Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Bachelor's diploma thesis

in the field of study Data Science

Impact of data balancing on model behaviour with Explainable Artificial Intelligence tools in imbalanced classification problems

Adrian Stańdo

student record book number 305755

thesis supervisor

dr Mustafa Çavuş

supervising professor

dr hab. inż. Przemysław Biecek

WARSAW 2023

Abstract

Impact of data balancing on model behaviour with Explainable Artificial Intelligence tools in imbalanced classification problems

This thesis analyses the differences in model behaviours after applying data balancing methods to train datasets in binary classification tasks. The main aim of the work is to find an answer to the question of whether these algorithms have an influence on model behaviours, and if so, on which main areas.

In order to do it, I resampled twenty-one datasets with six different balancing methods and five different Imbalance Ratio values. Afterwards, I trained the Random Forest and Weighted Random Forest models and I used three Explainable Artificial Intelligence methods, *Partial Dependence Profile*, *Accumulated Local Effects* and *Variable Importance*, to investigate model behaviours. The calculated explanations were compared using a novel metric *Standard Deviation of Differences* and the Wilcoxon statistical test.

Aiming to automate the steps described above, I created a package in Python called `edgaro`, which is available on GitHub and Python Package Index (PyPI). Additionally, the datasets used in the study are available through this package and in a dedicated GitHub repository.

The results show that the balancing methods do have an influence on model behaviours. In most cases, they contribute to the reduction of model bias towards the majority class. Nevertheless, there is also an example which indicates that resampling can change the relationships present in data. In addition, the results show that the safest balancing algorithm, in terms of keeping the model behaviour unchanged, is Random Undersampling.

This study proves that balancing techniques should be used with more caution as they can unexpectedly distort the true relationships in the data.

Keywords: Explainable Artificial Intelligence, imbalanced learning, classification, resampling methods

Streszczenie

Badanie wpływu metod balansownia zbiorów danych na zachowanie modelu przy pomocy narzędzi Wyjaśnialnego Uczenia Maszynowego w problemach klasyfikacji danych niezbalansowanych

W niniejszej pracy dyplomowej analizowane są różnice w zachowaniach modeli po zastosowaniu metod balansowania danych na zbiorach treningowych w zadaniach klasyfikacji binarnej. Głównym celem pracy jest znalezienie odpowiedzi na pytanie, czy algorytmy te mają wpływ na zachowania modeli, a jeśli tak, to na jakie główne obszary.

W tym celu zbalansowałem dwadzieścia jeden zbiorów danych sześcioma różnymi metodami balansowania i pięcioma różnymi wartościami Imbalance Ratio. Następnie, wytrenowałem modele Random Forest i Weighted Random Forest oraz wykorzystałem trzy metody Wyjaśnialnej Sztucznej Inteligencji, *Partial Dependence Profile*, *Accumulated Local Effects* i *Variable Importance*, aby zbadać zachowania modeli. Obliczone wyjaśnienia zostały porównane przy użyciu nowatorskiej metryki *Standard Deviation of Differences* oraz testu statystycznego Wilcoxona.

W celu zautomatyzowania opisanych powyżej kroków, stworzyłem pakiet w Pythonie o nazwie `edgaro`, który jest dostępny na GitHubie i w Python Package Index (PyPI). Dodatkowo, zbiory danych wykorzystane w badaniu są dostępne poprzez ten pakiet oraz dedykowane repozytorium na GitHubie.

Wyniki pokazują, że metody balansowania mają wpływ na zachowanie modelu. W większości przypadków przyczyniają się one do zmniejszenia stronniczości modelu w kierunku klasy większościowej. Niemniej jednak, istnieje również przykład, który wskazuje, że balansowanie może zmienić zależności występujące w danych. Ponadto, wyniki pokazują, że najbezpieczniejszym algorytmem balansowania, pod względem pozostawienia zachowania modelu bez zmian, jest Random Undersampling.

Niniejsze badanie dowodzi, że techniki balansowania powinny być stosowane z większą ostrożnością, ponieważ mogą nieoczekiwanie zniekształcić prawdziwe relacje w danych.

Słowa kluczowe: Wyjaśnialna Sztuczna Inteligencja, uczenie niezbalansowanie, klasyfikacja, metody balansowania

Contents

1. Introduction

Artificial Intelligence (AI) is and will remain a significant and ubiquitous part of human lives. It is utilised on a daily basis in many fields, for example, in road traffic analysis, medicine, financial markets, or as voice assistants. Its main aim is to provide people with as accurate suggestions and predictions as feasible.

AI solutions have been proven powerful many times. Nevertheless, the way these state-of-the-art and advanced tools make decisions can be incomprehensible for people. This inability to decipher what is hidden inside black-box AI was proved to be problematic [9, 15]. Consequently, these black-boxes, which may create predictions and suggestions on wrong assumptions, cannot be trusted and utilised. This is why Explainable Artificial Intelligence (XAI) has emerged in recent years. Its main goal is to make these black-boxes more transparent to people.

Machine Learning (ML) has been another prominent field of AI in recent years. It focuses on algorithms that find and learn associations from data. One of the most common tasks in ML is *classification*. Its goal is to assign to any given object a *class (label)* from a finite, predefined set. This thesis focuses on a *binary* classification task, where there are only two target classes.

One of the problems existing in the classification task, which has attracted many researchers in recent years, is the so-called *imbalanced data* problem. It happens when there is only one class of particular importance, but there are much fewer data examples available for this class than for the other. The class with a higher number of samples is called *majority class* and the other is called *minority class*. An example of such a problem is the *Mammography* dataset [16]. It contains data to identify *benign* and *malignant mammographic masses*. The point is to predict which type the patient suffers from. The *malignant mammographic masses* class is much smaller (about 42 times) and, at the same time, much more crucial for doctors and patients. The *imbalanced data* problem occurs very often in real-life applications; however, it may be challenging to train models on such data [43].

Many methods have been proposed to overcome this problem and they can be divided into two groups: algorithmic-level (e.g. [12, 20, 30]) and data-level (e.g. [11, 24]). The first focuses on developing improved methods and the second (*balancing methods*) on transforming the original dataset to make it more balanced. In this thesis, the latter group of approaches are investigated.

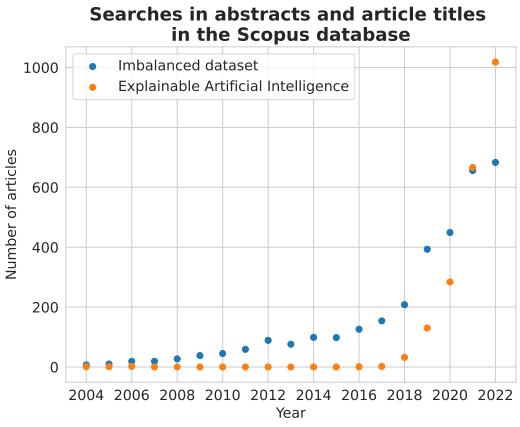
1.1. Motivation

It has been proved that using balancing techniques in ML problems leads to higher performance of models in terms of prediction power (e.g. [37, 42]). However, the question can be raised whether the model trained on a dataset after applying balancing methods (i.e. removing or adding artificial information) is reliable. This is where XAI methods can provide help.

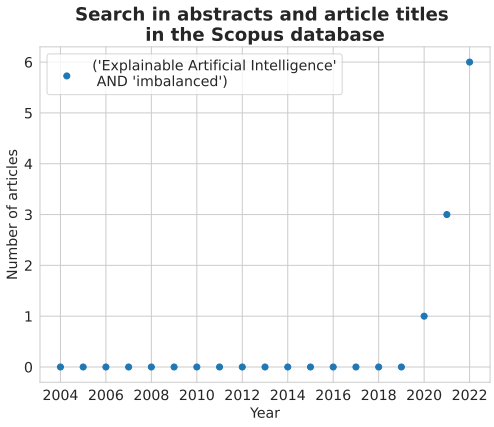
The authors of [10] studied model explanations for the xG (eXpected Goals) model. In the article, they posed the following question: what are the effects of data balancing on the model's behaviour? Unfortunately, this question was left unanswered. They did, however, include graphs showing indications that this phenomenon might be taking place and invited the scientific community to investigate this problem deeper. The curiosity about whether this phenomenon actually does occur is the main motivation that stands behind this work.

Another motivational factor is the fact that in recent years the fields of XAI and imbalanced learning have both been of interest to many scientists. Their rising popularity, evidenced by appearances in abstracts and article titles, is shown in Figure 1.1a. However, only a few have combined both fields in their articles (Figure 1.1b).

If it is proved and shown that the balancing methods do have an influence on the model behaviour, they will have to be used much more cautiously.



(a) Results of *Imbalanced dataset* and *Explainable artificial intelligence* searches.



(b) Result of *("Explainable artificial intelligence" AND "imbalanced")* search.

Figure 1.1: Results of searches in Scopus[1] database in abstracts and article titles.

[1] www.scopus.com

1.2. Contribution

The contribution of this thesis is a package called `edgaro` that automates the processes of data balancing, model training and model explanation for a set of datasets. The package focuses on binary classification tasks and allows applying the same methods with the same parameters to each of the datasets. Moreover, it provides methods to compare and visualise explanations, thus enabling a benchmark case study. The package is available on GitHub and Python Package Index (PyPI) platforms.

In this thesis, I introduce a novel metric to compare *global-level* explanations of two models. It will be used to analyse the differences between a model trained on an original dataset and on a dataset transformed with any balancing method. This metric is targeted to compare Partial Dependence Profiles (PDP) [19] or Accumulated Local Effects (ALE) [2]. These XAI methods create visual explanations in the form of line plots. However, until now, the plots have not been compared in any measurable way. Consequently, this is another contribution of the thesis.

Additionally, the models will be compared in terms of their Variable Importance (VI) [18] which is another XAI method. In [1] the authors used the Wilcoxon statistical test [41] to compare VI of models, and in this thesis, the same approach will be used.

Furthermore, a benchmark case study is conducted where the proposed metrics are used. They are applied to explanations of models trained on datasets transformed by Random Undersampling, Near Miss, Random Oversampling, Synthetic Minority Oversampling Technique (SMOTE), Borderline SMOTE and SMOTETomek (SMOTE + Tomek Links) methods using different target Imbalance Ratios. The datasets used in this study are available through the `edgaro` package and in a dedicated GitHub repository.

The main aim of this thesis is to answer the question of whether the data balancing methods change the model behaviour and what are the main areas (if any) affected by them. Thanks to the newly proposed metric, measurable differences can be extracted and the answer to this problem may be found.

Summary description of the contents of the thesis

The next section (Section 1.3) provides a brief summary of existing attempts to show the effects of data balancing methods. This paper is organised into four main Chapters. Chapter 2 describes the theoretical concepts that stand behind this thesis. Chapter 3 specifies the implementation details of the Python package. In Chapter 4, I present the results of a benchmarking case study. Finally, Chapter 5 provides conclusions and ideas for future research.

1.3. Related works

In the scientific world, it has been studied how data balancing influences and changes the performance of trained ML models (e.g. [13, 20]). Nevertheless, there have not been many attempts to investigate how and to what degree it changes the dataset and the relations between variables.

In the literature, one of the ideas was to investigate whether the samples generated in the process of oversampling are more likely to belong to the majority class than to the minority class (to which they were supposed to belong). The authors of [23] studied 70 most popular oversampling methods and developed a metric to measure that phenomenon. It turned out that almost all methods produce samples which are more likely to belong to the other class than the targeted one. They also concluded that: *"Oversampling in its current forms and methodologies is a misleading approach that should be avoided since it feeds the learning process with falsified instances that are pushed to be members of the minority class when they are most likely members of the majority."*

In [34] it was studied whether the balancing technique called SMOTE changes the correlations between features. The experiment was conducted only on one highly imbalanced dataset. The results showed that this algorithm was successful not only in eliminating imbalance but also preserving the original correlations. After that, the authors applied a few XAI methods to extract explanations of the model created on oversampled data. They underlined, however, that it could be done only because the feature correlations remained unchanged.

The first article that compared explanations of models trained on datasets after applying different balancing methods was [1]. The authors studied feature importance, which is one of the XAI methods, and compared them with a statistical test. The experiments were conducted on two datasets, connected with the blockchain field, using different variants of SMOTE algorithm. The results showed that one of the methods changed the feature importance on the test dataset in both cases.

Finally, the authors of [10] created an ML model for calculating xG (eXpected Goals). They used two balancing techniques, called Random Oversampling and Random Undersampling, to find the best-performing model. The main aim of their article was to study the explanations of the best model. However, they also compared the differences in explanations between models trained on the original, oversampled and downsampled datasets. The analysis of Partial Dependence Profiles (PDP), which is one of the XAI methods, revealed that these balancing methods might change the model behaviour. Nevertheless, they left that problem without a final answer.

2. Theory

This chapter provides the theoretical background behind the concepts described in the thesis.

2.1. Imbalanced classification

Imbalanced classification refers to the classification problem where the distribution of classes in data is not equal. It is a common practice to describe the degree of the imbalance with the ratio called Imbalance Ratio (IR).

Definition 2.1 (Imbalance Ratio). Consider an imbalanced binary classification problem. Let Y be the vector of target values and let 0 be the majority class. The Imbalance Ratio (IR) is the ratio of the number of the majority class observations to the number of the minority class observations in Y.

$$IR = \frac{\#\{i \in \mathbb{Z}: \quad Y_i = 0\}}{\#\{i \in \mathbb{Z}: \quad Y_i = 1\}}$$

Most current studies focus on datasets where the IR ranges from $4:1$ to $100:1$; however, in real-life applications it may range from $1000:1$ up to $5000:1$ [28].

One of the main issues about imbalanced learning is that most ML models assume an equal distribution of classes. Hence, if one of the classes is not adequately represented, the model will get biased towards the other class. An example of that is presented in Figure 2.1. It shows a result of a linear classifier in imbalanced classification - it is clear that the model will give poor predictions for the minority class.

Another well-known problem in this field is the so-called Accuracy Paradox. Accuracy is a standard measure of model performance (it is the ratio of correctly assigned observations in the test dataset to the number of observations in the dataset). This paradox states that a model with lower accuracy can have greater predictive power and be more accurate than a model with higher accuracy. Assume that there is a dataset with IR equal to $99:1$ and consider a dummy model that always predicts the majority class in the dataset. Such a model will have 0.99 accuracy, but it did not recognize any pattern. Hence, another model with lower accuracy,

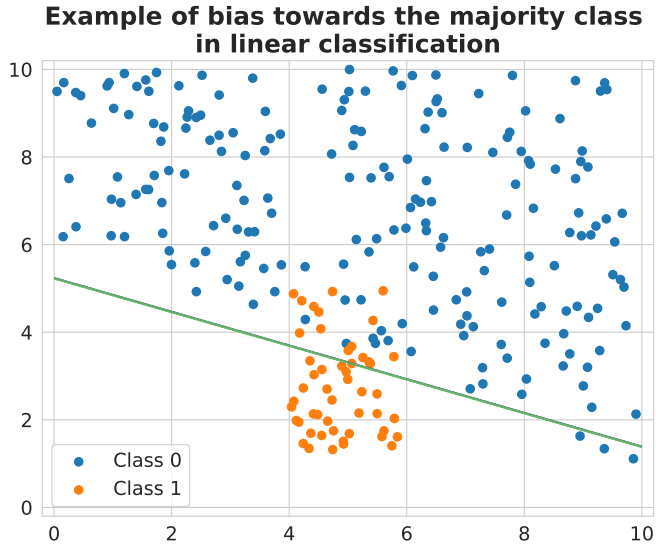


Figure 2.1: Example of bias towards the majority class in linear classification. The decision boundary (green line) gives poor predictions for the minority class.

but which learned something from the data, will be better and more powerful. This is why in the imbalanced classification other metrics are used, for example, *sensitivity* (Definition 2.2) or *balanced accuracy* (Definition 2.4) which will be mostly used in the experiments in Chapter 4.

Definition 2.2 (Sensitivity). Consider a binary classification problem. Let Y be the vector of target values and let $\hat{Y}$ be the vector of predicted values. Assume that 1 is the minority class. The Sensitivity is given by the formula:

$$Sensitivity(Y, \hat{Y}) = \frac{\#\{i \in \mathbb{Z} : \quad Y_i = \hat{Y}_i = 1\}}{\#\{i \in \mathbb{Z} : \quad Y_i = 1\}} .$$

It is also called true positive rate or recall as this metric measures the ratio of all correctly predicted minority class examples to all minority class observations in a dataset.

Definition 2.3 (Specificity). Consider a binary classification problem. Let Y be the vector of target values and let $\hat{Y}$ be the vector of predicted values. Assume that 0 is the majority class. The Specificity is given by the formula:

$$Specificity(Y, \hat{Y}) = \frac{\#\{i \in \mathbb{Z} : \quad Y_i = \hat{Y}_i = 0\}}{\#\{i \in \mathbb{Z} : \quad Y_i = 0\}} .$$

It is also called true negative rate or recall as this metric measures the ratio of all correctly predicted majority class examples to all majority class observations in a dataset.

Definition 2.4 (Balanced Accuracy). Consider a binary classification problem. Let Y be the vector of target values and let $\hat{Y}$ be the vector of predicted values. The Balanced Accuracy is given by the formula:

$$BalancedAccuracy(Y, \hat{Y}) = \frac{Sensitivity(Y, \hat{Y}) + Specificity(Y, \hat{Y})}{2} .$$

To sum up, a slight imbalance in a dataset is usually not concerning, nevertheless, a severe imbalance requires using specialized techniques. As it was mentioned in Chapter 1, there are two main types of approaches when dealing with that kind of task: algorithmic-level and data-level. In this thesis, I will focus on the latter, while the theoretical background of the most popular methods is provided in Section 2.2.

2.2. Balancing methods

There are two main types of data-level methods for dealing with imbalanced data: *oversampling* and *undersampling*. The aim of the first one is to make the training dataset bigger by adding the observations of the minority class. The other one focuses on removing some of the majority class observations from the data and hence making the training dataset smaller. There are also methods called hybrid techniques that combine both approaches - first, an oversampling algorithm is applied and then an undersampling method (e.g. [4]).

In this thesis, I focus on the six most popular balancing strategies - two undersampling (Random Undersampling and Near Miss, Sections 2.2.1 and 2.2.2), three oversampling (Random Oversample, SMOTE and Borderline SMOTE, Sections 2.2.3, 2.2.4 and 2.2.5) and one hybrid method (SMOTE + Tomek Links, Section 2.2.6).

2.2.1. Random Undersampling

It is the easiest undersampling method. It works by randomly selecting an observation from the training dataset and removing it. The process can be repeated until the desired IR is achieved. The main drawback of this approach is that by deleting the existing data some crucial information may be lost and hence finding the decision boundaries may be more challenging.

2.2.2. Near Miss

There are three algorithms called Near Miss, but in this thesis, I focus only on the first version which is called *NearMiss-1* [32]. This method focuses on finding the majority class examples which are the nearest to some of the minority class observations. More specifically, it finds the points whose average distance to k nearest minority class points is the smallest. Next, these observations are removed from the dataset. The selected points are very often placed near the decision boundaries and therefore this algorithm allows the models to focus on the minority class in the area close to the decision boundaries. The details are presented in Algorithm 2.1.

Algorithm 2.1: Near Miss 1

Data: $X_+ \in \mathbb{R}^{n_+ \times m}$, $X_- \in \mathbb{R}^{n_- \times m}$ - dataset matrixes with the majority and the minority class observations, - dataset matrix with the minority class observations, k - number of neighbours, T - number of observations to remove

Result: $z \in \mathbb{R}^T$ - row indexes of samples to be removed

$d[] \leftarrow$ an empty list of average distance of length n_+
for $i = 1 : n_+$ **do**
  $neighbours[] \leftarrow$ select k nearest neighbours of $X_+[i]$ from the minority class samples
  $L[] \leftarrow$ list of distances
  for $j = 1 : k$ **do**
    $L[k] \leftarrow distance(X_+[i], neighbours[j])$
  end
  $d[i] \leftarrow mean(L)$
end
$z \leftarrow min(d, T)$ *# indexes of T elements among d with the smallest value*

2.2.3. Random Oversampling

It is the easiest oversampling method - it works similarly to Random Undersampling (Section 2.2.1), but instead of removing the majority class observations, random observations from the minority class are duplicated. The rows for duplication are selected randomly with replacement. Consequently, there is no information loss and the transformed training dataset becomes bigger. However, this method might increase the likelihood of overfitting as exactly the same copies of the minority class observations are made. Furthermore, the transformed dataset increases the computing time for training an ML model whereas no additional information is added.

2.2.4. Synthetic Minority Oversampling Technique

The more sophisticated and also well-known oversampling method is the Synthetic Minority Oversampling Technique (SMOTE) [11]. In a nutshell, its aim is to produce a new, synthetic minority class observation by selecting a new point which is a convex combination of two existing observations in the minority class. The SMOTE method is described in the Algorithm 2.2.

Algorithm 2.2: Synthetic Minority Oversampling Technique

Data: $X \in \mathbb{R}^{n \times m}$ - dataset matrix, $Y \in \mathbb{R}^n$ - vector of target values, T - number of observations to generate, k - number of neighbours

Result: $Z \in \mathbb{R}^{T \times m}$ - generated samples

for $i = 1 : T$ **do**
- $z \leftarrow$ select a random sample from the minority class in X
- $neighbours[] \leftarrow$ select k nearest neighbours of z from the minority class observations
- $n \leftarrow$ select a random observation from $neighbours$
- $weights[] \leftarrow$ vector of m random numbers in range $(0, 1)$
- $Z[i] \leftarrow z + (z - n) \cdot weights^T$

end

The above method is only applicable to a continuous feature space. Nevertheless, *SMOTE-NC* algorithm was also developed to deal with datasets containing both categorical and continuous features. It works in the same way as *SMOTE* for continuous variables, but the categorical ones are populated according to the most common values in the point's neighbourhood.

2.2.5. Borderline Synthetic Minority Oversampling Technique

The Borderline Synthetic Minority Oversampling Technique (Borderline SMOTE) [21] is based on the SMOTE method. The main difference is that the points for generating new samples are not picked randomly, but only the ones in *danger* zone are chosen. For each minority class sample, the algorithm finds their p nearest neighbours. Then, the samples are classified as *noise* (if all neighbours are the majority class samples), *in danger* (if less than half, but not all, neighbours are the majority class samples) and *safe* (if all neighbours are the minority class samples). Next, for each sample in *danger*, SMOTE algorithm generates new samples using k nearest neighbours. The details are presented in Algorithm 2.3 - it is a simplified form of the algorithm where a maximum of one observation is generated per one minority observation.

Algorithm 2.3: Borderline Synthetic Minority Oversampling Technique

Data: $X \in \mathbb{R}^{n \times m}$ - dataset matrix, $X_+ \in \mathbb{R}^{n_+ \times m}$ - dataset matrix with the majority class observations, $X_- \in \mathbb{R}^{n_- \times m}$ - dataset matrix with the minority class observations, k - number of neighbours for generating samples, p - number of neighbours to determine if a minority sample is in *danger*

Result: $Z \in \mathbb{R}^{l \times m}$ - generated samples

```
for i = 1 : n_ do
    #select p nearest neighbours of X_[i] from X
    NB[] ← neighbours(X_[i], X, p)
    if classify(NB) ==' danger' then
        # generate a new sample using only minority class observations with X_[i] as a centre
        Z[i] ← SMOTE(X = X_, T = 1, k = k, z = X_[i])
    end
end
```

2.2.6. Synthetic Minority Oversampling Technique + Tomek Links

One of the well-known hybrid techniques is a combination of an oversampling method SMOTE and an undersampling method Tomek Links, which is called *SMOTETomek* [4]. The biggest disadvantage of SMOTE is that it is prone to creating noisy samples. For example, a lot of false samples can be generated from a few outliers in data. The answer to this problem is to use a method that cleans up the dataset after sampling with SMOTE - in this case, one of the undersampling methods - Tomek Links [38]. This method detects so-called Tomek's links defined in Definition 2.5 and removes them from the dataset.

Definition 2.5 (Tomek's link). Let a, b be two observations of different classes in X and $d(x, y)$ be a distance function. A Tomek's link exists between a and b if:

$$\forall c \in X \quad d(a,b) < d(a,c) \wedge d(a,b) < d(b,c) \ .$$

In other words, there is a Tomek's link between a and b if they are the nearest neighbours.

In essence, this method first uses the SMOTE technique to oversample the minority class and then cleans the dataset by removing all of Tomek's links.

In summary, the methods described in this section focus on two ideas. The first is based on randomness - observations are randomly duplicated, removed or new, synthetic points are generated from randomly selected points. The second idea is to take action only in the area close to the decision boundaries so that the ML algorithms have more data or cleaned space in the "critical" area.

2.3. Explainable Artificial Intelligence methods

In this thesis, three popular XAI techniques are used: PDP, ALE and VI. The first two are very similar as both create explanations based on the expected values of the model predictions. In addition, their outputs have the same form of a function of a value of the selected variable. Nevertheless, they are calculated in different ways. The third XAI method, VI, represents how the model performance changes when we remove a certain variable from the dataset. All these three methods are called *model-agnostic* methods as the explanations they provide are created with no assumption of model structure - they treat it as a black-box model.

The following sections provide definitions and theoretical background for these methods. They are based on the *Explanatory Model Analysis* book [7].

Definition 2.6. Let X be the dataset matrix, where rows represent observations and columns represent features. Let X^j be the j-th column of the matrix X. Let X_i be the i-th row (observation) of matrix X. Let Y be a binary vector of the target variable.

Definition 2.7. Let n be the number of rows and m the number of columns in X.

Definition 2.8. Let X^{-j} be the matrix X without j-th column.. Let $X^{j|=z}$ be the matrix X where all values in j-th column are replaced with value z.

Definition 2.9. Let X^{*j} be the matrix X where the j-th column is randomly permuted.

Definition 2.10. Let f represent a Machine Learning model and $\hat{Y} = f(X)$ the vector of model prediction for rows (observations) in X.

2.3.1. Partial Dependence Profile

Partial Dependence Profile (PDP) was first introduced in [19]. For many years, however, this technique was of no interest since it was overshadowed by another concept presented in this article. Nevertheless, with the progress of XAI, it is currently a popular technique available in many Python packages (e.g. [3, 35, 27]).

Definition 2.11 (Partial Dependence Profile). Partial Dependence Profile (PDP) for a model f and a variable j is a function of argument z defined as follows:

$$PDP(f, j, z) = E_{X^{-j}}\left[f(X^{j|=z})\right] .$$

In other words, the PDP value for the j-th column in the point z is an average prediction of model f when values in the j-th column are set to z. This is a theoretical concept, but in practice, we do not usually know the distribution of X^{-j}. As a result, it is estimated with a formula presented in Definition 2.12.

Definition 2.12 (PDP estimator). The estimator of PDP is the following formula:

$$\widehat{PDP}(f, j, z) = \frac{1}{n}\sum_{i=1}^{n} f(X_i^{j|=z}) .$$

2.3.2. Accumulated Local Effects

PDP is a very powerful tool; however, it also has some weaknesses. One of them is the fact that the explanation created by them may be misleading if explanatory features are correlated. Therefore, the Accumulated Local Effects (ALE) method was proposed [2]. It is an attention-worth alternative to PDP as they both produce the functions as outputs, but ALE is unbiased. Its implementation is available in for example [3, 27].

Definition 2.13 (Accumulated Local Effects). Accumulated Local Effects (ALE) for a model f and a variable j is a function of argument z defined as follows:

$$ALE(f, j, z) = \int_{z_0}^{z} \left(E_{X^{-j}}\left[\left\{ \frac{\partial f(u)}{\partial u^j} \right\}_{u=X^{j|=v}} \right] \right) dv + c .$$

The constant c is selected in a way that $E_{X^j}[ALE(f, j, X^j)] = 0$ and z_0 is a value close to the lower bound of the support of X^j.

In other words, $\left\{\frac{\partial f(u)}{\partial u^j}\right\}$ describes the local change of the model which is then averaged over the distribution of X^{-j} (expected value) and aggregated (integrated) over values from z_0 to z.

Definition 2.14 (ALE estimator). The estimator of ALE is the following formula:

$$\widehat{ALE}(f,j,z) = \sum_{k=1}^{K} \left(\frac{1}{\sum_l w_l^j(z_k)} \sum_{i=1}^{N} w_i^j(z_k) \left[f\left(X^{j|=z_k}\right) - f\left(X^{j|=z_k-\Delta}\right) \right] \right) + \hat{c} \ .$$

The constant K is the number of intermediate points, $(z_0, z_1, z_2, \ldots, z_K)$ are evenly distributed points in (z_0, z) interval with step $\Delta = (z - z_0)/K$. The weights $w_i^j(z_k)$ represent the distance between z_k and x_i^j. The constant $\hat{c}$ is selected in a way that $\sum_{i=1}^{N} ALE(f,j,X_i^j) = 0$.

Most often, for a categorical variable, an indicator function is used to calculate the weights in the equation above: $w_i^j(z) = \mathbb{1}_{z=x_i^j}$. In the case of a continuous variable, the Gaussian kernel is usually used: $w_i^j(z) = \Phi(z - x_i^j, 0, s)$, where $\Phi(*, 0, s)$ is a density function of a normal distribution with mean 0 and variance s^2. The parameter s is also called a smoothing factor.

2.3.3. Variable Importance

The previous two methods show how the model predictions change when the selected variable value changes. However, the Variable Importance technique focuses on creating one explanation for all variables in the model. The main idea is to permute each column in X multiple times and see how it affects the model performance. The method presented here was proposed for the first time in [18]. The detailed algorithm is presented in Algorithm 2.4.

Algorithm 2.4: Variable Importance

Data: $X \in \mathbb{R}^{n \times m}$ - dataset matrix, $Y \in \mathbb{R}^n$ - vector of target values, k - number of permutations, f - Machine Learning model

Result: $VI \in \mathbb{R}^m$ - Variable Importance values

$L_0 \leftarrow 1$ - $AUC(Y, \hat{Y})$ # *AUC stands for ROC AUC score*
for $j = 1 : m$ **do**
  $scores[] \leftarrow$ empty vector of length k
  for $i = 1 : k$ **do**
    $X^{*j} \leftarrow$ permute randomly j-th column of X
    $L_i \leftarrow 1 - AUC(Y, f(X^{*j}))$
    $scores[i] \leftarrow L_i - L_0$
  end
  $VI[j] \leftarrow mean(scores)$
end

2.4. Comparison metrics

In this section, the details concerning the comparison measures of the differences between model explanations are provided.

2.4.1. Standard Deviation of Differences

In this thesis, I propose a novel metric to compare the profiles of two models. Its aim is to compare the PDP or ALE curves in terms of the difference in their behaviour, not the position. It is because models can have different performances and hence the curves will be vertically moved. However, if the models have similar behaviours the differences between values should remain the same or similar at all points. The proposed metric is called Standard Deviation of Differences (SDD) and it is based on the standard deviation of the differences in curve values.

Definition 2.15 (SDD). Let f_1 and f_2 be two models trained on the same dataset. Let X be the dataset matrix and X^j be any column (variable). Let $h_1(z) = PDP(f_1, j, z)$ and $h_2(z) = PDP(f_2, j, z)$ be PDP (or ALE) curves. Let denote as $(x_1, x_2, \ldots, x_k)$ evenly distributed k points over the interval of X^j values. Then,

$$SDD(h_1, h_2, j, k) = std[\{h_1(x_i) - h_2(x_i)\}_{i=1,2,\ldots,k}] \ .$$

In the definition above, if $\forall z \ h_1(z) = h_2(z)$, which means that the curves are equal, the metric value equals $SDD(h_1, h_2, j, k) = 0$. Similarly, if $\exists c \ \forall z \ h_1(z) = h_2(z) + c$, which means that the curves are parallel (Figure 2.2a), $SDD(h_1, h_2, j, k) = 0$. This behaviour is expected as the metric is supposed to measure the changes in the shape of curves, not vertical offset. It is because the position of the PDP plot depends on the model accuracy, but only the changes in shape indicate changes in behaviour.

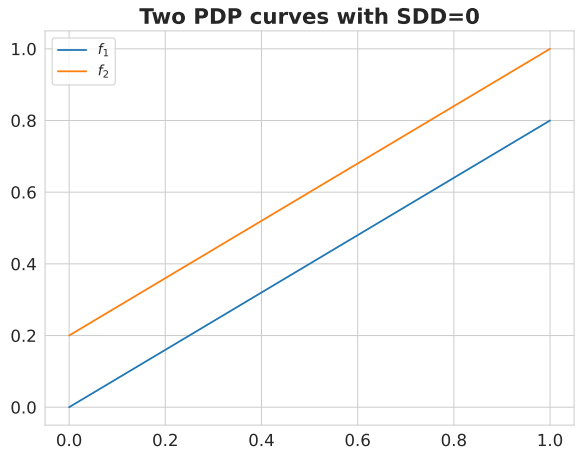
On the other hand, consider $(x_1, x_2, \ldots, x_{101}) = (\frac{0}{100}, \frac{2}{100}, \ldots, \frac{100}{100})$, $h_1(z) = z$, $h_2(z) = 1 - z$. In such a situation (Figure 2.2b), $SDD(h_1, h_2, j, 101) \approx 0.58$. As it is seen, the curves which have totally different behaviours have a high value of SDD.

SDD metric compares two models for one variable. To compare the behaviours of models in terms of all variables, the SDD values can be aggregated to $ASDD$ (Averaged SDD) values.

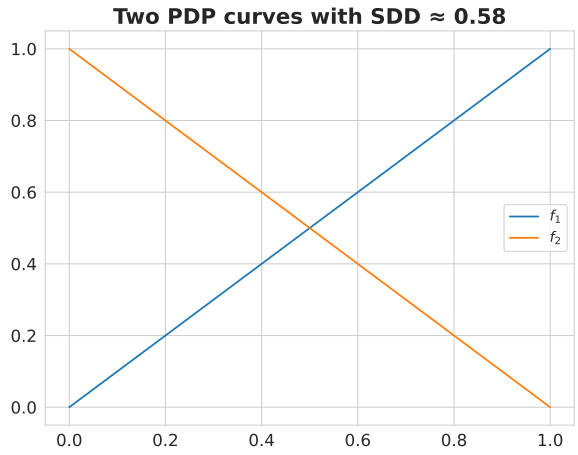
Definition 2.16 (ASDD). Let f_1 and f_2 be two models trained on the same dataset.

$$ASDD(f_1, f_2, k) = \frac{1}{m} \sum_{j=1}^{m} SDD\left(PDP(f_1, j, *), PDP(f_2, j, *), j, k\right) \ .$$

In the above equations, PDP can be replaced with ALE.



(a) Two example PDP curves with $SDD = 0$.



(b) Two example PDP curves with $SDD \approx 0.58$.

Figure 2.2: Exemplary PDP plots and the SDD values.

2.4.2. Wilcoxon test

It was proposed in [1] to use the Wilcoxon test [41] to compare models in terms of their Variable Importance profiles. The test is a signed-rank non-parametric statistical hypothesis test and it tests the null hypothesis that two related paired samples come from the same distribution. In the case of this thesis, it tests the null hypothesis that the Variable Importance profiles of the two models are the same.

The test statistic is calculated by following the Algorithm 2.5.

Algorithm 2.5: Wilcoxon test statistic

Data: $x, y \in \mathbb{R}^m$ - two related paired samples (VI values of two models)

Result: $T \in \mathbb{R}$ - test statistic and p-value

$d \leftarrow x - y$

$indexes \leftarrow sort(absolute\_value(d))$ *# indexes of sorted absolute values of d*

$T \leftarrow 0$

for $j = 1 : m$ **do**

  $rank \leftarrow j$

  $diff \leftarrow d[indexes[j]]$

  # sign(∗) function returns 1 *if input is positive, otherwise* -1

  $T \leftarrow T + sign(diff) * rank$

end

The *p-value* is extracted by comparing the test statistic T to its distribution. It can be calculated exactly for the small size of input samples; however, for the larger samples ($m > 20$) the normal distribution approximation is used.

In the experiments described in Chapter 4, the described test will be conducted multiple times and this is why a *p-value* adjustment should be taken into consideration. One of the most popular methods is the False Discovery Rate (FDR), which is designed to control the proportion of rejected null hypotheses that are false. The most popular algorithm is the Benjamini–Hochberg procedure [5, 6], which is described in Algorithm 2.6. After adjusting the *p-values*, the hypotheses are accepted or rejected using the new values with the original significance level α. In other words, all hypotheses with the original *p-value* equal or less j-th smallest original *p-value* are rejected, where $j = max\{k : sorted(p\_values)[k] \leq \frac{k}{m}\alpha\}$.

Algorithm 2.6: Benjamini–Hochberg False Discovery Rate

Data: $p \in \mathbb{R}^m$ - vector of *p-value*s, α - significance level

Result: $p_{adj} \in \mathbb{R}^m$ - adjusted *p-value*s

$indexes \leftarrow sort(p)$ *# indexes of sorted p vector in ascending order*

$k \leftarrow 0$

for $j = 1 : m$ **do**

  if $p[indexes[j]] \leq \frac{j}{m}\alpha$ **then**

    $k \leftarrow j$

  end

end

$p_{adj} \leftarrow p \cdot \frac{m}{k}$

3. Implementation in Python

The solution has a form of a Python package called `edgaro` (Explainable imbalanceD learninG compARatOr). It consists of four modules: *dataset*, *balancing*, *model* and *explain*. The tool aims to automate the process of data balancing with different methods, training ML models and calculating PDP/ALE/VI explanations for a set of models. The tool also provides the functionality to compare PDP/ALE/VI explanations so as to examine how the models differ from each other. The package is available on GitHub (`www.github.com/adrianstando/edgaro`) and Python Package Index (PyPI) (`www.pypi.org/project/edgaro`).

3.1. Technology selection

Programming language The package was written in *Python* language since now it is one of the most popular programming languages. Moreover, it is widely used in the data science field and many user-friendly libraries are available.

Data representation The *pandas* library [33] is used to represent data as it is the most popular tool for data processing. It contains many useful methods and implements the convenient concept of a dataframe. Moreover, many *Python* libraries for ML are integrated with this library.

Mathematical calculations A popular library which performs fast mathematical array-based calculations in *Python* is called *NumPy* [22]. This tool also provides a data representation format which can be simply transformed to *pandas* dataframe object.

Machine Learning models The most popular library for ML in *Python* is *scikit-learn* [35]. It implements many useful functions, metrics, data encoders and ML algorithms, for example, *Random Forest*.

Data balancing algorithms A library which provides tools when dealing with imbalanced classification problems is *imbalanced-learn* [29]. It contains implementations of many balancing methods, for example, *Random Under Sampling, Random Over Sampling* and *SMOTE*.

Model explanations *DALEX* [3] is a popular XAI library which implements many model explanations, for example PDP, ALE and VI.

Statistical procedures Two popular statistical libraries which contain implementations of the most important statistical tests and procedures are *scipy* [40] and *statsmodels* [36].

Data visualisation The *matplotlib* library [25] with *pyplot* submodule is the most popular library for data visualisation in *Python*.

Data source The sample dataset and benchmarking set of datasets are created on data available in *imbalanced-learn* library and *OpenML* platform [39]. It is an open platform for sharing datasets, algorithms and experiments. It has a *Python* client library [17] with the same name. This platform shares so-called *benchmark suites*, which are sets of *OpenML* datasets used for benchmarking purposes.

3.2. Package description

The `edgaro` Python package is the first to provide a user-friendly interface for balancing and training ML models for a number of datasets arranged in arrays or nested arrays. It allows using implementations from major libraries, *scikit-learn* and *imbalanced-learn*. The package also calculates, in the same form of arrays and nested arrays, explanations using the PDP, ALE or VI method and gives functions to compare them.

The `edgaro`'s capabilities can be easily extended to include other methods not available in *scikit-learn* or *imbalanced-learn* by creating a wrapper that extends abstract classes.

The package is divided into four modules:

- dataset - contains classes for representing a single dataset or an array of datasets; it also contains functions to create example datasets and an example benchmarking set of datasets,
- balancing - contains classes for defining a balancing method applicable to a single dataset or an array of datasets,
- model - contains classes for embodying an ML model to train and make predictions on a single dataset or an array of datasets,
- explain - contains classes for calculating explanations on a model or arrays of models and keeping them in the form of objects which can be easily compared and visualised.

In order to use all functionalities available in the package, the modules should be used in the order shown in Figure 3.1. Among them, only *balancing* module can be omitted in the data flow - it is possible only when a user already has datasets to which they want to apply further procedures.

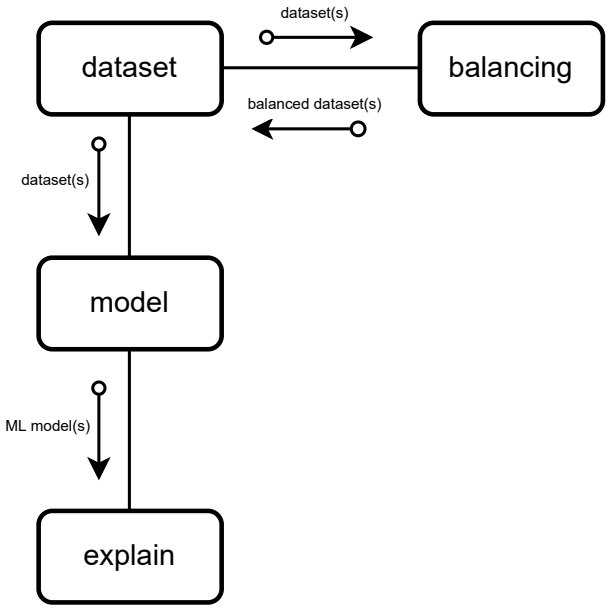


Figure 3.1: Structure chart of `edgaro` package.

3.3. Implementation details

This section provides the description and information about the main functionalities of classes from each of the created modules.

3.3.1. Dataset module

This module gives a user-friendly interface to create and manage objects containing data from one or many datasets. The `Dataset` object represents one dataset - predictors and target. The `DatasetArray` object represents a set (an array) of `Dataset` or `DatasetArray` objects. Additionally, the objects can be created from data written in files (*csv* or *pickle* - Python serialisation format), from datasets available in *OpenML* or from data available in *Python* session.

Apart from representing and keeping the data, these objects have also other functionalities, for example:

- basic data cleaning procedures - removing rows with `NaN` values and removing columns containing almost only `NaN` values,
- calculating Imbalance Ratio,
- checking whether an object represents a binary classification task,
- splitting themselves into train and test datasets (so that test datasets remain the same even after transformations with *balancing* module).

3.3.2. Balancing module

This module contains classes (`Transformer`, `TransformerArray` and their extensions) which enable the application of different balancing methods on `Dataset` and `DatasetArray` objects. It is possible to use custom balancing transformers (by extending the basic classes) or to use methods implemented in *imbalanced-learn* library (by using `TransformerFromIMBLEARN` class).

In the case of using a `DatasetArray`, the `TransformerArray` object applies the same balancing technique (defined as a `Transformer` object) to each of the `Dataset`s in the array. Moreover, the balancing method can be executed with a list of different (sets of) parameters (for example different output Imbalance Ratio) on the same input `Dataset` and `DatasetArray`. As a result, the package creates a bigger and higher-dimensional `DatasetArray`, which contains all datasets that underwent all the defined balancing transformations.

The one child class that is worth attention is `NestedAutomaticTransformer` class. This object inherits and behaves like `Transformer` class, however, it enables using a set of methods with different parameters in one object. It is a very useful class as far as benchmarking purposes are concerned since it also calculates Imbalance Ratio values automatically, based on a number of intermediate points, which is passed as an argument.

3.3.3. Model module

This module contains classes (`Model`, `ModelArray` and their extensions) which make it possible to train Machine Learning (ML) model(s) for a `Dataset` or `DatasetArray` object. It is possible to train custom models (by extending the basic classes) or to use ML algorithms implemented in *scikit-learn* library (by using `ModelFromSKLEARN` class).

Apart from training the ML models, the module automatically encodes nominal (categorical) columns and the target. In the field of imbalanced learning, it is a common practice that the minority class in target is represented by 1 and the majority class by 0 - and so it is done here.

The base classes (`Model` and `ModelArray`) also contain methods to calculate the model's performance metrics (by default on a test subset), for example, `accuracy, F1, precision, recall`. Apart from these, there are also available widely-used metrics in imbalanced learning, e.g. `balanced accuracy, F1-weighted` and `geometric mean`.

3.3.4. Explain module

This module contains classes (`Explainer` and `ExplainerArray`) which calculate PDP, ALE or VI explanation for `Model` or `ModelArray`. Next, the calculated explanations are stored in the dedicated classes: for PDP and ALE they are `ModelProfileExplanation` and `ModelProfileExplanationArray` classes, whereas for VI there are `ModelPartsExplanation` and `ModelPartsExplanationArray`.

The classes containing the results provide the methods to plot the explanations and to compare them using metrics described in Chapter 2. In addition, `ModelProfileExplanationArray` and `ModelPartsExplanationArray` classes can create summary plots for discovering higher-level relationships. These plots are in the form of boxplots, with metric values on the vertical axis and regex filters for model names or properties on the horizontal axis.

3.4. Examples of usage

In this section, basic usage examples of the `edagro` package are shown. For simplicity, in most cases, the default parameters will be used and the examples will be presented only on the *mammography* dataset and benchmarking set of datasets (proposed in Section 4.1).

In the examples below, only the Random Forest algorithm is used as an ML model. In real-life applications, it can be replaced with other models or *GridSearch* object to find the optimal model inside a parameter grid. The three most important use cases of the package will be presented below.

Use Case 1

A user has one dataset and wants to see how different balancing methods would change the model behaviour based on this one example. They can do it by running the Code 3.1.

Code 3.1: Use Case 1.

```
1 # loading the dataset and splitting it into train and test datasets
2 from edgaro.data.dataset import load_mammography
3 df = load_mammography()
4 df.train_test_split(test_size=0.2)
5
6 # using nested_transformer for balancing
7 # so as to include a set of standard methods
```

```
from edgaro.balancing.nested_transformer import BasicAutomaticTransformer
transformer = BasicAutomaticTransformer(
    keep_original_dataset=True, n_per_method=5)
transformer.fit(df)
df_transformed = balancing_array.transform(df)

# training and evaluating models
from edgaro.model.model import RandomForest
from edgaro.model.model_array import ModelArray
rf = ModelArray(RandomForest())
rf.fit(df_transformed)
rf.evaluate()

# calculating PDP explanations, visualising and comparing
from edgaro.explain.explainer_array import ExplainerArray
pdp = ExplainerArray(rf, explanation_type='PDP')
pdp.fit()
pdp_results = pdp.transform()
pdp_results.plot()
pdp_results[0][0].compare(pdp_results[0][1])
```

A user can also calculate and visualise the results of ALE and VI in the same manner - the only difference is the `explanation_type` argument value, as it is seen in Code 3.2.

Code 3.2: Use Case 1 - the usage of other explanation methods.

```
# calculating ALE explanations, visualising and comparing
ale = ExplainerArray(rf, explanation_type='ALE')
ale.fit()
ale_results = ale.transform()
ale_results.plot()
ale_results[0][0].compare(ale_results[0][1])

# calculating VI explanations, visualising and comparing
vi = ExplainerArray(rf, explanation_type='VI')
vi.fit()
vi_results = vi.transform()
vi_results.plot()
vi_results[0][0].compare(vi_results[0][1])
```

Use Case 2

A user has one dataset and wants to investigate the impact of one of the balancing methods with different parameters. They can do it by running the Code 3.3.

Code 3.3: Use Case 2.

```
# loading the dataset and splitting it into train and test datasets
from edgaro.data.dataset import load_mammography
df = load_mammography()
df.train_test_split(test_size=0.2)

# using RandomUnderSampler for balancing
from edgaro.balancing.transformer import RandomUnderSampler
from edgaro.balancing.transformer_array import TransformerArray
transformer = TransformerArray(RandomUnderSampler())
transformer.set_params(IR=[30, 20, 10, 5, 1])
transformer.fit(df)
df_transformed = balancing_array.transform(df)

# training and evaluating models
from edgaro.model.model import RandomForest
from edgaro.model.model_array import ModelArray
rf = ModelArray(RandomForest())
rf.fit(df_transformed)
rf.evaluate()

# calculating explanations
from edgaro.explain.explainer_array import ExplainerArray
pdp = ExplainerArray(rf, explanation_type='PDP')
pdp.fit()
pdp_results = pdp.transform()

# visualising and comparing explanations
pdp_results.plot()
pdp_results[0].compare(pdp_results[1])
```

Use Case 3

A user has many datasets and wants to investigate the impact of one chosen or many balancing methods. It is to find out whether there are any problems with certain methods in imbalanced classification. They can do it by running the Code 3.4.

Code 3.4: Use Case 3.

```
# loading the benchmarking dataset array
# and splitting it into train and test datasets
from edgaro.data.dataset_array import load_benchmarking_set
df = load_benchmarking_set()
df.train_test_split(test_size=0.2)

# using RandomUnderSampler for balancing
from edgaro.balancing.transformer import RandomUnderSampler
from edgaro.balancing.transformer_array import TransformerArray
transformer = TransformerArray(RandomUnderSampler())
transformer.set_params(IR=[1.1, 1.2, 1.3])
transformer.fit(df)
df_transformed = balancing_array.transform(df)

# using nested_transformer for balancing
# so as to include a set of standard methods
from edgaro.balancing.nested_transformer import BasicAutomaticTransformer
transformer = TransformerArray(BasicAutomaticTransformer(
    keep_original_dataset=True,
    n_per_method=5), set_suffixes=False)
transformer.fit(df)
df_transformed = balancing_array.transform(df)

# training and evaluating models
from edgaro.model.model import RandomForest
from edgaro.model.model_array import ModelArray
rf = ModelArray(RandomForest())
rf.fit(df_transformed)
rf.evaluate()
```

```
# calculating explanations
from edgaro.explain.explainer_array import ExplainerArray
pdp = ExplainerArray(rf, explanation_type='PDP')
pdp.fit()
pdp_results = pdp.transform()

# visualising and comparing explanations
pdp_results[0].plot()
pdp_results[0][[0, 1]].plot()
pdp_results.plot_summary()
pdp_results.plot_summary(model_filters = [
      'UNDERSAMPLING__Random',
      'OVERSAMPLING__Random',
      'OVERSAMPLING__SMOTE'
])
```

4. Experiments

In this chapter, I present the conducted experiments and their results. This case study aims to find an answer to the main question of the thesis of whether the data balancing methods affect model behaviours.

4.1. Datasets

For the experiments, I propose a new benchmarking set of datasets. It is made of three main sources: *OpenML-100* [8], *OpenML-CC18* [8] and the collection of datasets available in *imblearn* library which was proposed by [14].

The benchmarking set contains only datasets for binary classification tasks which have only continuous columns (categorical and nominal were removed), at least 1000 rows and the Imbalance Ratio of at least 1.5. The details are presented in Table 4.1. These datasets are also modified a little - the columns representing the same information were removed.

I propose this set because all of these datasets are open-source, well-known in the ML field, and their combination covers a big range of IR values (from slight imbalance with $IR = 1.54$ up to $IR = 129.53$). This study is intended to investigate the changes not only in highly imbalanced datasets (which are mainly available in *imblearn* library), but also in less imbalanced tasks - this is why the proposed benchmarking set is crucial.

This benchmarking set is available through the `edgaro` package and can be loaded using the Code 4.1 into the `DatasetArray` object. Moreover, it is also available in a dedicated GitHub repository (`www.github.com/adrianstando/imbalanced-benchmarking-set`).

Code 4.1: Loading the benchmarking set.

```
1 from edgaro.data.dataset_array import load_benchmarking_set
2 df = load_benchmarking_set()
```

Table 4.1: Proposed benchmarking set.

Dataset name	IR	Rows	Columns	Source
spambase	1.54	4601	55	OpenML-100, OpenML-CC18
MagicTelescope	1.84	19020	10	OpenML-100
steel-plates-fault	1.88	1941	13	OpenML-100, OpenML-CC18
qsar-biodeg	1.96	1055	17	OpenML-100, OpenML-CC18
phoneme	2.41	5404	5	OpenML-100
jm1	4.17	10880	17	OpenML-100, OpenML-CC18
SpeedDating	4.63	1048	18	OpenML-100
kc1	5.47	2109	17	OpenML-100, OpenML-CC18
churn	6.07	5000	8	OpenML-CC18
pc4	7.19	1458	12	OpenML-100, OpenML-CC18
pc3	8.77	1563	14	OpenML-100, OpenML-CC18
abalone	9.68	4177	7	imblearn
us_crime	12.29	1994	100	imblearn
yeast_ml8	12.58	2417	103	imblearn
pc1	13.40	1109	17	OpenML-100, OpenML-CC18
ozone-level-8hr	14.84	2534	72	imblearn, OpenML-100, OpenML-CC18
wilt	17.54	4839	5	OpenML-100, OpenML-CC18
wine_quality	25.77	4898	11	imblearn
yeast_me2	28.10	1484	8	imblearn
mammography	42.01	11183	6	imblearn
abalone_19	129.53	4177	7	imblearn

Table 4.2: Hyperparameter grid for RF with unbalanced class weights.

Hyperparameter	Values
max_depth	[5, 10, 30, None]
n_estimators	[50, 100, 300, 500]
class_weight	[None]
random_state	[42]

Table 4.3: Hyperparameter grid for RF with balanced class weights.

Hyperparameter	Values
max_depth	[5, 10, 30, None]
n_estimators	[50, 100, 300, 500]
class_weight	['balanced']
random_state	[42]

4.2. Experiments settings

Definition 4.1 (Unbalancedness). Let IR_{old} be the value of the Imbalance Ratio of the original dataset, before sampling. Let IR_{new} be the value of the Imbalance Ratio after applying a balancing method to it. The Unbalancedness value is defined as:

$$Unbalancedness = \frac{IR_{new} - 1}{IR_{old} - 1} \ .$$

The main aim of this rate is to capture the percentage differences in changes of IR. The form of the Unbalancedness rate is caused by the fact that the minimal possible IR value is 1. It is worth noting that the $Unbalancedness = 0$ means a perfectly balanced dataset.

The datasets listed in Table 4.1 were balanced using the six most popular balancing methods: Random Undersample, Near Miss, Random Oversample, SMOTE, Borderline SMOTE and SMOTETomek. They were described in detail in Section 2.2. For each of the datasets five intermediate evenly distributed IR values were chosen (I will refer to them as *Unbalancedness* rate with values of 0.8, 0.6, 0.4, 0.2, 0.0, as defined in the Definition 4.1). As a result, each dataset was balanced 30 times (five IR values for each balancing method).

The Machine Learning models were trained using the Random Forest (RF) implementation from *scikit-learn* library. The best models were found thanks to the usage of the *grid search* method with hyperparameter grid presented in Table 4.2, with five-fold *cross-validation* optimized for *balanced accuracy* metric.

In addition, Random Forest models with balanced class weights were trained to investigate the impact of bias towards the majority class in the experiments. The training procedure was the same as in the original case with the hyperparameter grid shown in Table 4.3.

All three types of XAI explanations (PDP, ALE and VI) for all trained models were calculated. The whole test dataset, extracted before applying any balancing method and with a class distribution equal to that of the training set, was used as a background dataset in each case (matrix X in the Definitions 2.11, 2.13 and Algorithm 2.4). This was to ensure that the models trained on the original and resampled data were compared on the same dataset.

4.3. Results

This section provides the experiment results. Each plot shows two groups which represent results derived from two kinds of models: Random Forest (RF) and Random Forest with balanced class weights (called here Weighted Random Forest, WRF). Both of them are analysed here since a crucial question can be posed in this study: if there are any changes in model behaviours, are they an effect of disturbed relationships in data or is it just models which get less biased? In order to find an answer, the results of an unbiased model have to be analysed, which in this study is a Weighted Random Forest.

4.3.1. Model performance analysis

The main aim of using the balancing methods is to increase the model performance. The distributions of Balanced Accuracy obtained in the experiments are presented in Figure 4.1. It is seen that the models trained on the original datasets with the unbiased estimator have higher performance than those trained with unmodified RF. Moreover, in most cases, the Balanced Accuracy of RF increases when the balancing methods are applied. However, there are also some lower values and outliers in the plots - it turns out that a few models overfitted after resampling. The Balanced Accuracy for these estimators on train datasets are in the range of $0.98 - 1.0$ for RF and in the range of $0.8 - 0.9$ for WRF, but the same score on the test dataset gives values below 0.5.

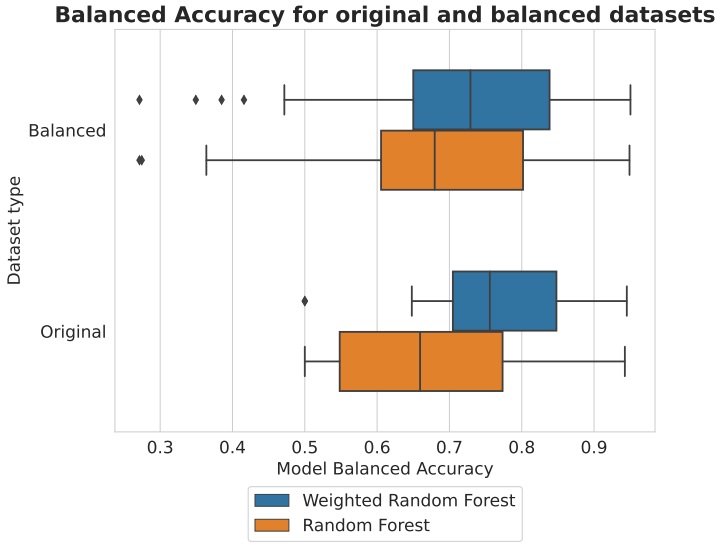
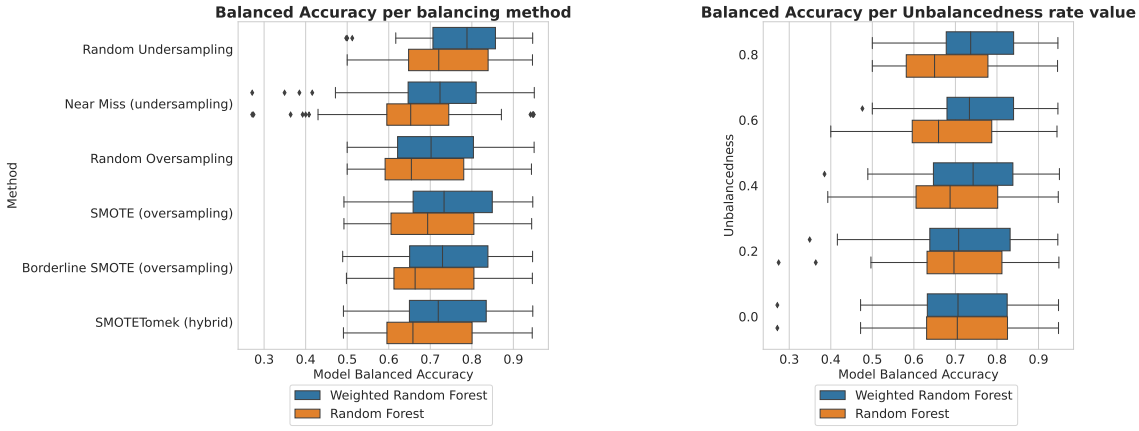


Figure 4.1: Balanced Accuracy results on test datasets.

It is also possible to look at these scores from different perspectives. In Figure 4.2a the plot shows the distribution of Balanced Accuracy for different balancing methods. It turns out that the overfitted models were created after using the Near Miss method. Furthermore, the diagram indicates that the models with the highest scores were trained after using the Random Undersampling technique.

Figure 4.2b presents the score distribution for the Unbalancedness rate value. Firstly, it can be observed that the most overfitted models were created with more resampled datasets. Secondly, the values remain similar for WRF models when the Unbalancedness value is reduced, whereas for RF it is rising. Thirdly, the results for $Unbalancedness = 0$ are identical as the class weights are equal in this case and the RF model is equivalent here to the WRF model.

To sum up, the RF models trained on balanced datasets have higher Balanced Accuracy than those trained on the original data. Also, the more balanced the dataset is, the higher the score. Nevertheless, the Near Miss method seems to have sometimes a negative influence. It must have removed some crucial observations from datasets as the model overfits. As far as WRF is concerned, the Balanced Accuracy remains stable when the Unbalancedness rate changes. Moreover, these models have a higher performance than the RF models.



(a) Results for balancing methods.

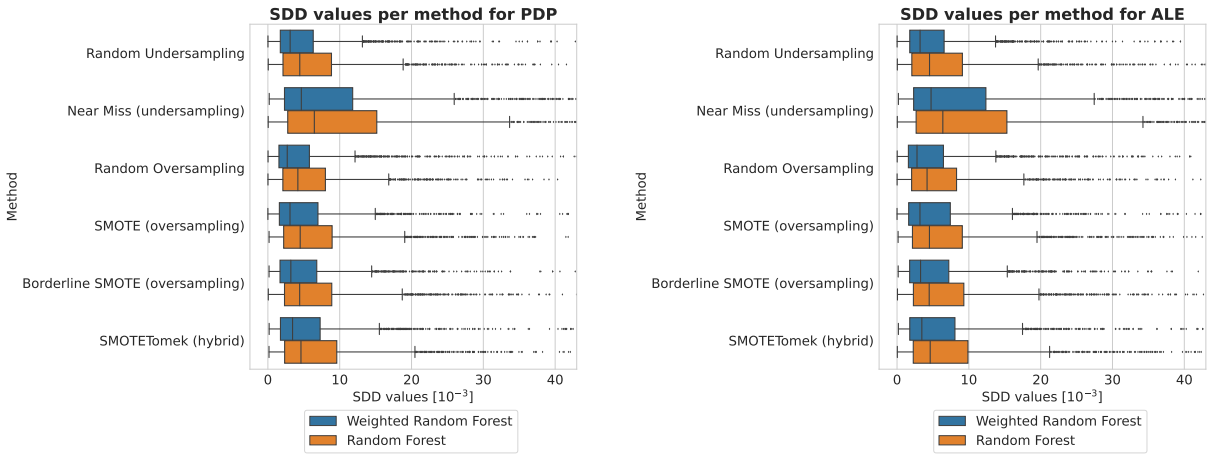
(b) Results for Unbalancedness rate values.

Figure 4.2: Balanced Accuracy results on test datasets for different filters.

4.3.2. Model profile analysis

The model profile analysis includes the analysis of PDP and ALE curves in terms of SDD values. The metric compares the behaviours of two models and in this analysis, the first one is always the model trained on the original, unbalanced dataset. The boxplots presented in this section have SDD values calculated for each column on the horizontal axis and aggregated groups of models on the vertical axis.

Figure 4.3 compares SDD values for each of the balancing methods in terms of PDP and ALE profiles. The values for WRF are lower than for RF in each case which suggests that WRF models are less changed than in the RF case. Moreover, the highest values are created by the Near Miss algorithm, which can be a result of overfitted models. The lowest values are for the models trained on datasets resampled by Random Oversampling. It is also seen that the plots generated by both methods are similar to each other.



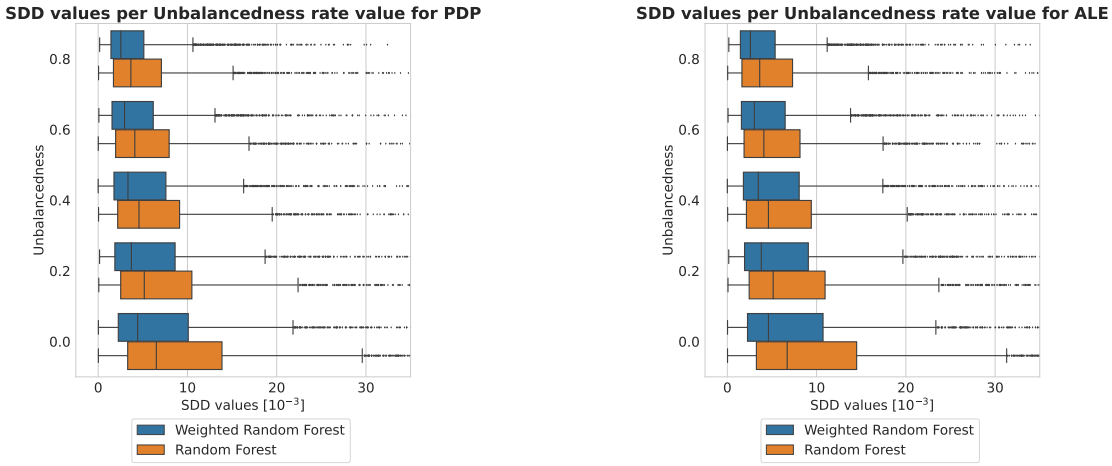
(a) Comparison of PDP for different methods.

(b) Comparison of ALE for different methods.

Figure 4.3: Comparison of model profiles on test datasets for different methods.

In Figure 4.4 the SDD values for different Unbalancedness rate values are shown. The main observation is that for both models (RF and WRF) the more the dataset is balanced (the lower the Unbalancedness rate), the more distant is the model from the original one. Moreover, the metric values for WRF are much lower than for RF. This indicates that the change in model behaviour is not only the result of reducing model bias - then the SDD values for WRF would not raise. Therefore it can be said that the balancing methods have an effect on the relationships in data.

Figure 4.6 presents the metric values for each of the datasets in the benchmarking set. It shows that almost in all cases a reduction in SDD is observed when WRF is used instead of RF. Moreover, there are only four datasets for which the metric value has been raised: *abalone*, *abalone_19*, *wilt* and *qsar-biodeg*. For the first two datasets overfitting is the cause of that phenomenon. However, for the other two, there is not a simple explanation and the *wilt* dataset is studied in more detail in Section 4.3.4.



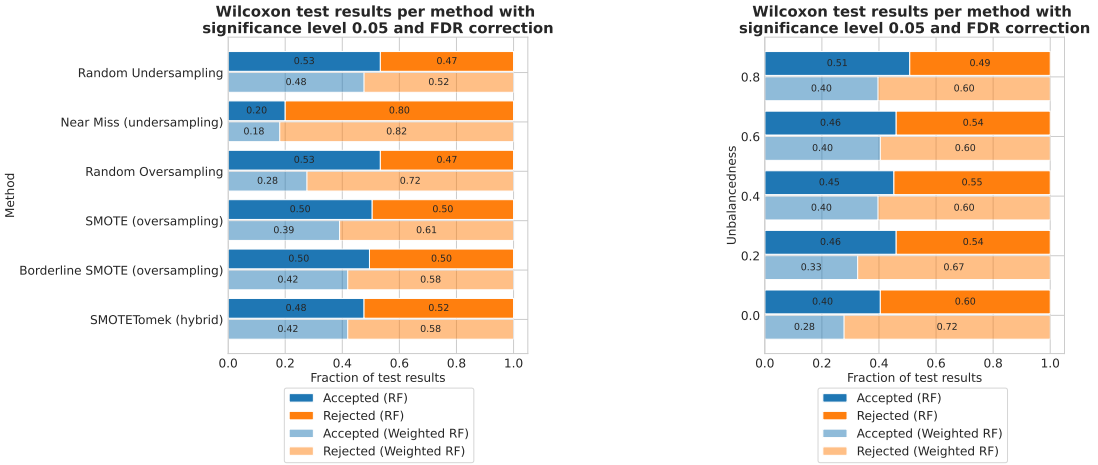
(a) Comparison of PDP for different Unbalancedness rate values.

(b) Comparison of ALE for different Unbalancedness rate values.

Figure 4.4: Comparison of model profiles on test datasets for different Unbalancedness rate values.

4.3.3. Model parts analysis

The model parts analysis includes the analysis of Variable Importance. The Wilcoxon test is used to test the null hypothesis that the VI profiles of the two models are the same. The first input VI profile is always of the model trained on the original dataset. The plots presented in this section show how many hypotheses were accepted and rejected in each group. This is done by calculating the p-values, applying the p-value adjustment algorithm called FDR, and making the decision whether to accept or reject the hypothesis with the significance level of 0.05.



(a) Comparison of Variable Importances for different methods.

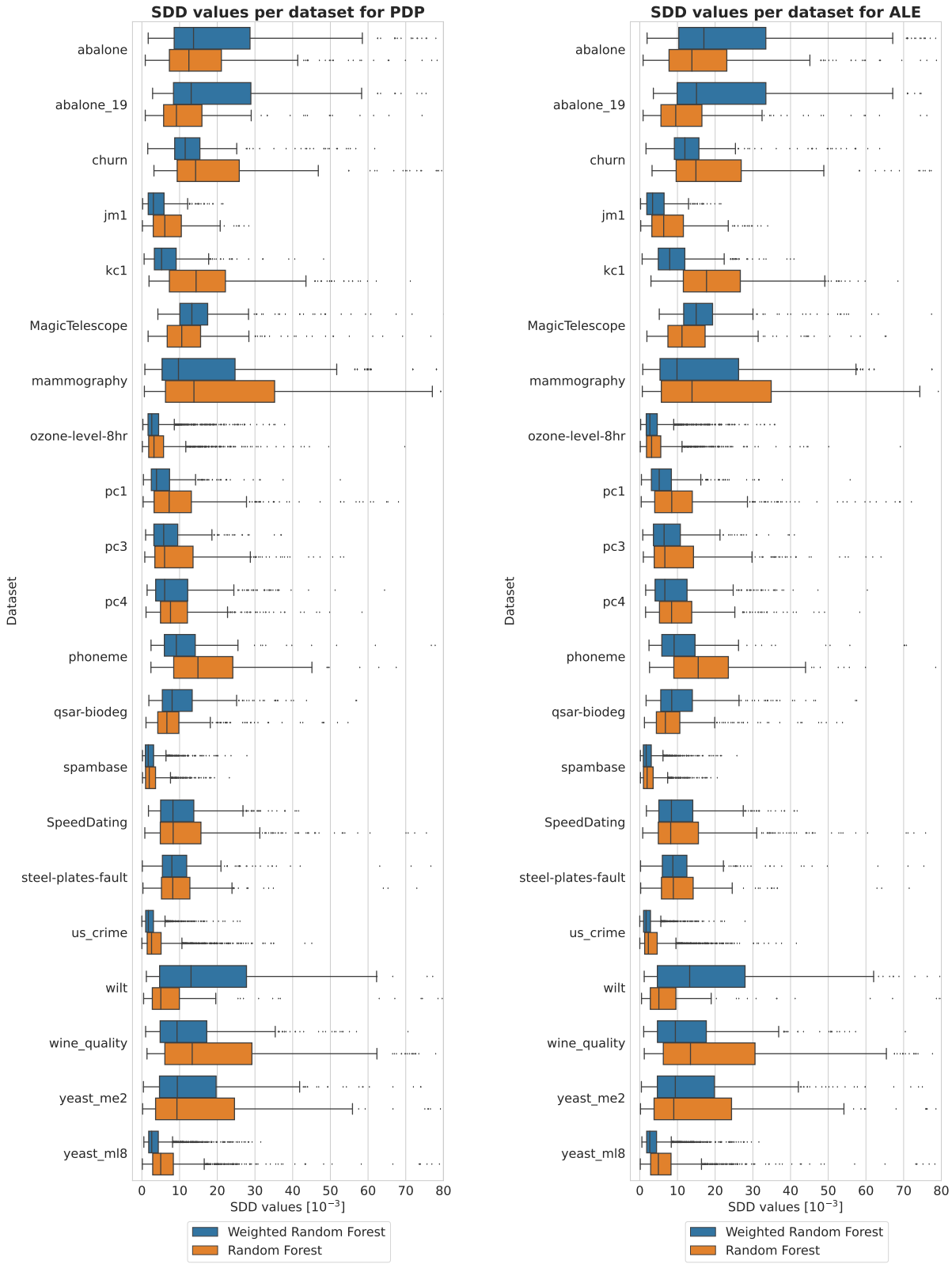
(b) Comparison of Variable Importances for different Unbalancedness values.

Figure 4.5: Comparison of model parts on test datasets for different filters.

Figure 4.5a presents the fraction of tests that have been accepted or rejected for each of the methods. The algorithm that clearly stands out in this chart is Near Miss - 80% of the tests are rejected for both RF and WRF models. This indicates that the Variable Importance profiles are changed most often after resampling with this technique. Another attention-worth observation is that in each case more tests were rejected for WRF than for RF models.

In Figure 4.5b the test results for different Unbalancedness rate values are compared. The fraction of rejected tests seems to be stable, slightly rising for RF models; however, it is clearly increasing in the WRF case. What is observed here suggests that the balancing methods affect also VI. If they did not have any influence, the fraction for WRF would remain at the same level as in the RF case. It is also interesting that bigger changes are observed in the WRF case, which suggests that the changes are not only affected by the model bias reduction, but also by changed relationships in the data.

Figure 4.7 shows the results for each dataset in the benchmarking set. The outcomes are completely different from those observed in SDD values. For example, the SDD values for *wilt* dataset rise after resampling, but in terms of VI, there is no difference. What is more, all of the tests in all cases are accepted which indicates completely no change. Another interesting example is *qsar-biodeg* dataset where all the hypotheses are accepted for the RF model, but almost all are rejected for WRF. This might show that the biased model (RF) has not changed, even though the relationships in the data might have changed (which was captured by WRF).



(a) Comparison of PDP for different datasets.

(b) Comparison of ALE for different datasets.

Figure 4.6: Comparison of model profiles on test datasets for different datasets.

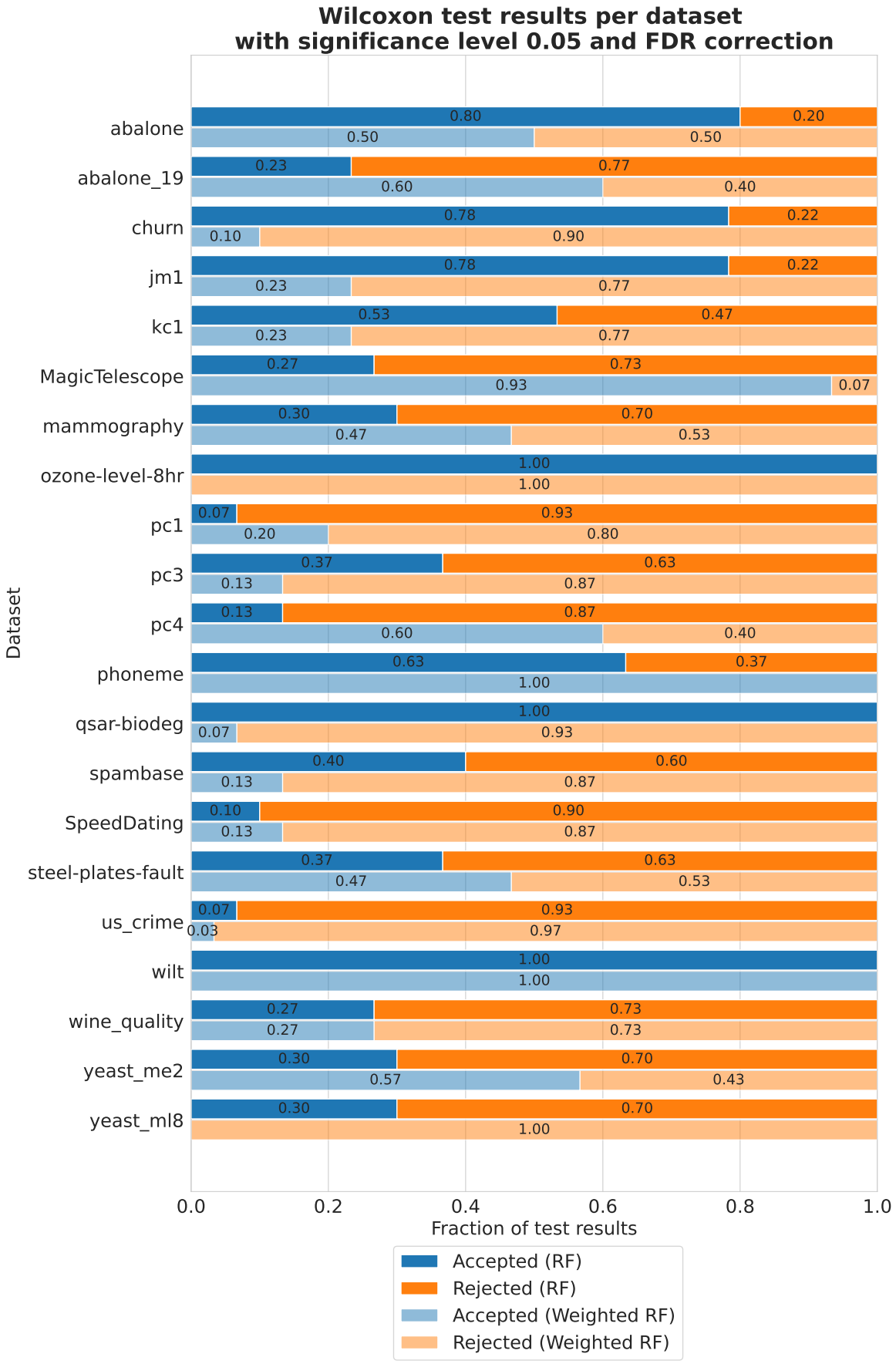


Figure 4.7: Comparison of Variable Importances for different datasets.

4.3.4. Interesting examples

In this section, the most interesting examples from the two previous sections will be studied in depth. The most noteworthy dataset in terms of comparing model profiles was the *wilt* dataset, whose SDD values increased after using the WRF model. The ALE plots of one of the variables, *Mean_ NIR*, are shown in Figure 4.11 (the PDP plot is omitted as both methods produce very similar outputs). Additionally, the curves are centred at the beginning of the range to make the comparison easier. For further analysis, it is crucial to know that all the models have high performance in terms of Balanced Accuracy, which is presented in Figure 4.8.

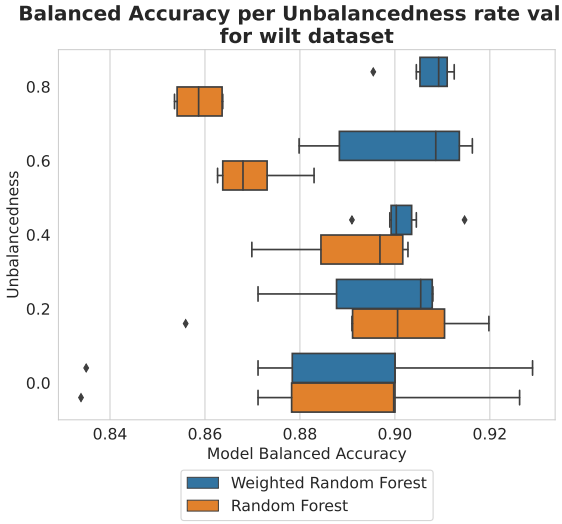
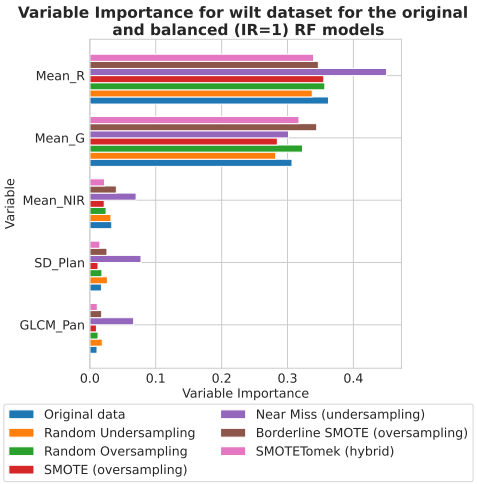


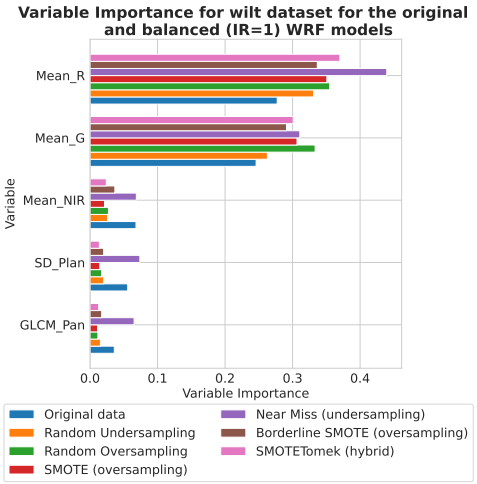
Figure 4.8: Balanced Accuracy for *wilt* dataset.

The *wilt* dataset [26] contains data for detecting diseased trees with the wilt disease. The dataset contains, among others, variables which are the mean spectral values of red, green, blue and near-infrared (NIR) bands in photos. According to the studies [31], the healthy plants reflect more NIR waves than the diseased. Consequently, the model trained on the original data with WRF (unbiased model trained on the unmodified dataset) seems to reflect true relationships in the data (the positive class represents an infected tree). However, not many other models produced similar outputs. Only one RF model trained on randomly undersampled data to $IR = 1$ has found the correct relationship. As far as WRF is considered, in almost all methods the more the dataset is balanced, the more it resembles an incorrect RF model. Nevertheless, a slight resampling does not seem to have an influence on behaviour.

The model part analysis has shown that it is worth exploring the details of the *wilt* and *qsar-biodeg* datasets. Even though there are significant changes in model behaviours in terms of PDP/ALE for *wilt* dataset, the Wilcoxon test has not revealed changes in Variable Importance. The results are correct as in all cases the order of variables in VI plots is the same (Figure 4.9).



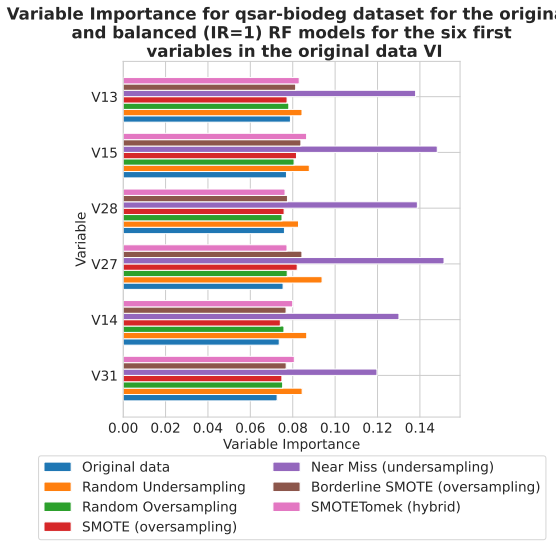
(a) Variable Importance for RF model.



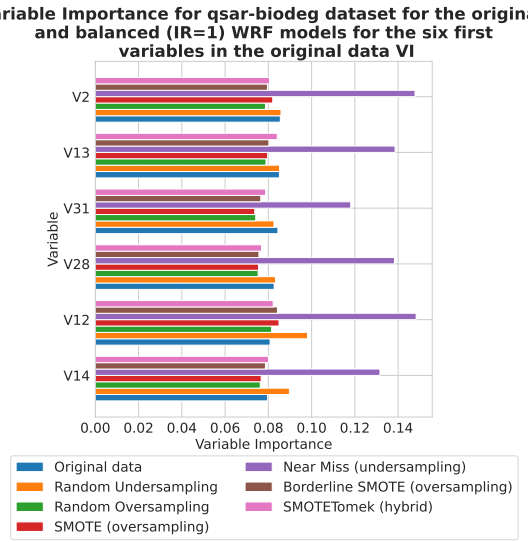
(b) Variable Importance for WRF model.

Figure 4.9: Variable Importance for *wilt* dataset for the original and balanced ($IR = 1$) datasets.

As far as *qsar-biodeg* dataset is considered, all hypotheses are accepted for the RF model, but almost all are rejected for WRF. The plots in Figure 4.10 presents the VI for the six first variables in the original datasets. Firstly, the orders in the original RF and WRF models are different as both plots show different variables on the vertical axis. Secondly, in both cases, the orders of variables are diverse - mainly because the differences between VI values are small. As a result, it is not possible to assess visually the Wilcoxon test results. It is based on ranks and therefore it only captures the differences in the order and probably these small differences in value cause the disorder.

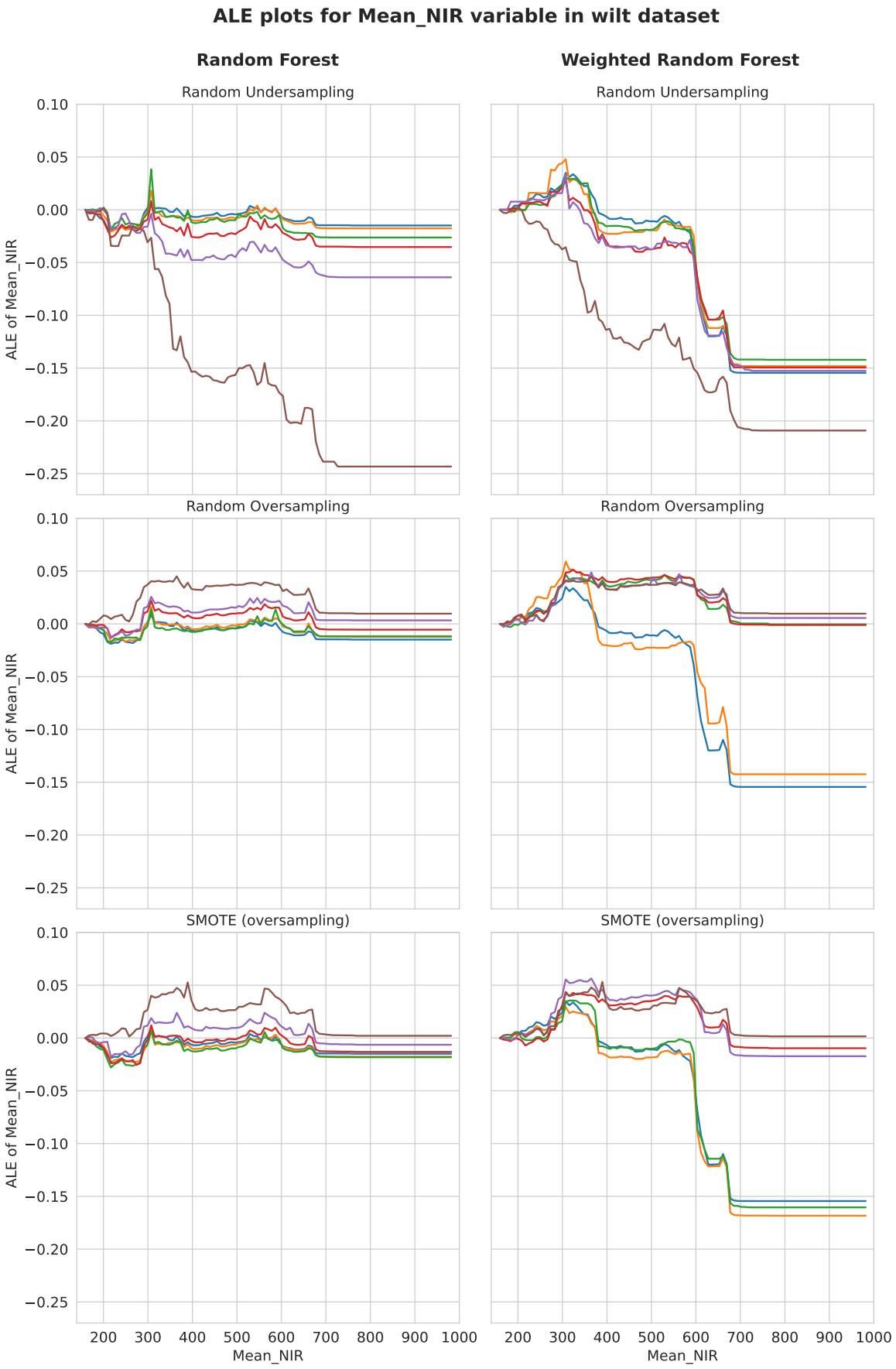


(a) Variable Importance for RF model.



(b) Variable Importance for WRF model.

Figure 4.10: Variable Importance for *qsar-biodeg* dataset for the original and balanced ($IR = 1$) datasets for the six first variables (out of seventeen) in the original data VI.


ALE plots for Mean_NIR variable in wilt dataset
Random Forest
Weighted Random Forest
Random Undersampling
Random Oversampling
SMOTE (oversampling)
ALE of Mean_NIR
Mean_NIR
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
200
300
400
500
600
700
800
900
1000

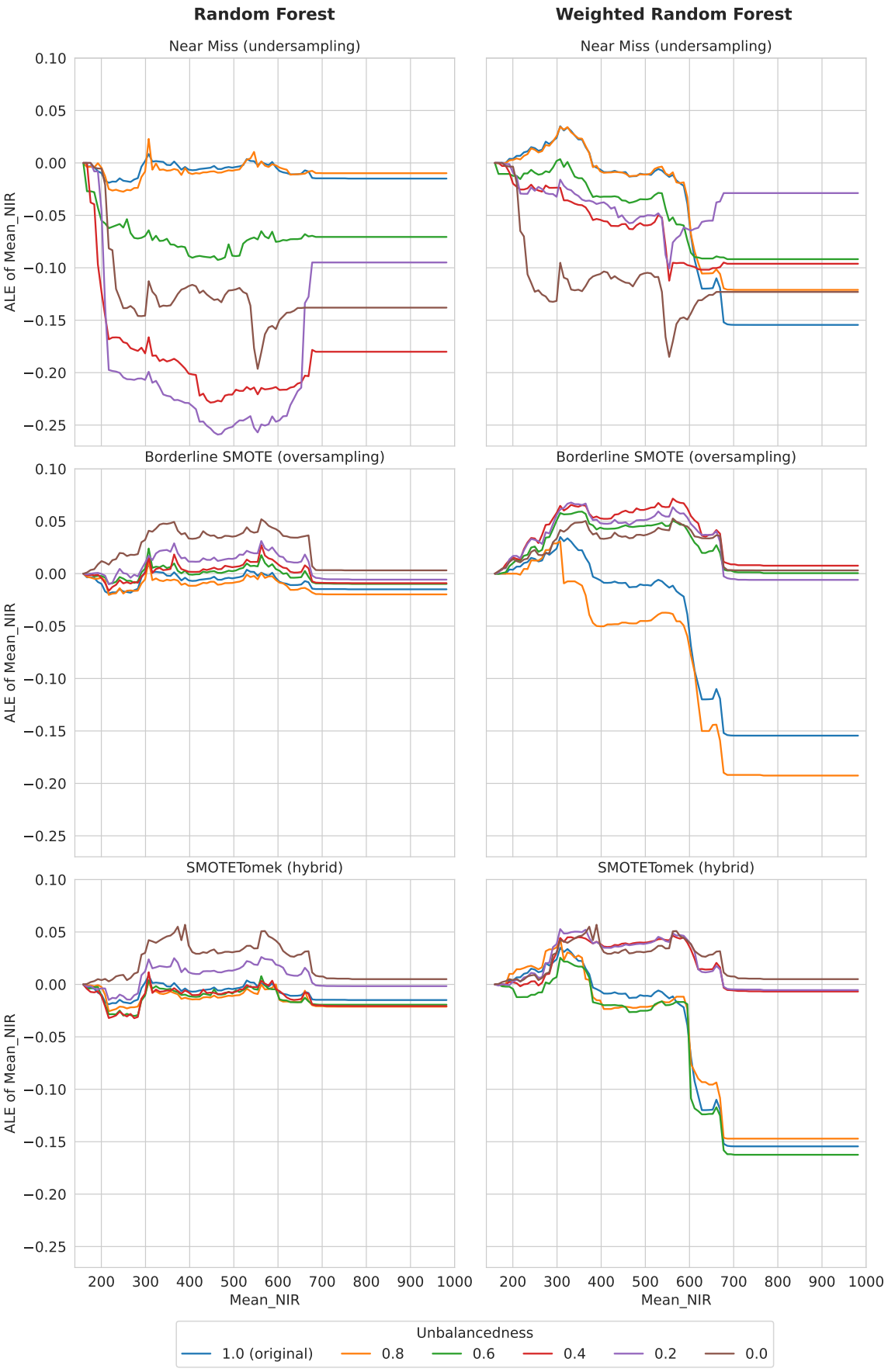


Figure 4.11: Centered ALE plots for *Mean_ NIR* variable in the *wilt* dataset for RF and WRF.

5. Conclusions

This work has attempted to answer the question of whether the balancing methods change the model behaviour and if so, what are the areas/datasets affected the most. I used two XAI techniques to investigate model behaviour and I proposed two methods to compare model explanations - one invented specifically for this work, based on standard deviation, and the other one based on the Wilcoxon test. To answer the main question of the thesis I created a benchmarking set, I applied to each dataset six different balancing methods with five different target Imbalance Ratios and I trained RF and WRF models.

5.1. Summary

The answer to the main question of this thesis is positive - the balancing methods do change the model behaviour. In most cases, they make the model less biased towards the majority class, which enables a model to extract existing relationships. In general, the more the dataset is changed, the more the model differs from its original. Nevertheless, it is not only a matter of bias - the balancing methods may change the relationships in data, as was shown for the *wilt* dataset. Therefore, it is important to remember that example and to always check the model explanations to see if it is reliable.

5.2. Discussion

In this thesis, it has been shown that the "safest" resampling method is Random Undersampling. Not only did it not affect the ALE explanation in *wilt* dataset for the WRF model, but also it made the RF model find the true relationship. On the other hand, the Near Miss method should be avoided as it can produce models which are overfitted and most of which differ from the original.

In this work, two methods for comparing model explanations were proposed. Each of them has some advantages and disadvantages, which are revealed in this study. Firstly, the SDD values

for PDP/ALE are good at capturing the changes in behaviour for each of the variables. However, the profiles can be misleading if there are outliers in the background dataset on which PDP/ALE are calculated - the output curves can be straight lines in large part. Moreover, the method does not take into account the Variable Importance - as a result, a variable which is less or not at all relevant has the same influence on the ASDD metric value as the most important one.

As far as the Wilcoxon test is considered, it is good at capturing the changes in the order of the variables in VI. It considers a bigger picture and does not reject the null hypothesis if only two unimportant variables are in a different order. On the other hand, it does not use all information available - it ignores the VI values, only signed differences. As a result, it may fail to capture similar VI if the differences in values are small and the order changes because of the randomness of the process of model training.

5.3. Future work

I foresee several possible directions for future work. Firstly, the benchmark study focused only on six balancing techniques. It could be interesting to repeat the experiments for a higher number of methods. Secondly, other ML models, for example, neural networks, can be studied to see whether the results are similar. Thirdly, other XAI methods could be used for the comparisons, for example, SHAP values. That study would also require developing new comparison metrics and, possibly, extending the `edgaro` package. Last but not least, one can focus on categorical and nominal variables in terms of model behaviour comparison. The developed SDD metric does not work well for such variables and possibly, another one should be developed. Another problem is also the shortage of balancing methods suitable for categorical or nominal variables.

Acknowledgements

This work was carried out with the support of the Laboratory of Bioinformatics and Computational Genomics and the High Performance Computing Center of the Faculty of Mathematics and Information Science, Warsaw University of Technology under computational grant number A-22-09.

Bibliography

[1] Ismail Alarab and Simant Prakoonwit. „Effect of data resampling on feature importance in imbalanced blockchain data: comparison studies of resampling techniques". In: *Data Science and Management* 5.2 (2022), pp. 66–76.

[2] Daniel W Apley and Jingyu Zhu. „Visualizing the effects of predictor variables in black box supervised learning models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.4 (2020), pp. 1059–1086.

[3] Hubert Baniecki et al. „dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python". In: *Journal of Machine Learning Research* 22.214 (2021), pp. 1–7.

[4] Gustavo E. A. P. A. Batista, Ana Lúcia Cetertich Bazzan, and Maria Carolina Monard. „Balancing Training Data for Automated Annotation of Keywords: a Case Study". In: *WOB*. 2003.

[5] Yoav Benjamini and Yosef Hochberg. „Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1 (1995), pp. 289–300. (Visited on 01/19/2023).

[6] Yoav Benjamini and Daniel Yekutieli. „The control of the false discovery rate in multiple testing under dependency". In: *The Annals of Statistics* 29.4 (2001), pp. 1165–1188.

[7] Przemyslaw Biecek and Tomasz Burzykowski. *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021.

[8] Bernd Bischl et al. *OpenML Benchmarking Suites*. 2017.

[9] Rich Caruana et al. „Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission". In: KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1721–1730.

[10] Mustafa Cavus and Przemysław Biecek. *Explainable expected goal models for performance analysis in football analytics*. 2022.

[11] N. V. Chawla et al. „SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357.

[12] Chao Chen and Leo Breiman. „Using Random Forest to Learn Imbalanced Data". In: *University of California, Berkeley* (Jan. 2004).

[13] Samrat Jayanta Dattagupta. „A performance comparison of oversampling methods for data generation in imbalanced learning tasks". MA thesis. 2018.

[14] Zejin Ding. „Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics". PhD thesis. Georgia State University, 2011.

[15] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017.

[16] Matthias Elter, Rüdiger Schulz-Wendtland, and Thomas Wittenberg. „The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process". In: *Medical physics* 34.11 (2007), pp. 4164–4172.

[17] Matthias Feurer et al. „Openml-python: an extensible python api for openml". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 4573–4577.

[18] 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.

[19] Jerome H. Friedman. „Greedy function approximation: A gradient boosting machine." In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232.

[20] Qinghua Gu et al. „A novel Random Forest integrated model for imbalanced data classification problem". In: *Knowledge-Based Systems* (2022), p. 109050.

[21] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. „Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *International conference on intelligent computing*. Springer. 2005, pp. 878–887.

[22] Charles R. Harris et al. „Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.

[23] Ahmad B. Hassanat et al. *Stop Oversampling for Class Imbalance Learning: A Critical Review*. 2022.

[24] 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.

[25] J. D. Hunter. „Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[26] Brian Alan Johnson, Ryutaro Tateishi, and Nguyen Thanh Hoan. „A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees". In: *International Journal of Remote Sensing* 34.20 (2013), pp. 6969–6982.

[27] Janis Klaise et al. „Alibi Explain: Algorithms for Explaining Machine Learning Models". In: *Journal of Machine Learning Research* 22.181 (2021), pp. 1–7.

[28] Bartosz Krawczyk. „Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5.4 (2016), pp. 221–232.

[29] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. „Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5.

[30] Yazhe Li, Niall Adams, and Tony Bellotti. „A Relabeling Approach to Handling the Class Imbalance Problem for Logistic Regression". In: *Journal of Computational and Graphical Statistics* 31.1 (2022), pp. 241–253.

[31] Christopher Lum et al. „Multispectral imaging and elevation mapping from an unmanned aerial system for precision agriculture applications". In: *Proceedings of the 13th International Conference on Precision Agriculture*. Vol. 31. 2016, pp. 4–10.

[32] Inderjeet Mani and I Zhang. „kNN approach to unbalanced data distributions: a case study involving information extraction". In: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126. ICML. 2003, pp. 1–7.

[33] Wes McKinney et al. „Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. 1. Austin, TX. 2010, pp. 51–56.

[34] Aum Patil, Aman Framewala, and Faruk Kazi. „Explainability of smote based oversampling for imbalanced dataset problems". In: *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE. 2020, pp. 41–45.

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

[36] Skipper Seabold and Josef Perktold. „statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference*. 2010.

[37] Sashank Sridhar and Sowmya Sanagavarapu. „Handling Data Imbalance in Predictive Maintenance for Machines using SMOTE-based Oversampling". In: *2021 13th International Conference on Computational Intelligence and Communication Networks (CICN)*. Sept. 2021, pp. 44–49.

[38] Ivan Tomek. „Two Modifications of CNN". In: *IEEE Transactions on Systems, Man, and Cybernetics*. IEEE. 1976, pp. 769–772.

[39] Joaquin Vanschoren et al. „OpenML: networked science in machine learning". In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60.

[40] Pauli Virtanen et al. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[41] Frank Wilcoxon. „Individual comparisons by ranking methods". In: *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

[42] Tao Wu et al. „Intrusion detection system combined enhanced random forest with SMOTE algorithm". In: *EURASIP Journal on Advances in Signal Processing* 2022.1 (May 2022), p. 39.

[43] Qiang Yang and Xindong Wu. „10 Challenging Problems in Data Mining Research". In: *Int. J. Inf. Technol. Decis. Mak.* 5 (2006), pp. 597–604.

List of symbols and abbreviations

AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
ML	Machine Learning
IR	Imbalance Ratio
PDP	Partial Dependence Profile
ALE	Accumulated Local Effects
VI	Variable Importance
RF	Random Forest
WRF	Random Forest with balanced class weights

List of Figures

List of Tables

List of Algorithms

List of Codes