Distance or similarity measures, such as Euclidian distance for continuous attributes or matching coefficients for categorical attributes, are used to calculate the outlierness of data points. Some prominent algorithms are k-Nearest Neighbors (k-NN), k-th Nearest Neighbor (k^{th}-NN), Local Outlier Factor (LOF), Connectivity-Based Outlier Factor (COF), and others [86, 88].

- **Clustering-based:** Many clustering-based anomaly detection methods are based on distinct assumptions. For instance, algorithms like DBSCAN, Robust Clustering using Links (ROCK), Shared Nearest Neighbor (SNN), and Find Outliers (FindOut) assume that normal data points are located in clusters, while anomalies are not associated with any specific cluster [88]. However, these algorithms are optimized more for cluster identification than anomaly detection. Other methods (e.g., Self Organizing Maps (SOM), K-means Clustering, and Expectation Maximization (EM)) categorize data points as anomalies if they are far from their closest cluster centroid. These clustering-based methods may also leverage labeled training data and therefore learn in a semi-supervised manner [88].

  A limitation of the first two groups of clustering-based methods arises when anomaly data points create clusters by themselves. In such scenarios, they will be most probably misclassified as normal points. In response, some methods assume that normal data points create large and dense clusters, while the anomaly points form smaller, sparser clusters [88]. Typically, these techniques establish thresholds for the size and density of clusters, below which all clusters are classified as anomalous. An example of an algorithm in this category is FindCBLOF (Find Cluster-Based Local Outlier Factor) [88].

- **Statistical algorithms:** These algorithms are based on the premise that normal data points are situated in "high-probability areas of a stochastic model", whereas anomalies are placed in the low-probability areas of such a model [88]. They include a statistical model that fits on the given data that represents the normal behavior. New, unseen data instances are then tested if they belong to the model or not. Well-known methods are Gaussian Model-Based such as the simple box plot rule technique, Grubb's test, and the student's t-test. Additionally, the regression-model-based techniques are used for time-series data (e.g., Autoregressive Integrated Moving Average (ARIMA), Autoregressive Moving Average (ARMA)). Another set of techniques includes histogram-based methods, also known as frequency- or count-based methods. These non-parametric techniques use histograms to establish the borders of normal behavior [88].
- **Subspace techniques:** As previously discussed, PCA is a feature extraction technique that projects data from one space to another with reduced dimensionality. It is used for anomaly detection in the way that anomalous data points deviate from the normal subspaces. However, this approach may not be optimal since the anomalies negatively impact the covariance matrix, leading to imprecise density estimation. An improved version, Robust Principal Component Analysis (rPCA), is proposed in [89].

Nonetheless, anomaly detection holds great importance in numerous fields, particularly in network security, where rare events such as unusual traffic patterns are often a sign of a critical situation like network attacks. Indeed, the authors in [90] have demonstrated that network attacks exhibit higher global outlierness compared to normal traffic. Given that the anomalies are typically in the minority, the link to imbalanced binary classification is evident.

For a further discussion of the individual algorithms and their comparison, please refer to [86, 88].

2.2.1.3 Semi-Supervised Learning

A combination of supervised and unsupervised learning is known as Semi-Supervised Learning (SSL). In this type of learning, there is a dataset that contains labeled, but also unlabeled examples. It is particularly often in domains where labeling data is extremely time- and money-consuming, such as speech analysis or article classification. The primary objective of SSL is to improve the prediction power of the model by incorporating unlabeled data alongside labeled data. By doing so, the model can gain a deeper understanding of the underlying data distribution [82].

For instance, there is a task where numerous articles need to be classified into different topic categories like robotics, cybersecurity, etc. A possible approach is to use the words that occur in these articles to train a supervised classifier [91]. From the training data, the classifier may learn that the articles containing the phrase "hash function" often belong to

the cybersecurity category. However, a great challenge arises when a cybersecurity article does not include this word but another one, e.g., "cryptography" that was not available in the training data. That article is most likely to be misclassified by the trained supervised classifier. In this scenario, unlabeled data may be useful. There might be articles in the unlabeled data that link "cryptography" to "hash function". For example, there are two unlabeled articles where the first one contains the "hash function" and "SHA-1" phrases, while the second includes "SHA-1" and "cryptography" terms. Leveraging this unlabeled data, the classifier may recognize the article containing the unseen word "cryptography" as belonging to the cybersecurity category [91].

There are three important assumptions important for SSL [91]:

- **Smoothness assumption:** "If two data instances are close in the feature space, they should have the same labels" [91].
- **Low-density assumption:** "The decision boundary should not pass through high-density regions in the feature space" [91].
- **Manifold assumption:** "Data instances belonging to the same low-dimensional manifold should have the same labels" [91]. For instance, there are data points in a 3-dimensional feature space that belong to the same 2-dimension manifold such as a sphere surface and therefore they should have the same labels [91].

The smoothness assumption can also be used in a transitive manner to label unlabeled data. For instance, there is a labeled instance x_1, and two unlabeled instances x_2 and x_3. If x_1 is close to x_2 while x_2 is close to x_3, yet x_1 and x_3 are not close, then according to the smoothness assumption, x_3 can be expected to have the same label as x_1. The second assumption is also very intuitive and supports the smoothness assumption. The decision boundary "should pass through the low-density areas" [91] because in this way the close data points will not receive different labels, which would happen if the boundary traversed high-density areas. Since the smoothness assumption only considers similar and close data points – typically not found in low-density areas – it is not violated in this scenario [91].

Additionally, it is crucial to emphasize that SSL is only valuable if the unlabeled data delivers information that is not already provided by labeled samples [91]. However, this information must be also useful for the labeling process.

The semi-supervised methods can be grouped as follows [91]:

1. **Inductive methods** generate similarly to supervised techniques a model capable of predicting the label of any data instance. Given sets of labeled and unlabeled instances, $X_L, X_U \subseteq X$, with Y_L representing the labels of labeled data X_L, these methods produce the models $f : X \to Y$ [91].

   Inductive methods include [91]:

- **Wrapper methods** include models that are trained using the labeled points and then these models are utilized to generate labels for unlabeled instances. Subsequently, the model is re-trained using pseudo-labeled data along with the originally labeled instances. Essentially, any supervised algorithm can be adapted for this purpose. Specific techniques designed for this approach are ASSEMBLE, SemiBoost, and others [91].
- **Unsupervised preprocessing** involves both unsupervised and supervised stages. In the unsupervised stage, methods either extract useful features from unlabeled data (feature extraction), cluster data (cluster-then-label) or initialize parameters of the learning procedure (pre-training). They can also be implemented using supervised classifiers. Semi-supervised autoencoders, deep belief networks, and stacked autoencoders belong to this group of methods [91].
- **Intrinsically semi-supervised methods** directly incorporate information from both labeled and unlabeled instances into the objective function. For instance, semi-supervised maximum-margin methods (SVM3, SVM4) are extensions of SVM where the margin is maximized by also considering unlabeled data along with labeled data [91].

2. **Transductive methods** do not offer a model, but instead directly provide predictions for the available unlabeled instances in X_U, Y_U. Therefore, they are strictly limited to instances from the training space. Since there are no models, data is forwarded from one point to another via connections, making graph-based methods a natural fit for this task [4].

Additionally, it is worth nothing that anomaly detection can be conducted in a semi-supervised manner, as previously discussed. In this approach, an algorithm learns only the normal state from the training data (no training instances that represent anomalies). Subsequently, anomalies are recognized as deviations from the learned normal profile [86]. This method is commonly known as one-class classification [92], and famous algorithms include one-class SVM (SVM3, SVM4) and autoencoders.

For further explanations of SSL and its methods, the interested reader is referred to [91].

2.2.1.4 Reinforcement Learning

In Reinforcement Learning (RL), there is a learner called *agent* that exists in an *environment*. The agent takes actions based on a *policy* and an input feature vector known as a *state*. Similar to the learning function of a model in supervised learning, the policy serves as a decision-making function. It evolves and is refined based on the *reward* received after each performed action [82, 93].

The agent gathers valuable experiences to optimize its behavior within the environment. At each state, the agent must select a proper action, and this decision-making problem can

be modeled by Markov Decision Process (MDP). MDP defines several critical parameters for modeling, including a finite set of states S, a finite set of actions A, a reward function R, and a state transition function Φ. The goal is defined by the reward function (R), which maps each possible state-pair pair to a score that indicates the desirability of that state. On the other hand, the transition function Φ estimates the probability of reaching the state s_2 when an action is performed in the state s_1. The agent can only take specific actions at each state, and therefore the set of possible actions for each state is different. The other important elements are the policy π and value function (V_π, Q_π). The agent behaves according to the policy that maps the states to actions. On the other hand, the value function estimates the quality of a state by defining the maximum possible reward that the agent could claim transitioning into that state. Similarly, the policy incorporates the Q-function, which assesses the quality of executing an action in a state $s \in S$ [94, 95].

It is also important to distinguish between *episodic* and *non-episodic* RL. An *episode* is defined as a subsequence of interactions within the environment. In episodic RL, the agent is reset to the initial state when an episode concludes. Episode terminations are denoted by the absorbing states, from which no further states and rewards are possible, regardless of the action taken. On the other hand, in non-episodic RL, there are no state resets [94].

The MDPs could be addressed through the following procedures [95, 96]:

1. **Dynamic Programming (DP)** approach is model-based, requiring a model that is used by the agent to estimate the environment's responses to its actions [96]. DP can calculate the optimal policy by leveraging the environment's model that is given in the form of MDP. Firstly, DP estimates the transition and reward functions, and using these estimations, it evaluates the value function. This process of estimating values based on the estimations of other values is known as *bootstrapping* [95]. DP is computationally expensive and impractical for large RL problems since the complete environment dynamics are needed [96].

2. **Monte Carlo (MC)** represents a model-free approach where the policy is learned through the agent's trial-and-error exploration, rather than relying on complete modeling of transition and reward functions in the environment. In this approach, experience samples (one experience sample includes: state, action, reward, and the following state) are required to train the agent. The reward and subsequent state are obtained from the environment that is usually represented by a model. However, the knowledge of the model is not utilized to optimize the policy and value functions of states. At the end of each episode, the returned samples from the environment are averaged, and the policy and value functions are updated. It is noteworthy that for a single state in an episode, either all visits or only the first one may be considered when averaging the returned samples. After a sufficient number of episodes, the policy should be optimized based on the agent's experience and received rewards. MC methods have a great advantage over DP methods as

they are model-free, and also the computational requirements are lower since the value functions in an episode are calculated only for the visited states [96].

3. **Temporal Differences (TD)** is one of the most widely used RL concepts. It is similar to DP due to the use of bootstrapping and previous value estimates. However, reward and transition functions are unknown and therefore it is like MC a model-free approach. Well-known algorithms are State-Action-Reward-State-Action (SARSA), Expected SARSA, and Q-learning.

The main objective of RL is to learn an optimal policy with an optimal value function. However, the recent development of deep RL introduces Goal-Conditional Reinforcement Learning (GCRL), where the policy is not only dependent on states but also on the goal being optimized. This goal may vary under different conditions [94]. A practical application of GCRL can be found in maze navigation tasks, where goals are represented by different maze locations [94].

The field of RL has improved greatly with the rise of deep neural networks. Neural networks help the agent to learn from images, leading to many practical applications such as autonomous driving and gaming. The pioneering algorithm that successfully combined RL with deep neural networks was Deep Q-Network (DQN). It was initially trained to play the Atari games demonstrating remarkable performance. Additionally, the fact that the deep reinforcement model AlphaGo defeated the world champion in Go, shows the significant progress and immense potential of this field [95, 94, 96].

2.2.2 Ensemble Learning

Ensemble learning is a type of supervised classification. It is a powerful learning paradigm in machine learning rooted in the divide-and-conquer principle. The fundamental concept is to combine predictions from multiple individual models to create a more robust and predictive model.
Figure 2.1 represents the fundamental structure of an ensemble learning model. The structure comprises base learners, which may be represented by identical models, such as decision trees, resulting in a *homogeneous* ensemble. In contrast, when the learners are structurally different, the ensemble is *heterogeneous*. Nevertheless, independent of their internal structure, all learners contribute to the same classification or regression task.
Base learners are also known as *weak learners* because they are slightly better than random guessing. However, their true power emerges when they are combined. One of the most important characteristics of base learners is diversity. Employed learners must differ in their approaches, otherwise, combining them will not result in substantial performance improvement. The three main classes of ensemble learning methods are bagging, boosting, and stacking [97].

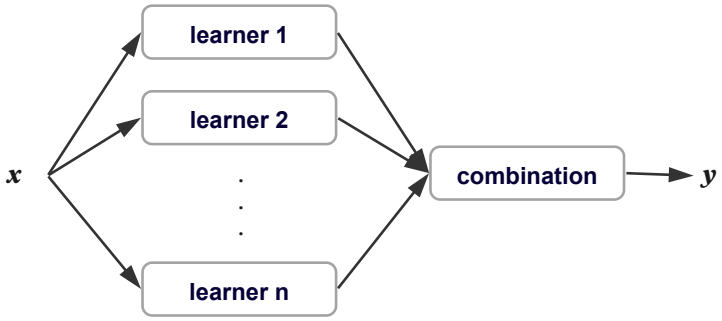

Figure 2.1: Basic ensemble structure (based on [97]). Each learner makes a prediction for a data instance x, and the final output y is determined by combining their predictions.

2.2.2.1 Bagging

Bootstrap Aggregating or Bagging, is an ensemble paradigm where the base learners are generated in parallel. Usually, the used weak learners are shallow decision trees. These trees are constructed to stop their splitting process after a few iterations, achieving a predictive performance slightly better than random guessing. The main idea is that by aggregation of many such trees, it is feasible to get a potent final learner that surpasses the predictive capabilities of these individual trees [82].

To ensure diversity and less correlation among the learners, one approach is to use different subsets of the original datasets for training each decision tree. However, many times real-world datasets might not be large enough to support this strategy. This emphasizes the significance of bagging as it offers a practical solution to this limitation [97].

The bagging process is illustrated in Figure 2.2. Firstly, multiple subsets are created from the original training dataset. Each of the subsets is used to train an individual model within the ensemble. The selection of samples is performed randomly and with replacement, meaning that a sample chosen from the training dataset remains available for selection in subsequent rounds. Therefore, a subset may contain duplicate samples. The final prediction for a data sample can be made by averaging predictions of all predictors in case of regression or taking a majority vote in the case of classification tasks [82].

An algorithm that builds upon the bagging, with a slight modification, is RF. In a dataset, if two features emerge as highly important, each decision tree within the ensemble may exclusively rely on these features for splitting. This could lead to high correlation among the trees and such correlated trees with the same incorrect predictions could prevail in the majority voting. To mitigate this negative effect, each decision tree in the RF at each splitting "sees" only a random subset of features.
This modification together with sampling with replacement, is the reason for the low variance of the final model and reduced overfitting effect[82].

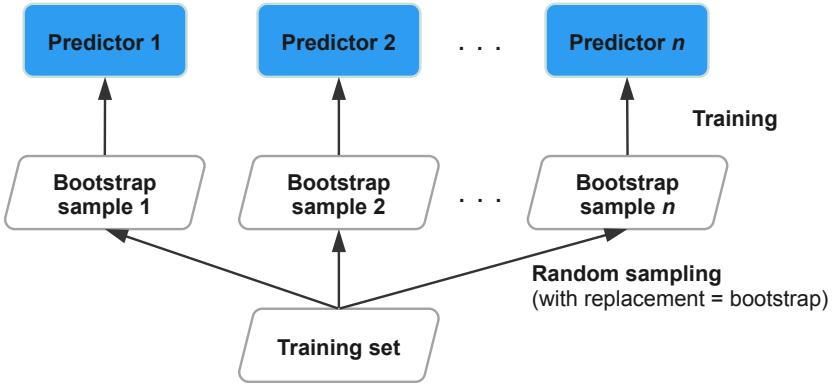

Figure 2.2: Training dataset sampling and training of base models in bagging (based on [98])

2.2.2.2 Boosting

Boosting is a sequential ensemble learning technique that employs a chain of weak learners, where each learner is supposed to correct the prediction errors made by its predecessor. The boosting model assigns greater importance to false predictions and the learning process is more devoted to those training samples. The well-known boosting algorithms are AdaBoost and the optimized variant of gradient boosting - XGB [82, 98]. In AdaBoost, a new model in sequence corrects the errors of its predecessor by assigning larger weights to misclassified training samples. The procedure begins with the initial model being fitted to the training data and generating predictions. Subsequently, the sample weights are adjusted based on the disparity between the predictions and the ground truth data labels. The subsequent classifier in the sequence is trained using the updated weights and, once again, it generates predictions and updates the weights. This iterative procedure continues until a predetermined number of models are constructed. The final prediction is generated by combining the weighted predictions from all trained predictors, wherein the more accurate predictors have larger weights. Weak models in AdaBoost can be selected freely, although decision trees are a commonly preferred choice [98].

On the other hand, there is a variant of boosting known as *gradient boosting*. This process uses the gradient descent technique to minimize the loss function concerning *residuals*. Residuals are calculated based on the current model's predictions and the ground truth labels. They will be passed as new data labels to the next model in sequence, which is supposed to correct the predictions of its predecessor. This procedure shifts the model in the right direction and addresses misclassified samples. It is necessary whenever a new learner is introduced to the boosting model. To mitigate the correlation between sequential learners, it is possible to perform subsampling similar to RF. This approach is known as *stochastic gradient boosting* and its well-known optimized version is XGB. Boosting compared to bagging tends to reduce the bias instead of variance since the

learning is devoted to fitting the underfitted samples from the previous learners [82].

2.2.2.3 Stacking

Stacking or stacked generalization [99] combines predictions of diverse models, going beyond trivial aggregation functions like hard voting or averaging. Instead, it employs a more complex *meta-learner* or *blender*, which learns to combine the decisions of first-level learners in the most optimal way. While it is not atypical to have more layers in stacked models, the most common approach includes two layers: the first-level learners and a meta-learner. Any model can be used as a meta-learner, such as linear regression, logistic regression, or RF. The stacking inference process for a regression task is represented in Figure 2.3. First-level learners generate predictions, which are used by blender to bring the final prediction (3.0) [100, 98].

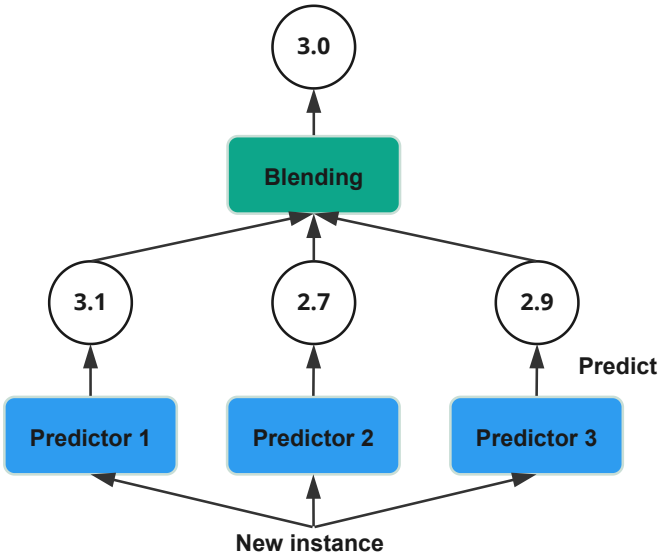

Figure 2.3: Combining predictions with a blender (based on [98])

During the training of a stacked ensemble, it is crucial to avoid any data leakage and potential biases. To achieve this, the training dataset should be divided into two distinct hold-out sets. The first hold-out set is used to train exclusively the first-level learners. Subsequently, the second hold-out set is used to train a meta-learner. The first-level learners generate predictions for the samples from a second hold-out set and they will be used as input training data vectors to the meta-learner. This dataset separation is important because utilizing the same dataset for both the training first-level learners and generating input data for meta-learners can induce significant bias in the training process, making overfitting inevitable [98].

2.3 Imbalanced Learning

An imbalanced dataset is a dataset with an unequal data distribution across different classes. This means that the number of data samples in a class is much lower than in other classes. It is an often issue in both binary and multiclass classification. In binary classification, the minor class is commonly termed the positive class, while the majority is the negative class. Class imbalance is frequent in various domains, such as fault detection, fraud detection, or medical diagnosis [101].

2.3.1 Problem of an Imbalanced Dataset

An illustration of an imbalanced dataset is shown in Figure 2.4, where it is evident that the number of samples belonging to the class "×" is significantly lower than those in class "•". Typically, the minority class contains the samples of interest. For instance, this dataset can be imagined as a data representation containing two tumor types, benign or malignant, where the malignant tumor is less common and therefore there are fewer data instances. However, the prediction of malignancy is of utmost importance in the medical field [102].

Observing the figure, it is very challenging to establish a decision boundary that would separate minor samples perfectly, and even very complex and accurate models have the similar problem. Usually, the established decision boundary results in over- or underfitting. The reason is that most classifiers tend to ignore minority samples, achieving high accuracy primarily for the majority class. During the learning process, they perform an optimization of the cost function focusing only on majority samples, consequently deeming minority samples as less important and misclassifying them. Therefore, it is necessary to address the class imbalance through a learning process where the classifier will be able to distinguish between majority and minority classes [102]. It is also important to emphasize that the ratio between minority and majority classes does not always adequately represent the real issue of class imbalance. In fact, the ratio itself does not significantly impact the learning process of the classifier. The greatest challenge arises when the number of samples in the minority class is insufficient to capture data patterns. This can be illustrated with an example: in a dataset there are 100 samples in the minority class and 10,000 in the majority class, resulting in a ratio of 1 to 100. On the other hand, in the second dataset, there are only 10 samples in the positive and 1,000 samples in the negative class. Despite the fact, that both datasets have the same ratio, it is highly unlikely that the algorithm will effectively learn to predict the positive class in the second dataset. Therefore, the *sample size* may be a more critical measure in an imbalanced dataset [102].

Additionally, the distribution of data instances in the feature space is extremely important. In some datasets, a linear decision boundary may exist that separates the classes perfectly. However, when samples from different classes are highly mixed and overlapping, finding an optimal decision boundary without under- or overfitting becomes a great difficulty.

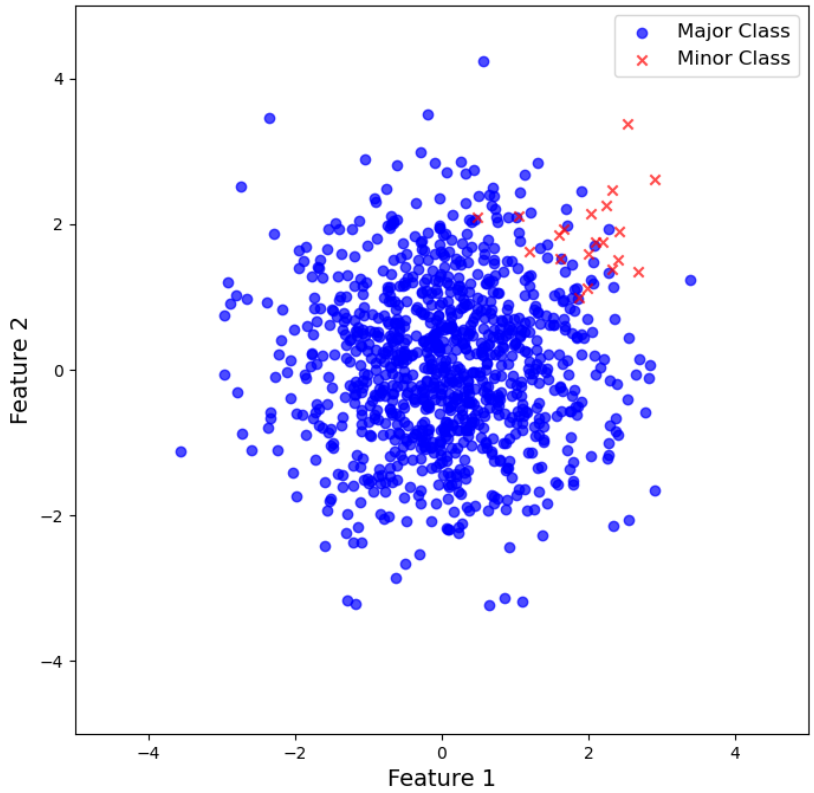

Figure 2.4: Example of an imbalanced dataset in binary classification (based on [102])

Another noteworthy challenge is dealing with small disjuncts within the data. Both phenomena can be seen in Figure 2.5 [102].

The final issue that has to be discussed and is particularly significant for Intrusion Detection System (IDS) is the **base-rate fallacy**. There are numerous demands for an IDS including efficiency, transparency, ease of use, interoperability, and more [103]. However, the primary demand for an IDS is its effectiveness, which involves a high detection rate of the intrusions while keeping the False Alarm Rate (FAR) at an acceptable level. Regrettably, this balance is not easily achievable and FAR might be the greatest limitation that an IDS may face. Difficulties are mainly caused by a phenomenon known as the base-rate fallacy, which requires such a low (often practically unreachable) FAR in order to achieve a high *Bayesian detection rate*, $P(Intrusion|Alarm)$. The probability that an alarm is triggered by an actual attack known as the Bayesian detection rate is

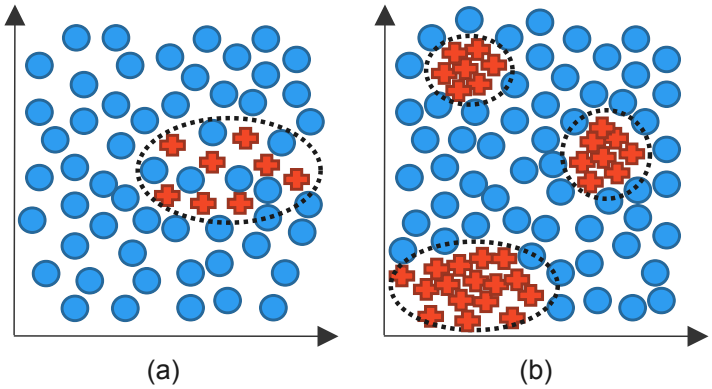

Figure 2.5: Possible challenges in an imbalanced dataset (a) Class overlapping, (b) Disjuncts (based on [102])

defined as [103]:

$$P(I \mid A) = \frac{P(I) \cdot P(A \mid I)}{P(I) \cdot P(A \mid I) + P(\neg I) \cdot P(A \mid \neg I)} \tag{2.1}$$

where I and $\neg I$ represent the presence or absence of an intrusion/attack, A and $\neg A$ denote the presence or absence of an intrusion alarm and $P(A \mid I)$ is *detection* or *true positive rate*. The FAR is the probability $P(A \mid \neg I)$, while $P(I)$ and $P(\neg I)$ represent the prior probabilities of an attack or normal traffic [103].

To illustrate this phenomenon, consider the following example. There is a dataset containing network traffic and a model achieves an accuracy rate of 99% in attack detection and the accuracy rate for classifying normal traffic as normal is also 99%. While these accuracy rates appear to be very high for both classes, they might be misleading when considering other data characteristics, such as the number of instances belonging to attacks and normal traffic. Imagine there are 100 attack instances and 100,000 normal instances. This means that the model accurately predicts 99 out of 100 attacks, but 990,000 out of 1,000,000 normal instances are detected as normal. This results in 10,000 of them being misclassified as attacks. Consequently, if the model classifies an instance as an attack, there is only a $99/(99 + 10,000) \approx 0.0098 = 0.98\%$ chance that it is truly an attack. This indicates that the model does not perform that well despite having a high detection rate for attacks. The main cause for this discrepancy is that the amount of normal traffic is far larger than the number of attacks, and therefore preliminarily low FAR (1%) has an enormous impact on the effectiveness of the model, even when its detection rate is very high.

The base-rate fallacy is an issue that is often missed when evaluating models on strongly imbalanced datasets. The author in [103] suggests that the FAR should be measured relative to the number of expected intrusions rather than concerning the maximal possible number of false alarms. Otherwise, the probability that a predicted attack is genuinely an

attack will be very low. The impact of higher FAR could be mitigated through increased percentage of attacks in the entire traffic and improved detection rate.

When Axelsson [103] first indicated the problem of the base-rate fallacy in the context of IDSs, data mining methods were able to achieve a minimal FAR of 10^{-3} [104, 105]. However, almost three decades ago, the existing research proposed methods that only marginally improved the FAR by a factor of 10^{1} [106, 105]. Considering the amount of network traffic at that time and now, this FAR improvement is negligible. The author in [105] illustrated an intrusion detection example at the packet level considering the volume of data of 2.5TB inbound per day (amount from 2011 based on [107]). With an average packet size of 870.607 bytes per packet [108], this results in approximately 119,648,296.15 packets per hour. Even if an optimistic FAR of 0.37% is taken [109], it would lead to 442,698.695 false alarms per hour. This problem becomes even greater with the fact that false positives must be examined by an analyst [105]. According to [110], an analyst can examine 12 alarms per hour, which results in a total of 36,892 analysts approximately. Anticipating that network traffic will continue to be composed largely of normal data in the future, IDS must persist in addressing this issue. The only feasible trade-off is to set an acceptable FAR as low as possible, nearly as low as prior probabilities of attacks $P(I)$ [103, 111].

Therefore, it is of great importance to address the problem of imbalanced data and the used techniques can be categorized into four groups [102]:

- **Algorithm level** is an internal approach that corrects data imbalance by altering the learning process of a classifier.
- **Data level** is an external approach that is suitable when the modifying of the internal structure and learning process of a classifier becomes too intricate. This approach involves various data resampling techniques, which either generate additional minority samples or remove the majority samples from a dataset. In the end, a completely balanced dataset or a predefined class ratio is achieved.
- **Cost-sensitive learning** combines algorithm and data level approaches to address imbalanced datasets. The larger weights are assigned to instances of minor classes, emphasizing their importance. The learning process is also adjusted to be more devoted to minor samples.
- **Ensemble based methods** address imbalanced classes by combining an ensemble algorithm with either a data level or cost-sensitive learning approach.

2.3.2 Data Level Balancing Techniques

A common approach to balance data includes data sampling techniques. It is an external approach that eliminates the need to alter the learning algorithms of classifiers. Resampling techniques can be classified into the following groups [102]:

- **Undersampling methods:** These methods remove data instances and create a subset of the original dataset.
- **Oversampling methods:** On the other hand, oversampling methods create a superset of the original dataset by replicating the existing or creating new data instances.
- **Hybrid methods:** Represent a combination of both undersampling and oversampling techniques. They perform oversampling of the minor class and undersampling of the major to achieve a defined class ratio.

2.3.2.1 Undersampling Techniques

Methods that eliminate samples from the major class in order to balance the class ratio are known as undersampling methods. The simplest, yet sometimes the most effective undersampling method is RUS. RUS is a non-heuristic method that can be used to achieve a specific balance ratio by random elimination of instances from the major class. However, the main drawback of this method is the potential removal of useful data from a dataset. To mitigate this issue, there is a need for a more sophisticated approach, that removes the major instances which potentially do not bring any useful information to the classifier [102].

Many heuristic undersampling methods are based on specific rules for instance removal. Examples include Tomek Links, Condensed Nearest Neighbor Rule (CNN), One-Sided Selection (OSS), and Neighborhood Cleaning Rule (NCR). Furthermore, there are more advanced data sampling techniques, which incorporate ensembles and clustering in their mechanism such as Instance Reduction by Undersampling Synthesis (IRUS), Cluster-based Oversampling and Undersampling (Cluster OSS), and Density-based-Safe-level Undersampling (DSUS) [102].

2.3.2.2 Oversampling Techniques

Random Oversampling (ROS) is another non-heuristic method used to achieve a defined balance ratio by randomly replicating samples from the minor class. ROS is as RUS straightforward to implement, but it has also a notable drawback including an increased risk of overfitting as the dataset contains duplicated instances [102].

Another heuristic technique of oversampling is SMOTE. In contrast to ROS which merely replicates original samples, SMOTE generates synthetic examples through interpolation between the existing samples in minor class. The underlying procedure of SMOTE is represented in Figure 2.6. Firstly, a minor sample x_i is randomly chosen as a starting point for generating new synthetic samples. Based on a distance metric, several neighbors from the same class are found - x_{i1}, x_{i2}, x_{i3}, and x_{i4}. Then, using random interpolation new synthetic examples are generated between the starting sample and the neighbors: r_1, r_2, r_3, and r_4 [102].

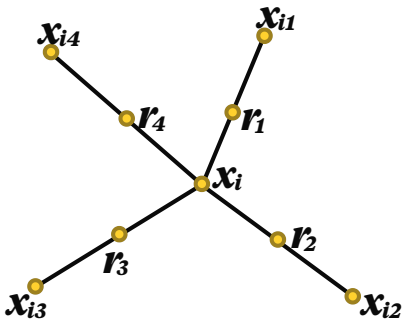

Figure 2.6: Generating synthetic data points with SMOTE (based on [102])

This procedure served as a basic concept to many other oversampling techniques, but it does come with certain limitations. One notable drawback is the potential introduction of noisy data instances and SMOTE does not consider the major samples or overlapping with the other class. In response, numerous researchers tried to improve the SMOTE algorithm and some of the famous SMOTE extensions are Borderline-SMOTE and Adaptive Synthetic Sampling Approach (ADASYN). In comparison to SMOTE, Borderline-SMOTE generates synthetic examples only based on the limit or border samples, where most of their neighbors are samples from the majority class. Consequently, these samples are more likely to be misclassified. On the other hand, ADASYN is based on adaptively generating minority examples, where more synthetic data is generated in the areas with a lower density of minor samples [112]. This approach reduces the likelihood of misclassification in those sparser regions [102].

2.3.3 Cost-Sensitive Learning

Cost-sensitive learning is another class imbalance strategy that introduces a cost to bias the learning process to minority samples. Instances that are misclassified receive higher costs or penalty terms, contributing to an increase in the overall total loss function that should be minimized by the classifier. Thus, the classifier will prioritize the minority class. Typically, classifiers use a 0-1 loss function where a wrongly classified sample is assigned a value of 1 and a correctly classified one is assigned a value of 0. When an algorithm predicts only the major samples correctly, it is sufficient for minimizing greatly the loss function. This assigning of the same costs to all classes is not objective in imbalanced data. Therefore, the primary objective of cost-sensitive learning is to assign different misclassification costs to different classes. This means that misclassifying a minor sample could result in, for instance, a tenfold increase in the loss function compared to misclassifying a major sample [102].

Cost-sensitive learning is incorporated in many classifiers such as decision trees. In decision trees, the two most important aspects of cost-sensitive learning are splitting criterion modification and instance weighting. The first approach is connected with the

feature cost known also as test cost, while the second is the misclassification cost of an instance, where higher weights are assigned to instances from classes that are more likely to be wrongly classified. Prominent algorithms that offer cost-sensitive learning are XGB and RF [102].

2.4 Machine Learning Interpretability

Machine learning algorithms are incredibly powerful for solving complex tasks and generating accurate predictions. However, they come with some limitations such as a lack of transparency [113]. This means that the internal structure and workings of these models are difficult to understand, leading to so-called black-box models. For instance, deep neural networks or random forests with thousands of decision trees are examples of non-transparent models [114, 115]. When models are not understandable, it leads to trust and verification issues, underlying the need for interpretability in machine learning. However, it is important to recognize a trade-off between model flexibility and interpretability. With increasing task complexity, it is necessary to use complex models to achieve high accuracy at the cost of reduced interpretability. Examples of such tasks include predicting the sentiment of a sentence or detecting various objects [116, 115]. Briefly, Miller defines in [117] interpretability "as the degree to which a human can comprehend the cause of a decision" [114].
Authors in [118] have defined several factors that can be optimized through interpretability:

- **Fairness:** Ensuring that predictions are not biased (e.g., racial biased).
- **Privacy:** Protecting sensitive information within the model.
- **Reliability/Robustness:** Small input alteration does not cause a great change in the output.
- **Causality:** Understanding causal relationships within data
- **Trust:** Interpretable models are more likely to be trusted

In some scenarios, interpretability may not be as important as predictive performance. For instance, in low-risk systems such as product recommenders and advertisement systems [114, 119], there is no need to understand the reason behind predictions as long as they are accurate. However, in fields such as finance, cybersecurity and medicine, a prediction must be completely comprehended and well-trusted. The consequences in these domains can result in large financial losses or even life-threatening situations.

The primary objective of interpretable methods is to explain a model based on explanations (Figure 2.7). Miller defines an explanation as an answer to why-question [114, 117].

Methods used to interpret machine learning models can be model-specific or model-agnostic [114]. Model-specific methods are restricted to specific models and they rely on

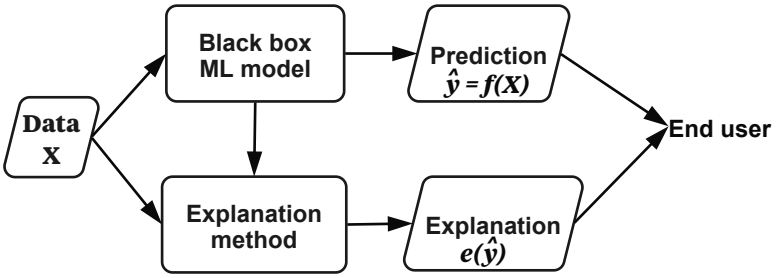

Figure 2.7: Explanation pipeline for a machine learning model (based on [114])

the model's internal structure to produce explanations (e.g., weights of a linear model). On the other hand, model-agnostic methods are more flexible and can be applied to any trained machine learning model [114, 119]. Some of these methods will be discussed in the following sections.

2.4.1 Model-Specific Methods

2.4.1.1 Tree's Feature Importance from Mean Decrease in Impurity

The concept of feature importance in decision trees was introduced by Breiman in [120]. The importance of a feature X_j can be measured as follows [121]:

$$\mathrm{Imp}\left(X_j\right) = \sum_{t \in \varphi} \Delta \mathrm{I}\left(\tilde{s}_{\mathrm{t}}^{\mathrm{j}}, \mathrm{t}\right). \tag{2.2}$$

The introduction of a surrogate split $\tilde{s}_{\mathrm{t}}^{\mathrm{j}}$, that represents the closest split to the actual split s_{t} at the node t, is essential to combat the *masking effects*. Masking effects occur when a variable X_{j_1} is never chosen for splitting at a node because it performs slightly worse than another variable X_{j_2}. In such scenarios, X_{j_1} might wrongly appear as completely unimportant. If variable X_{j_2} is removed, a new tree may choose variable X_{j_1} for splitting, resulting in almost similar performance as the original tree. Therefore, when a variable X_{j_1} is masked by X_{j_2}, and $\tilde{s}_{\mathrm{t}}^{\mathrm{j2}}$ is similar to s_{t}, then $\Delta \mathrm{I}\left(\tilde{s}_{\mathrm{t}}^{\mathrm{j2}}, \mathrm{t}\right)$ will be close to $\Delta \mathrm{I}\left(s_{\mathrm{t}}, \mathrm{t}\right)$ and this measure shows the actual importance of the X_{j_2} [121].

On the contrary, when multiple randomized trees are built in a random forest, masking effects are mitigated due to the random selection of variables for constructing each tree. Even when X_{j_1} is masked by X_{j_2}, X_{j_1} could still be chosen if X_{j_2} is not in the randomly selected subset. Furthermore, the bootstrap samples introduce more diversity among trees, decreasing the likelihood that X_{j_2} will consistently outperform X_{j_1} [121].

Breiman proposed in [17, 122] a novel measure for assessing the importance of X_j by adding up the weighted impurity decreases for all nodes t where X_{j_2} is used. The total sum is then averaged over all trees in the random forest [121]:

$$\mathrm{Imp}\left(X_j\right) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t \in \varphi_m} 1\left(j_t = j\right) \left[p(t) \Delta i\left(s_t, t\right)\right], \tag{2.3}$$

where $p(t)$ represents the proportion of samples reaching t node, and j_t signifies the variable utilized to split node t [121].

For classification trees, various impurity measures $i(t)$ can be used such as misclassification error, Gini index, and cross-entropy or deviance. However, the Gini index and cross-entropy are generally preferred over misclassification error due to their differentiability and heightened sensitivity to the changes in node probabilities [43].

The variable importance in gradient boosting can be constructed in the same way as for random forests, except that trees are built differently. The iterative boosting process could result in ignoring some variables completely which is less likely in the random forest due to the underlying bagging technique [43].

2.4.2 Model-Agnostic Methods

2.4.2.1 Permutation Feature Importance

PI is a model-agnostic approach used to assess a model's behavior. The importance of a feature is determined by measuring the decrease in a model's performance score when the feature values are randomly shuffled. This process disrupts the relationship between a feature and outcome and its importance can be seen directly in an increase of prediction error [119]. The term permutation importance was introduced by Breiman in [17].

Similar to Breiman's random forest permutation importance, the authors in [123] refined it to be the model-agnostic approach. The procedure for PI, as defined and described in [119, 123] is as follows:

Input: trained model $\hat{f}$, feature matrix X, target vector y, error measure $L(y, \hat{f})$.

1. Compute the original error $e_{\text{orig}} = L\left(Y, \hat{f}\left(X\right)\right)$ (e.g., log loss function for classification or mean square loss for regression)
2. For each feature $j \in \{1, \ldots, n\}$ do:
   - Create the feature matrix X_{perm}, where values of feature X_j are randomly permuted. This process destroys the relationship between the feature and outcome y.
   - Estimate the error when data matrix with permuted feature X_{perm} is used for prediction $e_{\text{perm}} = L\left(Y, \hat{f}\left(X_{\text{perm}}\right)\right)$.
   - The importance of feature X_j is defined as $FI_j = e_{\text{perm}}$ - e_{orig}, where permutation that caused higher prediction error is associated with more importance.

There is a question of whether it is a better way to assess feature importance using the training data used for the model training or unseen, validation data. According to Molnar in [119], both approaches have their limitations. However, the use of validation

data may be a better approach to achieve more generalizable feature importance that is not biased to the training process. When the same training data is used for prediction and error measurement in permutation importance, it may result in inaccurate feature attributions. This is particularly critical when the model is overfitting. However, the use of training data will reveal which features the model relies on. On the other hand, in practice, it is desirable to use as much training data as possible and it can be a good reason to avoid the use of validation data for permutation importance. Moreover, when the model is not overfitting, the importance of features should remain similar for both the training and validation data [119].

PI offers numerous benefits, including an easy interpretation, a global insight into the model's behavior, and it does not require model retraining [119]. Another significant advantage is that PI considers all interactions between permuted features and other features. When feature values are permuted, it not only destroys the relationship with the outcome but also all relationships with the other features. Consequently, feature importance does not contain only the importance of itself, but also the importance of its relationships with the other features. This explains why the importance of all features does not add up to the total performance drop. However, this effect could also be considered as a disadvantage [119].

The greatest drawback of PI is the creation of unrealistic data instances as a result of the random permutation of feature values [119]. This issue becomes evident when dealing with correlated features. To illustrate, consider the scenario where features X_1 and X_2 are positively correlated, meaning that there are no data instances where a large X_1 value occurs with a small value of X_2, or vice versa. However, when employing PI, this unnatural values pairing may occur, although it is completely unrealistic and does not reflect the true data distribution. This phenomenon is commonly known as model *extrapolation* because the model must predict data that is positioned outside the range of the used training data [124]. Furthermore, the authors in [124] demonstrated that PI assigns large weights to these extrapolated predictions, leading to inaccurate feature importance.
Another limitation of PI is that the correlated features split the importance since they provide similar information. When permuting one of the correlated features, the model can still access the same information through the other features. Consequently, this can demote important features from their initially high to the lower levels of importance, greatly impacting the results of feature selection process [119].

2.4.2.2 Shapley Values

The Shapley values, named after their creator L. Shapley, represent a method from a collational game theory [125]. They can be conceptualized as a scenario where a team of players cooperate to contribute to a payout. The problem of a fair payout distribution among the players is solved by Shapley values [119].

This concept is also used in the field of machine learning to clarify how the predictions

are made. The game is represented by a prediction task for a data instance, where the feature values are equivalent to the players and the prediction itself is the payout. The players - feature values collaborate to receive the gain that in the end should be fairly distributed among them. The gain is "the difference between an actual prediction and the average prediction across all instances" [119]. For each feature value, there is a single Shapley value, that is calculated as an "average marginal contribution of the feature value across all possible coalitions" (combinations of features) [119]. The final goal is to explain the gain and show how each feature contributed to prediction [119].

The following illustrative example shows the calculation of a Shapley value for a specific feature value. There is a regression task where it is needed to predict apartment prices. Each apartment has three features such as size, floor, and a categorical feature indicating whether pets are allowed or not. Consider a particular apartment with a size of 50 square meters, located on the first floor, and pets are allowed. To determine the Shapley value for the "pets are allowed" feature, it is needed to simulate a coalition. This coalition includes only the feature that represents the size of the apartment and the "pets are allowed" feature. The feature related to the floor level should be excluded from this coalition. To avoid completely removing this feature and thereby altering the input function, it is possible to simulate its removal. Firstly, a new apartment data instance is randomly selected from the dataset and its floor value is donated to the observed initially observed instance. By doing so, a simulated coalition is created where the floor feature is absent, as it does not retain its original value but instead has been assigned a random one. Therefore, instead of completely excluding the feature from the coalition, it gets a random value. Finally, the prediction is made using the values from the simulated coalition. The next step simulates the omission of the "pets are allowed" feature by replacing its original value with a value from the previously drawn apartment instance. Now, both the floor and pet features are selected randomly from the other instances. The prediction is made again and a contribution of feature value "pets are allowed" can be calculated as the difference between these two predictions. For greater accuracy, it is possible to perform apartment random sampling several times and calculate the average contribution. This entire process is repeated for all possible coalitions and the average contribution yields the Shapley value. In the end, the calculated Shapley shows how much the feature value "pets are allowed" contributed to the prediction of the observed apartment instance, compared to the average prediction of all instances in the dataset [119].

The mathematical representation of Shapley value is defined as follows [119]:

$$\phi_j(val) = \sum_{S \subseteq \{1,\dots,n\} \setminus \{j\}} \frac{|S|!(n-|S|-1)!}{n!} (\text{val}(S \cup \{j\}) - \text{val}(S)), \tag{2.4}$$

where S represents the features used in the model, x is the data instance that needs to be explained, and n is the number of features. $val_x(S)$ is "the prediction of feature values in the subset S, that are marginalized over the features that are not included in S" [119]:

$$\text{val}_x(S) = \int \hat{f}\,(x_1, \dots, x_n)\, d\mathbb{N}_{x \notin S} - E_X(\hat{f}(X)). \tag{2.5}$$

Additionally, Shapley value is the only attribution method with the following properties [119]:

1. **Efficiency:** The sum of feature attributions is equal to the difference between the prediction for an instance x and the average prediction

$$\sum_{j=1}^{n} \phi_j = \hat{f}(x) - E_X(\hat{f}(X)). \tag{2.6}$$

2. **Symmetry:** If two feature values j and k contribute the same to all possible coalitions, then they have the equal Shapley values.
If

$$\text{val}(S \cup \{j\}) = \text{val}(S \cup \{k\}) \tag{2.7}$$

for all

$$S \subseteq \{1, \ldots, n\} \backslash \{j, k\} \tag{2.8}$$

then

$$\phi_j = \phi_k \tag{2.9}$$

3. **Dummy:** Feature j that does not contribute to the prediction of any of possible coalitions have the Shapley value of 0:

$$\text{val}(S \cup \{j\}) = \text{val}(S) \tag{2.10}$$

for all

$$S \subseteq \{1, \ldots, n\} \tag{2.11}$$

then

$$\phi_j = 0 \tag{2.12}$$

4. **Additivity:** If there is a game with combined payouts $val + val^+$ then the corresponding Shapley value is

$$\phi_j + \phi_j^+ \tag{2.13}$$

The additivity property holds great importance, especially for the models like ensemble trees. In the context of a RF, the Shapley value for a feature can be calculated by taking the average of the Shapley values computed for individual trees.

The greatest disadvantage of Shapley values is extensive computational demand due to a large number of possible coalitions. The number of coalitions grows exponentially with the number of features 2^n. Therefore, the authors proposed in [126] an approximation with Monte-Carlo Sampling:

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}\left(x_{+j}^m\right) - \hat{f}\left(x_{-j}^m\right) \right) \tag{2.14}$$

where $\hat{f}(x^m_{+j})$ is the prediction for a data sample x, where the random number of feature values are replaced with the values from the randomly sampled instance z, but without replacing feature j value. The x^m_{-j} term is identical to x^m_{+j}, only the feature value x^m_j is also replaced with the corresponding value from z.
Within the coalitions, a feature exclusion is simulated by random assigning of the feature value from a drawn instance. However, this is not executed for all possible coalitions, but rather for a defined number of iterations M. In each iteration, a random subset of feature values is substituted with their corresponding values from the drawn instance z. The selection of which feature values to replace is determined by a randomly generated permutation of the feature values. Specifically, for all features that are left in permuted vector to the obtained feature, the original values are retained, while to the right, the values are taken from the instance z. In the end, the marginal contribution is calculated for each iteration and after M iterations, all contributions are averaged [119].

Another limitation of Shapley values share similarities to PI. When simulating the omission of a feature from a coalition, the replacement of the feature with a value from a random instance may also lead to the creation of unrealistic instances. This is especially relevant for correlated features [119].

2.4.2.3 Shapley Additive Explanations (SHAP)

SHAP is a Shapley-based explanation method introduced by Lundberg et al. [12]. This method integrates the principles of local surrogate models (LIME) with Shapley values. The novelty is that Shapley values are expressed as a linear model. The explanation for an instance x is defined as [119]:

$$g\left(z'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.15}$$

where g represent the explanation model, $z\prime \in \{0,1\}^M$ is the coalition vector, M is the maximum coalition size and $\phi_j \in \mathbb{R}$ is the Shapley value for the feature j. The coalition vector $z\prime$ consists only of zeros and ones, where 0 represents the absence and 1 the presence of a feature [119].

SHAP has three important properties [119]:

1. **Local accuracy:**

$$\hat{f}(x) = g\left(x'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{2.16}$$

   This property is similar to efficiency from Shapley values.

2. **Missingness:**

$$x_j\prime = 0 \implies \phi_j = 0 \tag{2.17}$$

A missing feature in the coalition gets the attribution of zero because it is represented as 0 in the coalition vector. This is not an inherent property of Shapley values. Ensuring missingness, the local accuracy principle cannot be hurt.

3. **Consistency:** If a model changes and the marginal contribution of a feature value increases or stays the same, the corresponding Shapley values will also increase or stay the same.

Lundberg et al. proposed two different Shapley values estimation approaches, KernelSHAP and TreeSHAP [12, 127].

KernelSHAP includes the following steps [119]:

- Sample coalitions $z_k\prime \in \{0,1\}^M, k \in \{1,\ldots,K\}$, where the present features are represented by ones and absent by zeros.
- Convert each coalition $z_k\prime$ to original data space and get a prediction $\hat{f} : \hat{f}(h_x(z_k\prime))$, where $h_x : \{0,1\}^M \rightarrow \mathbb{R}^n$. This means that h_x maps ones to original values from data instance x which is the one to be explained, while zeros are selected from another randomly sampled data instance.
- Compute the weight for each $z_k\prime$.
- Fit weighted linear model: where the instances are assigned weights according to weights attributed to coalition vectors. The largest weights are assigned to the coalitions with either few ones (small coalitions) or many ones (large coalitions). This weighting strategy allows learning of isolation effects of a feature. Additionally, when a single feature is absent from the coalition, it enables examination of its primary impact and interactions with the other features.
- Return Shapley values ϕ_k, the coefficients of the linear model.

To optimize computational resources during the coalition vector sampling process, a strong strategy is to start with sampling all combinations where only one feature is included or excluded. This process would then progressively continue with combinations involving two features being included or excluded, and so forth [119].

Given that the absence of a feature is again represented by a randomly drawn value, the KernelSHAP encounters the same issue of generating unrealistic instances as PI and Shapley values. One potential solution could involve conditional sampling, but this approach changes the game and may result in non-zero Shapley values for features that are not used by a model [119].

On the other hand, TreeSHAP is a SHAP estimation approach proposed in [127] specifically designed for decision trees, random forests, and gradient boosting trees. Its primary distinction to KernelSHAP is the use of conditional expectation instead of marginal.

However, a challenge arises when the features are correlated. Authors in [128] demonstrated that a feature not used by the model can still receive a non-zero Shapley value when it is correlated with another feature that the model does use. However, TreeSHAP enhanced significantly the computation efficiency. Computation complexity is reduced from $O(TL2^M)$ to $O(TLD^2)$, where T represents the number of trees, L denotes the number of leaves, and D is the maximal tree depth [119]. The conditional expectation is described extensively in the original paper [127].

SHAP can be also used to calculate the global feature importance [119]:

$$I_j = \frac{1}{m} \sum_{i=1}^{m} \left| \phi_j^{(i)} \right|. \tag{2.18}$$

For a given feature j, the mean value of all absolute Shapley values is calculated to determine the global feature importance.

In summary, the greatest limitations of KernelSHAP are its slow computational performance, and neglect of the feature dependencies, that lead to extrapolation of data instances. On the other hand, TreeSHAP is faster but it may produce incorrect feature attributions because it changes the value function and the game by using conditional explanations [119].

CHAPTER 3

Methodology & Experiments

In this chapter, we present the procedure followed in the conducted experiments. The schematic description of performed experiments is visible in Figure 3.1. All experiments were implemented in *Python* programming language.

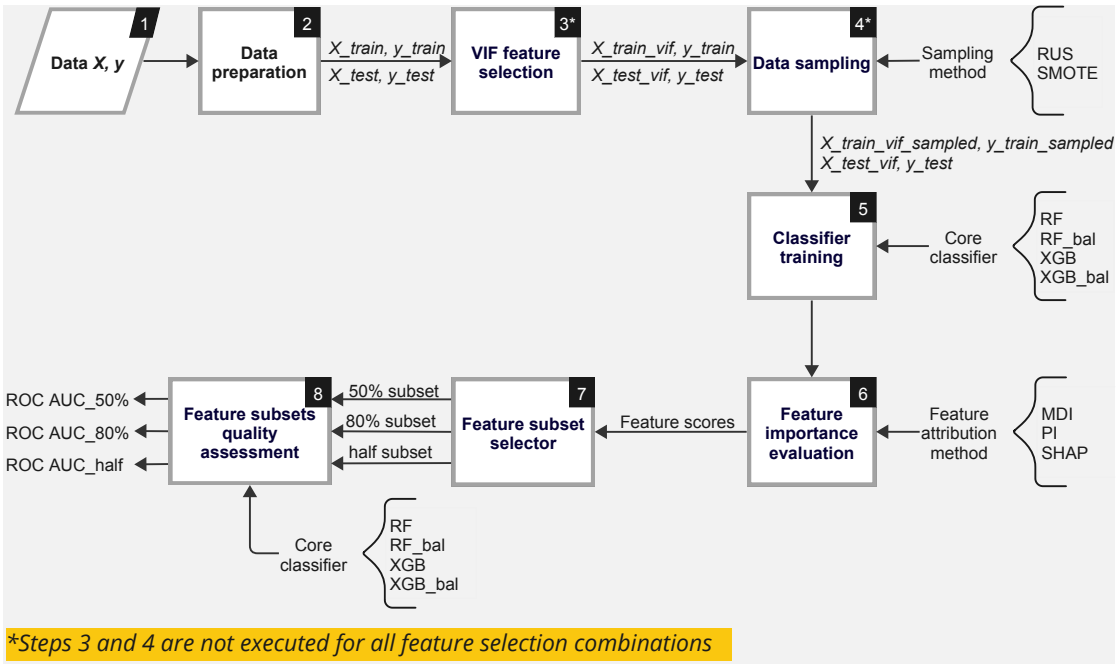

Figure 3.1: Schematic description of the conducted experiments

The schematic contains the following building blocks:

1. **Input data:** In our experiments, we used diverse imbalanced datasets for binary classification. Each dataset includes the input feature matrix X and class labels $y \in \{0, 1\}$. The instances that belong to the positive class are in the minority and the negative instances are in the majority.

2. **Data preparation:** The first step in our methodology is data preparation. We started by eliminating categorical and constant features from the feature matrix X. We also standardized the features and divided the input dataset into two sets: a training dataset X_train, y_train and a test dataset X_test, y_test. The split ratio between training and test data was 70/30.

3. **VIF feature selection:** In this step, which was optional for some feature selection combinations, we identified and removed multicollinear features based on VIF calculations performed on the training data, X_train. The same multicollinear features were also eliminated from the test dataset, X_test. In the end, there are two datasets without multicollinear features, X_train_vif and X_test_vif.

4. **Data sampling:** We applied RUS or SMOTE sampling techniques to address the issue of imbalanced data. The goal was to balance data distribution between the minor and major classes, achieving a ratio of 1:1. Data sampling was exclusively performed on the training dataset to prevent contamination of the test set and potential bias in the testing phase. This step was also optional for some of the tested feature selection combinations.

5. **Classifier training:** Since our feature selection relied on models, we selected, trained, and fine-tuned the core classifier for each feature selection combination. Possible core classifiers were RF, XGB, and their cost-sensitive variants - RF_bal and XGB_bal. It is noteworthy that the cost-sensitive versions were not combined with data sampling techniques from the previous step 4 since they internally address class imbalance through their learning algorithms.

6. **Feature importance evaluation:** In this phase we performed feature scoring on the training dataset using the previously trained classifiers and feature attributions methods. Feature attribution was performed using an internal method such as a tree's MDI, or by wrapping the classifier with model-agnostic methods like PI or SHAP. Each feature in the dataset received an importance score and the scores were sum-normalized to ensure consistent scaling across different attribution methods.

7. **Feature subset selector** With feature scores available, we sorted them by their importance and proceeded to select three distinct feature subsets: the first half of most important features and subsets containing features that contribute to at least 50% or 80% of the total feature importance.

8. **Feature subsets quality assessment** In the final step, we evaluated the quality of the selected subsets using the same core classifier chosen and utilized in steps 5 and 6. The classifier was trained and tested on each of the selected subsets from

step 7 and ROC AUC scores were recorded as performance metrics. We used the same classifier hyperparameters found in step 5.

In our experiments, we built various feature selection combinations selecting different methods within steps 3, 4, 5, and 6. All tested combinations are represented in Table 3.1. The notations used to describe each tested feature selection combination include the following parts:

- The core classier (RF, RF_bal, XGB, XGB_bal) which was used in step 5.
- The data sampling technique (RUS, SMOTE, none) which was applied to training data in step 4.
- The feature attribution method (MDI, PI, SHAP) which evaluated feature importance scores in step 6.
- The VIF feature selection if it was performed in step 3 to remove multicollinear features.

3.1 Dataset Preprocessing

To effectively address practical problems using machine learning, the two pivotal factors are the selection of a proper algorithm and obtaining quality data. Nowadays, there are many algorithms suitable for various problems. However, it is a great challenge to find proper datasets that accurately represent a problem and enable the algorithms to learn from it. The performance of many algorithms, regarding their complexity, can heavily depend on the quality and amount of input data, as it is shown in [129]. We can confidently say that sometimes data holds more importance than the choice of algorithm, as emphasized by Peter Norvig et al. in [130]. How to ensure the data quality is described in [131].

For this reason, we invested considerable effort in carefully selecting the diverse datasets from various domains. These datasets served as a solid representation of imbalanced binary classification tasks. The 17 datasets utilized in our experiments are visible in Table 3.2.

The versions without duplicates and normalization were used for the datasets from [133]. We performed feature selection with 36 different feature selection combinations across all 17 datasets, resulting in a total of 612 evaluations.

3.1.1 Scaling

One of the first steps in data preprocessing and feature engineering is bringing the data to a shape, that is more suitable for the learning process of a classifier. These techniques are commonly referred to as data normalization and standardization. Bringing the features

Feat. sel. comb.	Classifier	Data sampling	Feat. imp. method	VIF applied
RF_RUS_FeatImp	RF	RUS	MDI	No
RF_RUS_PI	RF	RUS	PI	No
RF_RUS_SHAP	RF	RUS	SHAP	No
RF_SMOTE_FeatImp	RF	SMOTE	MDI	No
RF_SMOTE_PI	RF	SMOTE	PI	No
RF_SMOTE_SHAP	RF	SMOTE	SHAP	No
RF_bal_FeatImp	RF_bal	None	MDI	No
RF_bal_PI	RF_bal	None	PI	No
RF_bal_SHAP	RF_bal	None	SHAP	No
RF_RUS_FeatImp_VIF	RF	RUS	MDI	Yes
RF_RUS_PI_VIF	RF	RUS	PI	Yes
RF_RUS_SHAP_VIF	RF	RUS	SHAP	Yes
RF_SMOTE_FeatImp_VIF	RF	SMOTE	MDI	Yes
RF_SMOTE_PI_VIF	RF	SMOTE	PI	Yes
RF_SMOTE_SHAP_VIF	RF	SMOTE	SHAP	Yes
RF_bal_FeatImp_VIF	RF_bal	None	MDI	Yes
RF_bal_PI_VIF	RF_bal	None	PI	Yes
RF_bal_SHAP_VIF	RF_bal	None	SHAP	Yes
XGB_RUS_FeatImp	XGB	RUS	MDI	No
XGB_RUS_PI	XGB	RUS	PI	No
XGB_RUS_SHAP	XGB	RUS	SHAP	No
XGB_SMOTE_FeatImp	XGB	SMOTE	MDI	No
XGB_SMOTE_PI	XGB	SMOTE	PI	No
XGB_SMOTE_SHAP	XGB	SMOTE	SHAP	No
XGB_bal_FeatImp	XGB_bal	None	MDI	No
XGB_bal_PI	XGB_bal	None	PI	No
XGB_bal_SHAP	XGB_bal	None	SHAP	No
XGB_RUS_FeatImp_VIF	XGB	RUS	MDI	Yes
XGB_RUS_PI_VIF	XGB	RUS	PI	Yes
XGB_RUS_SHAP_VIF	XGB	RUS	SHAP	Yes
XGB_SMOTE_FeatImp_VIF	XGB	SMOTE	MDI	Yes
XGB_SMOTE_PI_VIF	XGB	SMOTE	PI	Yes
XGB_SMOTE_SHAP_VIF	XGB	SMOTE	SHAP	Yes
XGB_bal_FeatImp_VIF	XGB_bal	None	MDI	Yes
XGB_bal_PI_VIF	XGB_bal	None	PI	Yes
XGB_bal_SHAP_VIF	XGB_bal	None	SHAP	Yes

Table 3.1: Description of feature selection combinations

to a similar scale accelerates the convergence of optimization techniques such as gradient descent, and reduces the impact of features with large values [134]. We standardized our data according to the formula:

$$X_j\_stand = \frac{X_j - \mu_j}{\sigma_j}, \tag{3.1}$$

where X_j is a feature j, μ_j represents the mean and σ_j is the standard deviation of the feature values. Standardization brings the data centered around the mean with a standard deviation of 1, ensuring that all features share the same magnitude. However,

Dataset	#points	#features	#major	#minor (%)
BreastW	683	9	444	239 (35%)
Cardiotocography	2114	21	1648	466 (22%)
Heartdisease	270	13	150	120 (44.4%)
Ionosphere	351	33	225	126 (35.9%)
Letter Recognition	1600	32	1500	100 (6.2%)
Mammography	11183	6	10923	260 (2.3%)
Mnist	7603	100	6903	700 (9.2%)
PageBlocks	5393	10	4883	510 (9.5%)
Pima	768	8	500	268 (34.9%)
Pendigits	6870	16	6714	156 (2.3%)
Satellite	6435	36	4399	2036 (31.6%)
Seismic	2584	14	2414	170 (6.6%)
SpamBase	4207	57	2528	1679 (39.9%)
Satimage-2	5803	36	5732	71 (1.2%)
Vertebral	240	6	210	30 (12.5%)
Vowels	1456	12	1406	50 (3.4%)
Waveform	3443	21	3343	100 (2.9%)

Table 3.2: Description of used datasets [132, 133]

data standardization may not be strictly requested when using classifiers like RF and XGB, which consist of decision trees that are invariant to data scaling [135, 136].

3.1.2 Train-Test Splitting

After training an algorithm, it is essential to evaluate its performance using test data. We split the data into training and test datasets. Typically, 80% of the dataset is used for training and the rest 20% for the test phase. However, our analysis is heavily dependent on the outcomes from the test data, and we wanted to achieve as variant results as possible. Therefore, we reduced the proportion of training data to 70% and increased the proportion for the test phase to 30%, making the training and test phase more challenging. Nevertheless, ensuring enough data for the training process is also very important, otherwise, the employed algorithms will not be able to learn effectively [98].

Another important aspect when dividing an imbalanced dataset is to ensure the same class ratio in the training and test sets as in the initial data. This is crucial when splitting data because due to random sampling, very few instances might be in the training or test set, and therefore model will not be able either to learn or to be tested effectively. This is also known as *stratified* sampling and it is usually preferred over random sampling.

3.1.3 Sampling

We used data sampling techniques such as RUS and SMOTE to achieve a 1:1 ratio between the majority and minority classes. Employing RUS, we eliminated random instances from the majority class to balance the data. On the other hand, with SMOTE we created new data instances based on the existing ones in the minority class. This enabled our classifiers to learn and distinguish the classes more effectively. To avoid any potential data bias between the test and training datasets, we applied these techniques exclusively to the training data.

The second approach for addressing the imbalanced datasets involved cost-sensitive versions of RF and XGB. Instead of altering the data distribution, we adapted the learning algorithms of our classifiers to be devoted more to the positive class. We configured the cost-sensitive version of RF classifier setting its *class_weight* parameter to *balanced* in *scikit-learn* package [137]. This setting automatically adjusts the weights assigned to positive and negative classes, making them inversely proportional to their frequencies in the training dataset. Similarly, we used the *pos_scale_weight* parameter in XGB to assign different weights to classes, enabling us to bias the learning process towards the minority class [138].

3.2 Multicollinearity

A multiple regression model is defined as:

$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + e \tag{3.2}$$

where y represents a dependent variable, $x_1, \ldots, x_n$ are independent variables or predictors, and the weights or regression coefficients of predictors are $\beta_1, \ldots, \beta_n$. e is an error term representing the difference between the estimated and actual value [139]. Several assumptions are vital for multiple regression models including a linear relationship between the outcome and the predictors and the absence of linear dependency or correlation between the independent variables [140]. Nevertheless, there is a condition where two or more predictors are linearly dependent and it is known as *multicollinearity* [141]. This phenomenon poses significant limitations in regression analysis, as it leads to an increase in the standard errors of coefficients. The regression coefficients are interpreted as the change in the dependent variable, y, resulting from one unit of change in the observed predictor while holding all others constant [142, 143]. However, this interpretation is impossible when predictors are correlated since changing one of the predictors leads to a change of others as well. This interplay may not only increase the standard errors in coefficients, but also their sign can be opposite to their true effect [139]. Consequentially, unreliable standard errors will lead to unstable p-values which are used to determine the significant difference of predictors [143]. In the end, certain predictors that should be statistically significant due to their strong correlation with the outcome may appear as insignificant [141, 144]. It is important to note, that although multicollinearity introduces

the issues regarding the estimation of the impact of a variable on the outcome [145], it may not impair the predictive performance of models [142].

In this work, we used tree-based methods such as RF and XGB which are non-parametric models and therefore, their predictive performance remains unaffected by multicollinearity [136]. However, when they are used to evaluate feature importance (for instance based on MDI), multicollinearity may adversely impact its stability. The independent effects of features that share the same information are hard to estimate. In a decision tree, the feature's importance may be reduced due to its correlation with other features. For instance, there are two correlated features A and B. If feature A is chosen as the feature that most effectively removes impurity at the observed node in a tree, the importance of feature B will be reduced, since the impurity it could remove is already eliminated by feature A. However, since RF performs bagging and uses different subsamples of data instances and features to construct each tree, it is less likely for correlated features to be included in the same subsample. However, the feature importance coefficients can still be impacted, and thus, the effect is not entirely eliminated.

For these reasons, the existing literature recommends addressing multicollinearity, and there are several possibilities [146]:

1. Multicollinearity often occurs due to a lack of information within a dataset to precisely investigate the underlying effects of features [143]. Therefore, since it is more the problem related to data rather than the model itself, the first option is to increase the sample size. However, this is sometimes impossible to do, or at very high costs [139]. Increasing the number of data instances introduces more data and feature variety, potentially leading to strengthened differences among the features. This diversity enables an easier and more precise estimation of the feature importance when dealing with ensemble trees such as RF. In linear regression, the benefits of a larger sample size are more pronounced since the standard errors are inversely proportional to the sample size. Therefore, an increase in the sample size will ultimately result in decreased standard errors and more precise and stable coefficient estimates. It is noteworthy that increasing the data sample size may only mitigate but not completely remove multicollinearity and its effects.

2. A more practical way is to remove the correlated predictors. This approach was used in our work.

3. Creating new variables by combining the existing variables is the last approach. For instance, we can create a new variable as a ratio between two correlated predictors and subsequently remove them.

4. Data standardization can also help to reduce the multicollinearity effects. However, this approach may only be beneficial for multiple linear regression models as discussed in [147, 146].

Generally, methods employed to mitigate multicollinearity can be classified into two groups [139]:

- **Variable selection methods**: A widely known dimension reduction technique, that is also a potential solution to multicollinearity is Principal Component Analysis (PCA) [148]. This statistical method creates uncorrelated principal components using two or more variables. A more robust method was proposed in [149], known as Partial Least Squares (PLS). PLS captures the characteristics of both dependent and independent variables, whereas PCA operates exclusively on the predictors.

- **Modified estimators**: These methods include ridge regression [150] or L2 regularization and lasso regression [151], known also as L1 regularization. They introduce regression coefficients along with a parameter λ as a penalty term to the loss function. The main difference between L1 and L2 regularization is that in L1 regression coefficients can be forced down to zero, making the interpretation and variable selection more approachable. However, the ridge regularization is considered more stable when dealing with the multicollinearity, since the effects are spread across all coefficients, while L1 seeks to eliminate some variables by forcing their coefficients to zero.

As previously discussed in 2.4.2, collinear features can adversely affect the feature attribution methods, such as MDI tree importance, PI, and SHAP importance. Therefore, we want to determine the degree to which multicollinearity impacts the tested feature selection combinations.

3.2.1 Variance Inflation Factor (VIF)

There are various methods to quantify collinearity among the features. One common approach is to measure pairwise correlations using a correlation matrix [139]. Usually, a correlation coefficient exceeding 0.8 is a sign of high correlation [152]. However, pairwise correlation coefficients are not practical, since they assess the correlation between only two features.

A more robust measure of multicollinearity is the VIF [142], calculated as follows:

$$\mathrm{VIF} = \frac{1}{1 - R_j^2}, \tag{3.3}$$

where R_j^2 is the coefficient of determination for the regression of a feature X_j on the remaining variables. The square root of VIF shows how much larger the standard error for the coefficients for a feature is, compared to a scenario in which that feature would not be correlated with the other features [141]. There is no strict threshold that confidently indicates the presence of multicollinearity, but the common thresholds are 5 or 10 [141, 153]. VIF is also known as a reciprocal value of Tolerance (TOL) [139]. Other common

methods to confirm the presence of multicollinearity are eigenvalues from the PCA and the Condition Index (CI) [139].

In our study, we used the VIF measure to eliminate the multicollinear features from the training and test datasets and the implemented algorithm is visible in Algorithm 1. Firstly, we calculate the VIF for each feature i in the training dataset X_train. Then, we find the feature with the maximal VIF and if its value exceeds the predetermined multicollinearity threshold, we remove that feature from both X_train and X_test datasets. After each elimination, we recalculate the VIF scores for the remaining features and iterate through the previous steps. The algorithm terminates when no feature in X_train has a VIF higher than the selected threshold. It is important to note, that VIF calculations were performed only on the training data.

Algorithm 1: VIF-based feature selection

```
Data: X_train, X_test
Result: X_train and X_test without multicollinear features
n ← number of features in X_train or X_test
vif_threshold ← 5
while True do
    vifs ← an empty list with n length
    for i ← 0 to n − 1 do
        vifs[i] ← variance_inflation_factor(X_train, i)
    end
    vifs_max ← maximal VIF in vifs
    feat_max ← feature name with the highest VIF in vifs
    if vif_max ≥ vif_threshold then
        remove feat_max from X_train, X_test
    end
    else
        break
    end
end
return X_train, X_test
```

3.3 Supervised Analysis

3.3.1 Hyperparameter Tuning

In machine learning, a learning algorithm comes with various parameters known as *hyperparameters* that need to be configured before model training. One of the essential steps in the training process is hyperparameter optimization or tuning. Hyperparameters are highly dependent on the input data and proper optimization is required to avoid issues like over- and underfitting. Overfitting is a problem where an algorithm performs

well on the training data, but poorly on unseen, test data. This means that the algorithm does not generalize well. Generally, the complex models are more prone to overfitting, while simple models are not powerful enough to catch the underlying data patterns and therefore they usually underfit.
It is advisable to train the model using the training data and determine the most optimal hyperparameters based on the performance of the model on a separate validation dataset. Using the test dataset for hyperparameter tuning can introduce a bias in the testing phase since the test set was part of the training process [98].

One of the greatest limitations of imbalanced datasets is the limited number of instances in the minority class. Thus, we divided our datasets into training and test datasets, where test datasets were also used for hyperparameter tuning of the core classifiers within the employed feature selection combinations. It is important to note that this approach does not significantly impact our analysis, since all combinations went under the same conditions through training and testing phases, enabling us to perform a fair comparative analysis.

There are various techniques to fine-tune a model, but the most used are grid search, random search, and Bayesian hyperparameter optimization.

Grid search is a suitable approach when the search space and the number of hyperparameters are relatively small. Otherwise, it can become excessively time-consuming as the search space and hyperparameters grow. In a scenario with only 3 hyperparameters, each with a search space of only 5 values, there would be a total of 125 combinations to evaluate. This means that we must train 125 distinct models. It becomes even more demanding when using cross-validation to train and assess the performance of each model [82].

To address this issue, more efficient techniques such as **random search**, or **Bayesian hyperparameter optimization** are used. In random search, rather than specifying discrete values for each hyperparameter, a statistical distribution is defined from which values are randomly sampled. It is also needed to determine the total number of iterations that indicate how many hyperparameter combinations will be evaluated [82].

The Bayesian technique takes the hyperparameter optimization step further. Instead of randomly sampling hyperparameter values from the search space, it uses past evaluation results and hyperparameter values to make better selections for future hyperparameter values. If a particular hyperparameter value has performed well in previous iterations, it is more likely to be chosen again in the following iterations. In this way, the objective function is optimized much faster [82].

In our work, we used *Optuna*, an automatic hyperparameter optimization software [154]. Optuna creates a *study* that optimizes an objective function within a specified number of *trials*. We specifically used the Tree-structured Parzen Estimator (TPE) that operates based on Bayesian optimization principles. It offers several benefits, including a define-by-run API (Application Programming Interface) that allows users to dynamically construct

the parameter search space, efficient implementation of both searching and pruning strategies, as well as easy-to-setup architecture [154].

Let $\mathcal{H}$ represent the hyperparameter space for a given core classifier in step 5. The hyperparameters for each classifier can be described as follows:

- **RF:**

$$\mathcal{H}_{\text{RF}} = \begin{cases} \textit{class\_weight} & : \text{None} \\ \textit{max\_depth} & : 10 \\ \textit{n\_estimators} & : [50, 500] \text{ (integer)} \\ \textit{max\_features} & : [0.1, 0.9] \text{ (float)} \\ \text{Other parameters} & : \text{Default from } \textit{scikit-learn 1.0.2} \text{ [137]} \end{cases}$$

- **RF_bal:**

$$\mathcal{H}_{\text{RF\_bal}} = \begin{cases} \textit{class\_weight} & : \text{balanced} \\ \textit{max\_depth} & : 10 \\ \textit{n\_estimators} & : [50, 500] \text{ (integer)} \\ \textit{max\_features} & : [0.1, 0.9] \text{ (float)} \\ \text{Other parameters} & : \text{Default from } \textit{scikit-learn 1.0.2} \text{ [137]} \end{cases}$$

- **XGB:**

$$\mathcal{H}_{\text{XGB}} = \begin{cases} \textit{subsample} & : [0.5, 1.0] \text{ (float)} \\ \textit{max\_depth} & : \{5, 6, \ldots, 18\} \\ \textit{min\_child\_weight} & : \{1, 2, \ldots, 20\} \\ \textit{learning\_rate} & : [0.01, 0.05] \text{ (logarithmic scale)} \\ \textit{n\_estimators} & : [50, 500] \text{ (integer)} \\ \textit{scale\_pos\_weight} & : \text{None} \\ \text{Other parameters} & : \text{Default from } \textit{xgboost 1.5.0} \text{ [138]} \end{cases}$$

- **XGB_bal:**

$$\mathcal{H}_{\text{XGB\_bal}} = \begin{cases} \textit{subsample} & : [0.5, 1.0] \text{ (float)} \\ \textit{max\_depth} & : \{5, 6, \ldots, 18\} \\ \textit{min\_child\_weight} & : \{1, 2, \ldots, 20\} \\ \textit{learning\_rate} & : [0.01, 0.05] \text{ (logarithmic scale)} \\ \textit{n\_estimators} & : [50, 500] \text{ (integer)} \\ \textit{scale\_pos\_weight} & : [1, 200] \text{ (integer)} \\ \text{Other parameters} & : \text{Default from } \textit{xgboost 1.5.0} \text{ [138]} \end{cases}$$

3.4 Feature Scoring

The importance of features was evaluated using the training dataset. In this way, we prevented potential bias and retained the test data only for performance assessment.

We assessed the importance of features using three different attribution methods:

- **MDI:** RF and XGB provide feature importance based on the MDI that is defined as the total decrease in node impurity averaged across all trees in the tree ensemble. This is an efficient approach since feature importance is calculated during the training process and no additional steps are required.

- **PI:** The second approach is PI which requires several parameters to generate robust feature importance scores. The importance of a feature is determined by measuring the decrease in performance when the feature values are randomly permuted. This method needs a trained classifier to assess the impact of feature permutations, a performance metric, and data. In our experiments, we used the ROC AUC score to measure the importance of permuted features. To ensure stability, we performed 10 permutations for each feature and then averaged their importance.

- **SHAP:** SHAP is rooted in a method from the game theory known as Shapley values. Since we used RF and XGB as core classifiers, we calculated efficiently the Shapley values using TreeSHAP implementation. We calculated the global feature importance by averaging the absolute Shapley values for each feature independently.

It is noteworthy that multicollinear features removed during the VIF selection process in step 3 of the described methodology did not receive any importance score. In other words, their importance score was automatically set to 0 and they were not included in any of the reduced feature subsets in step 7.

3.5 Feature Selection

After calculating the importance scores, we reduced dataset dimensionality by selecting the most significant features. We used three distinct criteria, each leading to the selection of a subset:

- **Best half features subset:** The best half subset includes at most the top $n/2$ features, where n represents the number of features in either X_train or X_train_vif depending on the feature selection combination being tested. It is noteworthy that a feature selection combination may evaluate less than half of the features with a score different from 0. In this scenario, we include only the features that have non-zero importance scores, but ensure that the total number of included features does not exceed $n/2$. For instance, a feature selection combination is applied to X_train with 6 features and the sum-normalized scores are $\{0.6, 0.4, 0.0, 0.0, 0.0, 0.0\}$. In this context, the half subset allows the selection of a maximum of 3 features. However, we include only the first two, since all other features are considered unimportant, and we cannot prioritize any of them for inclusion in the best half subset.

- **50% subset:** This subset contains the best features that collectively contribute to at least 50% of the overall feature importance. For instance, if a combination

evaluates 6 features with importance scores of $\{0.35, 0.30, 0.20, 0.15, 0.0, 0.0\}$, the subset would include the first two features, resulting in the cumulative importance of 65% of the total feature importance.

- **80% subset:** The 80% subset consists of features that combined contribute to at least 80% of total feature importance. The features are selected in the same manner as for 50% subsets with the difference of setting the minimal cumulative importance threshold to 80%.

3.6 Performance Evaluation

3.6.1 Metrics

A widely used metric for evaluating the performance of a classifier is accuracy. However, accuracy is not suitable when dealing with imbalanced datasets. Consider a dataset with 90 major samples and 10 samples in the minority class. If a model predicts all samples as negative (major class), it would achieve an accuracy of 90%. At first glance, this accuracy appears impressive, but it conceals the model's poor performance in distinguishing between the two classes. The accuracy metric only counts the number of total correctly classified, without considering the performance of each class individually. This limitation becomes evident in imbalanced learning, highlighting a need for more appropriate performance metrics [102].

A convenient way to evaluate the performance of a classifier is with the use of a *confusion matrix*. The columns of the matrix represent the predicted instances, while the rows represent the actual instances. The confusion matrix for the binary classification problem is shown in Figure 3.2.

		Predicted class	
		Positive	Negative
Actual class	Positive	TP	FN
	Negative	FP	TN

Figure 3.2: Confusion matrix for binary classification (based on [102])

Typically, in binary classification, the positive class corresponds to the minor class, and the negative to the major class. True Positives (TP) and True Negatives (TN) indicate correctly classified positive and negative instances, while False Positives (FP) and False Negatives (FN) indicate the negative/positive instances misclassified as positive/negative, respectively [102].

The following metrics can be defined using the confusion matrix [102]:

- **Accuracy** is defined as the percentage of correctly classified instances:

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.4}$$

However, accuracy is not a proper metric for the imbalanced data, as it can yield a high score if the model simply predicts the majority class all the time. Moreover, it assumes that the misclassification cost of positive and negative instances are equal, which is often unrealistic in real-world scenarios, such as e.g., predicting an attack in the network or that a patient has cancer. In these cases, we may tolerate more FP than FN.

- **Precision or Positive Predictive Value (PPV)** is a fraction of correctly classified positive predictions among all positive predictions:

$$\text{PPV} = \frac{TP}{TP + FP}. \tag{3.5}$$

- **Recall or True Positive Rate (TPR)** is defined as the fraction of correctly classified positive samples among all actual positive samples in the test set:

$$\text{TPR} = \frac{TP}{TP + FN}. \tag{3.6}$$

- **Specificity or True Negative Rate (TNR)** is a fraction of correctly classified negative samples among all actual negative samples in the test set:

$$\text{TNR} = \frac{TN}{TN + FP}. \tag{3.7}$$

- **False Positive Rate (FPR)** represents the fraction of misclassified negative samples, among all negative samples in the test set:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{3.8}$$

- **F-measure** is a metric that combines precision and recall, where the focus is more on the positive class. There is an option to assign higher importance to one of the terms, using the parameter β:

$$\text{F}_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3.9}$$

  Typically, β is set to 1, treating false negatives and false positives equally costly. If false positives are more costly, β could be set to 0.5 or if greater emphasis should be set on false negatives then β could be set to 2.

- **G-mean** is a metric that treats both classes as equally important. G-mean is defined as the geometric mean of the TPR and TNR:

$$\text{G-mean} = \sqrt{TPR \cdot TNR} \tag{3.10}$$

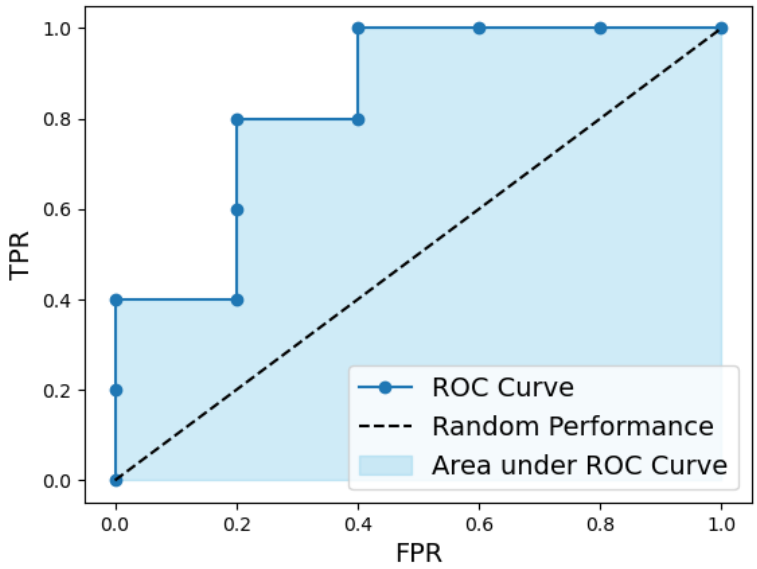

Figure 3.3: Example of a ROC graph (based on [102])

Compared to previous metrics, the **ROC AUC** operates with scoring predictions rather than discrete class labels. Many classifiers provide a numerical score for an instance indicating the likelihood that the instance belongs to the positive class. To obtain the class labels, a threshold must be defined and if the instance score is above the threshold it would be labeled as positive, otherwise as negative class. Adjusting this threshold can lead to different class labels, impacting performance [102].
ROC AUC is a graph performance evaluation method, where the x-axis represents FPR and y-axis the TPR. FPR quantifies the probability of a false alarm, meaning that a negative instance will be classified as a positive. By increasing the threshold for the class labeling, the FPR decreases, since only very few negative labels surpass the high threshold. On the other hand, TPR is a probability that a positive instance will be classified as a positive. Decreasing the threshold maximizes this metric but increases false positive rates. The ROC AUC considers both metrics and represents the probability that a randomly chosen positive instance will receive a higher score for belonging to the positive class than a randomly chosen negative instance. A classifier is expected to have ROC AUC score higher than 0.5 since it is the ROC AUC of a random classifier [102].

The ROC AUC's ability to work with scoring predictions introduces a level of granularity, that previous measures lack. An example of the ROC AUC score is represented in Figure 3.3. Each point in the figure represents a different value of the threshold, for which the FPR and TPR are calculated. The linear line represents the ROC curve for a random classifier and the goal for a good classifier is to have all plotted points above that curve. A larger area under the curve (the area in the blue color) indicates a better classifier. In

the ROC space, the lower left corner or origin represents the scenario where the classifier predicts always the negative class, while the top right corner represents a classifier that predicts only the positive class [102].

In this work, we used the ROC AUC measure to assess the performance of different feature selection combinations, since it is widely used in literature due to its robustness and effectiveness [155].

In our experiments, we tested various feature selection combinations that were constructed by combining 2 classifiers (RF, XGB), 3 data balancing techniques (RUS, SMOTE, and cost-sensitive learning), 3 feature importance methods (MDI, PI, and SHAP) and 2 variants (with or without VIF-based removal). This resulted in a total of 36 distinct feature selection combinations, which were evaluated on 17 imbalanced datasets. Each combination performed the selection of 3 distinct feature subsets (50%, 80%, and half) per dataset. The quality of each feature subset was assessed using the ROC AUC scores and therefore our analysis generated a total of 1,836 ROC AUC values (36 combinations $\times$ 3 feature subsets $\times$ 17 datasets). The results are presented and discussed in the following section.

CHAPTER 4

Results & Discussion

4.1 Comparative Analysis of Feature Selection Methods via ROC AUC score

In this section, we assess the quality of feature subsets selected by tested feature selection combinations. Each combination performs the selection of three distinct feature subsets - the best half features and subsets containing features that contribute at least 50% or 80% of the total feature importance. The core classifier (RF, XGB, RF_bal, XGB_bal) of each combination is used to evaluate the performance of the selected subsets. Our analysis involves 36 different feature selection combinations tested across 17 datasets and the goal is to identify a superior feature selection combination in terms of ROC AUC performance. All ROC AUC scores can be found in section 6.1.

We begin our analysis with 50% feature subsets. Firstly, we will describe the process behind the results represented in Table 4.1, which is also visually shown in Figures 4.1 and 4.2. It is important to emphasize that we computed mean ROC AUC scores separately for each subset selector (50%, 80%, and half subsets). Figure 4.1 illustrates exclusively the computation of the mean ROC AUC score for a specific combination (*feat_sel_c1*) when utilizing 50% feature subsets. The combination is applied across all 17 datasets, resulting in 17 distinct ROC AUC scores for 50% feature subsets. Averaging these scores yields the mean ROC AUC score for that combination. The same procedure is conducted for all other combinations (36 in total).
The computation underlying the mean ranks is somewhat more complicated and it is presented in Figure 4.2. Again, we observe only 50% feature subsets. For the first dataset (*Data_1*), we calculate the ROC AUC scores for each combination when 50% subsets are used. Subsequently, ranks are assigned to each combination based on their performance, with the lowest rank (0) given to the combination with the highest ROC AUC score. This ranking procedure is conducted across all datasets (17 in total) and we compute the average rank for each method. For instance, if there were 3 datasets a method could be

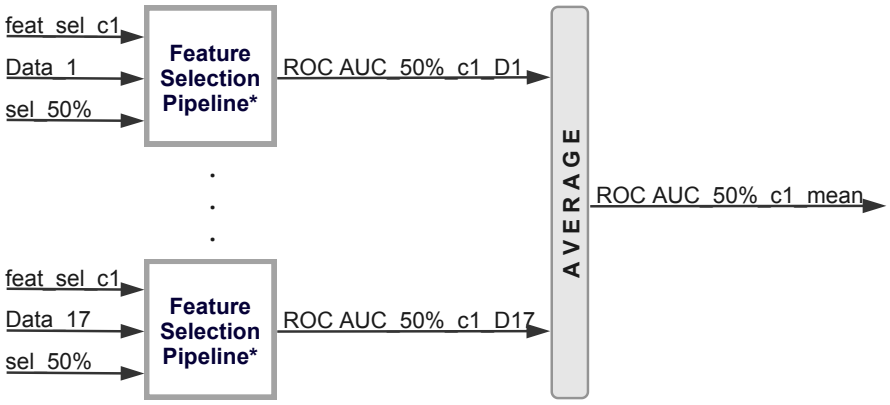

Figure 4.1: Computation of the mean ROC AUC for a feature selection combination across 17 datasets while considering 50% feature subsets. *This block represents the methodology presented in 3.1.

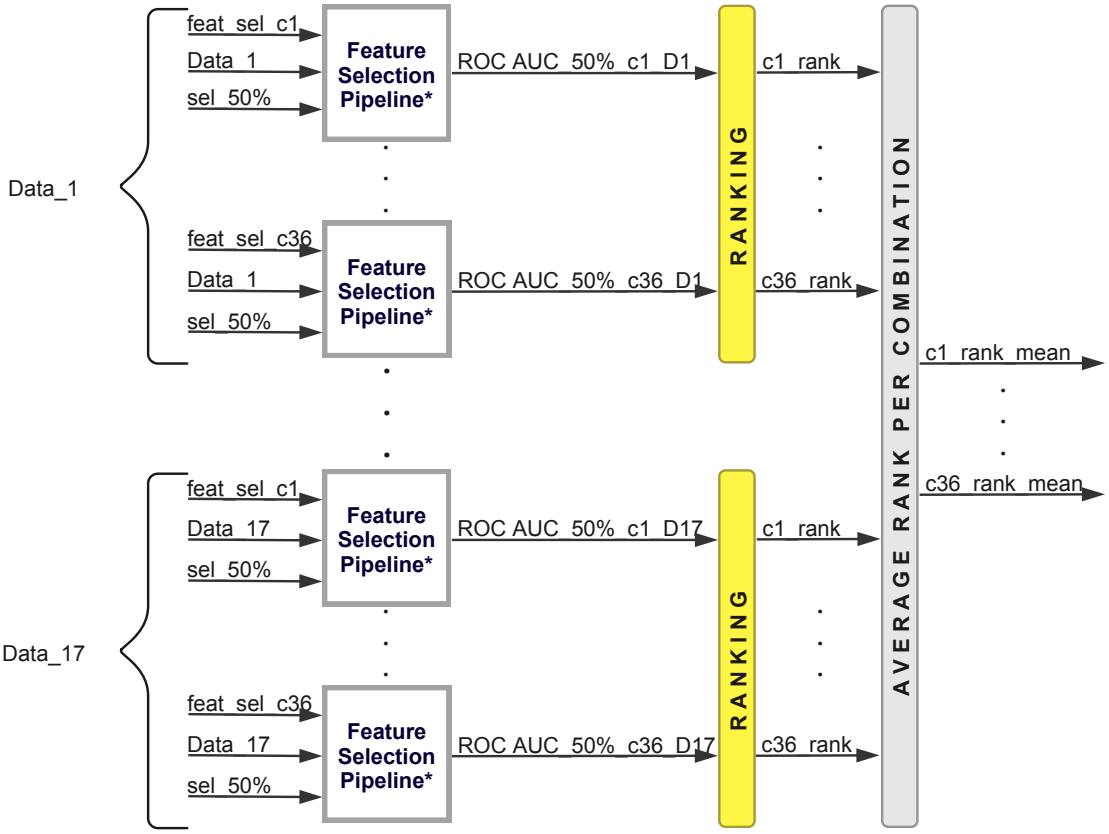

Figure 4.2: Computation of the mean ranks for all feature selection combinations across 17 datasets while considering 50% feature subsets. *This block represents the methodology presented in 3.1.

feat. sel. comb.	mean ROC AUC	mean rank ↓
XGB_SMOTE_SHAP	0.853	7.618
RF_RUS_FeatImp_VIF	0.847	9.676
XGB_SMOTE_SHAP_VIF	0.839	11.0
XGB_RUS_SHAP	0.839	11.088
XGB_bal_SHAP	0.846	11.147
XGB_bal_SHAP_VIF	0.848	11.265
RF_RUS_SHAP	0.841	11.382
RF_RUS_FeatImp	0.841	12.235
XGB_SMOTE_FeatImp	0.84	12.529
XGB_RUS_SHAP_VIF	0.836	12.618
RF_SMOTE_SHAP	0.823	13.353
XGB_bal_FeatImp	0.828	13.353
RF_RUS_SHAP_VIF	0.835	13.382
RF_SMOTE_FeatImp	0.822	13.412
XGB_RUS_FeatImp	0.827	14.588
RF_SMOTE_SHAP_VIF	0.82	16.382
XGB_RUS_FeatImp_VIF	0.814	16.824
XGB_SMOTE_FeatImp_VIF	0.813	17.235
RF_SMOTE_FeatImp_VIF	0.81	17.382
RF_bal_SHAP	0.786	17.912
XGB_bal_FeatImp_VIF	0.802	19.147
RF_bal_FeatImp	0.767	20.824
XGB_RUS_PI_VIF	0.79	20.941
XGB_RUS_PI	0.791	22.0
XGB_SMOTE_PI	0.782	22.471
RF_bal_SHAP_VIF	0.754	23.147
XGB_SMOTE_PI_VIF	0.777	23.206
RF_bal_FeatImp_VIF	0.739	24.794
XGB_bal_PI_VIF	0.725	26.059
RF_RUS_PI_VIF	0.772	26.088
RF_RUS_PI	0.77	26.235
XGB_bal_PI	0.704	26.588
RF_SMOTE_PI	0.731	29.118
RF_bal_PI_VIF	0.711	30.176
RF_bal_PI	0.716	30.382
RF_SMOTE_PI_VIF	0.729	30.441

Table 4.1: Summary of feature selection combinations for 50% feature subsets. Mean ROC AUC scores and ranks across datasets. A lower rank implicates a better performance.

ranked in the following way: in the first dataset, it had the best performance (rank 0), in the second dataset, it performed the worst (rank 35, if we were comparing 36 different methods), and in the third dataset, it exhibited the second-best performance (rank 1). Therefore, the average rank would be $(0 + 35 + 1)/3 = 12$. At the end, each combination receives a single mean rank. Almost an identical approach was applied to generate Tables 4.2 and 4.3, where instead of the 50% subsets, 80% or half feature subsets were used.

Table 4.1 reveals that the combination of XGB, SMOTE resampling, and SHAP feature importance method demonstrates the best results in terms of both mean ROC AUC scores and ranks. However, the mean rank of 7.618 indicates that this combination was not consistently the best in each dataset. In the second position, we find RF combined with RUS and its built-in feature importance estimation - MDI. For this combination a preliminary VIF multicollinearity correction was executed, to reduce potential issues that can introduce instability in feature coefficients. In the third place, we once again find XGB with SMOTE and SHAP, but now VIF multicollinearity correction has been also conducted before feature scoring.

In summary, XGB displays the best results, united with SHAP across all used class imbalance techniques: SMOTE, RUS, and cost-sensitive learning. On the other hand, RF performs the best with MDI importance, followed closely by SHAP. On the contrary, the least effective combinations are those using PI for feature attribution. Regarding class imbalance techniques, we can infer that XGB performs the best with SMOTE, followed by RUS. However, RF demonstrates the opposite trend. Both core algorithms perform worst with the balanced class weights. An interesting observation is that combinations incorporating VIF correction do not manage to select the most important features as well as non-VIF combinations.

In addition to the scores presented in Table 4.1, we want to analyze statistical differences between employed combinations. Hence, we perform the Wilcoxon signed-rank test analysis, which shows if there is a significant difference between a pair of combinations. The resulting statistical difference matrix is shown in Figure 4.3. Unfortunately, our examination did not reveal any combination that significantly differs from the others.

We continue our analysis with 80% feature subsets (Table 4.2). As expected, we observe a performance improvement due to an increased amount of feature importance within the subsets. This time, RF_RUS_SHAP emerges as the most effective combination, followed by balanced versions of XGB: XGB_bal_SHAP, XGB_bal_SHAP_VIF, XGB_bal_FeatImp. The SHAP method shows consistent efficiency and retains its position at the top. Unlike in 50% subsets, our current winner is even more convincing, with an average ranking of 6.706. XGB once again shows the most desirable results (independently of the used balancing technique) when combined with SHAP, followed by MDI importance. However, this time RF performs the best with SHAP, closely followed by MDI feature importance. PI is still the feature evaluation method with the worst performance. If we look closely at class imbalance techniques, XGB achieves the best results when combined with class weight balancing, followed closely by SMOTE and RUS. On the other hand, RF achieves its finest performance with RUS, followed by SMOTE

sampling. Finally, we can again confirm that VIF correction does not result in significant performance improvement.

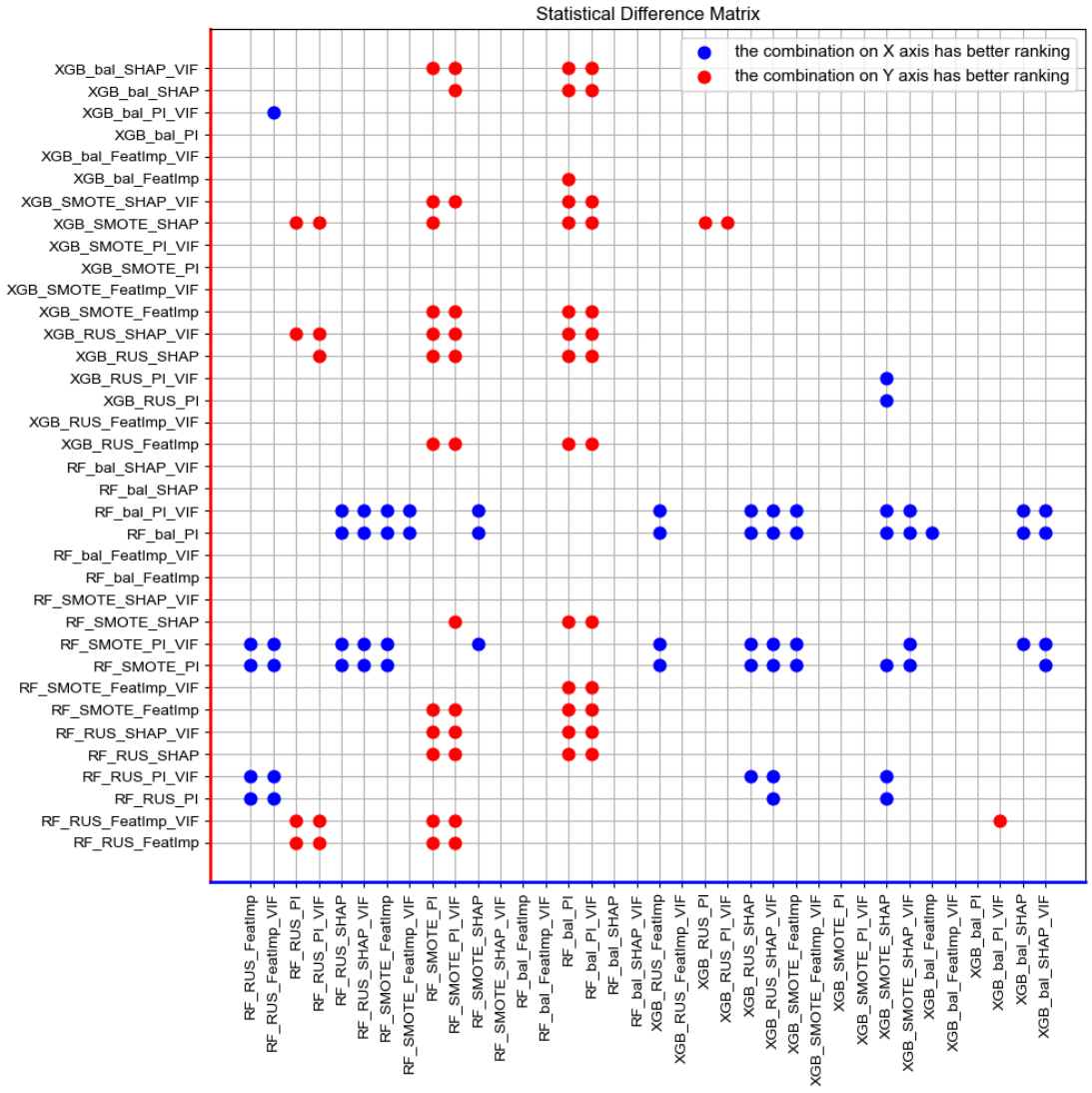

Figure 4.3: Pairwise statistical differences in ROC AUC scores for 50% feature subsets. A difference is marked by a circle. The color of the circle marks better-performing combination according to mean ranking across datasets.

The resulting statistical difference matrix for 80% subsets is shown in Figure 4.4. For a second time, our examination failed to reveal a combination that significantly differs from the rest. Nevertheless, compared to the analysis of 50% subsets, there is a stronger difference between the current winner and the remaining combinations.

Finally, we conclude our comparative analysis with the results of half feature subsets

feat. sel. comb.	mean ROC AUC	mean rank ↓
RF_RUS_SHAP	0.893	6.706
XGB_bal_SHAP	0.888	9.353
XGB_bal_SHAP_VIF	0.882	9.647
XGB_bal_FeatImp	0.882	9.647
RF_RUS_FeatImp	0.886	10.206
XGB_SMOTE_SHAP	0.876	10.235
RF_RUS_SHAP_VIF	0.88	10.265
RF_RUS_FeatImp_VIF	0.88	11.5
XGB_RUS_SHAP	0.875	11.676
XGB_SMOTE_FeatImp	0.878	11.706
XGB_bal_FeatImp_VIF	0.874	13.647
XGB_RUS_FeatImp	0.867	13.853
XGB_RUS_SHAP_VIF	0.873	14.382
RF_SMOTE_SHAP	0.854	15.147
RF_SMOTE_FeatImp	0.849	15.794
XGB_SMOTE_SHAP_VIF	0.872	15.794
RF_SMOTE_FeatImp_VIF	0.853	16.324
XGB_RUS_FeatImp_VIF	0.857	16.706
RF_SMOTE_SHAP_VIF	0.849	17.294
XGB_SMOTE_FeatImp_VIF	0.856	17.882
XGB_RUS_PI	0.831	21.735
RF_bal_SHAP	0.805	21.971
XGB_SMOTE_PI	0.836	22.5
RF_bal_FeatImp	0.804	23.5
XGB_RUS_PI_VIF	0.826	23.559
XGB_bal_PI_VIF	0.83	24.088
XGB_SMOTE_PI_VIF	0.821	24.853
RF_bal_FeatImp_VIF	0.799	25.441
RF_bal_SHAP_VIF	0.8	25.529
RF_RUS_PI_VIF	0.811	25.588
RF_SMOTE_PI_VIF	0.818	25.853
XGB_bal_PI	0.811	26.794
RF_RUS_PI	0.805	27.676
RF_SMOTE_PI	0.786	28.206
RF_bal_PI_VIF	0.764	30.147
RF_bal_PI	0.763	30.794

Table 4.2: Summary of feature selection combinations for 80% feature subsets. Mean ROC AUC scores and rank across datasets.

(Table 4.3). This time, there seems to be less variance in ROC AUC scores and ranks compared to previous subsets. To verify this, we use the mean and standard deviation of ROC AUC scores across all employed feature selection combinations for each dataset (Figure 4.5).

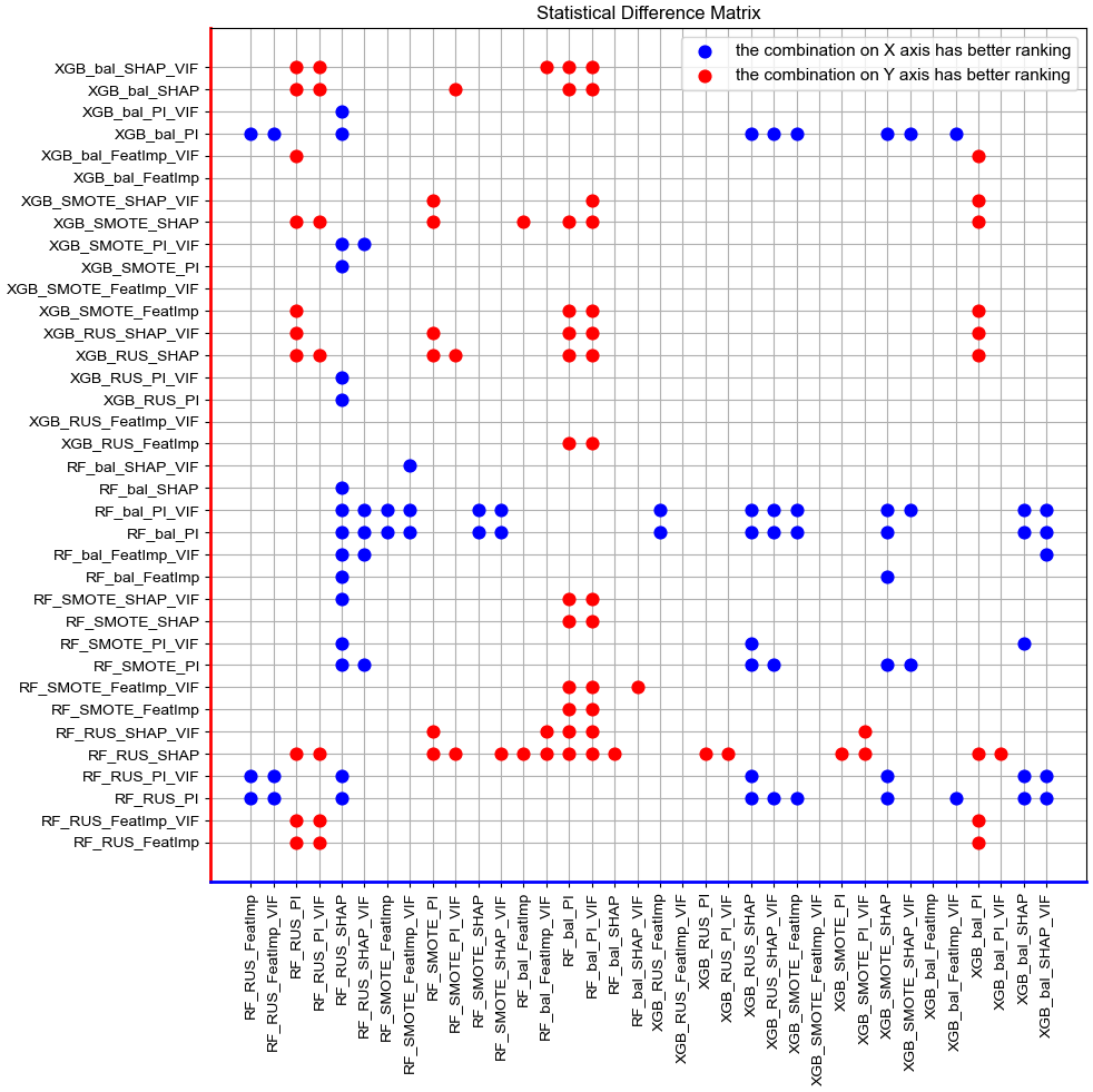

Figure 4.4: Pairwise statistical differences in ROC AUC scores for 80% feature subsets. A difference is marked by a circle. The color of the circle marks better-performing combination according to mean ranking across datasets.

It is evident that certain datasets such as *Vertebral*, *Letter Recognition*, and *Seismic* are more challenging, and therefore feature selection combinations exhibit weaker performance. However, the standard deviation is at its lowest for half subsets, indicating the smallest

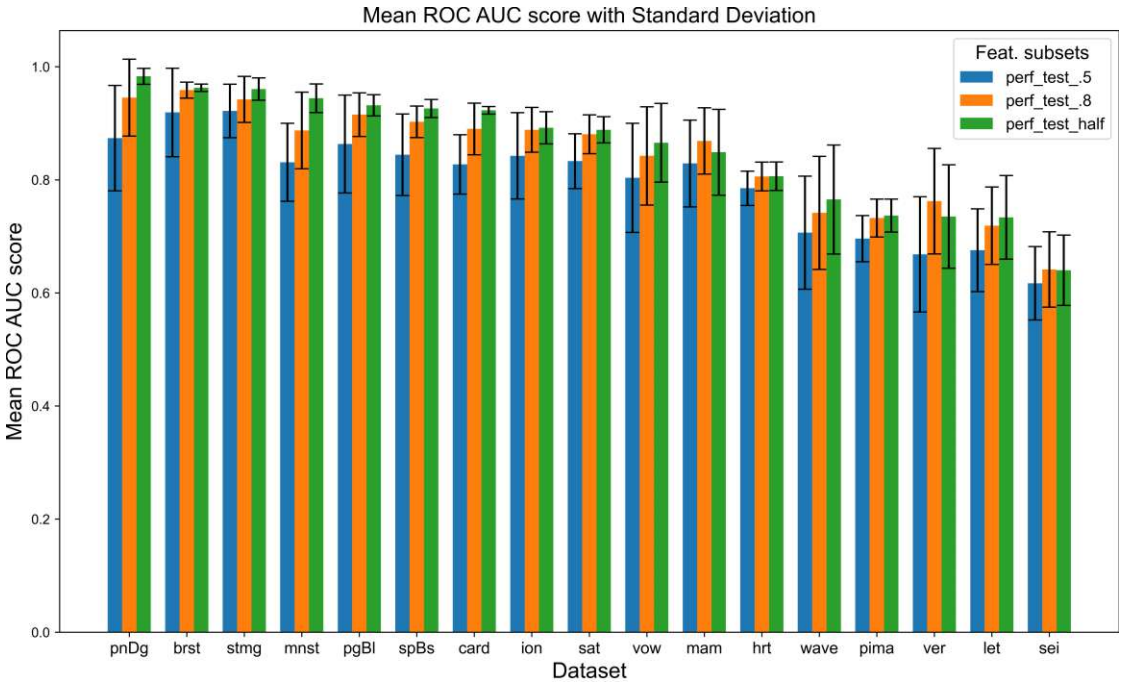

Figure 4.5: Mean ROC AUC score with standard deviation across all feature selection combinations for different datasets. Computation is performed separately for each feature subset (50%, 80%, half).

diversity in the performance across different feature selection combinations. Usually, the selected half feature subsets include the largest number of features (an example will be discussed later on). The expanded feature space increases the complexity of a model and enables the capture of more complex data patterns. It potentially leads to higher similarity in results across tested combinations. Moreover, the chance of selecting truly informative features using a method increases with the size of the selected subset. Ultimately, the similarity between the selected feature subsets increases.

To quantify the similarity between the selected feature subsets by distinct combinations, we followed the procedure demonstrated in Figure 4.6. We observe only the 50% feature subsets and a single dataset (*Data_1*). Each combination selects the 50% feature subset for the given dataset, resulting in 36 distinct feature sets. To assess the similarity between these subsets, we use *Jaccard Similarity* which is defined as the number of the shared elements that can be found in two sets (intersection) divided by the total number of unique elements in both sets (union).

Since Jaccard similarity applies only to two sets, we compare all selected subsets pairwise, considering all possible combinations. This results in a total of $\binom{36}{2} = 630$ combinations per dataset. Finally, we compute the average of these 630 Jaccard similarity coefficients to get a single value that represents the average similarity between selected subsets by feature selection combinations for the given dataset and selector. The same procedure

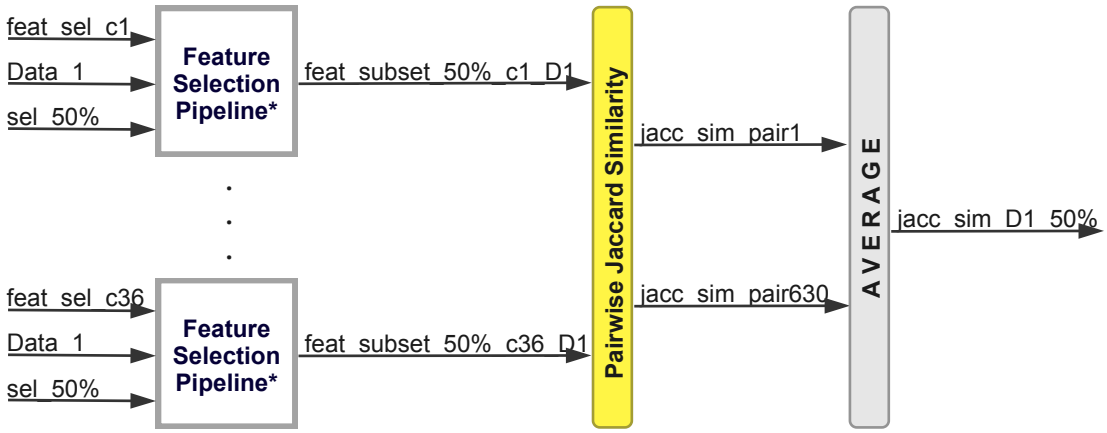

Figure 4.6: Computation of the Jaccard similarity among the selected 50% feature subsets by different feature selection combinations for a dataset. *This block represents the methodology presented in 3.1.

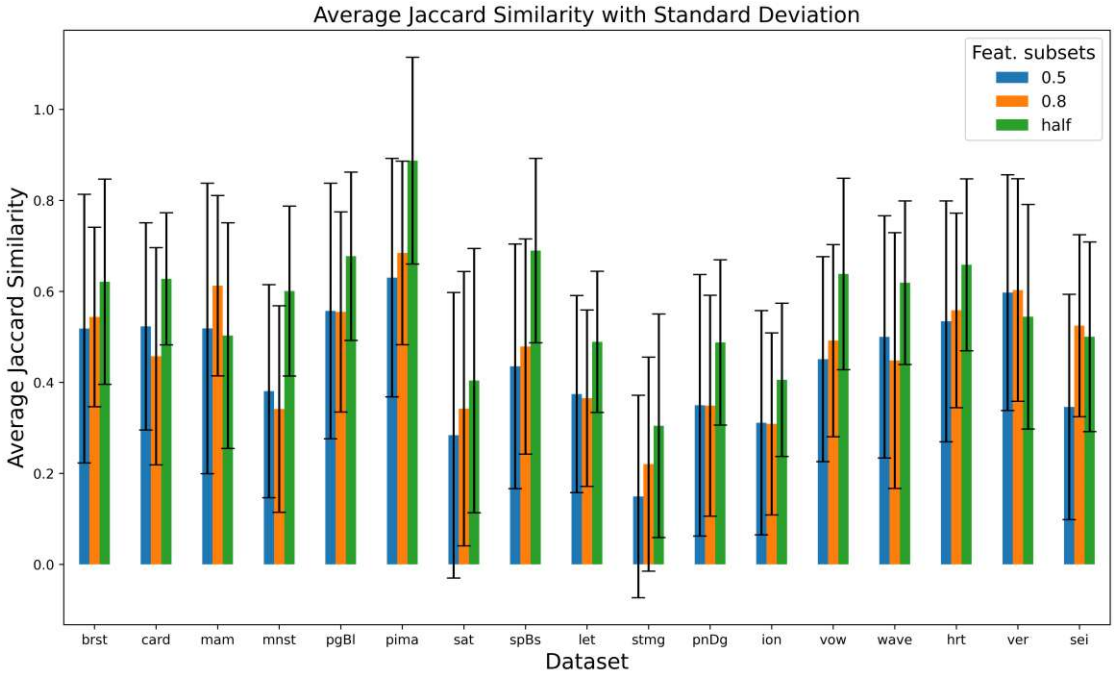

Figure 4.7: Average Jaccard similarity among the selected feature subsets by different feature selection combinations for different datasets

is done for 80% and half subsets, and results are presented in 4.7. It is visible that the Jaccard similarity is the highest for the half selection, indicating that the half feature subsets selected by different combinations are the most similar (compared to 50% and 80% selection). Furthermore, when observing only the half subsets, it is evident that the mean Jaccard coefficient varies across datasets, generally demonstrating moderate or low similarity (below 0.7 or 0.5). However, an interesting observation occurs when considering the standard deviation of ROC AUC scores (Figure 4.5) along with similarity results. Notably, low variance in the performance of different combinations for the half selection does not directly correlate with the high subsets similarity. For instance, the half similarity coefficient for the *Satimage* dataset is only about 0.33. However, the standard deviation of ROC AUC scores for the same dataset is very low. On the other hand, the half similarity for the *Vowels* dataset is more than double that of the *Satimage* dataset, but the variance of performance is much larger (normally, we would expect lower variance). This implies that we cannot establish a direct connection between the high subset similarity and the low variance in performance. Consequently, there may be another reason (e.g., characteristics of the used datasets) for the similar performance and the minimal differences in the Wilcoxon rank test, as presented in Figure 4.8.

We continue with the analysis of ROC AUC performance (Table 4.3). Similar to previous observations, RF_RUS_SHAP and XGB_SMOTE_SHAP continue to perform well. Intriguingly, the previously underperforming XGB_SMOTE_PI now shows the best results. We see similar trends regarding balancing techniques and VIF correction. The tested combinations lead to similar results when selecting the best half features and none of them is significantly superior to others (Figure 4.8). The overall variety of the presented ROC AUC scores was relatively low (Figure 4.5) and therefore we did not observe significant differences when conducting Wilcoxon rank tests. Since we were limited by the number of useful imbalanced datasets, we were not able to increase the statistical power of the performed analysis. However, RF balanced versions might not be the most appropriate choice for this task.

feat. sel. comb.	mean ROC AUC	mean rank ↓
XGB_SMOTE_PI	0.879	10.176
RF_RUS_SHAP	0.884	10.265
XGB_SMOTE_SHAP	0.878	10.412
RF_RUS_FeatImp	0.881	11.0
XGB_SMOTE_FeatImp	0.883	12.588
XGB_bal_SHAP	0.885	12.765
XGB_bal_PI	0.881	12.941
RF_RUS_FeatImp_VIF	0.871	13.353
XGB_RUS_SHAP	0.872	13.941
XGB_bal_FeatImp	0.873	14.412
RF_RUS_SHAP_VIF	0.869	14.765
RF_RUS_PI	0.874	14.912
RF_RUS_PI_VIF	0.869	15.176
XGB_RUS_PI	0.864	15.706
XGB_RUS_SHAP_VIF	0.869	17.471
XGB_RUS_FeatImp	0.865	18.412
XGB_RUS_PI_VIF	0.863	18.794
XGB_SMOTE_SHAP_VIF	0.871	19.0
RF_SMOTE_FeatImp	0.85	19.294
RF_SMOTE_SHAP	0.84	19.382
XGB_SMOTE_PI_VIF	0.868	19.5
RF_SMOTE_PI	0.844	19.559
XGB_bal_PI_VIF	0.865	19.912
XGB_bal_SHAP_VIF	0.868	19.912
XGB_bal_FeatImp_VIF	0.861	20.235
XGB_RUS_FeatImp_VIF	0.866	20.676
RF_SMOTE_PI_VIF	0.854	21.147
RF_SMOTE_SHAP_VIF	0.847	21.471
RF_SMOTE_FeatImp_VIF	0.847	21.706
XGB_SMOTE_FeatImp_VIF	0.859	22.529
RF_bal_FeatImp	0.799	25.941
RF_bal_PI_VIF	0.805	26.118
RF_bal_SHAP	0.805	26.294
RF_bal_PI	0.809	26.824
RF_bal_SHAP_VIF	0.794	27.265
RF_bal_FeatImp_VIF	0.788	32.147

Table 4.3: Summary of feature selection combinations for half feature subsets. Mean ROC AUC scores and ranks across datasets.

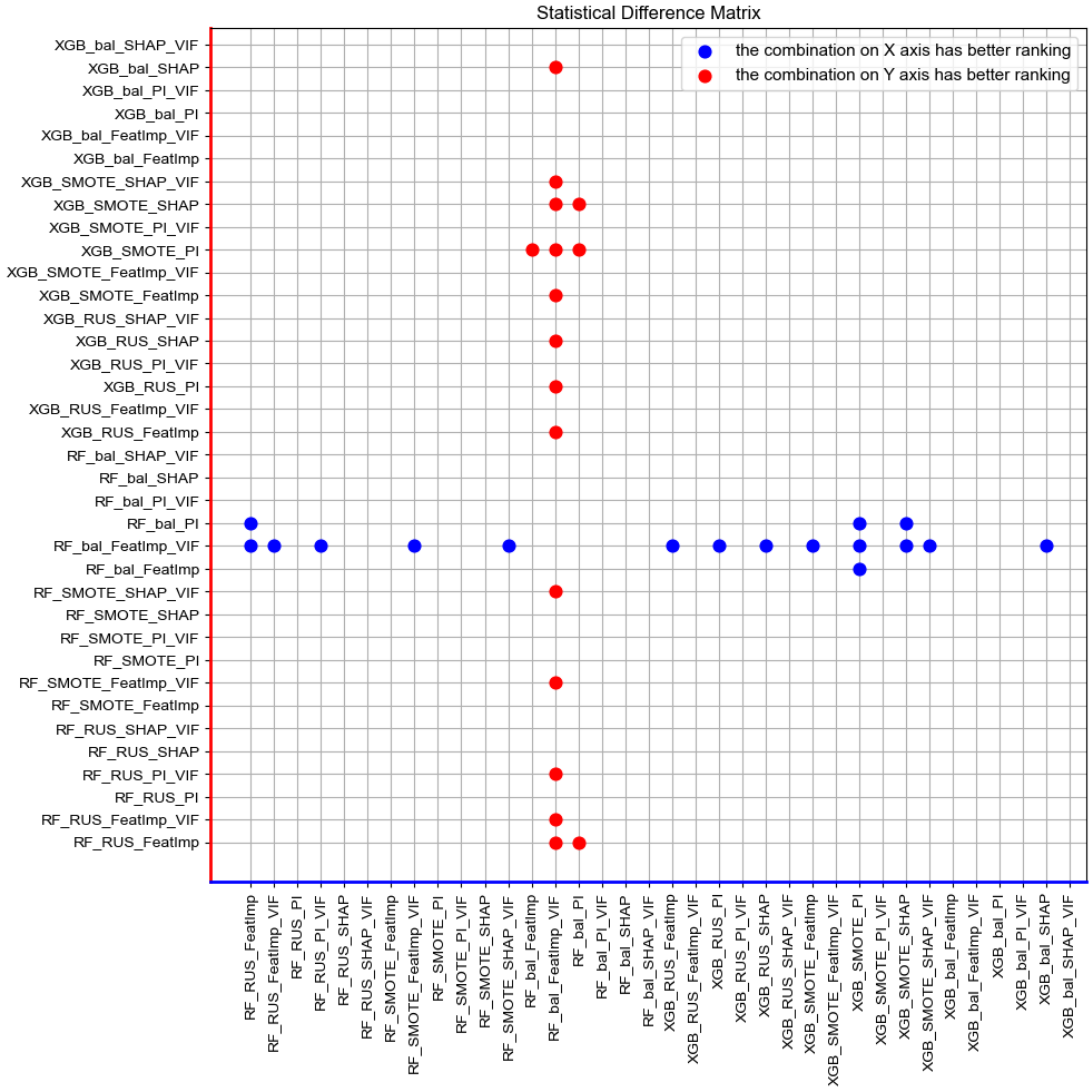

Figure 4.8: Pairwise statistical differences in ROC AUC scores for half feature subsets. A difference is marked by a circle. The color of the circle marks better-performing combination according to mean ranking across datasets.

4.2 Stability and Variance of Feature Scores

In this section, we analyze the stability of feature coefficients scored by different combinations. We expect from a stable and reliable combination that if a subset of chosen features weighs more in terms of accumulated feature importance, it will lead to better ROC AUC performance. Therefore, the 50% subsets should never outperform the 80% subsets scored by the same combination. For this analysis, we will observe results in Table 4.4.

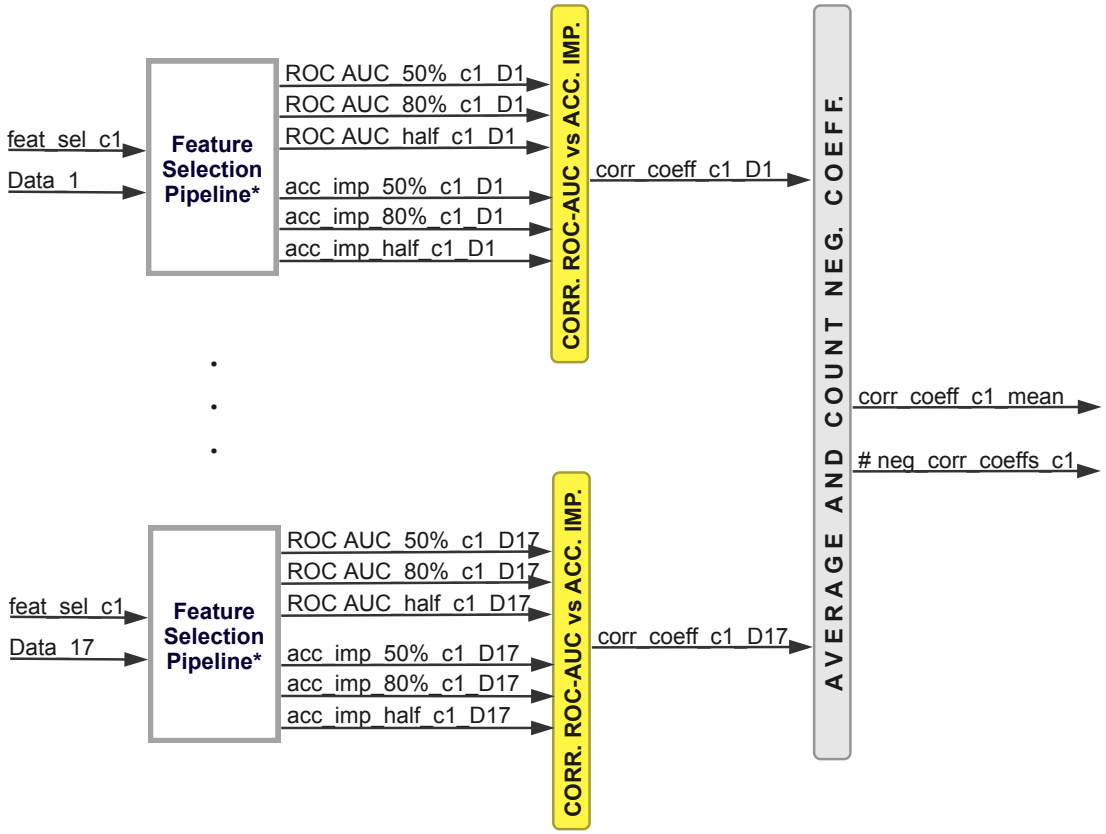

Figure 4.9: Computation of the mean correlation coefficient between accumulated feature scores and ROC AUC performance for a feature selection combination across all datasets. Additionally, we counted the number of negative correlation coefficients across all datasets.

To calculate the results presented in the table, we followed the procedure illustrated in Figure 4.9. We focus only on a single combination (*feat_sel_c1*) to explain the computation for a single row of Table 4.4. When applied to a dataset (*Data_1*), the combination selects three distinct feature subsets (50%, 80%, and half subset) and their quality is reflected by the corresponding ROC AUC scores. Additionally, each selected subset carries some weight in terms of the accumulated feature importance. Therefore, there are 3 ROC AUC scores and 3 accumulated feature importance scores and we

feat. sel. comb.	mean corr. coeff.		negative corr. instances	
	Pearson	Spearman ↑	Pearson	Spearman
RF_SMOTE_PI_VIF	0.978	0.971	0	0
RF_RUS_PI_VIF	0.937	0.971	0	0
RF_RUS_PI	0.932	0.963	0	0
XGB_RUS__SHAP	0.827	0.861	1	1
XGB_bal_PI_VF	0.827	0.853	1	1
XGB_bal_SHAP_VIF	0.815	0.853	1	1
XGB_RUS_Featlmp	0.829	0.845	1	0
RF_RUS_SHAP	0.894	0.837	0	0
XGB_bal_SHAP	0.864	0.829	1	1
XGB_bal_PI	0.839	0.824	1	1
XGB_RUS_PI	0.804	0.824	1	1
XGB_bal_Featlmp	0.879	0.816	0	1
XGB_SMOTE_PI	0.761	0.794	1	1
RF_RUS_Featlmp	0.85	0.786	1	1
XGB_SMOTE_Featlmp	0.849	0.786	1	1
XGB_RUS_FeatImp_VIF	0.813	0.781	1	1
RF_SMOTE_PI	0.809	0.765	2	2
RF_bal_SHAP_VIF	0.767	0.765	2	2
RF bal PI VIF	0.715	0.757	1	1
RF_RUS_SHAP_VIF	0.752	0.735	2	1
RF_bal_PI	0.73	0.735	1	1
RF_RUS_Featlmp_VIF	0.785	0.727	1	1
XGB_RUS_SHAP_VIF	0.725	0.706	2	2
XGB_bal_Featlmp_VIF	0.66	0.698	3	2
XGB_SMOTE_Featlmp_VIF	0.741	0.676	2	2
XGB_SMOTE_PI_VIF	0.738	0.676	1	2
XGB_SMOTE_SHAP	0.71	0.661	2	2
XGB_RUS_PI_VIF	0.618	0.618	3	3
RF_bal_Featlmp_VIF	0.631	0.61	3	3
RF_SMOTE_Featlmp_VIF	0.711	0.602	2	3
RF_SMOTE_SHAP_VF	0.62	0.594	3	3
RF_bal_Featlmp	0.597	0.588	3	3
RF_SMOTE_Featlmp	0.644	0.572	3	3
RF_SMOTE_SHAP	0.616	0.551	3	3
RF_bal_SHAP	0.529	0.522	4	4
XGB_SMOTE_SHAP_VIF	0.489	0.484	4	4

Table 4.4: Correlation analysis between accumulated feature scores and ROC AUC performance. Pearson and Spearman coefficients, along with the number of negative correlation coefficients.

calculate the correlation between them. This process is carried out across all datasets and the mean correlation coefficient is calculated. Additionally, we count the number of

negative correlation coefficients for a single combination across all datasets. Negative correlation coefficients imply that there were subsets where higher accumulated feature importance is associated with lower ROC AUC performance. The entire procedure is repeated for all feature selection combinations (36 in total) and results are documented in Table 4.4. Both Pearson and Spearman correlation coefficients were computed.

From the results we can derive that only RF shows consistently the expected behavior across all datasets: RF_SMOTE_PI_VIF, RF_RUS_PI_VIF, RF_RUS_PI, RF_RUS_SHAP. The first three methods have higher correlation coefficients meaning that there is a stronger ROC AUC increase when we use a subset with the larger importance. XGB balanced and RUS versions also show higher mean correlation coefficients, but these combinations tend to break the expected behavior.

Additionally, we present the mean correlation coefficients with standard deviation for each dataset in Figure 4.10. Simply, we averaged all correlation coefficients across the tested combinations for each dataset individually. Notably, there is a large discrepancy in stability among the tested combinations for certain datasets such as *Heartdisease*, *Mammography*, *Seismic*, *Letter Recognition* and others. Moreover, we observe lower mean correlation coefficients for these datasets, implicating overall poor stability in their feature coefficients. On the other hand, for *SpamBase*, *Satellite*, *Pima*, and others, all tested combinations demonstrate very high stability.

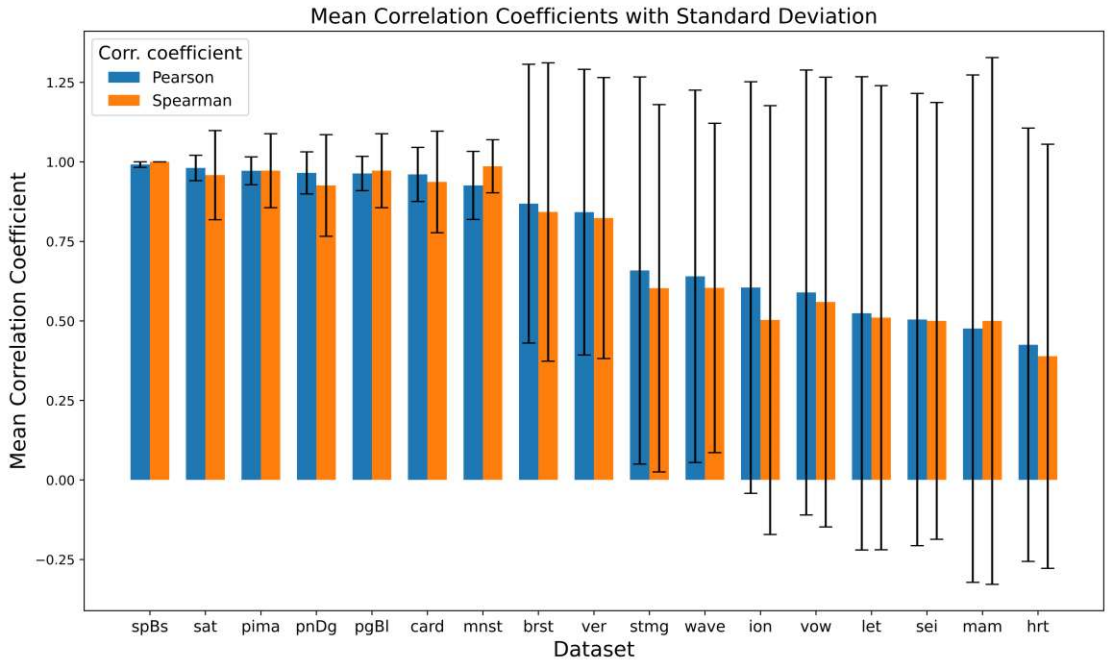

Figure 4.10: Mean correlation coefficients with standard deviation for different datasets. The procedure presented in Figure 4.9 is repeated for all 36 feature selection combinations, resulting in 36 different correlation coefficients per dataset and the mean is taken.

To assess a method's ability to distinguish between highly and less significant features, we use the Coefficient of Variation across a set of feature scores attributed by the method. The average value across all datasets is calculated and results are presented in Table 4.5. A higher coefficient of variation implies greater variability in feature scores.
Clearly, PI is the most discriminatory method when compared to the other two - SHAP and MDI importance, which show rather similar ability in distinguishing highly important features. In our experiments, we used a decrease in ROC AUC score resulting from feature permutation as an indicator of its importance. This evaluation method assigns high importance strictly to those features, that have a direct impact on the ROC AUC score. It appears that the number of such features per dataset is low. Figures 4.11, 4.12, and 4.13 show why PI combinations exhibited poor performance on 50% and 80% datasets, as a large amount of feature importance was attributed to a small set of features. We consider the *Ionosphere* dataset and XGB_SMOTE combinations as an example. Notably, SHAP and MDI importance manifest smooth feature scoring, resulting in much larger 50%, 80%, and half subsets. However, we see from our results in 4.1, that this has a great impact on PI combinations on 50% and 80% subsets but not on the half subsets. The half feature subsets include the highest number of features and this is a trend observed across multiple datasets and combinations. Thereby, we do not only have the best results for PI combinations on the half subsets in previous sections, but also for the other combinations. Although selected half subsets are much smaller for XGB_SMOTE_PI it still manages to have excellent results, in this example, XGB_SMOTE_PI achieves 0.901 ROC AUC score compared to 0.906 and 0.921 achieved by MDI and SHAP, respectively.

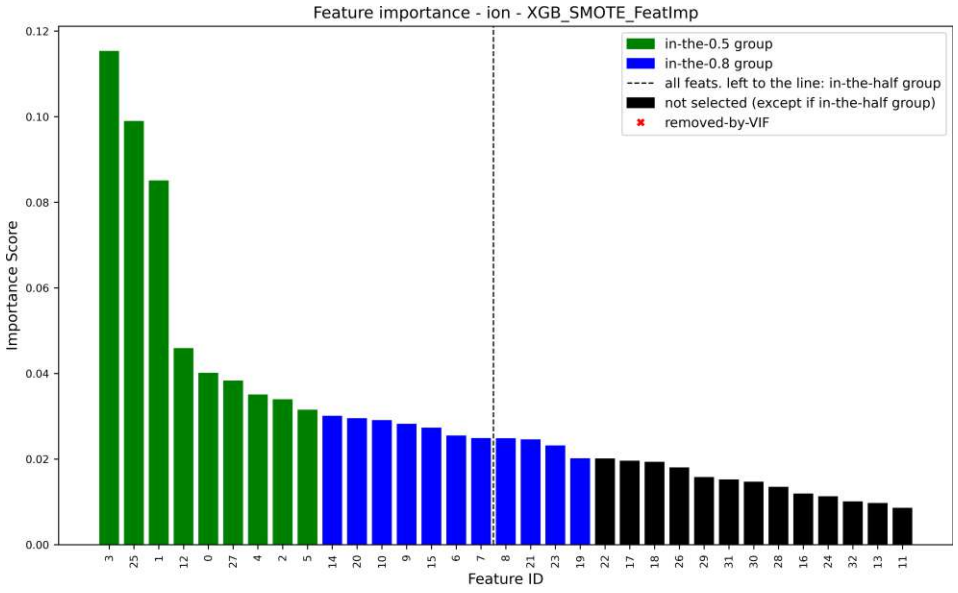

Figure 4.11: Feature importance scores by XGB_SMOTE_FeatImp on *Ionosphere*. Please note that this feature selection combination does not incorporate VIF feature selection and therefore the graph does not show any VIF-removed features.

feat. sel. comb.	mean coeff. of variation ↑
RF_RUS_PI	3.253
RF_bal_PI	2.977
RF_SMOTE_PI	2.664
XGB_RUS_PI	2.601
XGB_bal_PI	2.578
XGB_SMOTE_PI	2.571
RF_RUS_PI_VIF	2.398
RF_SMOTE_PI_VIF	2.218
XGB_SMOTE_PI_VIF	2.21
RF_bal_PI_VIF	2.188
XGB_bal_PI_VIF	2.111
XGB_RUS_PI_VIF	2.093
XGB_SMOTE_FeatImp_VIF	1.458
XGB_SMOTE_FeatImp	1.43
RF_bal_FeatImp	1.371
XGB_RUS_FeatImp_VIF	1.259
XGB_RUS_FeatImp	1.243
XGB_SMOTE_SHAP	1.188
RF_bal_FeatImp_VIF	1.17
XGB_RUS_SHAP	1.131
RF_bal_SHAP	1.126
XGB_SMOTE_SHAP_VIF	1.069
RF_RUS_SHAP	1.065
RF_RUS_FeatImp	1.057
RF_SMOTE_FeatImp	1.028
XGB_bal_SHAP	0.997
XGB_RUS_SHAP_VIF	0.978
RF_SMOTE_FeatImp_VIF	0.975
RF_SMOTE_SHAP	0.975
RF_bal_SHAP_VIF	0.965
XGB_bal_FeatImp	0.947
RF_SMOTE_SHAP_VIF	0.928
RF_RUS_SHAP_VIF	0.901
XGB_bal_SHAP_VIF	0.883
XGB_bal_FeatImp_VIF	0.841
RF_RUS_FeatImp_VIF	0.781

Table 4.5: Mean Coefficient of Variation of feature importance scores

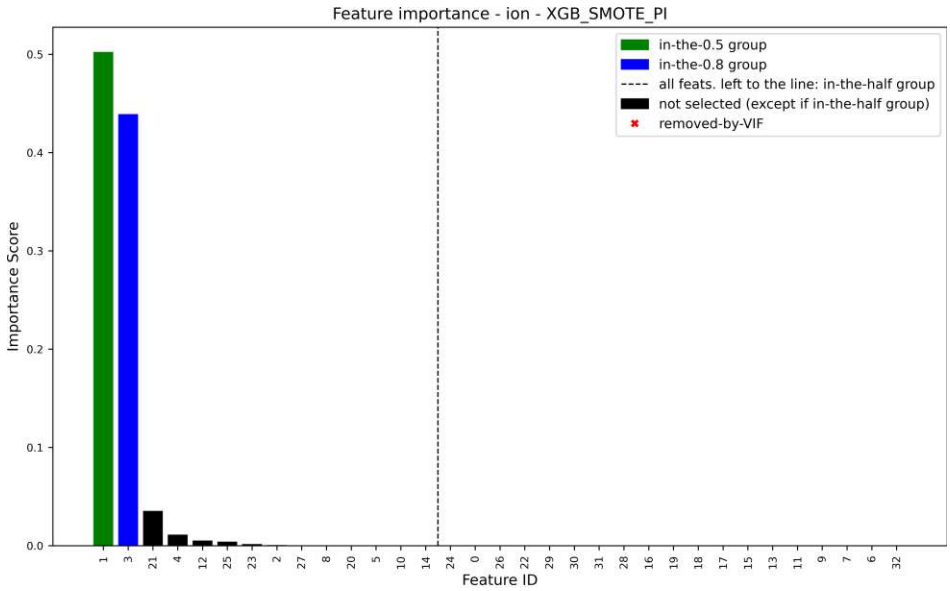

Figure 4.12: Feature importance scores by XGB_SMOTE_PI on *Ionosphere*. In our experiments, we specifically include features in a subset if they hold an importance higher than 0. As a result, in this scenario, the number of features in the half subset is reduced to 14 instead of the initial 16.

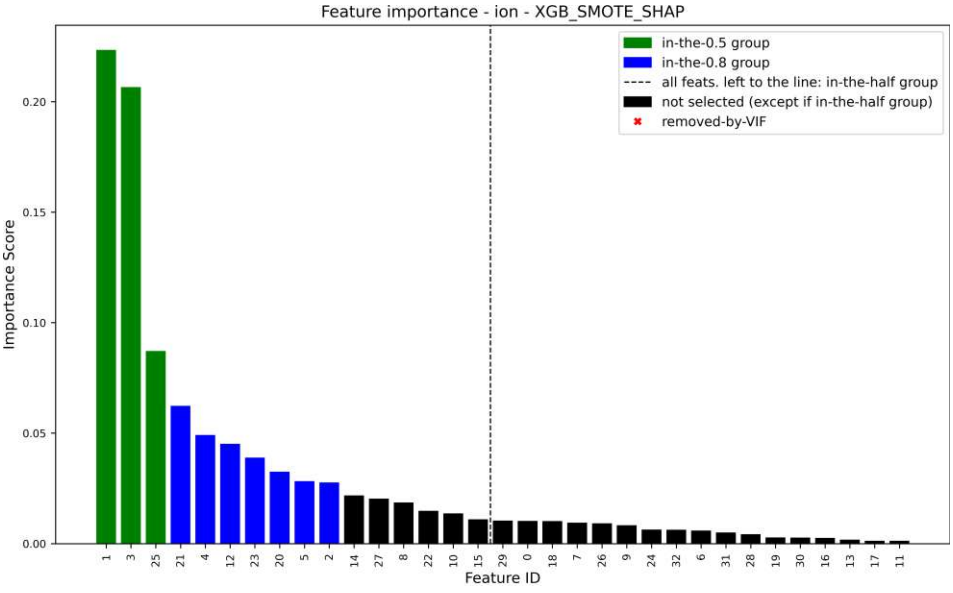

Figure 4.13: Feature importance scores by XGB_SMOTE_SHAP on *Ionosphere*. Please note that this feature selection combination does not incorporate VIF feature selection and therefore the graph does not show any VIF-removed features.

4.3 Implications of Various Factors on Feature Selection

At this point, we will investigate the individual impacts of the core classifier, sampling techniques, VIF correction, and feature evaluation methods - SHAP, PI, and MDI importance. Our focus is to analyze how the introduction of these factors influences ROC AUC performance and stability of feature scores.

feat. sel. comb.		mean corr. coeff.				Coeff. of Var.	
		Pearson		Spearman			
		RF	XGB	RF	XGB	RF	XGB
RF_RUS_Featlmp	XGB_RUS_Featlmp	**0.85**	0.83	0.79	**0.84**	1.06	**1.24**
RF_RUS_Featlmp_VIF	XGB_RUS_Featlmp_VIF	0.78	**0.81**	0.73	**0.78**	0.78	**1.26**
RF_RUS_PI	XGB_RUS_PI	**0.93**	0.8	**0.96**	0.82	**3.25**	2.6
RF_RUS_PI_VIF	XGB_RUS_PI_VIF	**0.94**	0.62	**0.97**	0.62	**2.4**	2.09
RF_RUS_SHAP	XGB_RUS_SHAP	**0.89**	0.83	0.84	**0.86**	1.06	**1.13**
RF_RUS_SHAP_VIF	XGB_RUS_SHAP_VIF	**0.75**	0.72	**0.74**	0.71	0.9	**0.98**
RF_SMOTE_Featlmp	XGB_SMOTE_Featlmp	0.64	**0.85**	0.57	**0.79**	1.03	**1.43**
RF_SMOTE_Featlmp_VIF	XGB_SMOTE_Featlmp_VIF	0.71	**0.74**	0.6	**0.68**	0.98	**1.46**
RF_SMOTE_PI	XGB_SMOTE_PI	**0.81**	0.76	0.76	**0.79**	**2.66**	2.57
RF_SMOTE_PI_VIF	XGB_SMOTE_PI_VIF	**0.98**	0.74	**0.97**	0.68	**2.22**	2.21
RF_SMOTE_SHAP	XGB_SMOTE_SHAP	0.62	**0.71**	0.55	**0.66**	0.98	**1.19**
RF_SMOTE_SHAP_VIF	XGB_SMOTE_SHAP_VIF	**0.62**	0.49	**0.59**	0.48	0.93	**1.07**
RF_bal_Featlmp	XGB_bal_Featlmp	0.6	**0.88**	0.59	**0.82**	**1.37**	0.95
RF_bal_Featlmp_VIF	XGB_bal_FeatImp_VIF	0.63	**0.66**	0.61	**0.7**	**1.17**	0.84
RF_bal_PI	XGB_bal_PI	0.73	**0.84**	0.74	**0.82**	**2.98**	2.58
RF_bal_PI_VIF	XGB_bal_PI_VIF	0.72	**0.83**	0.76	**0.85**	**2.19**	2.11
RF_bal_SHAP	XGB_bal_SHAP	0.53	**0.86**	0.52	**0.83**	**1.13**	1.0
RF_bal_SHAP_VIF	XGB_bal_SHAP_VIF	0.77	**0.82**	0.76	**0.85**	**0.96**	0.88

Table 4.6: Comparison of XGB and RF combinations using correlation and variation coefficients. The larger of the two scores is marked.

4.3.1 Ensemble Model

We summarize results from sections 4.1 and 4.2 into Tables 4.6 and 4.8 for an easier comparison between RF and XGB. To ensure a fair assessment, we analyze each combination side by side by changing only the core classifier as the part that we are interested in.

We observe in Table 4.8 a potential advantage of RF over XGB when combined with the RUS. However, in combination with SMOTE and particularly for cost-sensitive versions, XGB demonstrates superior performance. It is a similar trend if we observe the stability of these combinations using correlation coefficients in Table 4.6. Therefore, RF is more stable than XGB in union with the RUS technique, while this does not hold for the cost-sensitive versions, as XGB is clearly more reliable.

In terms of feature discrimination, a significant difference exists only for the cost-sensitive versions, where the RF_bal is more discriminant than XGB_bal.

4.3.2 Balancing Technique

As previously discussed, XGB consistently performs best with SMOTE and cost-sensitive versions, whereas RF is more effective when combined with the RUS sampling technique

feat. sel. comb.		mean corr. coeff.				Coeff. of Var.	
		Pearson		Spearman			
		VIF	NO	VIF	NO	VIF	NO
RF_RUS_FeatImp_VIF	RF_RUS_FeatImp	0.78	**0.85**	0.73	**0.79**	0.78	**1.06**
RF_RUS_PI_VIF	RF_RUS_PI	**0.94**	0.93	**0.97**	0.96	2.4	**3.25**
RF_RUS_SHAP_VIF	RF_RUS_SHAP	0.75	**0.89**	0.74	**0.84**	0.9	**1.06**
RF_SMOTE_Featlmp_VIF	RF_SMOTE_FeatImp	**0.71**	0.64	**0.6**	0.57	0.98	**1.03**
RF_SMOTE_PI_VIF	RF_SMOTE_PI	**0.98**	0.81	**0.97**	0.76	2.22	**2.66**
RF_SMOTE SHAP_VIF	RF_SMOTE_SHAP	0.62	0.62	**0.59**	0.55	0.93	**0.98**
RF_bal_FeatImp_VIF	RF_bal_FeatImp	**0.63**	0.6	**0.61**	0.59	1.17	**1.37**
RF_bal_PI_VIF	RF_bal_PI	0.72	**0.73**	**0.76**	0.74	2.19	**2.98**
RF_bal_SHAP_VIF	RF_bal_SHAP	**0.77**	0.53	**0.76**	0.52	0.96	**1.13**
XGB_RUS_FeatImp_VIF	XGB_RUS_FeatImp	0.81	**0.83**	0.78	**0.84**	**1.26**	1.24
XGB_RUS_PI_VIF	XGB_RUS_PI	0.62	**0.8**	0.62	**0.82**	2.09	**2.6**
XGB_RUS_SHAP_VIF	XGB_RUS_SHAP	0.72	**0.83**	0.71	**0.86**	0.98	**1.13**
XGB_SMOTE_Featlmp_VIF	XGB_SMOTE_Featlmp	0.74	**0.85**	0.68	**0.79**	**1.46**	1.43
XGB_SMOTE_PI_VIF	XGB_SMOTE_PI	0.74	**0.76**	0.68	**0.79**	2.21	**2.57**
XGB_SMOTE_SHAP_VIF	XGB_SMOTE_SHAP	0.49	**0.71**	0.48	**0.66**	1.07	**1.19**
XGB_bal_FeatImp_VIF	XGB_bal_Featlmp	0.66	**0.88**	0.7	**0.82**	0.84	**0.95**
XGB_bal_PI_VIF	XGB_bal_Pl	0.83	**0.84**	**0.85**	0.82	2.11	**2.58**
XGB_bal_SHAP_VIF	XGB_bal_SHAP	0.82	**0.86**	**0.85**	0.83	0.88	**1.0**

Table 4.7: Comparison of combinations with and without VIF using correlation and variation coefficients

(Tables 4.8 and 4.6).

4.3.3 Variance Inflation Factor (VIF)

Tables 4.9 and 4.7 show that the implemented VIF method does not consistently lead to better ROC AUC scores nor higher stability in feature scores. According to the coefficient of variation, the combinations without VIF seem to be more discriminant. The only combination that benefits from VIF in all segments is RF_SMOTE_PI_VIF. Nevertheless, the two best combinations involve VIF corrections and demonstrate high stability.

Since the incorporation of VIF-based removal into the tested combinations did not improve performance, there is a question of whether the features removed by VIF-based combinations are also eliminated by the corresponding non-VIF combinations, resulting in equal (or nearly equal) feature subsets. For instance, does XGB_SMOTE_SHAP select the equal (or nearly equal) feature subsets as its VIF-based version - XGB_SMOTE_SHAP_VIF? To address this question, we use a process similar to the one presented in Figure 4.6. However, this time, we do not compute Jaccard similarity coefficients between all possible pairs, but only between VIF combinations and their corresponding non-VIF versions (e.g., XGB_SMOTE_SHAP vs XGB_SMOTE_SHAP_VIF, RF_SMOTE_SHAP vs RF_SMOTE_SHAP_VIF and so forth). Consequently, rather than evaluating 630 pairs, we compute the similarity coefficients for only 18 and take the mean value. This analysis is conducted across all datasets and results are presented in Figure 4.14.

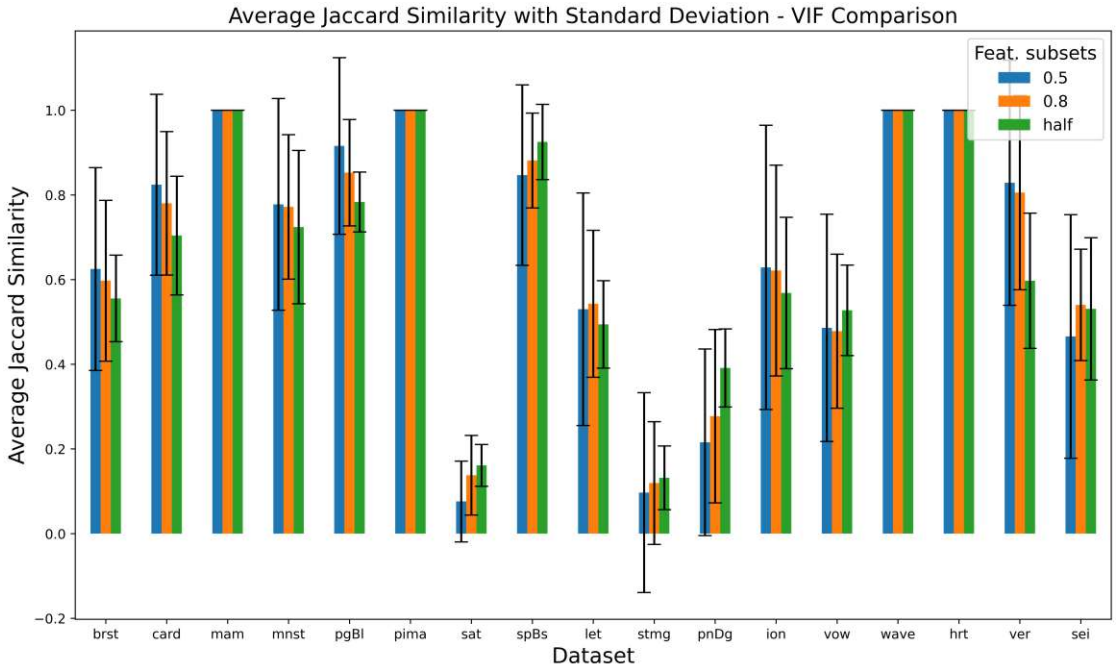

Figure 4.14: Average Jaccard similarity among the selected feature subsets by different feature selection combinations for different datasets - VIF comparison. Similarity coefficients are computed pairwise between feature subsets selected by VIF and corresponding non-VIF feature selection combinations.

It is evident that similarity is now more pronounced compared to the scenario where all possible combinations were considered. We observe perfect similarity (selected feature subsets are identical for all 18 pairs) for *Mammography*, *Pima*, *Waveform*, and *Heartdisease* datasets. However, these results are not relevant since these datasets had no multicollinear features, and therefore VIF selection could not remove any of them. For other datasets, results differ. Notably, there is a high similarity (over 0.7) for datasets *Cardiotocography*, *Mnist*, *PageBlocks*, *SpamBase*, and *Vertebral*. On the other hand, we observe moderate similarity (between 0.5 and 0.7) for *BreastW*, *Ionosphere*, and *Seismic*. However, *Vowels*, *Satellite*, *Satimage-2*, and *Pendigits* datasets demonstrate lower similarity (below 0.5) between the selected subsets. Based on these observations, we cannot claim that VIF and non-VIF combinations consistently select equal or nearly equal feature subsets, at least in our experiments.

Considering that VIF-based combinations did not consistently lead to performance improvements or more stable feature coefficients, it is not necessary to explicitly address the multicollinear features when using the tested combinations (at least for the investigated datasets). Explicit removal of multicollinear features may discard some useful information and therefore our recommendation is to allow the models alone to handle multicollinearity.

4.3.4 Feature Scoring Technique

From our previous sections, it is clear that XGB performs the best with SHAP, closely followed by MDI importance for 50% and 80% datasets. The same trend holds for the RF algorithm. However, we notice some deviation from this pattern in half subsets, where PI can also have strong results. Finally, PI, followed by SHAP is a more stable choice for RF combinations. If we consider the discriminatory power of tested combinations, PI is the most discriminant feature attribution method.

		mean scores - 50% subsets				mean scores - 80% subsets				mean scores - half subsets			
		ROC AUC		ranking		ROC AUC		ranking		ROC AUC		ranking	
feat. sel. comb.		RF	XGB	RF	XGB	RF	XGB	RF	XGB	RF	XGB	RF	XGB
RF_RUS_Featlmp	XGB_RUS_Featlmp	**0.84**	0.83	**12.24**	14.59	**0.89**	0.87	**10.21**	13.85	**0.88**	0.86	**11.0**	18.41
RF_RUS_Featlmp_VIF	XGB_RUS_Featlmp_VIF	**0.85**	0.81	**9.68**	16.82	**0.88**	0.86	**11.5**	16.71	0.87	0.87	**13.35**	20.68
RF_RUS_PI	XGB_RUS_PI	0.77	**0.79**	26.24	**22.0**	0.8	**0.83**	27.68	**21.74**	**0.87**	0.86	**14.91**	15.71
RF_RUS_PI_VIF	XGB_RUS_PI_VIF	0.77	**0.79**	26.09	**20.94**	**0.81**	0.83	25.59	**23.56**	**0.87**	0.86	**15.18**	18.79
RF_RUS_SHAP	XGB_RUS_SHAP	0.84	0.84	11.38	**11.09**	**0.89**	0.88	**6.71**	11.68	**0.88**	0.87	**10.26**	13.94
RF_RUS_SHAP_VIF	XGB_RUS_SHAP_VIF	0.84	0.84	13.38	**12.62**	**0.88**	0.87	**10.26**	14.38	0.87	0.87	**14.76**	17.47
RF_SMOTE_Featlmp	XGB_SMOTE_Featlmp	0.82	**0.84**	13.41	**12.53**	0.85	**0.88**	15.79	**11.71**	0.85	**0.88**	19.29	**12.59**
RF_SMOTE_Featlmp_VIF	XGB_SMOTE_Featlmp_VIF	0.81	0.81	17.38	**17.24**	0.85	**0.86**	**16.32**	17.88	0.85	**0.86**	**21.71**	22.53
RF_SMOTE_PI	XGB_SMOTE_PI	0.73	**0.78**	29.12	**22.47**	0.79	**0.84**	28.21	**22.5**	0.84	**0.88**	19.56	**10.18**
RF_SMOTE_PI_VIF	XGB_SMOTE_PI_VIF	0.73	**0.78**	30.44	**23.21**	0.82	0.82	25.85	**24.85**	0.85	**0.87**	21.15	**19.5**
RF_SMOTE_SHAP	XGB_SMOTE_SHAP	0.82	**0.85**	13.35	**7.62**	0.85	**0.88**	15.15	**10.24**	0.84	**0.88**	19.38	**10.41**
RF_SMOTE_SHAP_VIF	XGB_SMOTE_SHAP_VIF	0.82	**0.84**	16.38	**11.0**	0.85	**0.87**	17.29	**15.79**	0.85	**0.87**	21.47	**19.0**
RF_bal_Featlmp	XGB_bal_Featlmp	0.77	**0.83**	20.82	**13.35**	0.8	**0.88**	23.5	**9.65**	0.8	**0.87**	25.94	**14.41**
RF_bal_Featlmp_VIF	XGB_bal_FeatImp_VIF	0.74	**0.8**	24.79	**19.15**	0.8	**0.87**	25.44	**13.65**	0.79	**0.86**	32.15	**20.24**
RF_bal_PI	XGB_bal_PI	**0.72**	0.7	30.38	**26.59**	0.76	**0.81**	30.79	**26.79**	0.81	**0.88**	26.82	**12.94**
RF_bal_PI_VIF	XGB_bal_PI_VIF	0.71	**0.72**	30.18	**26.06**	0.76	**0.83**	30.15	**24.09**	0.8	**0.86**	26.12	**19.91**
RF_bal_SHAP	XGB_bal_SHAP	0.79	**0.85**	17.91	**11.15**	0.8	**0.89**	21.97	**9.35**	0.8	**0.88**	26.29	**12.76**
RF_bal_SHAP_VIF	XGB_bal_SHAP_VIF	0.75	**0.85**	23.15	**11.26**	0.8	**0.88**	25.53	**9.65**	0.79	**0.87**	27.26	**19.91**

Table 4.8: Comparison of XGB and RF combinations. Mean ROC AUC and ranking scores for 50%, 80%, half subsets. The better of the two scores is marked.

feat. sel. comb.	mean scores - 50% subsets				mean scores - 80% subsets				mean scores - half subsets			
	ROC AUC		ranking		ROC AUC		ranking		ROC AUC		ranking	
	VIF	NO	VIF	NO	VIF	NO	VIF	NO	VIF	NO	VIF	NO
RF_RUS_FeatImp_VIF	**0.85**	0.84	**9.68**	12.24	0.88	**0.89**	11.5	**10.21**	0.87	**0.88**	13.35	**11.0**
RF_RUS_PI_VIF	0.77	0.77	**26.09**	26.24	**0.81**	0.8	**25.59**	27.68	0.87	0.87	15.18	**14.91**
RF_RUS_SHAP_VIF	0.84	0.84	13.38	**11.38**	0.88	**0.89**	10.26	**6.71**	0.87	**0.88**	14.76	**10.26**
RF_SMOTE_FeatImp_VIF	0.81	**0.82**	17.38	**13.41**	0.85	0.85	16.32	**15.79**	0.85	0.85	21.71	**19.29**
RF_SMOTE_PI_VIF	0.73	0.73	30.44	**29.12**	**0.82**	0.79	**25.85**	28.21	**0.85**	0.84	21.15	**19.56**
RF_SMOTE_SHAP_VIF	0.82	0.82	16.38	**13.35**	0.85	0.85	17.29	**15.15**	**0.85**	0.84	21.47	**19.38**
RF_bal_FeatImp_VIF	0.74	**0.77**	24.79	**20.82**	0.8	0.8	25.44	**23.5**	0.79	**0.8**	32.15	**25.94**
RF_bal_PI_VIF	0.71	**0.72**	**30.18**	30.38	0.76	0.76	**30.15**	30.79	0.8	**0.81**	**26.12**	26.82
RF_bal_SHAP_VIF	0.75	**0.79**	23.15	**17.91**	0.8	0.8	25.53	**21.97**	0.79	**0.8**	27.26	**26.29**
XGB_RUS_FeatImp_VIF	0.81	**0.83**	16.82	**14.59**	0.86	**0.87**	16.71	**13.85**	**0.87**	0.86	20.68	**18.41**
XGB_RUS_PI_VIF	0.79	0.79	**20.94**	22.0	0.83	0.83	23.56	**21.74**	0.86	0.86	18.79	**15.71**
XGB_RUS_SHAP_VIF	0.84	0.84	12.62	**11.09**	0.87	**0.88**	14.38	**11.68**	0.87	0.87	17.47	**13.94**
XGB_SMOTE_FeatImp_VIF	0.81	**0.84**	17.24	**12.53**	0.86	**0.88**	17.88	**11.71**	0.86	**0.88**	22.53	**12.59**
XGB_SMOTE_PI_VIF	0.78	0.78	23.21	**22.47**	0.82	**0.84**	24.85	**22.5**	0.87	**0.88**	19.5	**10.18**
XGB_SMOTE_SHAP_VIF	0.84	**0.85**	11.0	**7.62**	0.87	**0.88**	15.79	**10.24**	0.87	**0.88**	19.0	**10.41**
XGB_bal_FeatImp_VIF	0.8	**0.83**	19.15	**13.35**	0.87	**0.88**	13.65	**9.65**	0.86	**0.87**	20.24	**14.41**
XGB_bal_PI_VIF	**0.72**	0.7	**26.06**	26.59	**0.83**	0.81	**24.09**	26.79	0.86	**0.88**	19.91	**12.94**
XGB_bal_SHAP_VIF	0.85	0.85	11.26	**11.15**	0.88	**0.89**	9.65	**9.35**	0.87	**0.88**	19.91	**12.76**

Table 4.9: Comparison of combinations with and without VIF. Mean ROC AUC and ranking scores for 50%, 80%, half subsets

CHAPTER 5

Conclusion

5.1 Conclusions

In this work, we tested various feature selection combinations to address the problem of feature selection in imbalanced datasets for binary classification. To accomplish this, we used tree ensembles (RF and XGB) combined with data balancing techniques (RUS, SMOTE, cost-sensitive learning) and feature attribution methods (MDI, PI, and SHAP). Our experiments include 36 distinct feature selection combinations, which were evaluated on 17 imbalanced datasets. We used ROC AUC scores - a metric widely used in literature for the imbalanced data, to evaluate the performance of tested combinations. We also conducted a stability analysis where we compared the performance of three selected feature subsets (best half features, 50% subset, and 80% subset) relative to their cumulative feature importance. Lastly, we explored the discriminatory power of the employed feature selection combinations and the impact of multicollinear features using VIF.

The following conclusions emerged from the conducted experiments:

- XGB and RF in combination with a data balancing technique are promising methods for an effective feature selection across diverse imbalanced datasets. The combinations XGB_SMOTE_SHAP, XGB_bal_SHAP and RF_RUS_SHAP consistently demonstrate a strong ROC AUC performance across all evaluated feature subsets.
- RF_SMOTE_PI_VIF stands out as the most stable combination, manifesting the strongest correlation between feature importance scores and ROC AUC performance. Other stable RF combinations are RF_RUS_PI_VIF, RF_RUS_PI, and RF_RUS_SHAP. On the other hand, the most reliable XGB combinations are XGB_RUS_FeatImp, XGB_bal_FeatImp, and XGB_SMOTE_PI. Combining RUS or SMOTE with PI results in stable feature importance scores.

- RF_RUS_PI is the most powerful combination in distinguishing between highly and less important features. Similarly, the most discriminant XGB method is XGB_RUS_PI. In general, PI combinations are more discriminant compared to those relying on SHAP and MDI feature importance.
- The RF classifier leads to better performance and is more stable than XGB, when combined with the RUS technique. On the other hand, the cost-sensitive version of XGB, along with the combination involving SMOTE, shows better performance and stability compared to RF combinations. We do not advise RF_bal for the feature selection. Therefore, the selection of a balancing technique depends strongly on the core classifier.
- The VIF measure does not consistently improve ROC AUC performance, nor feature scoring stability. On the contrary, most of the combinations show better results without the VIF. The only combination that benefits from VIF in all segments is RF_SMOTE_PI_VIF.
- PI is the most discriminant feature attribution method. SHAP and MDI feature importance display smooth feature scoring. Regarding stability, PI and SHAP are the best choices for RF. On the other hand, MDI feature importance can also show strong reliability when used with XGB. Every feature scoring technique can yield high ROC AUC scores depending on the used combination and subset selection criteria. We do not advise 50% and 80% cut-off criteria for PI combinations, because of poor performance.

In our experiments, we used a variety of different data sets and feature selections. However, all statements provided here can be only made about the analyzed data sets with the feature selections and may be different if data sets or features differ.

We summarize our observations, suggestions, and recommendations considering the initially stated research questions and goals:

- **R1: Accuracy**

  XGB_SMOTE_SHAP, XGB_bal_SHAP, and RF_RUS_SHAP are the preferred and most prominent combinations.

- **R2: Stability**

  To maximize stability and performance, RF should be used with RUS, while XGB should be used in its cost-sensitive version (XGB_bal) together with SMOTE.

- **R3: Discriminant power**

  PI shows the most discriminant scores, i.e., it is the approach that most clearly separates the most important features from the less relevant ones.

- **R4: Particular impacts**

- Both RF and XGB can perform equally well, depending on the other elements employed in the feature selection pipeline.
- All data-balancing techniques can perform equally well when combined with the right elements in the feature selection pipeline. RUS performs best when combined with RF and SHAP, SMOTE is effective with XGB and SHAP, and the recommendation is to use only the XGB cost-sensitive version - XGB_bal and also with SHAP.
- There are no significant advantages obtained by the use of VIF-based feature selection. It is not necessary to explicitly address multicollinearity when using ensemble trees (RF, XGB) in combination with data balancing techniques (RUS, SMOTE, cost-sensitive learning) and feature attribution methods (MDI, PI, and SHAP) to generate high-performing and stable feature subsets.
- PI shows the most discriminant feature importance scores, while SHAP and MDI feature importance display smooth feature scoring. Every feature scoring technique can yield high ROC AUC scores, depending on the specific combination and subset selection criteria.

5.2 Future Work

In this work we addressed the problem of imbalanced data in binary classification; however, there is still much research to be done on the topic of imbalanced data, for instance, in multiclass classification, which commonly includes several minority and majority classes with strongly divergent cardinalities [156].

Nevertheless, the natural continuance of this work would involve an analysis of the feature selection combinations regarding the characteristics of the used datasets, such as dataset geometry, dimensions, class types and overlapping [157]. Although we attempted to collect a wide range of imbalanced datasets, our results could be more significant by increasing the number of proper imbalanced datasets. Additionally, valuable insights could be gained by testing novel combinations to address the problems related to multicollinearity and imbalanced data. For instance, many extensions of standard SMOTE oversampling technique could be examined and compared to our results: Borderline-SMOTE, Adjusting the Direction of the Synthetic Minority ClasS Examples (ADOMS), ADASYN, Safe-Level SMOTE, Density-Based SMOTE (DBSMOTE), and many others. In our work, we used the non-heuristic RUS technique, however, there are many other heuristic techniques such as Tomek Links (TL), CNN, OSS, Class Purity Maximization (CPM), and even more advanced undersampling techniques like Evolutionary Undersampling, ACOSampling, Cluster-Based Evolutionary Undersampling (CBEUS), and others [102]. An extensive comparison of various over- and undersampling techniques is conducted in [158, 159, 160]. The class imbalance problem could be also addressed by combining over- and undersampling techniques, as Estabrooks et al. [161] and Barandela et al. [162] suggested.

Moreover, an ensemble approach can be employed to select the most important features. An interesting approach is proposed in [163], where the imbalanced binary problem is broken into multiple balanced binary problems without altering the original data distribution. The major instances are divided into several data non-overlapping subsets, with minor instances added to each dataset to achieve a ratio of 1:1. Subsequently, multiple classifiers are used to learn from these diverse balanced datasets and perform feature scoring. Finally, the feature coefficients obtained from multiple models for a single feature can be merged into one score in an ensemble manner. Another similar technique is EM1vs1 [164], where the imbalanced binary dataset is transformed into a balanced multi-class problem. Additionally, alternative measures such as the Condition Index – a function of eigenvalues (λ) that represent the variance of the linear combination of variables [165] – can be used to detect multicollinearity.

Furthermore, it may be beneficial to investigate the impact of stability measures on the tested feature selection combinations proposed in [166]. It would be interesting to explore the potential impact of stability selection proposed by Meinshausen et al. [16], where the training data is divided into smaller subsamples and only the consistently important features across all subsamples are included in the final feature subset. The question remains whether the outcomes obtained through this method would be different from ours.

CHAPTER 6

Appendix

6.1 ROC AUC Performance of Feature Selection Combinations

ROC AUC scores achieved by the core classifiers of each feature selection combination, on the full set of features and the selected 50%, 80%, and half feature subsets are shown in the Tables 6.1 - 6.17.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.974	0.608	0.952	0.949	0.868	0.994	0.96	0.91	1.0	0.967
RF_RUS_PI	RF	1.0	0.974	0.849	0.922	0.896	0.849	0.922	0.896	0.992	1.0	0.967
RF_RUS_SHAP	RF	1.0	0.974	0.679	0.994	0.96	0.827	1.0	0.967	0.827	1.0	0.967
RF_SMOTE_FeatImp	RF	1.0	0.974	0.654	0.995	0.96	0.808	0.997	0.964	0.808	0.997	0.964
RF_SMOTE_PI	RF	1.0	0.974	0.705	0.942	0.859	0.826	0.981	0.942	0.958	1.0	0.964
RF_SMOTE_SHAP	RF	1.0	0.974	0.598	0.995	0.967	0.818	1.0	0.967	0.723	1.0	0.967
RF_bal_FeatImp	RF	1.0	0.974	0.64	0.995	0.953	0.859	1.0	0.974	0.786	0.995	0.964
RF_bal_PI	RF	1.0	0.974	0.755	0.915	0.896	0.86	0.975	0.942	0.989	0.998	0.96
RF_bal_SHAP	RF	1.0	0.974	0.635	0.994	0.957	0.819	1.0	0.96	0.749	0.998	0.96
RF_RUS_FeatImp_VIF	RF	1.0	0.974	0.751	0.973	0.952	0.833	0.997	0.97	0.89	1.0	0.97
RF_RUS_PI_VIF	RF	1.0	0.974	0.607	0.922	0.896	0.879	0.973	0.952	0.996	1.0	0.967
RF_RUS_SHAP_VIF	RF	1.0	0.974	0.611	0.973	0.952	0.851	1.0	0.967	0.851	1.0	0.967
RF_SMOTE_FeatImp_VIF	RF	1.0	0.967	0.776	0.981	0.943	0.866	0.997	0.964	0.911	1.0	0.964
RF_SMOTE_PI_VIF	RF	1.0	0.967	0.911	0.981	0.946	0.911	0.981	0.946	0.995	1.0	0.957
RF_SMOTE_SHAP_VIF	RF	1.0	0.967	0.572	0.981	0.943	0.811	1.0	0.964	0.811	1.0	0.964
RF_bal_FeatImp_VIF	RF	1.0	0.974	0.752	0.975	0.942	0.841	0.997	0.964	0.894	1.0	0.957
RF_bal_PI_VIF	RF	1.0	0.974	0.895	0.977	0.942	0.895	0.977	0.942	0.994	1.0	0.967
RF_bal_SHAP_VIF	RF	1.0	0.974	0.611	0.975	0.942	0.834	1.0	0.967	0.834	1.0	0.967
XGB_RUS_FeatImp	XGB	0.991	0.974	0.637	0.952	0.949	0.841	0.979	0.97	0.841	0.979	0.97
XGB_RUS_PI	XGB	0.991	0.974	0.672	0.922	0.896	0.894	0.979	0.97	0.948	0.979	0.974
XGB_RUS_SHAP	XGB	0.991	0.974	0.51	0.973	0.964	0.829	0.979	0.974	0.829	0.979	0.974
XGB_SMOTE_FeatImp	XGB	0.984	0.974	0.513	0.961	0.935	0.864	0.979	0.964	0.864	0.979	0.964
XGB_SMOTE_PI	XGB	0.984	0.974	0.644	0.939	0.859	0.905	0.977	0.964	0.948	0.976	0.974
XGB_SMOTE_SHAP	XGB	0.984	0.974	0.514	0.973	0.946	0.847	0.981	0.964	0.765	0.979	0.964
XGB_bal_FeatImp	XGB	0.986	0.974	0.636	0.933	0.91	0.854	0.984	0.97	0.791	0.976	0.952
XGB_bal_PI	XGB	0.986	0.974	0.768	0.961	0.952	0.871	0.977	0.959	0.933	0.984	0.963
XGB_bal_SHAP	XGB	0.986	0.974	0.536	0.961	0.952	0.804	0.984	0.963	0.804	0.984	0.963
XGB_RUS_FeatImp_VIF	XGB	1.0	0.978	0.575	0.928	0.911	0.875	0.991	0.953	0.875	0.991	0.953
XGB_RUS_PI_VIF	XGB	1.0	0.978	0.722	0.964	0.953	0.911	0.988	0.953	0.972	0.994	0.95
XGB_RUS_SHAP_VIF	XGB	1.0	0.978	0.66	0.988	0.953	0.818	0.994	0.95	0.818	0.994	0.95
XGB_SMOTE_FeatImp_VIF	XGB	0.997	0.974	0.64	0.945	0.89	0.862	0.989	0.964	0.905	0.995	0.957
XGB_SMOTE_PI_VIF	XGB	0.997	0.974	0.533	0.937	0.859	0.899	0.977	0.953	0.991	0.994	0.96
XGB_SMOTE_SHAP_VIF	XGB	0.997	0.974	0.579	0.977	0.953	0.828	0.994	0.96	0.828	0.994	0.96
XGB_bal_FeatImp_VIF	XGB	1.0	0.978	0.639	0.953	0.911	0.872	0.984	0.952	0.872	0.984	0.952
XGB_bal_PI_VIF	XGB	1.0	0.978	0.575	0.5	0.5	0.836	0.977	0.949	0.995	0.997	0.96
XGB_bal_SHAP_VIF	XGB	1.0	0.978	0.505	0.977	0.949	0.863	1.0	0.974	0.784	0.997	0.96

Table 6.1: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *BreastW*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.941	0.526	0.998	0.838	0.816	1.0	0.92	0.816	1.0	0.92
RF_RUS_PI	RF	1.0	0.941	0.811	0.888	0.828	0.811	0.888	0.828	0.998	1.0	0.934
RF_RUS_SHAP	RF	1.0	0.941	0.573	0.998	0.841	0.812	1.0	0.93	0.871	1.0	0.924
RF_SMOTE_FeatImp	RF	1.0	0.94	0.542	0.941	0.851	0.816	1.0	0.919	0.877	1.0	0.922
RF_SMOTE_PI	RF	1.0	0.94	0.841	0.922	0.824	0.841	0.922	0.824	1.0	1.0	0.933
RF_SMOTE_SHAP	RF	1.0	0.94	0.562	0.978	0.843	0.818	1.0	0.935	0.876	1.0	0.936
RF_bal_FeatImp	RF	1.0	0.935	0.52	0.865	0.837	0.819	1.0	0.92	0.874	1.0	0.928
RF_bal_PI	RF	1.0	0.935	0.501	0.773	0.766	0.923	0.865	0.837	0.999	1.0	0.923
RF_bal_SHAP	RF	1.0	0.935	0.517	0.957	0.822	0.827	1.0	0.923	0.885	1.0	0.934
RF_RUS_FeatImp_VIF	RF	1.0	0.943	0.57	0.998	0.903	0.845	1.0	0.909	0.797	1.0	0.924
RF_RUS_PI_VIF	RF	1.0	0.943	0.584	0.771	0.766	0.813	0.888	0.824	0.994	1.0	0.93
RF_RUS_SHAP_VIF	RF	1.0	0.943	0.579	0.998	0.837	0.818	1.0	0.928	0.857	1.0	0.921
RF_SMOTE_FeatImp_VIF	RF	1.0	0.936	0.5	0.923	0.827	0.837	1.0	0.907	0.902	1.0	0.923
RF_SMOTE_PI_VIF	RF	1.0	0.936	0.758	0.923	0.827	0.834	0.941	0.85	0.999	1.0	0.923
RF_SMOTE_SHAP_VIF	RF	1.0	0.936	0.598	0.978	0.844	0.821	1.0	0.92	0.901	1.0	0.924
RF_bal_FeatImp_VIF	RF	1.0	0.935	0.523	0.865	0.837	0.835	1.0	0.908	0.898	1.0	0.913
RF_bal_PI_VIF	RF	1.0	0.935	0.502	0.773	0.766	0.877	0.865	0.837	0.997	1.0	0.931
RF_bal_SHAP_VIF	RF	1.0	0.935	0.537	0.977	0.881	0.825	1.0	0.907	0.91	1.0	0.931
XGB_RUS_FeatImp	XGB	1.0	0.933	0.527	0.919	0.842	0.827	1.0	0.915	0.827	1.0	0.915
XGB_RUS_PI	XGB	1.0	0.933	0.578	0.771	0.766	0.934	0.882	0.808	1.0	1.0	0.921
XGB_RUS_SHAP	XGB	1.0	0.933	0.541	0.994	0.82	0.808	1.0	0.918	0.808	1.0	0.918
XGB_SMOTE_FeatImp	XGB	0.997	0.948	0.535	0.929	0.859	0.809	0.992	0.925	0.856	0.993	0.921
XGB_SMOTE_PI	XGB	0.997	0.948	0.826	0.91	0.832	0.826	0.91	0.832	0.991	0.993	0.925
XGB_SMOTE_SHAP	XGB	0.997	0.948	0.51	0.965	0.873	0.801	0.993	0.925	0.801	0.993	0.925
XGB_bal_FeatImp	XGB	1.0	0.944	0.526	0.992	0.895	0.813	0.998	0.94	0.744	0.997	0.924
XGB_bal_PI	XGB	1.0	0.944	0.512	0.623	0.622	0.968	0.856	0.817	0.999	0.999	0.924
XGB_bal_SHAP	XGB	1.0	0.944	0.539	0.991	0.891	0.82	1.0	0.934	0.745	0.999	0.93
XGB_RUS_FeatImp_VIF	XGB	1.0	0.931	0.588	0.919	0.844	0.8	0.998	0.91	0.832	0.998	0.919
XGB_RUS_PI_VIF	XGB	1.0	0.931	0.527	0.771	0.766	0.858	0.88	0.816	0.998	0.998	0.915
XGB_RUS_SHAP_VIF	XGB	1.0	0.931	0.55	0.991	0.821	0.837	0.998	0.915	0.837	0.998	0.915
XGB_SMOTE_FeatImp_VIF	XGB	0.994	0.938	0.553	0.931	0.847	0.805	0.989	0.908	0.805	0.989	0.908
XGB_SMOTE_PI_VIF	XGB	0.994	0.938	0.731	0.912	0.832	0.839	0.931	0.847	0.979	0.992	0.919
XGB_SMOTE_SHAP_VIF	XGB	0.994	0.938	0.504	0.957	0.86	0.816	0.992	0.919	0.816	0.992	0.919
XGB_bal_FeatImp_VIF	XGB	0.998	0.942	0.54	0.794	0.756	0.81	0.995	0.929	0.81	0.995	0.929
XGB_bal_PI_VIF	XGB	0.998	0.942	0.907	0.854	0.819	0.907	0.854	0.819	0.99	0.996	0.92
XGB_bal_SHAP_VIF	XGB	0.998	0.942	0.566	0.989	0.904	0.824	0.996	0.94	0.783	0.995	0.911

Table 6.2: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Cardiotocography*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.844	0.549	1.0	0.794	0.873	1.0	0.783	0.634	1.0	0.819
RF_RUS_PI	RF	1.0	0.844	0.523	0.756	0.744	0.824	0.774	0.736	0.986	1.0	0.772
RF_RUS_SHAP	RF	1.0	0.844	0.575	0.869	0.775	0.807	1.0	0.831	0.744	1.0	0.761
RF_SMOTE_FeatImp	RF	1.0	0.856	0.531	1.0	0.814	0.848	1.0	0.844	0.616	1.0	0.806
RF_SMOTE_PI	RF	1.0	0.856	0.69	0.79	0.733	0.898	0.81	0.783	0.997	1.0	0.794
RF_SMOTE_SHAP	RF	1.0	0.856	0.576	0.881	0.786	0.807	1.0	0.806	0.746	1.0	0.794
RF_bal_FeatImp	RF	1.0	0.842	0.526	1.0	0.814	0.854	1.0	0.792	0.618	1.0	0.789
RF_bal_PI	RF	1.0	0.842	0.617	0.754	0.794	0.858	0.792	0.783	0.969	1.0	0.808
RF_bal_SHAP	RF	1.0	0.842	0.55	0.86	0.764	0.844	1.0	0.811	0.729	1.0	0.817
RF_RUS_FeatImp_VIF	RF	1.0	0.844	0.549	1.0	0.794	0.873	1.0	0.783	0.634	1.0	0.819
RF_RUS_PI_VIF	RF	1.0	0.844	0.523	0.756	0.744	0.824	0.774	0.736	0.986	1.0	0.772
RF_RUS_SHAP_VIF	RF	1.0	0.844	0.575	0.869	0.775	0.807	1.0	0.831	0.744	1.0	0.761
RF_SMOTE_FeatImp_VIF	RF	1.0	0.856	0.531	1.0	0.814	0.848	1.0	0.844	0.616	1.0	0.806
RF_SMOTE_PI_VIF	RF	1.0	0.856	0.69	0.79	0.733	0.898	0.81	0.783	0.997	1.0	0.794
RF_SMOTE_SHAP_VIF	RF	1.0	0.856	0.576	0.881	0.786	0.807	1.0	0.806	0.746	1.0	0.794
RF_bal_FeatImp_VIF	RF	1.0	0.842	0.526	1.0	0.814	0.854	1.0	0.792	0.618	1.0	0.789
RF_bal_PI_VIF	RF	1.0	0.842	0.617	0.754	0.794	0.858	0.792	0.783	0.969	1.0	0.808
RF_bal_SHAP_VIF	RF	1.0	0.842	0.55	0.86	0.764	0.844	1.0	0.811	0.729	1.0	0.817
XGB_RUS_FeatImp	XGB	0.857	0.856	0.555	0.839	0.814	0.842	0.827	0.831	0.721	0.857	0.808
XGB_RUS_PI	XGB	0.857	0.856	0.557	0.756	0.744	0.868	0.839	0.814	0.956	0.851	0.831
XGB_RUS_SHAP	XGB	0.857	0.856	0.619	0.839	0.814	0.823	0.81	0.819	0.886	0.851	0.831
XGB_SMOTE_FeatImp	XGB	0.886	0.867	0.57	0.852	0.808	0.842	0.852	0.839	0.719	0.852	0.853
XGB_SMOTE_PI	XGB	0.886	0.867	0.567	0.771	0.744	0.89	0.852	0.808	0.947	0.857	0.842
XGB_SMOTE_SHAP	XGB	0.886	0.867	0.609	0.852	0.814	0.82	0.857	0.811	0.877	0.843	0.817
XGB_bal_FeatImp	XGB	0.895	0.811	0.535	0.769	0.747	0.808	0.85	0.822	0.808	0.85	0.822
XGB_bal_PI	XGB	0.895	0.811	0.683	0.832	0.814	0.827	0.831	0.808	0.907	0.879	0.767
XGB_bal_SHAP	XGB	0.895	0.811	0.582	0.832	0.814	0.817	0.892	0.786	0.817	0.892	0.786
XGB_RUS_FeatImp_VIF	XGB	0.857	0.856	0.555	0.839	0.814	0.842	0.827	0.831	0.721	0.857	0.808
XGB_RUS_PI_VIF	XGB	0.857	0.856	0.557	0.756	0.744	0.868	0.839	0.814	0.956	0.851	0.831
XGB_RUS_SHAP_VIF	XGB	0.857	0.856	0.619	0.839	0.814	0.823	0.81	0.819	0.886	0.851	0.831
XGB_SMOTE_FeatImp_VIF	XGB	0.886	0.867	0.57	0.852	0.808	0.842	0.852	0.839	0.719	0.852	0.853
XGB_SMOTE_PI_VIF	XGB	0.886	0.867	0.567	0.771	0.744	0.89	0.852	0.808	0.947	0.857	0.842
XGB_SMOTE_SHAP_VIF	XGB	0.886	0.867	0.609	0.852	0.814	0.82	0.857	0.811	0.877	0.843	0.817
XGB_bal_FeatImp_VIF	XGB	0.895	0.811	0.535	0.769	0.747	0.808	0.85	0.822	0.808	0.85	0.822
XGB_bal_PI_VIF	XGB	0.895	0.811	0.683	0.832	0.814	0.827	0.831	0.808	0.907	0.879	0.767
XGB_bal_SHAP_VIF	XGB	0.895	0.811	0.582	0.832	0.814	0.817	0.892	0.786	0.817	0.892	0.786

Table 6.3: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Heartdisease*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.931	0.521	0.994	0.883	0.81	0.994	0.933	0.751	1.0	0.911
RF_RUS_PI	RF	1.0	0.931	0.765	0.994	0.777	1.0	0.994	0.83	1.0	0.994	0.83
RF_RUS_SHAP	RF	1.0	0.931	0.513	0.994	0.875	0.811	1.0	0.94	0.758	1.0	0.925
RF_SMOTE_FeatImp	RF	1.0	0.914	0.533	1.0	0.901	0.814	1.0	0.927	0.735	1.0	0.914
RF_SMOTE_PI	RF	1.0	0.914	0.616	0.924	0.665	0.836	0.987	0.836	1.0	1.0	0.914
RF_SMOTE_SHAP	RF	1.0	0.914	0.527	1.0	0.893	0.814	1.0	0.901	0.747	1.0	0.901
RF_bal_FeatImp	RF	1.0	0.934	0.612	0.987	0.911	0.811	1.0	0.895	0.887	1.0	0.895
RF_bal_PI	RF	1.0	0.934	0.724	0.873	0.809	0.888	0.981	0.911	1.0	1.0	0.908
RF_bal_SHAP	RF	1.0	0.934	0.534	0.987	0.911	0.816	1.0	0.908	0.868	1.0	0.901
RF_RUS_FeatImp_VIF	RF	1.0	0.925	0.506	1.0	0.892	0.821	1.0	0.903	0.673	1.0	0.912
RF_RUS_PI_VIF	RF	1.0	0.925	0.5	1.0	0.878	0.818	1.0	0.905	0.864	1.0	0.924
RF_RUS_SHAP_VIF	RF	1.0	0.925	0.529	1.0	0.87	0.812	1.0	0.92	0.718	1.0	0.906
RF_SMOTE_FeatImp_VIF	RF	1.0	0.927	0.554	0.987	0.903	0.807	1.0	0.914	0.883	1.0	0.901
RF_SMOTE_PI_VIF	RF	1.0	0.927	0.665	0.885	0.695	0.816	0.946	0.829	1.0	1.0	0.893
RF_SMOTE_SHAP_VIF	RF	1.0	0.927	0.557	0.994	0.861	0.818	1.0	0.921	0.871	1.0	0.901
RF_bal_FeatImp_VIF	RF	1.0	0.934	0.628	0.987	0.903	0.813	1.0	0.908	0.891	1.0	0.882
RF_bal_PI_VIF	RF	1.0	0.934	0.611	0.873	0.809	0.86	0.981	0.911	1.0	1.0	0.908
RF_bal_SHAP_VIF	RF	1.0	0.934	0.531	0.987	0.903	0.814	1.0	0.895	0.864	1.0	0.895
XGB_RUS_FeatImp	XGB	0.994	0.894	0.5	0.96	0.875	0.81	0.994	0.902	0.793	0.994	0.902
XGB_RUS_PI	XGB	0.994	0.894	0.685	0.898	0.757	0.856	0.949	0.874	0.999	0.994	0.887
XGB_RUS_SHAP	XGB	0.994	0.894	0.522	0.966	0.883	0.805	0.994	0.902	0.87	0.994	0.887
XGB_SMOTE_FeatImp	XGB	1.0	0.914	0.524	0.994	0.899	0.812	1.0	0.914	0.719	1.0	0.906
XGB_SMOTE_PI	XGB	1.0	0.914	0.502	0.844	0.659	0.942	0.914	0.808	1.0	1.0	0.901
XGB_SMOTE_SHAP	XGB	1.0	0.914	0.517	0.949	0.883	0.801	0.997	0.914	0.902	1.0	0.921
XGB_bal_FeatImp	XGB	0.997	0.943	0.544	0.975	0.896	0.804	0.99	0.889	0.785	0.99	0.925
XGB_bal_PI	XGB	0.997	0.943	0.56	0.943	0.777	0.989	0.955	0.833	1.0	0.975	0.818
XGB_bal_SHAP	XGB	0.997	0.943	0.512	0.987	0.881	0.802	0.99	0.89	0.802	0.99	0.89
XGB_RUS_FeatImp_VIF	XGB	1.0	0.916	0.564	0.977	0.889	0.806	0.994	0.896	0.842	0.994	0.889
XGB_RUS_PI_VIF	XGB	1.0	0.916	0.74	0.949	0.758	0.871	0.977	0.874	1.0	0.994	0.868
XGB_RUS_SHAP_VIF	XGB	1.0	0.916	0.546	0.989	0.875	0.824	0.994	0.894	0.824	0.994	0.894
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.933	0.528	1.0	0.884	0.829	1.0	0.878	0.899	1.0	0.864
XGB_SMOTE_PI_VIF	XGB	1.0	0.933	0.678	0.927	0.659	0.85	0.943	0.801	1.0	1.0	0.836
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.933	0.544	0.997	0.877	0.823	1.0	0.892	0.8	1.0	0.892
XGB_bal_FeatImp_VIF	XGB	1.0	0.93	0.535	0.997	0.892	0.806	1.0	0.903	0.768	1.0	0.87
XGB_bal_PI_VIF	XGB	1.0	0.93	0.841	0.962	0.786	0.841	0.962	0.786	1.0	0.981	0.82
XGB_bal_SHAP_VIF	XGB	1.0	0.93	0.512	0.994	0.862	0.812	1.0	0.952	0.788	1.0	0.931

Table 6.4: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Ionosphere*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.82	0.507	1.0	0.74	0.812	1.0	0.8	0.812	1.0	0.8
RF_RUS_PI	RF	1.0	0.82	0.766	0.693	0.676	0.962	0.836	0.707	1.0	1.0	0.787
RF_RUS_SHAP	RF	1.0	0.82	0.566	1.0	0.77	0.81	1.0	0.792	0.875	1.0	0.801
RF_SMOTE_FeatImp	RF	1.0	0.796	0.51	0.995	0.696	0.801	1.0	0.766	0.872	1.0	0.763
RF_SMOTE_PI	RF	1.0	0.796	0.665	0.952	0.52	0.839	0.989	0.677	1.0	1.0	0.728
RF_SMOTE_SHAP	RF	1.0	0.796	0.574	0.998	0.696	0.804	1.0	0.797	0.879	1.0	0.746
RF_bal_FeatImp	RF	1.0	0.633	0.508	0.985	0.7	0.801	1.0	0.667	0.824	1.0	0.65
RF_bal_PI	RF	1.0	0.633	0.611	0.795	0.653	0.947	0.954	0.619	1.0	1.0	0.617
RF_bal_SHAP	RF	1.0	0.633	0.557	0.985	0.717	0.809	1.0	0.65	0.85	1.0	0.65
RF_RUS_FeatImp_VIF	RF	1.0	0.777	0.533	1.0	0.664	0.818	1.0	0.763	0.738	1.0	0.696
RF_RUS_PI_VIF	RF	1.0	0.777	0.893	0.693	0.676	0.893	0.693	0.676	1.0	1.0	0.713
RF_RUS_SHAP_VIF	RF	1.0	0.777	0.545	0.993	0.667	0.815	1.0	0.667	0.837	1.0	0.69
RF_SMOTE_FeatImp_VIF	RF	1.0	0.698	0.549	0.998	0.678	0.811	1.0	0.748	0.811	1.0	0.748
RF_SMOTE_PI_VIF	RF	1.0	0.698	0.704	0.967	0.612	0.941	0.989	0.66	1.0	1.0	0.712
RF_SMOTE_SHAP_VIF	RF	1.0	0.698	0.524	0.997	0.646	0.82	1.0	0.698	0.841	1.0	0.699
RF_bal_FeatImp_VIF	RF	1.0	0.6	0.506	0.998	0.561	0.815	1.0	0.567	0.728	1.0	0.583
RF_bal_PI_VIF	RF	1.0	0.6	0.604	0.795	0.532	0.827	0.941	0.677	1.0	1.0	0.583
RF_bal_SHAP_VIF	RF	1.0	0.6	0.509	0.97	0.579	0.813	1.0	0.583	0.813	1.0	0.583
XGB_RUS_FeatImp	XGB	0.986	0.746	0.511	0.943	0.731	0.818	0.957	0.74	0.765	0.957	0.716
XGB_RUS_PI	XGB	0.986	0.746	0.512	0.779	0.649	0.801	0.921	0.731	0.983	0.979	0.727
XGB_RUS_SHAP	XGB	0.986	0.746	0.507	0.886	0.674	0.802	0.971	0.716	0.869	0.979	0.724
XGB_SMOTE_FeatImp	XGB	0.995	0.779	0.535	0.987	0.707	0.802	0.99	0.792	0.802	0.99	0.792
XGB_SMOTE_PI	XGB	0.995	0.779	0.558	0.949	0.56	0.903	0.979	0.726	0.99	0.993	0.763
XGB_SMOTE_SHAP	XGB	0.995	0.779	0.526	0.985	0.78	0.812	0.995	0.73	0.812	0.995	0.73
XGB_bal_FeatImp	XGB	0.953	0.871	0.524	0.918	0.758	0.82	0.949	0.866	0.67	0.928	0.774
XGB_bal_PI	XGB	0.953	0.871	0.671	0.729	0.62	0.836	0.858	0.697	0.983	0.954	0.854
XGB_bal_SHAP	XGB	0.953	0.871	0.506	0.837	0.764	0.805	0.956	0.868	0.868	0.954	0.88
XGB_RUS_FeatImp_VIF	XGB	1.0	0.73	0.519	0.979	0.752	0.817	0.993	0.742	0.69	0.986	0.762
XGB_RUS_PI_VIF	XGB	1.0	0.73	0.587	0.693	0.676	0.913	0.864	0.67	1.0	0.993	0.662
XGB_RUS_SHAP_VIF	XGB	1.0	0.73	0.572	0.95	0.69	0.829	0.993	0.717	0.879	0.993	0.717
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.732	0.506	0.998	0.698	0.811	1.0	0.731	0.844	1.0	0.766
XGB_SMOTE_PI_VIF	XGB	1.0	0.732	0.766	0.967	0.63	0.924	0.987	0.628	1.0	1.0	0.766
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.732	0.534	0.999	0.631	0.82	1.0	0.749	0.766	1.0	0.766
XGB_bal_FeatImp_VIF	XGB	0.971	0.821	0.514	0.95	0.832	0.805	0.969	0.75	0.589	0.96	0.81
XGB_bal_PI_VIF	XGB	0.971	0.821	0.762	0.72	0.603	0.881	0.877	0.722	0.986	0.966	0.859
XGB_bal_SHAP_VIF	XGB	0.971	0.821	0.512	0.913	0.782	0.828	0.963	0.797	0.828	0.963	0.797

Table 6.5: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Letter Recognition*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	0.986	0.925	0.527	0.948	0.886	0.802	0.984	0.9	0.666	0.948	0.918
RF_RUS_PI	RF	0.986	0.925	0.616	0.948	0.876	0.851	0.989	0.902	0.754	0.984	0.879
RF_RUS_SHAP	RF	0.986	0.925	0.536	0.948	0.876	0.812	0.956	0.913	0.68	0.948	0.91
RF_SMOTE_FeatImp	RF	0.993	0.898	0.567	0.993	0.878	0.862	0.993	0.845	0.567	0.993	0.878
RF_SMOTE_PI	RF	0.993	0.898	0.646	0.993	0.781	0.895	0.993	0.852	0.646	0.993	0.781
RF_SMOTE_SHAP	RF	0.993	0.898	0.588	0.99	0.839	0.882	0.993	0.887	0.588	0.99	0.839
RF_bal_FeatImp	RF	0.989	0.768	0.64	0.947	0.742	0.861	0.989	0.748	0.752	0.989	0.73
RF_bal_PI	RF	0.989	0.768	0.51	0.944	0.688	0.822	0.948	0.749	0.822	0.948	0.749
RF_bal_SHAP	RF	0.989	0.768	0.596	0.989	0.634	0.879	0.989	0.748	0.596	0.989	0.634
RF_RUS_FeatImp_VIF	RF	0.986	0.925	0.527	0.948	0.886	0.802	0.984	0.9	0.666	0.948	0.918
RF_RUS_PI_VIF	RF	0.986	0.925	0.616	0.948	0.876	0.851	0.989	0.902	0.754	0.984	0.879
RF_RUS_SHAP_VIF	RF	0.986	0.925	0.536	0.948	0.876	0.812	0.956	0.913	0.68	0.948	0.91
RF_SMOTE_FeatImp_VIF	RF	0.993	0.898	0.567	0.993	0.878	0.862	0.993	0.845	0.567	0.993	0.878
RF_SMOTE_PI_VIF	RF	0.993	0.898	0.646	0.993	0.781	0.895	0.993	0.852	0.646	0.993	0.781
RF_SMOTE_SHAP_VIF	RF	0.993	0.898	0.588	0.99	0.839	0.882	0.993	0.887	0.588	0.99	0.839
RF_bal_FeatImp_VIF	RF	0.989	0.768	0.64	0.947	0.742	0.861	0.989	0.748	0.752	0.989	0.73
RF_bal_PI_VIF	RF	0.989	0.768	0.51	0.944	0.688	0.822	0.948	0.749	0.822	0.948	0.749
RF_bal_SHAP_VIF	RF	0.989	0.768	0.596	0.989	0.634	0.879	0.989	0.748	0.596	0.989	0.634
XGB_RUS_FeatImp	XGB	0.956	0.926	0.535	0.843	0.808	0.862	0.951	0.896	0.79	0.931	0.895
XGB_RUS_PI	XGB	0.956	0.926	0.694	0.901	0.895	0.81	0.931	0.895	0.81	0.931	0.895
XGB_RUS_SHAP	XGB	0.956	0.926	0.584	0.901	0.895	0.813	0.92	0.918	0.7	0.918	0.915
XGB_SMOTE_FeatImp	XGB	0.97	0.939	0.535	0.84	0.839	0.827	0.942	0.885	0.827	0.942	0.885
XGB_SMOTE_PI	XGB	0.97	0.939	0.553	0.885	0.912	0.833	0.951	0.871	0.703	0.922	0.898
XGB_SMOTE_SHAP	XGB	0.97	0.939	0.525	0.885	0.912	0.903	0.96	0.896	0.665	0.922	0.898
XGB_bal_FeatImp	XGB	0.974	0.946	0.626	0.909	0.849	0.901	0.971	0.931	0.626	0.909	0.849
XGB_bal_PI	XGB	0.974	0.946	0.567	0.923	0.787	0.89	0.964	0.891	0.733	0.95	0.872
XGB_bal_SHAP	XGB	0.974	0.946	0.54	0.906	0.826	0.842	0.961	0.914	0.72	0.93	0.857
XGB_RUS_FeatImp_VIF	XGB	0.956	0.926	0.535	0.843	0.808	0.862	0.951	0.896	0.79	0.931	0.895
XGB_RUS_PI_VIF	XGB	0.956	0.926	0.694	0.901	0.895	0.81	0.931	0.895	0.81	0.931	0.895
XGB_RUS_SHAP_VIF	XGB	0.956	0.926	0.584	0.901	0.895	0.813	0.92	0.918	0.7	0.918	0.915
XGB_SMOTE_FeatImp_VIF	XGB	0.97	0.939	0.535	0.84	0.839	0.827	0.942	0.885	0.827	0.942	0.885
XGB_SMOTE_PI_VIF	XGB	0.97	0.939	0.553	0.885	0.912	0.833	0.951	0.871	0.703	0.922	0.898
XGB_SMOTE_SHAP_VIF	XGB	0.97	0.939	0.525	0.885	0.912	0.903	0.96	0.896	0.665	0.922	0.898
XGB_bal_FeatImp_VIF	XGB	0.974	0.946	0.626	0.909	0.849	0.901	0.971	0.931	0.626	0.909	0.849
XGB_bal_PI_VIF	XGB	0.974	0.946	0.567	0.923	0.787	0.89	0.964	0.891	0.733	0.95	0.872
XGB_bal_SHAP_VIF	XGB	0.974	0.946	0.54	0.906	0.826	0.842	0.961	0.914	0.72	0.93	0.857

Table 6.6: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Mammography*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.972	0.546	0.988	0.906	0.804	1.0	0.958	0.956	1.0	0.962
RF_RUS_PI	RF	1.0	0.972	0.689	0.846	0.796	0.965	0.882	0.8	1.0	1.0	0.947
RF_RUS_SHAP	RF	1.0	0.972	0.539	0.988	0.907	0.807	1.0	0.95	0.965	1.0	0.966
RF_SMOTE_FeatImp	RF	1.0	0.934	0.529	0.966	0.835	0.804	1.0	0.904	0.967	1.0	0.934
RF_SMOTE_PI	RF	1.0	0.934	0.835	0.899	0.742	0.835	0.899	0.742	1.0	1.0	0.928
RF_SMOTE_SHAP	RF	1.0	0.934	0.51	0.976	0.86	0.805	1.0	0.909	0.97	1.0	0.932
RF_bal_FeatImp	RF	1.0	0.901	0.551	0.871	0.785	0.815	0.999	0.878	0.977	1.0	0.897
RF_bal_PI	RF	1.0	0.901	0.786	0.856	0.763	0.937	0.919	0.783	1.0	1.0	0.897
RF_bal_SHAP	RF	1.0	0.901	0.551	0.98	0.819	0.803	1.0	0.884	0.973	1.0	0.892
RF_RUS_FeatImp_VIF	RF	1.0	0.97	0.506	0.995	0.911	0.812	1.0	0.958	0.943	1.0	0.963
RF_RUS_PI_VIF	RF	1.0	0.97	0.667	0.846	0.796	0.964	0.882	0.802	1.0	1.0	0.953
RF_RUS_SHAP_VIF	RF	1.0	0.97	0.536	0.996	0.917	0.803	1.0	0.938	0.965	1.0	0.967
RF_SMOTE_FeatImp_VIF	RF	1.0	0.936	0.53	0.956	0.811	0.812	1.0	0.911	0.972	1.0	0.926
RF_SMOTE_PI_VIF	RF	1.0	0.936	0.796	0.899	0.746	0.903	0.951	0.805	1.0	1.0	0.935
RF_SMOTE_SHAP_VIF	RF	1.0	0.936	0.535	0.977	0.858	0.809	1.0	0.931	0.973	1.0	0.933
RF_bal_FeatImp_VIF	RF	1.0	0.903	0.517	0.871	0.785	0.804	0.999	0.883	0.976	1.0	0.893
RF_bal_PI_VIF	RF	1.0	0.903	0.75	0.856	0.766	0.946	0.918	0.785	1.0	1.0	0.907
RF_bal_SHAP_VIF	RF	1.0	0.903	0.545	0.98	0.816	0.812	1.0	0.897	0.974	1.0	0.9
XGB_RUS_FeatImp	XGB	1.0	0.965	0.539	0.911	0.819	0.816	0.994	0.909	0.995	1.0	0.964
XGB_RUS_PI	XGB	1.0	0.965	0.777	0.881	0.805	0.967	0.917	0.813	1.0	1.0	0.943
XGB_RUS_SHAP	XGB	1.0	0.965	0.512	0.998	0.914	0.814	1.0	0.96	0.996	1.0	0.962
XGB_SMOTE_FeatImp	XGB	0.991	0.949	0.523	0.915	0.849	0.803	0.984	0.936	0.983	0.991	0.949
XGB_SMOTE_PI	XGB	0.991	0.949	0.636	0.879	0.817	0.885	0.905	0.84	1.0	0.991	0.946
XGB_SMOTE_SHAP	XGB	0.991	0.949	0.578	0.93	0.865	0.808	0.985	0.933	0.998	0.991	0.946
XGB_bal_FeatImp	XGB	0.995	0.976	0.515	0.937	0.919	0.812	0.984	0.964	0.94	0.992	0.973
XGB_bal_PI	XGB	0.995	0.976	0.675	0.667	0.603	0.87	0.85	0.804	1.0	0.995	0.975
XGB_bal_SHAP	XGB	0.995	0.976	0.532	0.981	0.95	0.816	0.992	0.964	0.978	0.995	0.973
XGB_RUS_FeatImp_VIF	XGB	1.0	0.97	0.507	0.977	0.858	0.804	0.997	0.946	0.978	1.0	0.968
XGB_RUS_PI_VIF	XGB	1.0	0.97	0.83	0.879	0.807	0.83	0.879	0.807	1.0	1.0	0.948
XGB_RUS_SHAP_VIF	XGB	1.0	0.97	0.535	0.989	0.893	0.802	1.0	0.964	0.993	1.0	0.965
XGB_SMOTE_FeatImp_VIF	XGB	0.99	0.948	0.533	0.88	0.817	0.807	0.98	0.935	0.989	0.99	0.948
XGB_SMOTE_PI_VIF	XGB	0.99	0.948	0.679	0.88	0.817	0.812	0.893	0.816	1.0	0.99	0.95
XGB_SMOTE_SHAP_VIF	XGB	0.99	0.948	0.518	0.907	0.836	0.807	0.977	0.915	0.998	0.991	0.947
XGB_bal_FeatImp_VIF	XGB	0.998	0.974	0.51	0.928	0.858	0.806	0.988	0.95	0.925	0.996	0.962
XGB_bal_PI_VIF	XGB	0.998	0.974	0.666	0.862	0.74	0.897	0.893	0.806	1.0	0.998	0.974
XGB_bal_SHAP_VIF	XGB	0.998	0.974	0.5	0.987	0.935	0.805	0.997	0.961	0.975	0.998	0.971

Table 6.7: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Mnist*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.954	0.554	1.0	0.934	0.831	1.0	0.948	0.658	1.0	0.936
RF_RUS_PI	RF	1.0	0.954	0.719	0.993	0.884	0.829	0.997	0.919	0.965	1.0	0.939
RF_RUS_SHAP	RF	1.0	0.954	0.605	1.0	0.935	0.828	1.0	0.947	0.683	1.0	0.938
RF_SMOTE_FeatImp	RF	1.0	0.944	0.606	0.998	0.902	0.88	1.0	0.942	0.797	1.0	0.931
RF_SMOTE_PI	RF	1.0	0.944	0.863	0.971	0.824	0.863	0.971	0.824	0.99	1.0	0.936
RF_SMOTE_SHAP	RF	1.0	0.944	0.584	0.998	0.902	0.852	1.0	0.947	0.785	1.0	0.931
RF_bal_FeatImp	RF	1.0	0.91	0.711	0.992	0.838	0.849	1.0	0.89	0.891	1.0	0.906
RF_bal_PI	RF	1.0	0.91	0.667	0.901	0.839	0.868	0.992	0.841	0.99	1.0	0.906
RF_bal_SHAP	RF	1.0	0.91	0.535	0.992	0.841	0.86	1.0	0.903	0.86	1.0	0.903
RF_RUS_FeatImp_VIF	RF	1.0	0.953	0.613	1.0	0.94	0.84	1.0	0.954	0.613	1.0	0.94
RF_RUS_PI_VIF	RF	1.0	0.953	0.695	0.978	0.875	0.863	0.999	0.93	0.863	0.999	0.93
RF_RUS_SHAP_VIF	RF	1.0	0.953	0.569	0.999	0.92	0.87	1.0	0.951	0.693	1.0	0.94
RF_SMOTE_FeatImp_VIF	RF	1.0	0.945	0.61	0.998	0.898	0.82	1.0	0.938	0.718	0.999	0.923
RF_SMOTE_PI_VIF	RF	1.0	0.945	0.814	0.972	0.824	0.814	0.972	0.824	0.971	1.0	0.935
RF_SMOTE_SHAP_VIF	RF	1.0	0.945	0.629	0.998	0.898	0.842	1.0	0.95	0.765	1.0	0.935
RF_bal_FeatImp_VIF	RF	1.0	0.91	0.662	0.992	0.841	0.836	1.0	0.878	0.836	1.0	0.878
RF_bal_PI_VIF	RF	1.0	0.91	0.698	0.901	0.839	0.86	0.992	0.841	0.976	1.0	0.903
RF_bal_SHAP_VIF	RF	1.0	0.91	0.547	0.992	0.841	0.815	1.0	0.877	0.815	1.0	0.877
XGB_RUS_FeatImp	XGB	0.994	0.943	0.566	0.958	0.89	0.801	0.985	0.938	0.801	0.985	0.938
XGB_RUS_PI	XGB	0.994	0.943	0.715	0.899	0.857	0.863	0.958	0.89	0.968	0.992	0.949
XGB_RUS_SHAP	XGB	0.994	0.943	0.575	0.958	0.89	0.82	0.983	0.931	0.876	0.985	0.948
XGB_SMOTE_FeatImp	XGB	0.978	0.95	0.619	0.943	0.893	0.802	0.972	0.937	0.857	0.974	0.944
XGB_SMOTE_PI	XGB	0.978	0.95	0.754	0.922	0.859	0.93	0.943	0.893	0.982	0.974	0.953
XGB_SMOTE_SHAP	XGB	0.978	0.95	0.655	0.943	0.893	0.828	0.972	0.937	0.889	0.972	0.933
XGB_bal_FeatImp	XGB	0.984	0.953	0.607	0.965	0.94	0.805	0.976	0.955	0.727	0.972	0.941
XGB_bal_PI	XGB	0.984	0.953	0.707	0.5	0.5	0.884	0.962	0.911	0.958	0.982	0.96
XGB_bal_SHAP	XGB	0.984	0.953	0.561	0.96	0.93	0.84	0.984	0.955	0.755	0.978	0.947
XGB_RUS_FeatImp_VIF	XGB	0.994	0.944	0.652	0.955	0.885	0.828	0.976	0.94	0.828	0.976	0.94
XGB_RUS_PI_VIF	XGB	0.994	0.944	0.756	0.899	0.857	0.867	0.955	0.885	0.941	0.98	0.945
XGB_RUS_SHAP_VIF	XGB	0.994	0.944	0.546	0.955	0.885	0.87	0.985	0.942	0.786	0.98	0.945
XGB_SMOTE_FeatImp_VIF	XGB	0.979	0.951	0.672	0.943	0.891	0.85	0.973	0.936	0.85	0.973	0.936
XGB_SMOTE_PI_VIF	XGB	0.979	0.951	0.744	0.919	0.859	0.92	0.943	0.891	0.965	0.972	0.93
XGB_SMOTE_SHAP_VIF	XGB	0.979	0.951	0.659	0.943	0.891	0.841	0.972	0.93	0.841	0.972	0.93
XGB_bal_FeatImp_VIF	XGB	0.972	0.954	0.541	0.903	0.882	0.885	0.967	0.945	0.7	0.955	0.938
XGB_bal_PI_VIF	XGB	0.972	0.954	0.713	0.585	0.583	0.803	0.918	0.888	0.931	0.958	0.942
XGB_bal_SHAP_VIF	XGB	0.972	0.954	0.664	0.947	0.921	0.857	0.968	0.945	0.764	0.963	0.941

Table 6.8: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *PageBlocks*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.782	0.602	1.0	0.732	0.821	1.0	0.763	0.728	1.0	0.757
RF_RUS_PI	RF	1.0	0.782	0.652	0.767	0.619	0.815	0.997	0.673	0.96	1.0	0.757
RF_RUS_SHAP	RF	1.0	0.782	0.533	0.997	0.673	0.886	1.0	0.763	0.779	1.0	0.757
RF_SMOTE_FeatImp	RF	1.0	0.747	0.593	1.0	0.726	0.878	1.0	0.737	0.717	1.0	0.736
RF_SMOTE_PI	RF	1.0	0.747	0.662	0.834	0.644	0.896	1.0	0.726	0.969	1.0	0.736
RF_SMOTE_SHAP	RF	1.0	0.747	0.529	0.994	0.677	0.864	1.0	0.714	0.788	1.0	0.736
RF_bal_FeatImp	RF	1.0	0.721	0.624	1.0	0.666	0.825	1.0	0.704	0.748	1.0	0.685
RF_bal_PI	RF	1.0	0.721	0.626	0.759	0.645	0.908	1.0	0.666	0.979	1.0	0.685
RF_bal_SHAP	RF	1.0	0.721	0.55	0.994	0.673	0.809	1.0	0.685	0.809	1.0	0.685
RF_RUS_FeatImp_VIF	RF	1.0	0.782	0.602	1.0	0.732	0.821	1.0	0.763	0.728	1.0	0.757
RF_RUS_PI_VIF	RF	1.0	0.782	0.652	0.767	0.619	0.815	0.997	0.673	0.96	1.0	0.757
RF_RUS_SHAP_VIF	RF	1.0	0.782	0.533	0.997	0.673	0.886	1.0	0.763	0.779	1.0	0.757
RF_SMOTE_FeatImp_VIF	RF	1.0	0.747	0.593	1.0	0.726	0.878	1.0	0.737	0.717	1.0	0.736
RF_SMOTE_PI_VIF	RF	1.0	0.747	0.662	0.834	0.644	0.896	1.0	0.726	0.969	1.0	0.736
RF_SMOTE_SHAP_VIF	RF	1.0	0.747	0.529	0.994	0.677	0.864	1.0	0.714	0.788	1.0	0.736
RF_bal_FeatImp_VIF	RF	1.0	0.721	0.624	1.0	0.666	0.825	1.0	0.704	0.748	1.0	0.685
RF_bal_PI_VIF	RF	1.0	0.721	0.626	0.759	0.645	0.908	1.0	0.666	0.979	1.0	0.685
RF_bal_SHAP_VIF	RF	1.0	0.721	0.55	0.994	0.673	0.809	1.0	0.685	0.809	1.0	0.685
XGB_RUS_FeatImp	XGB	0.821	0.782	0.56	0.794	0.74	0.854	0.821	0.769	0.69	0.81	0.757
XGB_RUS_PI	XGB	0.821	0.782	0.709	0.789	0.747	0.87	0.807	0.766	0.87	0.807	0.766
XGB_RUS_SHAP	XGB	0.821	0.782	0.575	0.789	0.747	0.823	0.807	0.766	0.823	0.807	0.766
XGB_SMOTE_FeatImp	XGB	0.817	0.773	0.602	0.786	0.748	0.889	0.816	0.773	0.704	0.803	0.764
XGB_SMOTE_PI	XGB	0.817	0.773	0.663	0.73	0.653	0.808	0.771	0.735	0.956	0.803	0.764
XGB_SMOTE_SHAP	XGB	0.817	0.773	0.638	0.771	0.735	0.892	0.803	0.764	0.892	0.803	0.764
XGB_bal_FeatImp	XGB	0.814	0.77	0.566	0.741	0.71	0.904	0.797	0.737	0.566	0.741	0.71
XGB_bal_PI	XGB	0.814	0.77	0.689	0.7	0.699	0.881	0.796	0.721	0.881	0.796	0.721
XGB_bal_SHAP	XGB	0.814	0.77	0.558	0.7	0.699	0.827	0.796	0.721	0.827	0.796	0.721
XGB_RUS_FeatImp_VIF	XGB	0.821	0.782	0.56	0.794	0.74	0.854	0.821	0.769	0.69	0.81	0.757
XGB_RUS_PI_VIF	XGB	0.821	0.782	0.709	0.789	0.747	0.87	0.807	0.766	0.87	0.807	0.766
XGB_RUS_SHAP_VIF	XGB	0.821	0.782	0.575	0.789	0.747	0.823	0.807	0.766	0.823	0.807	0.766
XGB_SMOTE_FeatImp_VIF	XGB	0.817	0.773	0.602	0.786	0.748	0.889	0.816	0.773	0.704	0.803	0.764
XGB_SMOTE_PI_VIF	XGB	0.817	0.773	0.663	0.73	0.653	0.808	0.771	0.735	0.956	0.803	0.764
XGB_SMOTE_SHAP_VIF	XGB	0.817	0.773	0.638	0.771	0.735	0.892	0.803	0.764	0.892	0.803	0.764
XGB_bal_FeatImp_VIF	XGB	0.814	0.77	0.566	0.741	0.71	0.904	0.797	0.737	0.566	0.741	0.71
XGB_bal_PI_VIF	XGB	0.814	0.77	0.689	0.7	0.699	0.881	0.796	0.721	0.881	0.796	0.721
XGB_bal_SHAP_VIF	XGB	0.814	0.77	0.558	0.7	0.699	0.827	0.796	0.721	0.827	0.796	0.721

Table 6.9: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Pima*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.993	0.589	1.0	0.901	0.822	1.0	0.991	0.919	1.0	0.99
RF_RUS_PI	RF	1.0	0.993	0.914	0.986	0.885	0.914	0.986	0.885	1.0	1.0	0.989
RF_RUS_SHAP	RF	1.0	0.993	0.515	1.0	0.891	0.845	1.0	0.992	0.929	1.0	0.992
RF_SMOTE_FeatImp	RF	1.0	0.989	0.541	1.0	0.872	0.824	1.0	0.989	0.773	1.0	0.989
RF_SMOTE_PI	RF	1.0	0.989	0.826	0.941	0.676	0.826	0.941	0.676	1.0	1.0	0.989
RF_SMOTE_SHAP	RF	1.0	0.989	0.517	1.0	0.975	0.833	1.0	1.0	0.778	1.0	0.989
RF_bal_FeatImp	RF	1.0	0.989	0.547	1.0	0.946	0.838	1.0	0.957	0.891	1.0	0.979
RF_bal_PI	RF	1.0	0.989	0.561	0.936	0.664	0.992	1.0	0.966	1.0	1.0	0.957
RF_bal_SHAP	RF	1.0	0.989	0.574	1.0	0.966	0.828	1.0	0.978	0.88	1.0	0.978
RF_RUS_FeatImp_VIF	RF	1.0	0.994	0.545	1.0	0.925	0.86	1.0	0.991	0.761	1.0	0.992
RF_RUS_PI_VIF	RF	1.0	0.994	0.922	0.995	0.813	0.922	0.995	0.813	1.0	1.0	0.992
RF_RUS_SHAP_VIF	RF	1.0	0.994	0.607	1.0	0.957	0.83	1.0	0.99	0.83	1.0	0.99
RF_SMOTE_FeatImp_VIF	RF	1.0	0.989	0.546	1.0	0.907	0.834	1.0	0.978	0.751	1.0	0.978
RF_SMOTE_PI_VIF	RF	1.0	0.989	0.793	1.0	0.812	0.99	1.0	0.974	1.0	1.0	0.988
RF_SMOTE_SHAP_VIF	RF	1.0	0.989	0.604	1.0	0.974	0.802	1.0	0.988	0.802	1.0	0.988
RF_bal_FeatImp_VIF	RF	1.0	0.947	0.505	0.996	0.693	0.843	1.0	0.904	0.896	1.0	0.957
RF_bal_PI_VIF	RF	1.0	0.947	0.724	1.0	0.656	0.987	1.0	0.924	0.998	1.0	0.947
RF_bal_SHAP_VIF	RF	1.0	0.947	0.662	1.0	0.924	0.841	1.0	0.946	0.841	1.0	0.946
XGB_RUS_FeatImp	XGB	1.0	0.991	0.551	0.945	0.908	0.804	0.991	0.987	0.914	0.991	0.988
XGB_RUS_PI	XGB	1.0	0.991	0.92	0.927	0.938	0.92	0.927	0.938	1.0	1.0	0.99
XGB_RUS_SHAP	XGB	1.0	0.991	0.661	0.95	0.907	0.803	0.991	0.987	0.974	1.0	0.99
XGB_SMOTE_FeatImp	XGB	1.0	0.999	0.624	0.936	0.935	0.846	0.999	0.982	0.966	1.0	0.999
XGB_SMOTE_PI	XGB	1.0	0.999	0.823	0.936	0.935	0.823	0.936	0.935	1.0	1.0	1.0
XGB_SMOTE_SHAP	XGB	1.0	0.999	0.505	0.999	0.996	0.818	1.0	1.0	0.818	1.0	1.0
XGB_bal_FeatImp	XGB	1.0	0.999	0.617	0.988	0.783	0.824	0.999	0.986	0.936	0.999	0.989
XGB_bal_PI	XGB	1.0	0.999	0.86	0.969	0.838	0.86	0.969	0.838	1.0	1.0	0.999
XGB_bal_SHAP	XGB	1.0	0.999	0.581	0.998	0.985	0.838	1.0	0.999	0.877	1.0	0.999
XGB_RUS_FeatImp_VIF	XGB	0.986	0.981	0.528	0.931	0.855	0.811	0.977	0.96	0.88	0.982	0.979
XGB_RUS_PI_VIF	XGB	0.986	0.981	0.61	0.931	0.855	0.88	0.954	0.891	0.986	0.982	0.981
XGB_RUS_SHAP_VIF	XGB	0.986	0.981	0.661	0.977	0.958	0.848	0.982	0.983	0.848	0.982	0.983
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.978	0.731	0.928	0.86	0.812	0.984	0.878	0.961	0.999	0.956
XGB_SMOTE_PI_VIF	XGB	1.0	0.978	0.698	0.928	0.86	0.91	0.976	0.845	1.0	1.0	0.978
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.978	0.644	0.976	0.845	0.824	0.998	0.961	0.964	1.0	0.977
XGB_bal_FeatImp_VIF	XGB	1.0	1.0	0.566	0.994	0.734	0.809	1.0	0.966	0.92	1.0	0.989
XGB_bal_PI_VIF	XGB	1.0	1.0	0.728	0.999	0.87	0.992	1.0	0.956	1.0	1.0	0.989
XGB_bal_SHAP_VIF	XGB	1.0	1.0	0.525	1.0	0.955	0.819	1.0	1.0	0.73	1.0	0.978

Table 6.10: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Pendigits*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.924	0.512	0.998	0.863	0.806	1.0	0.91	0.757	1.0	0.918
RF_RUS_PI	RF	1.0	0.924	0.897	0.812	0.805	0.897	0.812	0.805	1.0	1.0	0.917
RF_RUS_SHAP	RF	1.0	0.924	0.517	1.0	0.879	0.809	1.0	0.914	0.758	1.0	0.913
RF_SMOTE_FeatImp	RF	1.0	0.921	0.533	1.0	0.864	0.801	1.0	0.911	0.748	1.0	0.903
RF_SMOTE_PI	RF	1.0	0.921	0.535	0.875	0.808	0.829	0.956	0.821	1.0	1.0	0.899
RF_SMOTE_SHAP	RF	1.0	0.921	0.528	1.0	0.867	0.816	1.0	0.911	0.744	1.0	0.91
RF_bal_FeatImp	RF	1.0	0.91	0.502	0.992	0.865	0.805	1.0	0.895	0.773	1.0	0.893
RF_bal_PI	RF	1.0	0.91	0.948	0.808	0.806	0.948	0.808	0.806	1.0	1.0	0.899
RF_bal_SHAP	RF	1.0	0.91	0.515	0.996	0.845	0.815	1.0	0.902	0.764	1.0	0.896
RF_RUS_FeatImp_VIF	RF	1.0	0.904	0.569	0.993	0.866	0.895	1.0	0.894	0.569	0.993	0.866
RF_RUS_PI_VIF	RF	1.0	0.904	0.552	0.886	0.816	0.83	0.998	0.893	0.758	0.992	0.874
RF_RUS_SHAP_VIF	RF	1.0	0.904	0.599	0.987	0.876	0.811	0.999	0.892	0.599	0.987	0.876
RF_SMOTE_FeatImp_VIF	RF	1.0	0.906	0.542	0.968	0.828	0.835	0.999	0.895	0.645	0.988	0.871
RF_SMOTE_PI_VIF	RF	1.0	0.906	0.654	0.924	0.806	0.864	0.988	0.873	0.864	0.988	0.873
RF_SMOTE_SHAP_VIF	RF	1.0	0.906	0.551	0.968	0.828	0.857	0.999	0.895	0.669	0.988	0.871
RF_bal_FeatImp_VIF	RF	1.0	0.895	0.55	0.88	0.807	0.804	0.996	0.877	0.723	0.99	0.853
RF_bal_PI_VIF	RF	1.0	0.895	0.665	0.88	0.807	0.888	0.983	0.87	0.888	0.983	0.87
RF_bal_SHAP_VIF	RF	1.0	0.895	0.624	0.952	0.819	0.818	0.996	0.874	0.742	0.983	0.868
XGB_RUS_FeatImp	XGB	1.0	0.923	0.502	0.98	0.86	0.803	0.999	0.911	0.747	0.999	0.902
XGB_RUS_PI	XGB	1.0	0.923	0.635	0.872	0.827	0.828	0.956	0.859	0.988	0.999	0.913
XGB_RUS_SHAP	XGB	1.0	0.923	0.519	0.987	0.884	0.816	0.998	0.916	0.764	0.999	0.914
XGB_SMOTE_FeatImp	XGB	1.0	0.925	0.504	0.984	0.868	0.812	1.0	0.917	0.782	1.0	0.911
XGB_SMOTE_PI	XGB	1.0	0.925	0.866	0.909	0.808	0.866	0.909	0.808	0.999	1.0	0.923
XGB_SMOTE_SHAP	XGB	1.0	0.925	0.528	0.998	0.866	0.818	1.0	0.917	0.742	1.0	0.919
XGB_bal_FeatImp	XGB	1.0	0.933	0.509	1.0	0.908	0.809	1.0	0.922	0.646	1.0	0.923
XGB_bal_PI	XGB	1.0	0.933	0.549	0.691	0.65	0.815	0.965	0.816	1.0	1.0	0.914
XGB_bal_SHAP	XGB	1.0	0.933	0.501	1.0	0.895	0.806	1.0	0.927	0.698	1.0	0.914
XGB_RUS_FeatImp_VIF	XGB	0.962	0.898	0.544	0.865	0.832	0.824	0.942	0.884	0.743	0.927	0.871
XGB_RUS_PI_VIF	XGB	0.962	0.898	0.697	0.865	0.832	0.819	0.906	0.85	0.894	0.927	0.871
XGB_RUS_SHAP_VIF	XGB	0.962	0.898	0.581	0.865	0.832	0.862	0.942	0.88	0.789	0.927	0.871
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.895	0.525	0.918	0.817	0.81	0.992	0.867	0.724	0.983	0.85
XGB_SMOTE_PI_VIF	XGB	1.0	0.895	0.584	0.918	0.817	0.848	0.98	0.862	0.848	0.98	0.862
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.895	0.584	0.964	0.853	0.806	0.993	0.88	0.698	0.984	0.866
XGB_bal_FeatImp_VIF	XGB	1.0	0.898	0.581	0.972	0.855	0.811	0.998	0.888	0.581	0.972	0.855
XGB_bal_PI_VIF	XGB	1.0	0.898	0.595	0.772	0.7	0.85	0.978	0.87	0.85	0.978	0.87
XGB_bal_SHAP_VIF	XGB	1.0	0.898	0.573	0.947	0.834	0.875	0.998	0.894	0.682	0.978	0.87

Table 6.11: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Satellite*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	0.992	0.714	0.649	0.996	0.654	0.86	0.996	0.725	0.77	0.992	0.691
RF_RUS_PI	RF	0.992	0.714	0.503	0.992	0.515	0.886	0.996	0.682	0.935	0.996	0.688
RF_RUS_SHAP	RF	0.992	0.714	0.558	0.996	0.699	0.875	0.992	0.705	0.672	0.996	0.677
RF_SMOTE_FeatImp	RF	1.0	0.598	0.521	1.0	0.61	0.853	1.0	0.596	0.64	1.0	0.591
RF_SMOTE_PI	RF	1.0	0.598	0.534	0.988	0.6	0.826	1.0	0.57	0.936	1.0	0.59
RF_SMOTE_SHAP	RF	1.0	0.598	0.56	0.993	0.566	0.824	1.0	0.58	0.652	1.0	0.584
RF_bal_FeatImp	RF	1.0	0.529	0.515	1.0	0.519	0.843	1.0	0.548	0.758	1.0	0.549
RF_bal_PI	RF	1.0	0.529	0.571	0.952	0.525	0.81	1.0	0.519	0.962	1.0	0.539
RF_bal_SHAP	RF	1.0	0.529	0.58	1.0	0.525	0.885	1.0	0.539	0.691	1.0	0.527
RF_RUS_FeatImp_VIF	RF	1.0	0.71	0.622	1.0	0.665	0.909	1.0	0.687	0.792	1.0	0.657
RF_RUS_PI_VIF	RF	1.0	0.71	0.582	0.983	0.61	0.847	1.0	0.673	0.847	1.0	0.673
RF_RUS_SHAP_VIF	RF	1.0	0.71	0.616	1.0	0.677	0.866	1.0	0.672	0.744	1.0	0.673
RF_SMOTE_FeatImp_VIF	RF	1.0	0.601	0.541	1.0	0.59	0.848	1.0	0.585	0.695	1.0	0.6
RF_SMOTE_PI_VIF	RF	1.0	0.601	0.53	0.797	0.559	0.896	1.0	0.586	0.798	1.0	0.586
RF_SMOTE_SHAP_VIF	RF	1.0	0.601	0.622	0.993	0.501	0.874	1.0	0.586	0.758	1.0	0.586
RF_bal_FeatImp_VIF	RF	0.996	0.537	0.62	0.996	0.52	0.911	0.996	0.529	0.774	0.996	0.519
RF_bal_PI_VIF	RF	0.996	0.537	0.717	0.978	0.561	0.868	0.996	0.515	0.924	0.996	0.529
RF_bal_SHAP_VIF	RF	0.996	0.537	0.603	0.996	0.515	0.862	0.996	0.509	0.734	0.996	0.517
XGB_RUS_FeatImp	XGB	0.727	0.707	0.608	0.723	0.66	0.843	0.727	0.665	0.843	0.727	0.665
XGB_RUS_PI	XGB	0.727	0.707	0.61	0.723	0.642	0.835	0.727	0.676	0.914	0.714	0.685
XGB_RUS_SHAP	XGB	0.727	0.707	0.66	0.723	0.66	0.864	0.714	0.685	0.864	0.714	0.685
XGB_SMOTE_FeatImp	XGB	0.834	0.712	0.543	0.763	0.663	0.815	0.822	0.706	0.76	0.802	0.681
XGB_SMOTE_PI	XGB	0.834	0.712	0.538	0.76	0.642	0.829	0.806	0.663	0.829	0.806	0.663
XGB_SMOTE_SHAP	XGB	0.834	0.712	0.58	0.779	0.682	0.831	0.816	0.697	0.749	0.813	0.695
XGB_bal_FeatImp	XGB	0.883	0.722	0.577	0.87	0.708	0.835	0.881	0.735	0.577	0.87	0.708
XGB_bal_PI	XGB	0.883	0.722	0.529	0.761	0.517	0.804	0.861	0.657	0.87	0.876	0.718
XGB_bal_SHAP	XGB	0.883	0.722	0.557	0.847	0.653	0.881	0.883	0.715	0.78	0.875	0.713
XGB_RUS_FeatImp_VIF	XGB	0.748	0.702	0.651	0.723	0.665	0.886	0.748	0.678	0.772	0.748	0.658
XGB_RUS_PI_VIF	XGB	0.748	0.702	0.635	0.748	0.658	0.832	0.744	0.665	0.832	0.744	0.665
XGB_RUS_SHAP_VIF	XGB	0.748	0.702	0.52	0.693	0.644	0.812	0.727	0.686	0.812	0.727	0.686
XGB_SMOTE_FeatImp_VIF	XGB	0.831	0.69	0.64	0.765	0.675	0.805	0.824	0.662	0.805	0.824	0.662
XGB_SMOTE_PI_VIF	XGB	0.831	0.69	0.628	0.771	0.643	0.866	0.824	0.662	0.866	0.824	0.662
XGB_SMOTE_SHAP_VIF	XGB	0.831	0.69	0.614	0.765	0.675	0.848	0.824	0.662	0.848	0.824	0.662
XGB_bal_FeatImp_VIF	XGB	0.897	0.715	0.509	0.87	0.681	0.891	0.896	0.689	0.641	0.885	0.696
XGB_bal_PI_VIF	XGB	0.897	0.715	0.619	0.846	0.649	0.871	0.885	0.69	0.871	0.885	0.69
XGB_bal_SHAP_VIF	XGB	0.897	0.715	0.654	0.874	0.685	0.901	0.887	0.692	0.782	0.874	0.677

Table 6.12: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Seismic*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.938	0.528	0.998	0.885	0.803	1.0	0.921	0.916	1.0	0.932
RF_RUS_PI	RF	1.0	0.938	0.597	0.903	0.788	0.82	0.993	0.882	1.0	1.0	0.931
RF_RUS_SHAP	RF	1.0	0.938	0.518	0.997	0.899	0.804	0.999	0.927	0.937	1.0	0.934
RF_SMOTE_FeatImp	RF	1.0	0.936	0.522	0.995	0.886	0.815	1.0	0.918	0.932	1.0	0.931
RF_SMOTE_PI	RF	1.0	0.936	0.548	0.92	0.794	0.836	0.977	0.861	1.0	1.0	0.93
RF_SMOTE_SHAP	RF	1.0	0.936	0.515	0.986	0.891	0.81	0.999	0.924	0.949	1.0	0.932
RF_bal_FeatImp	RF	1.0	0.931	0.523	0.998	0.893	0.804	1.0	0.925	0.916	1.0	0.934
RF_bal_PI	RF	1.0	0.931	0.514	0.908	0.749	0.812	0.996	0.899	1.0	1.0	0.929
RF_bal_SHAP	RF	1.0	0.931	0.513	0.996	0.902	0.814	1.0	0.924	0.94	1.0	0.928
RF_RUS_FeatImp_VIF	RF	1.0	0.937	0.532	0.997	0.898	0.805	1.0	0.924	0.917	1.0	0.934
RF_RUS_PI_VIF	RF	1.0	0.937	0.592	0.903	0.786	0.818	0.991	0.862	1.0	1.0	0.933
RF_RUS_SHAP_VIF	RF	1.0	0.937	0.519	0.997	0.9	0.819	0.999	0.93	0.936	1.0	0.933
RF_SMOTE_FeatImp_VIF	RF	1.0	0.934	0.537	0.997	0.895	0.813	1.0	0.927	0.921	1.0	0.932
RF_SMOTE_PI_VIF	RF	1.0	0.934	0.533	0.927	0.751	0.842	0.995	0.883	1.0	1.0	0.931
RF_SMOTE_SHAP_VIF	RF	1.0	0.934	0.545	0.997	0.895	0.808	0.999	0.921	0.942	1.0	0.931
RF_bal_FeatImp_VIF	RF	1.0	0.935	0.509	0.996	0.898	0.806	1.0	0.924	0.914	1.0	0.929
RF_bal_PI_VIF	RF	1.0	0.935	0.501	0.9	0.79	0.844	0.996	0.891	1.0	1.0	0.931
RF_bal_SHAP_VIF	RF	1.0	0.935	0.506	0.996	0.901	0.803	1.0	0.922	0.936	1.0	0.933
XGB_RUS_FeatImp	XGB	0.993	0.935	0.527	0.937	0.88	0.802	0.954	0.907	0.921	0.982	0.92
XGB_RUS_PI	XGB	0.993	0.935	0.514	0.956	0.838	0.807	0.981	0.897	0.998	0.993	0.93
XGB_RUS_SHAP	XGB	0.993	0.935	0.524	0.977	0.9	0.801	0.994	0.912	0.965	0.992	0.927
XGB_SMOTE_FeatImp	XGB	0.997	0.935	0.507	0.947	0.879	0.806	0.967	0.915	0.953	0.994	0.922
XGB_SMOTE_PI	XGB	0.997	0.935	0.526	0.914	0.795	0.836	0.992	0.907	0.998	0.997	0.932
XGB_SMOTE_SHAP	XGB	0.997	0.935	0.51	0.992	0.901	0.811	0.997	0.929	0.955	0.997	0.936
XGB_bal_FeatImp	XGB	1.0	0.939	0.515	0.813	0.774	0.803	0.928	0.862	0.889	0.945	0.866
XGB_bal_PI	XGB	1.0	0.939	0.611	0.743	0.635	0.845	0.967	0.828	1.0	1.0	0.93
XGB_bal_SHAP	XGB	1.0	0.939	0.537	0.993	0.893	0.801	1.0	0.926	0.928	1.0	0.929
XGB_RUS_FeatImp_VIF	XGB	0.993	0.935	0.527	0.937	0.88	0.802	0.954	0.907	0.921	0.982	0.92
XGB_RUS_PI_VIF	XGB	0.993	0.935	0.514	0.956	0.838	0.807	0.981	0.897	0.998	0.993	0.93
XGB_RUS_SHAP_VIF	XGB	0.993	0.935	0.524	0.977	0.9	0.801	0.994	0.912	0.965	0.992	0.927
XGB_SMOTE_FeatImp_VIF	XGB	0.997	0.934	0.501	0.919	0.85	0.805	0.966	0.906	0.958	0.995	0.925
XGB_SMOTE_PI_VIF	XGB	0.997	0.934	0.578	0.971	0.827	0.831	0.994	0.896	0.999	0.997	0.931
XGB_SMOTE_SHAP_VIF	XGB	0.997	0.934	0.512	0.992	0.899	0.81	0.997	0.921	0.962	0.997	0.933
XGB_bal_FeatImp_VIF	XGB	1.0	0.935	0.51	0.817	0.778	0.803	0.925	0.859	0.891	0.945	0.86
XGB_bal_PI_VIF	XGB	1.0	0.935	0.582	0.743	0.635	0.824	0.967	0.828	1.0	1.0	0.928
XGB_bal_SHAP_VIF	XGB	1.0	0.935	0.536	0.993	0.894	0.818	1.0	0.925	0.929	1.0	0.928

Table 6.13: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *SpamBase*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.975	0.557	1.0	0.942	0.816	1.0	0.974	0.94	1.0	0.974
RF_RUS_PI	RF	1.0	0.975	1.0	0.96	0.898	1.0	0.96	0.898	1.0	1.0	0.97
RF_RUS_SHAP	RF	1.0	0.975	0.541	1.0	0.973	0.814	1.0	0.973	0.89	1.0	0.974
RF_SMOTE_FeatImp	RF	1.0	0.976	0.521	1.0	0.951	0.825	1.0	0.975	0.924	1.0	0.975
RF_SMOTE_PI	RF	1.0	0.976	0.526	1.0	0.974	0.842	1.0	0.976	1.0	1.0	0.975
RF_SMOTE_SHAP	RF	1.0	0.976	0.533	1.0	0.976	0.819	1.0	0.976	0.84	1.0	0.976
RF_bal_FeatImp	RF	1.0	0.952	0.52	0.996	0.923	0.807	1.0	0.928	0.981	1.0	0.952
RF_bal_PI	RF	1.0	0.952	0.998	0.929	0.893	0.998	0.929	0.893	1.0	1.0	0.928
RF_bal_SHAP	RF	1.0	0.952	0.533	1.0	0.928	0.83	1.0	0.976	0.93	1.0	0.952
RF_RUS_FeatImp_VIF	RF	1.0	0.972	0.636	1.0	0.971	0.901	1.0	0.971	0.636	1.0	0.971
RF_RUS_PI_VIF	RF	1.0	0.972	0.822	0.96	0.866	0.822	0.96	0.866	0.975	1.0	0.971
RF_RUS_SHAP_VIF	RF	1.0	0.972	0.562	1.0	0.97	0.895	1.0	0.97	0.678	1.0	0.971
RF_SMOTE_FeatImp_VIF	RF	1.0	0.974	0.569	0.999	0.926	0.823	1.0	0.973	0.71	1.0	0.973
RF_SMOTE_PI_VIF	RF	1.0	0.974	0.631	0.996	0.806	0.949	1.0	0.972	0.987	1.0	0.973
RF_SMOTE_SHAP_VIF	RF	1.0	0.974	0.602	1.0	0.974	0.868	1.0	0.973	0.696	1.0	0.973
RF_bal_FeatImp_VIF	RF	1.0	0.929	0.564	1.0	0.929	0.809	1.0	0.928	0.689	1.0	0.928
RF_bal_PI_VIF	RF	1.0	0.929	0.755	0.923	0.872	0.849	1.0	0.928	0.959	1.0	0.928
RF_bal_SHAP_VIF	RF	1.0	0.929	0.643	1.0	0.904	0.881	1.0	0.929	0.643	1.0	0.904
XGB_RUS_FeatImp	XGB	0.99	0.974	0.515	0.98	0.97	0.813	0.99	0.974	0.9	0.99	0.974
XGB_RUS_PI	XGB	0.99	0.974	1.025	0.95	0.909	1.025	0.95	0.909	1.062	0.98	0.922
XGB_RUS_SHAP	XGB	0.99	0.974	0.513	0.97	0.923	0.801	0.99	0.973	0.941	0.99	0.974
XGB_SMOTE_FeatImp	XGB	1.0	0.976	0.695	0.997	0.927	0.802	0.998	0.925	0.985	1.0	0.975
XGB_SMOTE_PI	XGB	1.0	0.976	0.539	0.992	0.923	0.847	0.997	0.901	1.0	1.0	0.975
XGB_SMOTE_SHAP	XGB	1.0	0.976	0.525	0.999	0.974	0.816	1.0	0.976	0.961	1.0	0.976
XGB_bal_FeatImp	XGB	1.0	0.976	0.536	0.998	0.974	0.81	0.999	0.976	0.964	1.0	0.976
XGB_bal_PI	XGB	1.0	0.976	0.772	0.93	0.897	0.881	0.997	0.925	1.0	1.0	0.976
XGB_bal_SHAP	XGB	1.0	0.976	0.516	1.0	0.928	0.811	1.0	0.976	0.846	0.999	0.976
XGB_RUS_FeatImp_VIF	XGB	1.0	0.973	0.869	0.92	0.811	0.869	0.92	0.811	0.96	1.0	0.971
XGB_RUS_PI_VIF	XGB	1.0	0.973	0.5	0.94	0.881	0.926	0.99	0.968	0.98	0.99	0.971
XGB_RUS_SHAP_VIF	XGB	1.0	0.973	0.519	0.99	0.968	0.859	1.0	0.971	0.652	1.0	0.971
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.974	0.672	0.993	0.854	0.838	1.0	0.927	0.895	1.0	0.926
XGB_SMOTE_PI_VIF	XGB	1.0	0.974	0.879	0.996	0.853	0.879	0.996	0.853	0.994	1.0	0.95
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.974	0.607	1.0	0.973	0.816	1.0	0.951	0.745	1.0	0.95
XGB_bal_FeatImp_VIF	XGB	1.0	0.976	0.65	0.999	0.949	0.912	0.999	0.926	0.798	0.999	0.949
XGB_bal_PI_VIF	XGB	1.0	0.976	0.5	0.928	0.868	0.862	0.999	0.949	0.991	1.0	0.951
XGB_bal_SHAP_VIF	XGB	1.0	0.976	0.587	0.999	0.926	0.814	1.0	0.951	0.707	1.0	0.951

Table 6.14: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Satimage-2*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.889	0.535	1.0	0.698	0.815	1.0	0.817	0.688	1.0	0.73
RF_RUS_PI	RF	1.0	0.889	0.77	1.0	0.556	0.887	1.0	0.698	0.954	1.0	0.81
RF_RUS_SHAP	RF	1.0	0.889	0.574	1.0	0.698	0.855	1.0	0.865	0.737	1.0	0.81
RF_SMOTE_FeatImp	RF	1.0	0.897	0.545	0.997	0.706	0.833	1.0	0.778	0.707	1.0	0.786
RF_SMOTE_PI	RF	1.0	0.897	0.8	0.993	0.635	0.8	0.993	0.635	0.938	1.0	0.786
RF_SMOTE_SHAP	RF	1.0	0.897	0.571	0.997	0.738	0.832	1.0	0.841	0.723	1.0	0.706
RF_bal_FeatImp	RF	1.0	0.714	0.589	1.0	0.492	0.853	1.0	0.603	0.735	1.0	0.476
RF_bal_PI	RF	1.0	0.714	0.798	1.0	0.46	0.902	1.0	0.603	0.962	1.0	0.714
RF_bal_SHAP	RF	1.0	0.714	0.581	1.0	0.706	0.836	1.0	0.603	0.722	1.0	0.714
RF_RUS_FeatImp_VIF	RF	1.0	0.881	0.622	1.0	0.698	0.914	1.0	0.81	0.622	1.0	0.698
RF_RUS_PI_VIF	RF	1.0	0.881	0.8	1.0	0.556	0.877	1.0	0.698	0.877	1.0	0.698
RF_RUS_SHAP_VIF	RF	1.0	0.881	0.629	1.0	0.698	0.913	1.0	0.81	0.629	1.0	0.698
RF_SMOTE_FeatImp_VIF	RF	1.0	0.944	0.516	1.0	0.698	0.864	1.0	0.817	0.516	1.0	0.698
RF_SMOTE_PI_VIF	RF	1.0	0.944	0.692	1.0	0.556	0.808	1.0	0.825	0.808	1.0	0.825
RF_SMOTE_SHAP_VIF	RF	1.0	0.944	0.565	1.0	0.825	0.87	1.0	0.825	0.565	1.0	0.825
RF_bal_FeatImp_VIF	RF	1.0	0.714	0.615	1.0	0.492	0.911	1.0	0.659	0.615	1.0	0.492
RF_bal_PI_VIF	RF	1.0	0.714	0.771	1.0	0.46	0.879	1.0	0.603	0.879	1.0	0.603
RF_bal_SHAP_VIF	RF	1.0	0.714	0.607	1.0	0.651	0.895	1.0	0.659	0.607	1.0	0.651
XGB_RUS_FeatImp	XGB	1.0	0.865	0.54	1.0	0.706	0.802	1.0	0.794	0.687	1.0	0.714
XGB_RUS_PI	XGB	1.0	0.865	0.587	0.929	0.587	0.852	1.0	0.659	0.852	1.0	0.659
XGB_RUS_SHAP	XGB	1.0	0.865	0.624	1.0	0.714	0.887	1.0	0.802	0.624	1.0	0.714
XGB_SMOTE_FeatImp	XGB	0.915	0.857	0.558	0.85	0.857	0.827	0.908	0.794	0.693	0.901	0.841
XGB_SMOTE_PI	XGB	0.915	0.857	0.751	0.816	0.722	0.851	0.85	0.857	0.921	0.905	0.786
XGB_SMOTE_SHAP	XGB	0.915	0.857	0.54	0.816	0.722	0.827	0.905	0.786	0.827	0.905	0.786
XGB_bal_FeatImp	XGB	0.956	0.929	0.511	0.946	0.77	0.806	0.956	0.865	0.681	0.956	0.857
XGB_bal_PI	XGB	0.956	0.929	0.82	0.881	0.706	0.82	0.881	0.706	0.972	0.956	0.857
XGB_bal_SHAP	XGB	0.956	0.929	0.614	0.922	0.77	0.886	0.956	0.849	0.751	0.956	0.857
XGB_RUS_FeatImp_VIF	XGB	1.0	0.825	0.64	0.976	0.675	0.808	1.0	0.675	0.64	0.976	0.675
XGB_RUS_PI_VIF	XGB	1.0	0.825	0.7	0.929	0.587	0.912	0.976	0.675	0.912	0.976	0.675
XGB_RUS_SHAP_VIF	XGB	1.0	0.825	0.63	0.976	0.675	0.821	1.0	0.77	0.63	0.976	0.675
XGB_SMOTE_FeatImp_VIF	XGB	1.0	0.857	0.724	1.0	0.683	0.862	1.0	0.722	0.724	1.0	0.683
XGB_SMOTE_PI_VIF	XGB	1.0	0.857	0.762	0.99	0.548	0.866	1.0	0.817	0.866	1.0	0.817
XGB_SMOTE_SHAP_VIF	XGB	1.0	0.857	0.631	1.0	0.817	0.92	1.0	0.905	0.631	1.0	0.817
XGB_bal_FeatImp_VIF	XGB	0.908	0.881	0.523	0.867	0.69	0.898	0.888	0.873	0.724	0.925	0.762
XGB_bal_PI_VIF	XGB	0.908	0.881	0.646	0.867	0.69	0.883	0.888	0.873	0.783	0.925	0.762
XGB_bal_SHAP_VIF	XGB	0.908	0.881	0.678	0.844	0.81	0.886	0.888	0.873	0.678	0.844	0.81

Table 6.15: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Vertebral*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.911	0.559	1.0	0.889	0.85	1.0	0.911	0.727	1.0	0.898
RF_RUS_PI	RF	1.0	0.911	0.887	1.0	0.823	0.887	1.0	0.823	1.0	1.0	0.907
RF_RUS_SHAP	RF	1.0	0.911	0.514	1.0	0.881	0.833	1.0	0.893	0.768	1.0	0.9
RF_SMOTE_FeatImp	RF	1.0	0.799	0.559	1.0	0.867	0.818	1.0	0.731	0.864	1.0	0.731
RF_SMOTE_PI	RF	1.0	0.799	0.509	1.0	0.771	0.828	1.0	0.867	1.0	1.0	0.732
RF_SMOTE_SHAP	RF	1.0	0.799	0.6	1.0	0.821	0.853	1.0	0.733	0.853	1.0	0.733
RF_bal_FeatImp	RF	1.0	0.733	0.534	1.0	0.665	0.845	1.0	0.767	0.845	1.0	0.767
RF_bal_PI	RF	1.0	0.733	0.857	1.0	0.665	0.857	1.0	0.665	1.0	1.0	0.733
RF_bal_SHAP	RF	1.0	0.733	0.652	1.0	0.833	0.805	1.0	0.7	0.876	1.0	0.733
RF_RUS_FeatImp_VIF	RF	1.0	0.933	0.592	1.0	0.89	0.802	1.0	0.929	0.802	1.0	0.929
RF_RUS_PI_VIF	RF	1.0	0.933	0.794	1.0	0.823	0.909	1.0	0.859	1.0	1.0	0.96
RF_RUS_SHAP_VIF	RF	1.0	0.933	0.514	1.0	0.859	0.852	1.0	0.929	0.852	1.0	0.929
RF_SMOTE_FeatImp_VIF	RF	1.0	0.767	0.57	1.0	0.782	0.863	1.0	0.832	0.797	1.0	0.863
RF_SMOTE_PI_VIF	RF	1.0	0.767	0.673	1.0	0.715	0.854	1.0	0.781	0.986	1.0	0.93
RF_SMOTE_SHAP_VIF	RF	1.0	0.767	0.52	1.0	0.897	0.858	1.0	0.765	0.795	1.0	0.83
RF_bal_FeatImp_VIF	RF	1.0	0.733	0.579	1.0	0.633	0.822	1.0	0.833	0.822	1.0	0.833
RF_bal_PI_VIF	RF	1.0	0.733	0.656	1.0	0.598	0.809	1.0	0.565	0.974	1.0	0.833
RF_bal_SHAP_VIF	RF	1.0	0.733	0.608	1.0	0.565	0.849	1.0	0.8	0.849	1.0	0.8
XGB_RUS_FeatImp	XGB	1.0	0.908	0.616	0.914	0.88	0.822	0.971	0.838	0.882	0.971	0.88
XGB_RUS_PI	XGB	1.0	0.908	0.607	0.914	0.841	0.87	0.914	0.874	0.966	1.0	0.859
XGB_RUS_SHAP	XGB	1.0	0.908	0.532	0.914	0.874	0.829	0.971	0.905	0.829	0.971	0.905
XGB_SMOTE_FeatImp	XGB	0.998	0.933	0.517	0.954	0.868	0.827	0.995	0.898	0.827	0.995	0.898
XGB_SMOTE_PI	XGB	0.998	0.933	0.543	0.827	0.779	0.937	0.982	0.891	1.0	0.998	0.899
XGB_SMOTE_SHAP	XGB	0.998	0.933	0.555	0.964	0.87	0.894	0.998	0.899	0.894	0.998	0.899
XGB_bal_FeatImp	XGB	1.0	0.933	0.72	0.999	0.795	0.864	1.0	0.763	0.959	1.0	0.832
XGB_bal_PI	XGB	1.0	0.933	0.746	1.0	0.586	0.894	1.0	0.862	1.0	1.0	0.9
XGB_bal_SHAP	XGB	1.0	0.933	0.553	1.0	0.758	0.853	1.0	0.9	0.853	1.0	0.9
XGB_RUS_FeatImp_VIF	XGB	1.0	0.963	0.589	0.986	0.89	0.832	1.0	0.967	0.674	1.0	0.959
XGB_RUS_PI_VIF	XGB	1.0	0.963	0.543	0.814	0.826	0.841	0.914	0.871	1.0	1.0	0.955
XGB_RUS_SHAP_VIF	XGB	1.0	0.963	0.562	0.914	0.871	0.861	1.0	0.955	0.922	1.0	0.968
XGB_SMOTE_FeatImp_VIF	XGB	0.996	0.962	0.595	0.922	0.902	0.822	0.978	0.911	0.822	0.978	0.911
XGB_SMOTE_PI_VIF	XGB	0.996	0.962	0.668	0.901	0.891	0.923	0.962	0.904	0.984	0.985	0.889
XGB_SMOTE_SHAP_VIF	XGB	0.996	0.962	0.602	0.962	0.904	0.824	0.985	0.889	0.824	0.985	0.889
XGB_bal_FeatImp_VIF	XGB	0.993	0.962	0.583	0.956	0.784	0.848	0.988	0.927	0.778	0.99	0.861
XGB_bal_PI_VIF	XGB	0.993	0.962	0.523	0.902	0.804	0.829	0.961	0.828	0.982	0.988	0.858
XGB_bal_SHAP_VIF	XGB	0.993	0.962	0.627	0.961	0.828	0.907	0.99	0.861	0.907	0.99	0.861

Table 6.16: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Vowels*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

		all feat.		50% subset			80% subset			half subset		
feat_sel_comb	core_cl	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc	acc_feat_imp	tr_roc	ts_roc
RF_RUS_FeatImp	RF	1.0	0.899	0.549	1.0	0.805	0.812	1.0	0.849	0.759	1.0	0.857
RF_RUS_PI	RF	1.0	0.899	0.919	0.993	0.721	0.919	0.993	0.721	1.0	1.0	0.843
RF_RUS_SHAP	RF	1.0	0.899	0.575	1.0	0.769	0.822	1.0	0.885	0.845	1.0	0.881
RF_SMOTE_FeatImp	RF	1.0	0.773	0.539	1.0	0.66	0.814	1.0	0.689	0.814	1.0	0.689
RF_SMOTE_PI	RF	1.0	0.773	0.598	0.986	0.584	0.803	1.0	0.743	0.997	1.0	0.738
RF_SMOTE_SHAP	RF	1.0	0.773	0.529	1.0	0.69	0.819	1.0	0.687	0.793	1.0	0.668
RF_bal_FeatImp	RF	1.0	0.633	0.525	1.0	0.496	0.817	1.0	0.583	0.84	1.0	0.583
RF_bal_PI	RF	1.0	0.633	0.79	0.954	0.56	0.937	1.0	0.496	0.998	1.0	0.599
RF_bal_SHAP	RF	1.0	0.633	0.551	1.0	0.514	0.811	1.0	0.599	0.779	1.0	0.582
RF_RUS_FeatImp_VIF	RF	1.0	0.899	0.549	1.0	0.805	0.812	1.0	0.849	0.759	1.0	0.857
RF_RUS_PI_VIF	RF	1.0	0.899	0.919	0.993	0.721	0.919	0.993	0.721	1.0	1.0	0.843
RF_RUS_SHAP_VIF	RF	1.0	0.899	0.575	1.0	0.769	0.822	1.0	0.885	0.845	1.0	0.881
RF_SMOTE_FeatImp_VIF	RF	1.0	0.773	0.539	1.0	0.66	0.814	1.0	0.689	0.814	1.0	0.689
RF_SMOTE_PI_VIF	RF	1.0	0.773	0.598	0.986	0.584	0.803	1.0	0.743	0.997	1.0	0.738
RF_SMOTE_SHAP_VIF	RF	1.0	0.773	0.529	1.0	0.69	0.819	1.0	0.687	0.793	1.0	0.668
RF_bal_FeatImp_VIF	RF	1.0	0.633	0.525	1.0	0.496	0.817	1.0	0.583	0.84	1.0	0.583
RF_bal_PI_VIF	RF	1.0	0.633	0.79	0.954	0.56	0.937	1.0	0.496	0.998	1.0	0.599
RF_bal_SHAP_VIF	RF	1.0	0.633	0.551	1.0	0.514	0.811	1.0	0.599	0.779	1.0	0.582
XGB_RUS_FeatImp	XGB	1.0	0.876	0.545	0.986	0.732	0.804	0.993	0.796	0.868	1.0	0.798
XGB_RUS_PI	XGB	1.0	0.876	0.549	0.943	0.747	0.86	0.993	0.747	0.977	0.986	0.843
XGB_RUS_SHAP	XGB	1.0	0.876	0.551	0.986	0.8	0.827	0.993	0.799	0.827	0.993	0.799
XGB_SMOTE_FeatImp	XGB	0.981	0.867	0.51	0.938	0.755	0.815	0.976	0.827	0.734	0.969	0.809
XGB_SMOTE_PI	XGB	0.981	0.867	0.627	0.84	0.8	0.802	0.898	0.768	0.967	0.972	0.809
XGB_SMOTE_SHAP	XGB	0.981	0.867	0.561	0.908	0.791	0.812	0.964	0.822	0.862	0.971	0.825
XGB_bal_FeatImp	XGB	0.992	0.865	0.531	0.97	0.736	0.812	0.986	0.811	0.716	0.986	0.823
XGB_bal_PI	XGB	0.992	0.865	0.768	0.887	0.772	0.917	0.946	0.721	0.997	0.985	0.825
XGB_bal_SHAP	XGB	0.992	0.865	0.538	0.968	0.788	0.813	0.987	0.807	0.813	0.987	0.807
XGB_RUS_FeatImp_VIF	XGB	1.0	0.876	0.545	0.986	0.732	0.804	0.993	0.796	0.868	1.0	0.798
XGB_RUS_PI_VIF	XGB	1.0	0.876	0.549	0.943	0.747	0.86	0.993	0.747	0.977	0.986	0.843
XGB_RUS_SHAP_VIF	XGB	1.0	0.876	0.551	0.986	0.8	0.827	0.993	0.799	0.827	0.993	0.799
XGB_SMOTE_FeatImp_VIF	XGB	0.981	0.867	0.51	0.938	0.755	0.815	0.976	0.827	0.734	0.969	0.809
XGB_SMOTE_PI_VIF	XGB	0.981	0.867	0.627	0.84	0.8	0.802	0.898	0.768	0.967	0.972	0.809
XGB_SMOTE_SHAP_VIF	XGB	0.981	0.867	0.561	0.908	0.791	0.812	0.964	0.822	0.862	0.971	0.825
XGB_bal_FeatImp_VIF	XGB	0.992	0.865	0.531	0.97	0.736	0.812	0.986	0.811	0.716	0.986	0.823
XGB_bal_PI_VIF	XGB	0.992	0.865	0.768	0.887	0.772	0.917	0.946	0.721	0.997	0.985	0.825
XGB_bal_SHAP_VIF	XGB	0.992	0.865	0.538	0.968	0.788	0.813	0.987	0.807	0.813	0.987	0.807

Table 6.17: Train and test ROC AUC scores on the selected 50%, 80%, and half feature subsets for *Waveform*. The assessment of subset quality is conducted by using the combination's core classifier. The cumulative feature importance of the subsets is also visible.

List of Figures

List of Tables

List of Algorithms

Acronyms

ADASYN Adaptive Synthetic Sampling Approach. 30, 83

ANOVA Analysis of Variance. 11

Cluster OSS Cluster-based Oversampling and Undersampling. 29

CNN Condensed Nearest Neighbor Rule. 29, 83

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 15, 16

DP Dynamic Programming. 20, 21

DSUS Density-based-Safe-level Undersampling. 29

FAR False Alarm Rate. 26–28

FindOut Find Outliers. 16

FPR False Positive Rate. 55

GA Genetic Algorithms. 12

GCRL Goal-Conditional Reinforcement Learning. 21

ICA Independent Component Analysis. 14

IDS Intrusion Detection System. 26, 28

IRUS Instance Reduction by Undersampling Synthesis. 29

LIME Local Interpretable Model-agnostic Explanations. 37

MARS Multivariate Adaptive Regression Spline. 13

MC Monte Carlo. 20, 21

MDI Mean Decrease in Impurity. vii, ix, 4, 42, 43, 47, 48, 52, 56, 60, 72, 75, 78, 81–83

MDP Markov Decision Process. 20

MIC Maximal Information Coefficient. 11

NCR Neighborhood Cleaning Rule. 29

OSS One-Sided Selection. 29, 83

PCA Principal Component Analysis. 14, 17, 48, 49

PI Permutation Feature Importance. vii, ix, 3, 4, 33, 34, 37, 38, 42, 43, 48, 52, 56, 60, 72, 75, 78, 81–83

PLS Partial Least Squares. 48

RF Random Forest. 4, 22–24, 31, 36, 42, 43, 45–47, 51, 52, 56, 57, 60, 66, 71, 75, 78, 79, 81–83, 105

RF_bal Random Forest Balanced Cost-Sensitive Version. 4, 42, 43, 51, 57, 75, 82

RFE Recursive Feature Elimination. 12

RL Reinforcement Learning. 19–21

ROC AUC Receiver Operating Characteristic Area Under the Curve. 5, 43, 52, 55–64, 66–72, 75, 76, 79–83, 85–103, 105, 106

ROCK Robust Clustering using Links. 16

ROS Random Oversampling. 29

RUS Random Undersampling. vii, ix, 4, 29, 42, 43, 46, 56, 60, 71, 75, 81–83

SA Simulated Annealing. 12

SARSA State-Action-Reward-State-Action. 21

SHAP SHapley Additive exPlanations. vii, ix, 4, 37–39, 42, 43, 48, 52, 56, 60, 72, 75, 78, 81–83

SMOTE Synthetic Minority Oversampling Technique. vii, ix, 4, 29, 30, 42, 43, 46, 56, 60, 75, 81–83

SNN Shared Nearest Neighbor. 16

SSL Semi-Supervised Learning. 17–19

SVD Singular Value Decomposition. 14

SVM Suppor Vector Machine. 15, 19

TD Temporal Differences. 21

TNR True Negative Rate. 54

TPR True Positive Rate. 54, 55

VIF Variance Inflation Factor. vii, ix, 4, 42, 43, 48, 49, 52, 56, 60, 61, 66, 75–77, 80–83, 104, 105

XAI eXplainable Artificial Intelligence. 2

XGB XGBoost. 4, 23, 31, 42, 43, 45–47, 51, 52, 56, 57, 60, 71, 75, 78, 79, 81–83, 105

XGB_bal XGBoost Balanced Cost-Sensitive Version. 4, 42, 43, 51, 57, 75, 82, 83

Bibliography

[1] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[2] Richard Bellman and Robert Kalaba. "On adaptive control processes". In: *IRE Transactions on Automatic Control* 4.2 (1959), pp. 1–9.

[3] Rok Blagus and Lara Lusa. "SMOTE for high-dimensional class-imbalanced data". In: *BMC bioinformatics* 14 (2013), pp. 1–16.

[4] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. "Experimental perspectives on learning from imbalanced data". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 935–942.

[5] Nadeem Qazi and Kamran Raza. "Effect of feature selection, SMOTE and under sampling on class imbalance classification". In: *2012 UKSim 14th International Conference on Computer Modelling and Simulation*. IEEE. 2012, pp. 145–150.

[6] Tawfiq Hasanin et al. "Investigating random undersampling and feature selection on bioinformatics big data". In: *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE. 2019, pp. 346–356.

[7] Ismael Ramos-Pérez et al. "When is resampling beneficial for feature selection with imbalanced wide data?" In: *Expert Systems with Applications* 188 (2022), p. 116015.

[8] Firuz Kamalov, Amir F Atiya, and Dina Elreedy. "Partial resampling of imbalanced data". In: *arXiv preprint arXiv:2207.04631* (2022).

[9] Guo Haixiang et al. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.12.035. URL: https://www.sciencedirect.com/science/article/pii/S0957417416307175.

[10] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable ai: A review of machine learning interpretability methods". In: *Entropy* 23.1 (2020), p. 18.

[11] Jie Zhang and Zong-ming Zhang. “Ethics and governance of trustworthy medical artificial intelligence”. In: *BMC Medical Informatics and Decision Making* 23.1 (2023), p. 7.

[12] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems* 30 (2017).

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[14] Harsha Nori et al. “Interpretml: A unified framework for machine learning interpretability”. In: *arXiv preprint arXiv:1909.09223* (2019).

[15] Erjola Zeraliu. *Comparison of ensemble-based feature selection methods for binary classification of imbalanced data sets*. eng. Wien, 2021.

[16] Nicolai Meinshausen and Peter Bühlmann. “Stability selection”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 72.4 (2010), pp. 417–473.

[17] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[18] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[19] Donald E Farrar and Robert R Glauber. “Multicollinearity in regression analysis: the problem revisited”. In: *The Review of Economic and Statistics* (1967), pp. 92–107.

[20] Sotiris B Kotsiantis. “Decision trees: a recent overview”. In: *Artificial Intelligence Review* 39 (2013), pp. 261–283.

[21] Ming Li et al. “Estimating annual runoff in response to forest change: A statistical method based on random forest”. In: *Journal of Hydrology* 589 (2020), p. 125168.

[22] L Ladha and T Deepa. “Feature selection methods and algorithms”. In: *International journal on computer science and engineering* 3.5 (2011), pp. 1787–1797.

[23] Jundong Li et al. “Feature selection: A data perspective”. In: *ACM computing surveys (CSUR)* 50.6 (2017), pp. 1–45.

[24] Vipin Kumar and Sonajharia Minz. “Feature selection: a literature review”. In: *SmartCR* 4.3 (2014), pp. 211–229.

[25] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media, 2012.

[26] Narendra and Fukunaga. “A branch and bound algorithm for feature subset selection”. In: *IEEE Transactions on computers* 100.9 (1977), pp. 917–922.

[27] Justin Doak. “An evaluation of feature selection methods and their application to computer security”. In: *Techninal Report CSE-92-18* (1992).

[28] Gilles Brassard and Paul Bratley. *Fundamentals of algorithmics*. Prentice-Hall, Inc., 1996.

[29] Hussein Almuallim and Thomas G Dietterich. "Learning boolean concepts in the presence of many irrelevant features". In: *Artificial intelligence* 69.1-2 (1994), pp. 279–305.

[30] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[31] Anil Bhattacharyya. "On a measure of divergence between two statistical populations defined by their probability distribution". In: *Bulletin of the Calcutta Mathematical Society* 35 (1943), pp. 99–110.

[32] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[33] Mark A Hall. "Correlation-based feature selection for machine learning". PhD thesis. The University of Waikato, 1999.

[34] Mark A Hall. "Correlation-based feature selection of discrete and numeric class machine learning". In: (2000).

[35] M.S. Raza and U. Qamar. *Understanding and Using Rough Set Based Feature Selection: Concepts, Techniques and Applications*. Springer Nature Singapore, 2019. ISBN: 9789813291669. URL: https://books.google.at/books?id=OOqqDwAAQBAJ.

[36] Lei Xu, Pingfan Yan, and Tong Chang. "Best first strategy for feature selection". In: *9th international conference on pattern recognition*. IEEE Computer Society. 1988, pp. 706–707.

[37] Leon Bobrowski. "Feature selection based on some homogeneity coefficient". In: *[1988 Proceedings] 9th International Conference on Pattern Recognition*. IEEE. 1988, pp. 544–546.

[38] Pat Langley and Stephanie Sage. "Oblivious decision trees and abstract cases". In: *Working notes of the AAAI-94 workshop on case-based reasoning*. Seattle, WA. 1994, pp. 113–117.

[39] Kenji Kira and Larry A Rendell. "The feature selection problem: Traditional methods and a new algorithm". In: *Proceedings of the tenth national conference on Artificial intelligence*. 1992, pp. 129–134.

[40] Huan Liu, Hiroshi Motoda, and Lei Yu. "Feature selection with selective sampling". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. 2002, pp. 395–402.

[41] Pavel Pudil and J Hovovicova. "Novel methods for subset selection with respect to problem knowledge". In: *IEEE Intelligent Systems and their Applications* 13.2 (1998), pp. 66–74.

[42] Jakub Segen. “Feature selection and constructive inference”. In: *Proceedings of Seventh International Conference on Pattern Recognition*. 1984, pp. 1344–1346.

[43] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846. URL: https://books.google.at/books?id=eBSgoAEACAAJ.

[44] Jacob Sheinvald, Byron Dom, and Wayne Niblack. “A modeling approach to feature selection”. In: *[1990] Proceedings. 10th International Conference on Pattern Recognition*. Vol. 1. IEEE. 1990, pp. 535–539.

[45] Claire Cardie. “Using decision trees to improve case-based learning”. In: *Proceedings of the tenth international conference on machine learning*. 1993, pp. 25–32.

[46] Daphne Koller, Mehran Sahami, et al. “Toward optimal feature selection”. In: *ICML*. Vol. 96. 28. 1996, p. 292.

[47] Lei Yu and Huan Liu. “Feature selection for high-dimensional data: A fast correlation-based filter solution”. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003, pp. 856–863.

[48] Rich Caruana and Dayne Freitag. “Greedy attribute selection”. In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 28–36.

[49] Alan Miller. *Subset selection in regression*. CRC Press, 2002.

[50] Anthony N Mucciardi and Earl E Gose. “A comparison of seven techniques for choosing subsets of pattern recognition properties”. In: *IEEE Transactions on Computers* 100.9 (1971), pp. 1023–1031.

[51] Noam Slonim et al. “Discriminative feature selection via multiclass variable memory Markov model”. In: *EURASIP Journal on Advances in Signal Processing* 2003 (2003), pp. 1–10.

[52] Hussein Almuallim, Thomas G Dietterich, et al. “Learning With Many Irrelevant Features.” In: *AAAI*. Vol. 91. Citeseer. 1991, pp. 547–552.

[53] Huan Liul, Hiroshi Motoda, and Manoranjan Dash. “A monotonic measure for optimal feature selection”. In: *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings 10*. Springer. 1998, pp. 101–106.

[54] Arlindo L Oliveira and Alberto Sangiovanni-Vincentelli. “Constructive induction using a non-greedy strategy for feature selection”. In: *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 355–360.

[55] Yong Rui, Thomas S Huang, and Shih-Fu Chang. “Image retrieval: Current techniques, promising directions, and open issues”. In: *Journal of visual communication and image representation* 10.1 (1999), pp. 39–62.

[56] Manoranjan Dash. “Feature selection via set cover”. In: *Proceedings 1997 IEEE knowledge and data engineering exchange workshop*. IEEE. 1997, pp. 165–171.

[57] Nick Littlestone. “Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm”. In: *Machine learning* 2 (1988), pp. 285–318.

[58] Huan Liu, Rudy Setiono, et al. “A probabilistic approach to feature selection-a filter solution”. In: *ICML*. Vol. 96. 1996, pp. 319–327.

[59] Iman Foroutan and Jack Sklansky. “Feature selection for automatic classification of non-gaussian data”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 17.2 (1987), pp. 187–198.

[60] M Ichino and J Sklansky. “Feature selection for linear classifier”. In: *Proceedings of the Seventh International Conference on Pattern Recognition*. Vol. 1. 1984, pp. 124–127.

[61] Manabu Ichino and Jack Sklansky. “Optimum feature selection by zero-one integer programming”. In: *IEEE transactions on systems, man, and cybernetics* 5 (1984), pp. 737–746.

[62] Pierre A Devijver and Josef Kittler. “Pattern recognition: A statistical approach”. In: *(No Title)* (1982).

[63] Pedro M Domingos. “Why Does Bagging Work? A Bayesian Account and its Implications.” In: *KDD*. Citeseer. 1997, pp. 155–158.

[64] Andrew W Moore and Mary S Lee. “Efficient algorithms for minimizing cross validation error”. In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 190–198.

[65] Moninder Singh and Gregory M Provan. “Efficient learning of selective Bayesian network classifiers”. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. 1996, pp. 453–461.

[66] Huan Liu and Rudy Setiono. “Feature selection and classification–a probabilistic wrapper approach”. In: *Industrial and Engineering Applications or Artificial Intelligence and Expert Systems*. CRC Press, 2022, pp. 419–424.

[67] David B Skalak. “Prototype and feature selection by sampling and random mutation hill climbing algorithms”. In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 293–301.

[68] Haleh Vafaie, Ibrahim F Imam, et al. “Feature selection methods: genetic algorithms vs. greedy-like search”. In: *Proceedings of the international conference on fuzzy and intelligent control systems*. Vol. 51. 1994, p. 28.

[69] David J Stracuzzi and Paul E Utgoff. “Randomized variable elimination”. In: *The Journal of Machine Learning Research* 5 (2004), pp. 1331–1362.

[70] Sanmay Das. “Filters, wrappers and a boosting-based hybrid for feature selection”. In: *Icml*. Vol. 1. Citeseer. 2001, pp. 74–81.

[71] Steven Loscalzo, Lei Yu, and Chris Ding. "Consensus group stable feature selection". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 567–576.

[72] M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman & Hall/CRC Data Science Series. CRC Press, 2019. ISBN: 9781351609463. URL: https://books.google.at/books?id=q5alDwAAQBAJ.

[73] Jiliang Tang, Salem Alelyani, and Huan Liu. "Feature selection for classification: A review". In: *Data classification: Algorithms and applications* (2014), p. 37.

[74] Quanquan Gu and Jiawei Han. "Towards feature selection in network". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. 2011, pp. 1175–1184.

[75] Jiliang Tang and Huan Liu. "Feature selection with linked data in social media". In: *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM. 2012, pp. 118–128.

[76] Jundong Li et al. "Robust unsupervised feature selection on networked data". In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM. 2016, pp. 387–395.

[77] Gregory Faletto and Jacob Bien. "Cluster Stability Selection". In: *arXiv preprint arXiv:2201.00494* (2022).

[78] Gao Wang et al. "A simple new approach to variable selection in regression, with application to genetic fine mapping". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.5 (2020), pp. 1273–1300.

[79] Félix Iglesias and Tanja Zseby. "Analysis of network traffic features for anomaly detection". In: *Machine Learning* 101 (2015), pp. 59–84.

[80] Isabelle Guyon and André Elisseeff. "An introduction to feature extraction". In: *Feature extraction: foundations and applications*. Springer, 2006, pp. 1–25.

[81] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. "A survey of feature selection and feature extraction techniques in machine learning". In: *2014 science and information conference*. IEEE. 2014, pp. 372–378.

[82] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN: 9781999579517. URL: https://books.google.at/books?id=0jbxwQEACAAJ.

[83] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques". In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[84] Douglas M Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.

[85] Félix Iglesias Vázquez et al. "Anomaly detection in streaming data: A comparison and evaluation study". In: *Expert Systems with Applications* 233 (2023), p. 120994.

[86] Markus Goldstein and Seiichi Uchida. “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data”. In: *PloS one* 11.4 (2016), e0152173.

[87] Lukas Ruff et al. “A unifying review of deep and shallow anomaly detection”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 756–795.

[88] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[89] Roland Kwitt and Ulrich Hofmann. “Unsupervised anomaly detection in network traffic by means of robust PCA”. In: *2007 International Multi-Conference on Computing in the Global Information Technology (ICCGI'07)*. IEEE. 2007, pp. 37–37.

[90] Félix Iglesias et al. “Are network attacks outliers? A study of space representations and unsupervised algorithms”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 159–175.

[91] Jesper E Van Engelen and Holger H Hoos. “A survey on semi-supervised learning”. In: *Machine learning* 109.2 (2020), pp. 373–440.

[92] Mary M Moya and Don R Hush. “Network constraints and multi-objective optimization for one-class classification”. In: *Neural networks* 9.3 (1996), pp. 463–474.

[93] José R Vázquez-Canteli and Zoltán Nagy. “Reinforcement learning for demand response: A review of algorithms and modeling techniques”. In: *Applied energy* 235 (2019), pp. 1072–1089.

[94] Zhuangdi Zhu et al. “Transfer learning in deep reinforcement learning: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[95] Eduardo F Morales and Hugo Jair Escalante. “A brief introduction to supervised, unsupervised, and reinforcement learning”. In: *Biosignal processing and classification using computational learning and intelligence*. Elsevier, 2022, pp. 111–129.

[96] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. “Reinforcement Learning Algorithms: A brief survey”. In: *Expert Systems with Applications* (2023), p. 120495.

[97] Z.H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CHAPMAN & HALL/CRC MACHINE LEA. Taylor & Francis, 2012. ISBN: 9781439830031. URL: https://books.google.at/books?id=BDB50Ev2ur4C.

[98] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019. ISBN: 9781492032618. URL: https://books.google.at/books?id=HHetDwAAQBAJ.

[99] David H Wolpert. “Stacked generalization”. In: *Neural networks* 5.2 (1992), pp. 241–259.

[100] C. Zhang and Y. Ma. *Ensemble Machine Learning: Methods and Applications*. Springer New York, 2012. ISBN: 9781441993250. URL: https://books.google.at/books?id=CjAs4stLXhAC.

[101] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. "A systematic review on imbalanced data challenges in machine learning: Applications and solutions". In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–36.

[102] A. Fernández et al. *Learning from Imbalanced Data Sets*. Springer International Publishing, 2018. ISBN: 9783319980744. URL: https://books.google.at/books?id=8Fp0DwAAQBAJ.

[103] Stefan Axelsson. "The base-rate fallacy and the difficulty of intrusion detection". In: *ACM Transactions on Information and System Security (TISSEC)* 3.3 (2000), pp. 186–205.

[104] Wenke Lee and Salvatore Stolfo. "Data mining approaches for intrusion detection". In: (1998).

[105] Robert F Erbacher. "Base-rate fallacy redux and a deep dive review in cybersecurity". In: *arXiv preprint arXiv:2203.08801* (2022).

[106] Osman Salem et al. "Flooding attacks detection in traffic of backbone networks". In: *2011 IEEE 36th Conference on Local Computer Networks*. IEEE. 2011, pp. 441–449.

[107] A Villa and Elizabeth Varki. "Characterization of a campus internet workload". In: *Proceedings of CATA*. 2012, pp. 140–148.

[108] Piotr Jurkiewicz, Grzegorz Rzym, and Piotr Boryło. "How many mice make an elephant? modelling flow length and size distribution of internet traffic". In: *arXiv preprint arXiv:1809.03486* (2018).

[109] S Prabavathy, K Sundarakantham, and S Mercy Shalinie. "Design of cognitive fog computing for intrusion detection in Internet of Things". In: *Journal of Communications and Networks* 20.3 (2018), pp. 291–298.

[110] Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. "Optimal scheduling of cybersecurity analysts for minimizing risk". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.4 (2017), pp. 1–32.

[111] Ciza Thomas. "Improving intrusion detection for imbalanced network traffic". In: *Security and communication Networks* 6.3 (2013), pp. 309–324.

[112] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee. 2008, pp. 1322–1328.

[113] Mengnan Du, Ninghao Liu, and Xia Hu. "Techniques for interpretable machine learning". In: *Communications of the ACM* 63.1 (2019), pp. 68–77.

[114] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. "Machine learning interpretability: A survey on methods and metrics". In: *Electronics* 8.8 (2019), p. 832.

[115] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Model-agnostic interpretability of machine learning". In: *arXiv preprint arXiv:1606.05386* (2016).

[116] Alex A Freitas. "Comprehensible classification models: a position paper". In: *ACM SIGKDD explorations newsletter* 15.1 (2014), pp. 1–10.

[117] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial intelligence* 267 (2019), pp. 1–38.

[118] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[119] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[120] L. Breiman. *Classification and Regression Trees*. Wadsworth/Thomson Learning, 1984. URL: https://books.google.at/books?id=8E9QAQAACAAJ.

[121] Gilles Louppe. "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502* (2014).

[122] Leo Breiman. "Manual on setting up, using, and understanding random forests v3. 1". In: *Statistics Department University of California Berkeley, CA, USA* 1.58 (2002), pp. 3–42.

[123] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously." In: *J. Mach. Learn. Res.* 20.177 (2019), pp. 1–81.

[124] Giles Hooker, Lucas Mentch, and Siyu Zhou. "Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance". In: *Statistics and Computing* 31 (2021), pp. 1–16.

[125] Lloyd S Shapley et al. "A value for n-person games". In: (1953).

[126] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41 (2014), pp. 647–665.

[127] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. "Consistent individualized feature attribution for tree ensembles". In: *arXiv preprint arXiv:1802.03888* (2018).

[128] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. "Feature relevance quantification in explainable AI: A causal problem". In: *International Conference on artificial intelligence and statistics*. PMLR. 2020, pp. 2907–2916.

[129] Michele Banko and Eric Brill. "Scaling to very very large corpora for natural language disambiguation". In: *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*. 2001, pp. 26–33.

[130] Alon Halevy, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE intelligent systems* 24.2 (2009), pp. 8–12.

[131] Abhinav Jain et al. “Overview and importance of data quality for machine learning tasks”. In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 3561–3562.

[132] Shebuti Rayana. *ODDS Library*. 2016. URL: https://odds.cs.stonybrook.edu.

[133] Guilherme O Campos et al. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. In: *Data mining and knowledge discovery* 30 (2016), pp. 891–927.

[134] Rodolfo Bonnin. *Machine Learning for Developers: Uplift your regular applications with the power of statistics, analytics, and machine learning*. Packt Publishing Ltd, 2017.

[135] Andreas C Müller and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. " O’Reilly Media, Inc.", 2016.

[136] Shovan Chowdhury et al. “Evaluation of tree based regression over multiple linear regression for non-normally distributed data in battery performance”. In: *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. IEEE. 2022, pp. 17–25.

[137] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[138] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[139] Jireh Yi-Le Chan et al. “Mitigating the multicollinearity problem and its machine learning approach: a review”. In: *Mathematics* 10.8 (2022), p. 1283.

[140] Jason W Osborne and Elaine Waters. “Four assumptions of multiple regression that researchers should always test”. In: *Practical assessment, research, and evaluation* 8.1 (2019), p. 2.

[141] Jamal I Daoud. “Multicollinearity and regression analysis”. In: *Journal of Physics: Conference Series*. Vol. 949. 1. IOP Publishing. 2017, p. 012009.

[142] M.H. Kutner. *Applied Linear Statistical Models*. McGrwa-Hill international edition. McGraw-Hill Irwin, 2005. ISBN: 9780071122214. URL: https://books.google.at/books?id=0xqCAAAACAAJ.

[143] Kristina P Vatcheva et al. “Multicollinearity in regression analyses conducted in epidemiologic studies”. In: *Epidemiology (Sunnyvale, Calif.)* 6.2 (2016).

[144] McKee J McClendon. *Multiple regression and causal analysis*. Waveland Press, 2002.

[145] Ranjit Kumar Paul. “Multicollinearity: Causes, effects and remedies”. In: *IASRI, New Delhi* 1.1 (2006), pp. 58–65.

[146] Vincent Abiodun Micheal and Alfred Adewole Abiodun. “Estimation of regression coefficients in the presence of multicollinearity”. In: *Social and Basic Sciences Research Review* 2.10 (2014), pp. 404–415.

[147] Oscar L Olvera Astivia and Edward Kroc. “Centering in multiple regression does not always reduce multicollinearity: How to tell when your estimates will not benefit from centering”. In: *Educational and Psychological Measurement* 79.5 (2019), pp. 813–826.

[148] SQ Lafi and JB Kaneene. “An explanation of the use of principal-components analysis to detect and correct for multicollinearity”. In: *Preventive Veterinary Medicine* 13.4 (1992), pp. 261–275.

[149] Herman Wold. “Soft modelling: the basic design and some extensions”. In: *Systems under indirect observation, Part II* (1982), pp. 36–37.

[150] AE Horel. “Application of ridge analysis to regression problems”. In: *Chemical Engineering Progress* 58 (1962), pp. 54–59.

[151] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.

[152] Charlotte H Mason and William D Perreault Jr. “Collinearity, power, and interpretation of multiple regression analysis”. In: *Journal of marketing research* 28.3 (1991), pp. 268–280.

[153] Sanford Weisberg. *Applied linear regression*. Vol. 528. John Wiley & Sons, 2005.

[154] Takuya Akiba et al. “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[155] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert systems with applications* 73 (2017), pp. 220–239.

[156] Haseeb Ali et al. “Imbalance class problems in data mining: A review”. In: *Indonesian Journal of Electrical Engineering and Computer Science* 14.3 (2019), pp. 1560–1571.

[157] Boutkhoum Omar et al. “Minimizing the overlapping degree to improve class-imbalanced learning under sparse feature selection: application to fraud detection”. In: *IEEE Access* 9 (2021), pp. 28101–28110.

[158] Adnan Amin et al. “Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study”. In: *Ieee Access* 4 (2016), pp. 7940–7957.

[159] Anjana Gosain and Saanchi Sardana. “Handling class imbalance problem using oversampling techniques: A review”. In: *2017 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2017, pp. 79–85.

[160] Matloob Khushi et al. “A comparative performance analysis of data resampling methods on imbalance medical data”. In: *IEEE Access* 9 (2021), pp. 109960–109975.

[161] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. “A multiple resampling method for learning from imbalanced data sets”. In: *Computational intelligence* 20.1 (2004), pp. 18–36.

[162] Ricardo Barandela et al. “The imbalanced training sample problem: Under or over sampling?” In: *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004. Proceedings*. Springer. 2004, pp. 806–814.

[163] Zhongbin Sun et al. “A novel ensemble method for classifying imbalanced data”. In: *Pattern Recognition* 48.5 (2015), pp. 1623–1637.

[164] Zhongbin Sun, Qinbao Song, and Xiaoyan Zhu. “Using coding-based ensemble learning to improve software defect prediction”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1806–1817.

[165] Noora Shrestha. “Detecting multicollinearity in regression analysis”. In: *American Journal of Applied Mathematics and Statistics* 8.2 (2020), pp. 39–42.

[166] Utkarsh Mahadeo Khaire and R Dhanalakshmi. “Stability of feature selection algorithm: A review”. In: *Journal of King Saud University-Computer and Information Sciences* 34.4 (2022), pp. 1060–1073.