UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Towards the automatic integration of missing patterns in unbalanced classification

Auteur·e(s) : Hadrien PLANCQ
Promoteur·rice(s) : Bertrand LEBICHOT
Année académique 2021-2022
Travail de fin d'études (TFE) en vue d'obtenir le titre de
Master (60) en Sciences de Gestion
Horaire de jour

Abstract

Payment card fraud is a significant problem for business owners, card issuers, and transactional services firms, resulting in significant financial losses each year. With the growing volume of data created by payment card transactions, it has become impossible for a human analyst to identify fraudulent patterns in transaction datasets, which are frequently characterized by a huge number of samples and multiple dimensions. As a result, during the last decade, the development of payment card fraud detection systems has shifted to approaches based on machine learning techniques, which automate the process of detecting fraudulent patterns from vast amounts of data.

However, not all steps can be automated. Some conclusions cannot yet be made efficiently by machine learning. More specifically, ML algorithms have a lot of difficulty dealing with class imbalance. Indeed, transaction data contains far more genuine transactions than fraudulent ones: in a real-world dataset, the percentage of fraudulent transactions is often considerably under 1%. Learning from imbalanced data is difficult since most learning algorithms struggle with big class disparities. Class imbalance necessitates the adoption of additional learning procedures such as sampling or weighting.

In this work, we will try to improve the efficiency of existing machine learning models as well as to create new ones with the goal of pushing for the total automation of this process, removing the need for human operators. Moreover, this study aims to try to draw conclusions on the avenues to be explored in future research concerning the detection of fraud in transactions.

Contents

Introduction

Payment card fraud is a significant problem for business owners, card issuers, and transactional services firms, resulting in significant financial losses each year. Indeed, global card fraud losses climbed from $9.84 billion in 2011 to $27.85 billion in 2018, according to the Nilson 2019 research, and are anticipated to reach more than $40 billion by 2027 [1].

It is well recognized that detecting fraudulent behaviors in payment card transactions is a difficult task. With the growing volume of data created by payment card transactions, it has become impossible for a human analyst to identify fraudulent patterns in transaction datasets, which are frequently characterized by a huge number of samples and multiple dimensions. As a result, during the last decade, the development of payment card fraud detection systems has shifted to approaches based on machine learning techniques, which automate the process of detecting fraudulent patterns from vast amounts of data [2].

Machine learning techniques have been integrated into credit card fraud detection systems, greatly improving their ability to detect fraud and assisting payment processors in identifying unlawful transactions. Despite the fact that the number of fraudulent transactions has gradually climbed in recent years, the percentage of fraud losses began to drop in 2016, a trend that can be linked to the growing deployment of machine learning solutions [1].

However, not all steps can be automated. On Figure 1, you can see the different steps that compose a card fraud detection system. The first two layers (Terminal and Transaction Blocking Rules) are executed in real-time (i.e. within milliseconds and before authorization). The next two layers (Scoring Rules and Data-Driven Model), are executed in near real-time to potentially block the card and prevent additional frauds. Finally, the last layer (Investigators) is the only one requiring human intervention and is carried out offline [3].

The reason why the last step of this process is not automated is that some conclusions cannot yet be made efficiently by machine learning. More specifi-

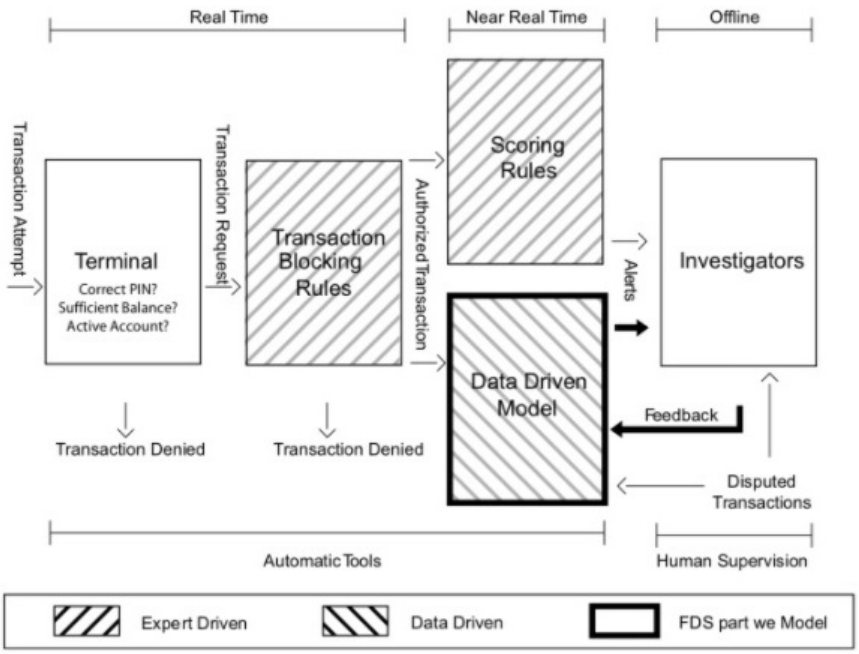


Figure 1: The layers of control in an card fraud detection system [3].

cally, ML algorithms have a lot of difficulty dealing with class imbalance. Indeed, transaction data contains far more genuine transactions than fraudulent ones: in a real-world dataset, the percentage of fraudulent transactions is often considerably under 1%. Learning from imbalanced data is difficult since most learning algorithms struggle with big class disparities. Class imbalance necessitates the adoption of additional learning procedures such as sampling or weighting.

In this work, we will try to improve the efficiency of existing machine learning models as well as to create new ones with the goal of pushing for the total automation of this process, removing the need for human operators (the Investigators step in Figure 1). It should be noted, however, that our approach is only focused on replacing the investigator by a machine learning model. Indeed, before reaching this stage, a transformation of the transaction data is already performed based on several information such as user habits, amounts, recipients, geolocation, ect. The data that our model receives is the same data and indicators that an investigator could receive in Figure 1. As pointed out by Kriviko and Wei et al., this data is of crucial importance in fraud detection and it is an approach that combines this analysis of customer habits with machine learning that will be the most effective in detecting fraudulent transactions [4][5]. The model implemented in this work is therefore intended to be used on pre-processed data, such as those generated on a regular basis by banks in their quest of fraud detection.

Haixiang et al. presents a variety of possible approaches to deal with imbalance classification in general, several of which will be explored in this work [6]. Other works also provide similar overviews of possible solutions [7][8]. Le Borgne, Sibilini, Lebichot et al. presents an overview of the problem of detecting fraudulent transactions and lays the groundwork for the various models that have already proven their worth in this type of problem [9].

Although already well studied in the literature, fraud detection is a constantly changing field. Indeed, transaction and fraud patterns change over time, fraudsters adopt new techniques as the old ones become obsolete. It is therefore important to continue research in this area.

The rest of this work is composed as follows : Chapter 1 will present the different challenges we face with respect to imbalanced classification as well as the metrics used to evaluate the results of our models. In Chapter 2, a presentation of the different models implemented will be made as well as an analysis of the optimal parameters of these models. Finally, in Chapter 3, we will present the results of our different models and try to draw conclusions on the avenues to be explored in future research concerning the detection of fraud in transactions.

Chapter 1

The problem with imbalanced classification

In this chapter, we will explain in detail the problem with imbalanced datasets. We will also look at the different metrics that can be relevant to evaluate a model trained on such datasets. Finally, we will present the datasets on which we have decided to work in this study.

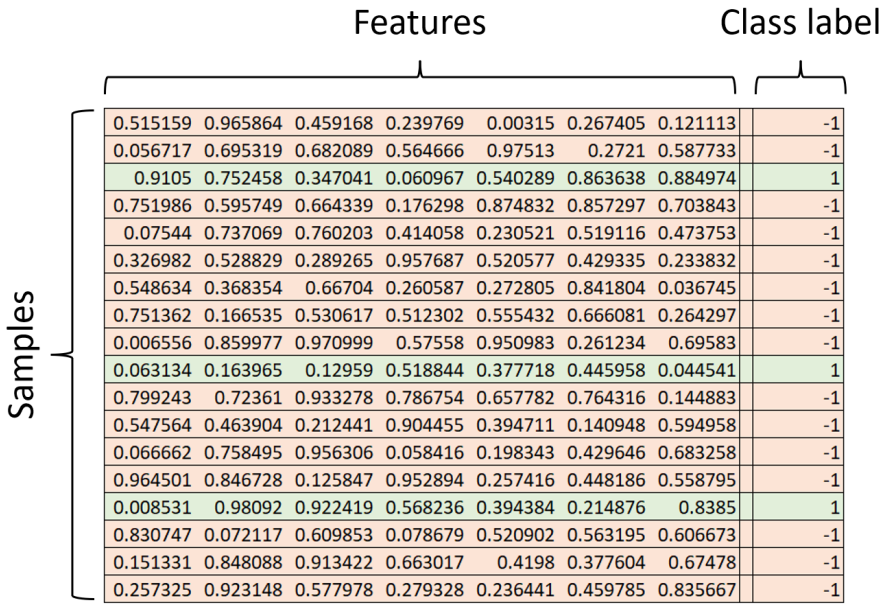
Imbalanced classification problems

Classification is the principle of assigning a class label to each observation. These labels can be multiple, this is called a multi-class classification problem. On the other hand, when the problem has only 2 classes it is called a binary classification problem. In this work, we will focus on binary classification problems because, in the context of fraud detection, there are only two classes of transactions: fraudulent and non-fraudulent.

The number of examples that belong to each class can be called the class distribution. Unbalanced classification refers to a problem where the number of examples in one class is much higher than in the other, creating an imbalance. In other words, the class distribution is skewed. It is common to describe class imbalance in terms of a ratio : an unbalanced binary classification problem with an imbalance of 1 to 50 (1:50) means that for every example in one class, there are 50 examples in the other class.

In general, when the problem has only a slight imbalance between classes, it can be treated as a normal classification problem. On the other hand, severe class imbalance can be difficult to model and requires the use of more advanced techniques. In the case of fraud detection, the imbalance is very consequent. Indeed, the imbalance ratio can range from 1:1000 to 1:5000 [10].

On Figure 1.1, you can see an illustration of a generic imbalanced dataset. We notice that there are many more -1 labels (non-fraudulent) than 1 labels (fraudulent). Moreover, the samples have continuous features as a real bank transaction dataset would.



0.515159	0.965864	0.459168	0.239769	0.00315	0.267405	0.121113	-1
0.056717	0.695319	0.682089	0.564666	0.97513	0.2721	0.587733	-1
0.9105	0.752458	0.347041	0.060967	0.540289	0.863638	0.884974	1
0.751986	0.595749	0.664339	0.176298	0.874832	0.857297	0.703843	-1
0.07544	0.737069	0.760203	0.414058	0.230521	0.519116	0.473753	-1
0.326982	0.528829	0.289265	0.957687	0.520577	0.429335	0.233832	-1
0.548634	0.368354	0.66704	0.260587	0.272805	0.841804	0.036745	-1
0.751362	0.166535	0.530617	0.512302	0.555432	0.666081	0.264297	-1
0.006556	0.859977	0.970999	0.57558	0.950983	0.261234	0.69583	-1
0.063134	0.163965	0.12959	0.518844	0.377718	0.445958	0.044541	1
0.799243	0.72361	0.933278	0.786754	0.657782	0.764316	0.144883	-1
0.547564	0.463904	0.212441	0.904455	0.394711	0.140948	0.594958	-1
0.066662	0.758495	0.956306	0.058416	0.198343	0.429646	0.683258	-1
0.964501	0.846728	0.125847	0.952894	0.257416	0.448186	0.558795	-1
0.008531	0.98092	0.922419	0.568236	0.394384	0.214876	0.8385	1
0.830747	0.072117	0.609853	0.078679	0.520902	0.563195	0.606673	-1
0.151331	0.848088	0.913422	0.663017	0.4198	0.377604	0.67478	-1
0.257325	0.923148	0.577978	0.279328	0.236441	0.459785	0.835667	-1

Figure 1.1: Illustration of a generic imbalanced dataset.

When working with an imbalanced classification problem, the minority class is of the most interest. This means that a model's skill in correctly predicting the class label or probability for the minority class is more important than the majority class. Indeed, in the case of fraud detection, although being a very small minority, fraudulent transactions accounted for $27.85 billion of loss in 2018 worldwide and this number is in growing each year.

The minority class is harder to predict because there are few examples of this class, by definition. This means it is more challenging for a model to learn the characteristics of examples from this class, and to differentiate examples from this class from the majority class.

The abundance of examples from the majority class can swamp the minority class. Most machine learning algorithms for classification are designed and demonstrated on problems that assume an equal distribution of classes. This means that a naive application of a model may focus on learning the characteristics of the

abundant observations only, neglecting the examples from the minority class that is, in fact, of more interest and whose predictions are more valuable [11]. More intuitively, imagine a classification problem with a ratio of 1:100. A model that characterises all inputs as belonging to the majority class would therefore have an accuracy of over 99%. It is therefore easy to understand the difficulty of the problem. Moreover, there is a need to use more advanced metrics than the accuracy, which we will discuss in the next section.

Metrics used to evaluate the performance of our model

As explained earlier, the metrics used to assess the effectiveness of our model must take into account class imbalance as well as the fact that it is the minority class that is most important to predict effectively.

Therefore, in this work we will evaluate our results through 5 metrics. Some of these metrics take class imbalance into account in order to emphasise the importance of a good prediction of minority class samples, such as the balanced accuracy score (BAS). However, other metrics do not specifically take class imbalance into account, such as the accuracy (ACC), F1-score (F1-score), ROCAUC score (ROCAUC). Indeed, although we are mainly interested in predicting the minority class efficiently, it is important to be able to assess the overall performance of our model as well, we don't want the model to always predict the sample to be in the minority class. A final metric that will be used is the average precision (AP), which will be used in the precision-recall curve when we compare the different models implemented.

You can see in Table 1.1 a short explanation of these different metrics. Precision P is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p) : $P = \frac{T_p}{T_p+F_p}$. Recall R is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n) : $R = \frac{T_p}{T_p+F_n}$.

Selection of datasets

For privacy concerns, it was not possible to carry out the experiments on a real dataset of bank transactions. Indeed, this kind of data being quite sensitive for a bank, they are only willing to give access to it in rare occasions and not for a small work like this one.

balanced accuracy score (BAS)	The average of recall obtained on each class. Also equivalent to accuracy with class-balanced sample weights.
ROCAUC score (ROCAUC)	The Area Under the Receiver Operating Characteristic Curve (ROC AUC), calculated from prediction scores.
F1-score (F1-score)	The harmonic mean of precision and recall : $F_1 = 2\frac{P \times R}{P+R}$
accuracy (ACC)	The number of correct predictions over the total number of predictions.
average precision (AP)	Average precision summarizes a precision-recall plot as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight : $AP = \sum_n (R_n - R_{n-1}) P_n$ where P_n and R_n are the precision and recall of the nth threshold.

Table 1.1: Short explanation of the different metrics used in this report.

However, it is useful to look at the characteristics of this type of dataset. Indeed, these datasets are not simply a list of transactions, the bank first performs transformations on it to make it more usable. Using behavioral models based on demographic information, expert features or RFMs [12], the bank will transform each transaction into a series of indicators in the form of floats.

Therefore, in this work we will work on a small group of datasets having some characteristics similar to a bank transaction dataset. These caracteristics are :

- 2 classes (binary classification problem).
- Less than 100 features.
- Continuous variables.

As a result of these conditions, we were able to establish a list of 11 datasets available on `Openml` on which we will work [13]. These datasets as well as their properties are presented in the Table 1.2.

Dataset	Features	Samples			Ratio
		Majority	Minority	Total	
kc2	21	415	107	522	25.78%
climate-model-(...)	20	494	46	540	9.31%
blood-transfusion-(...)	4	570	178	748	31.23%
pc1	21	1032	77	1109	7.46%
pc4	37	1280	178	1458	13.91%
pc3	37	1403	160	1563	11.40%
kc1	21	1783	326	2109	18.28%
ozone-level-(...)	72	2374	160	2534	6.74%
wilt	5	4578	261	4839	5.70%
churn	20	4293	707	5000	16.47%
phoneme	5	3818	1586	5404	41.54%

Table 1.2: The 11 datasets studied in this work

Chapter 2

Overview of the various implemented models

In this chapter, we will look at the different models that have been implemented to find a solution to the problem presented above. The 7 models implemented have been selected to cover a variety of fundamentally different models. For example, it is not the intention here to test several undersampling models because, despite small differences, these models will give a fairly similar result. Figure 2.1 gives an overview of the different classes of models that can be considered [7].

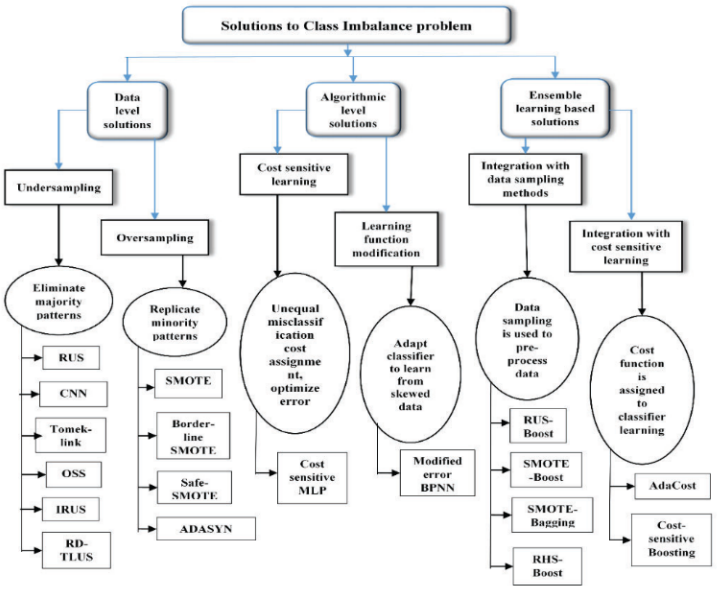


Figure 2.1: Types of available solutions to class imbalance problem [7]

In this work, at least one method from each category (data level solutions,

algorithmic level solutions and ensemble learning based solutions) was considered. More specifically, models using Random Undersampling (RUS), Synthetic Minority Oversampling Technique (SMOTE), a Cost-Sensitive Random Forest (Cost-Sensitive RF) and Random Undersampling Boosting (RUS-Boost) were implemented.

In addition, 3 other models have been added. The first one, eXtreme Gradient Boosting (XGBoost), has been implemented because this model represents the state of the art in machine learning at the moment. The other two models are custom models, derived from RUS, named Error Model and Shapley Error Model. We will come back in more detail to the functioning of these two models later in this chapter. It is worth noting that unless otherwise stated, all these models use a Random Forest as their core model. This is indeed an efficient model and easy to combine with the various solutions to class imbalance.

For each of these models, a number of hyper-parameters can be optimised. However, as we work on different datasets than the one for which the model is created, we will remain unspecific and only focus on one hyper-parameter each time, the one related to the weight or the number of samples of each class. More precisely, you can find in Table 2.1 the hyper-parameter studied for each model.

In the rest of this chapter, we will look at these 7 models one by one to find the optimal hyper-parameters for each of them. This analysis will be carried out on $n = 10$ repetitions for each of the hyper-parameter values considered in order to reduce the randomness in the result and to be able to observe the standard deviation on each metric. In addition, the Nemenyi statistical test will be used to evaluate the different hyper-parameter values and assess whether their contribution is significant enough to be anything other than random. This evaluation will be done through critical difference (CD) graphs, which is used to display the differences in method performance [14]. In order to draw a conclusion from a Nemenyi test, the values of the two models must be at least a critical difference (CD) apart. Finally, we will also explain in detail how the two custom models work.

2.1 Random Undersampling (RUS)

Random Undersampling is the most basic undersampling method, which is a data level solution to imbalance. It offers to randomly eliminate the majority class instances according to a pre-defined sampling rate. This rate is the parameter on which we conducted an analysis : the ratio of the number of samples in the mi-

Model	Hyper-parameter	Range
RUS	The ratio of the number of samples in the minority class over the number of samples in the majority class after undersampling.	$[\frac{1}{2}, \mathbf{1}, 2, 4, 8]$
SMOTE	The ratio of the number of samples in the minority class over the number of samples in the majority class after resampling, represented by the parameter `sampling_strategy` in `imblearn.over_sampling.SMOTE`	[0.5,0.6,0.7,0.8, 0.9,**1.0**]
RUSBoost	The ratio of the number of samples in the minority class over the number of samples in the majority class after resampling, represented by the parameter `sampling_strategy` in `imblearn.ensemble.RUSBoostClassifier`	[0.5,0.6,0.7,0.8, 0.9,**1.0**]
Cost-sensitive RF	The weight associated to the minority class, represented by `class_weight` in `sklearn.ensemble.RandomForestClassifier`. The weight of the majority class is 1.	[1,2,**4**,8,16]
XGBoost	The ratio of the weights associated to each class, represented by `scale_pos_weight` in `xgboost.XGBClassifier`	[1,2,4,8,**16**,32,64]
Error Model	The fraction of samples removed by RUS that are restored in the sample, see section 2.6 for technical explainations	[0,0.05,0.1,**0.15**, 0.20,0.25,0.30, 0.35,0.40,0.45, 0.50,0.55,0.60, 0.65,0.70]
Shapley Error Model	The number of times we remove redondant samples using their shapley values, see section 2.7 for technical explainations	[0,**1**,2]

Table 2.1: The hyper-parameters tweaked for each of the models

nority class over the number of samples in the majority class after undersampling was varied following the values [$\frac{1}{2}$,1,2,4,8].

Hyper-parameter tuning

In Figure A.1 in Appendix you can see that the 4 Nemenyi tests performed put forward two values of the parameter which are systematically the best whatever the metric used. These values are 1 and $\frac{1}{2}$. In order to decide between these two values, it is interesting to focus on the balanced accuracy score metric, which is the most representative of the efficiency of the model on the minority class. Indeed, on Figure B.1b in the Appendix, we notice that the average balanced accuracy score over the 10 iterations is higher for the value 1 than for the value 1/2 for this metric. It is therefore 1 that will be used as the value of this parameter in the rest of this work. The boxplots of these experiments are available in Figure B.1 in the Appendix.

2.2 SMOTE

SMOTE is a model that aims to oversample the minority class, which is a data level solution to imbalance. It consists of synthesising new examples from existing ones. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. For this implementation, `imblearn.over_sampling.SMOTE` was used. The hyper-parameter studied is the ratio of the number of samples in the minority class over the number of samples in the majority class after resampling, ranging from 0.5 to 1.0.

Hyper-parameter tuning

In Figure A.2 in Appendix, you can see that the Nemenyi test is not conclusive for the F1-score metric and that it tends to indicate that smaller values of the parameter are statistically better for the accuracy (ACC) and ROCAUC score (ROCAUC) metrics. This last result is quite logical as these metrics are representative of the efficiency of the model as a whole, and therefore mainly of its efficiency on the majority class. Less sample generation in the minority class will therefore tend to favour a good prediction of the majority class and thus favorise these metrics.

However, we observe that Nemenyi's test of the BAS metric is conclusive and tells us that the closer this hyper-parameter is to 1.0, the higher the BAS will be. As this metric is directly linked to the efficiency of the model on the minority class,

the one we are trying to predict in the most efficient way, the value of the hyper-parameter we have decided to select will be 1.0. SMOTE will therefore generate as many samples in the minority class as there are samples in the majority class. The boxplots of these experiments are available in Figure B.2 in the Appendix.

2.3 RUS-Boost

RUS-Boost is an ensemble learning based solution that uses a combination of RUS and the standard boosting procedure AdaBoost to better model the minority class by removing majority class samples. For this implementation, `imblearn.ensemble.RUSBoostClassifier` was used. The hyper-parameter studied is the ratio of the number of samples in the minority class over the number of samples in the majority class after undersampling, ranging from 0.5 to 1.0.

Hyper-parameter tuning

In Figure A.3 in Appendix, you can see that the Nemenyi test is not conclusive for the metric BAS. As for the metric F1-score, the only conclusion we can draw is that the value 0.5 for this parameter gives a statistically better model than the parameter 1.0. These results tell us that changing this parameter does not have a significant influence on the efficiency of our model in predicting the minority class elements.

However, we notice that for the accuracy and ROCAUC metrics, Nemenyi tests indicate that smaller values of the parameter are statistically better. This last result is quite logical as these metrics are representative of the efficiency of the model as a whole, and therefore mainly of its efficiency on the majority class. Less undersampling of the majority class will therefore tend to favour a good prediction of the majority class and thus favorise this metric.

We are primarily interested in predicting the minority class samples efficiently and these statistical tests do not allow us to draw any conclusions on the effect of this parameter in this respect. Therefore, the value chosen for this parameter will be the one used by default in the literature, i.e. 1.0, which represents an undersampling of the majority class until we have as much samples as in the minority class. The boxplots of these experiments are available in Figure B.3 in the Appendix.

2.4 Cost-Sensitive Random Forest

Cost-Sensitive RF, an algorithmic level solution to imbalance data, is a Random Forest model where the weights associated to each class are modified to give more importance to the minority class. The hyper-parameter studied for this model is the weight associated to the minority class, with the weight associated to the majority class being always 1. The tested values are $[1, 2, 4, 8, 16]$.

Hyper-parameter tuning

In A.4 in Appendix you can see that the Nemenyi test is not conclusive for the metric ROCAUC score. As for the 3 other metrics, we can observe that concerning F1-score, the value 4 is statistatically better than the value 1, concerning accuracy the value 1,2 and 4 are statistatically better than the value 16, concerning balanced accuracy score the value 2,4 and 8 are statistatically better than the value 1. The value 4 being among the best for each of these 3 metrics, it is this value that will be selected for our parameter. The boxplots of these experiments are available in Figure B.4 in the Appendix.

2.5 XGBoost

XGBoost, is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way [15]. This model is the state of the art in the field of machine learning at the moment. For this implementation, `xgboost.XGBClassifier` was used. The hyper-parameter studied is the ratio of the weights associated to each class, the tested values are $[1, 2, 4, 8, 16, 32, 64]$.

Hyper-parameter tuning

In Figure A.5 in the Appendix you can see that the Nemenyi test is not conclusive for the metric F1-score. As for the accuracy and the ROCAUC score, we can observe that the lower the value of this parameter the best it is. This is quite comprehensive as these metrics are representative of the efficiency of the model as a whole, and therefore mainly of its efficiency on the majority class. Less weight for the minority class will therefore tend to favour a good prediction of the majority class and thus favorise these metrics.

Finally, the BAS metric tends to indicate that the values 8, 16, 32 and 64 for this parameter are statisically better than the value 1. However, on Figure B.5b

in the Appendix, we notice that the average balanced accuracy score over the 10 iterations is higher for the value 16 than for the values 8, 32 and 64 for this metric. It is therefore 16 that will be used as the value of this parameter in the rest of this work. This represents a minority class with a weight 16 times the weight of the majority class. The boxplots of these experiments are available in Figure B.5 in the Appendix.

2.6 Error Model

The Error Model is a custom model derived from RUS. Indeed, this model is based on the observation that during the random undersampling, there is a risk that some samples that are important for the final model to be efficient are eliminated. The Error Model therefore reintegrates in a second step some eliminated samples which could be the most misclassified by the model.

In order to understand in depth how this model works, we will now look at it step by step with an example. In Figure 2.2, you can see a schematic diagram of how this model works. We will now go through these steps one by one.

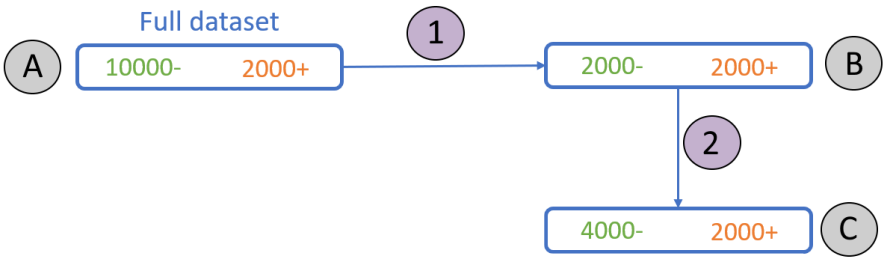


Figure 2.2: Illustration of how the Error model works

- **Step A :** The example dataset used here is a hypothetical dataset containing 10000 genuine transactions and 2000 fraudulent transactions. This dataset is separated into train, validation and test sets.
- **Step 1 :** In this step, a RUS (see Section 2.1) is performed to bring the majority sample size in the train set down to the same level as the minority sample.
- **Step B :** The training dataset now has an equal number of samples in the majority class as in the minority class. A Random Forest model is trained on it and validated on the validation set.

- **Step 2 :** In this step, we reintegrate some samples eliminated from the majority class that could be the most misclassified by the model.

  To do so, we define an error metric as $1 - (pred_{proba} * pred)$, with $pred$ and $pred_{proba}$ being respectively the prediction of the model (0 or 1) and the probability that the prediction is 1. In other words, this error will be equal to 1 for the samples of the majority class and $1 - pred_{proba}$ for the samples of the minority class. It is this error function that will be used as a ranking to choose which samples are to be reintegrated into the dataset. This sample selection step is sometimes done by hand to detect in which region of the space the undersampled model behaves badly. However, the objective of this work is to automate this step in order to remove the need for a human operator.

  Then, we train a Random Forest model on the validation set whose objective is to estimate the error. We thus obtain a model trained to estimate if a sample belongs to the majority class. This model is then used on the train set in order to estimate the samples belonging most certainly to the majority class.

  Finally, we select a percentage of these samples most likely to be of the majority class. This percentage is modifiable and an analysis of the effect of this parameter on the overall efficiency of the model will be carried out in the rest of this section. In the example in Figure 2.2, this percentage is 20% (20% of the majority class is selected to be reintegrated into the sample).

- **Step C :** In this last step, we start from the train set and we redo a RUS as in step 1, except that this time we artificially add the samples selected in the previous step before training our model. We thus performed a RUS to which we add the most representative samples of the majority class so that the latter is predicted efficiently despite a significant decrease in its size due to undersampling.

Hyper-parameter tuning

As explained earlier, the parameter on which this analysis will be performed is the fraction of the sample removed by RUS that will be reintegrated into the sample. This fraction will be varied from 0% (being exactly the same as the RUS model) to 70% of the removed samples reintegrated to the set.

One could ask why not also play on the same parameter as with RUS (see Section 2.1). Indeed, since RUS is an integral part of the Error Model, this parameter also plays a role in the latter. However, we hypothesized that since these models are very similar, the parameter selected for RUS, i.e. 100%, is best suited for maximum efficiency in the Error Model.

In Figure A.6 in Appendix you can observe that the Nemenyi tests are inconclusive for all metrics except for the accuracy. Indeed, for the 3 other metrics all the values of the parameter are in a range smaller than a critical difference (CD). For the accuracy, we observe a tendency indicating that the higher the value of the parameter, the higher its ACC. This last result is quite logical as this metric is representative of the efficiency of the model as a whole, and therefore mainly of its efficiency on the majority class. The more samples from the majority class are reintroduced, the more it will favour a good prediction of the majority class and thus favorise this metric.

In this work, we are primarily interested in effective prediction of the minority class, and this hyper-parameter does not seem to have an effect on this. Indeed, the most important metric, the balanced accuracy score (BAS), does not vary significantly when the studied parameter varies. Therefore, in order to select a value for this parameter, we will look at three points of importance. First, the goal of this custom model is not to do exactly the same thing as a RUS, so we will not select too small values for this parameter. Secondly, reintroducing too many samples from the majority class tends to make the model slower while negating the beneficial effect of RUS on the final model result. Finally, we observe that in terms of accuracy (ACC) only the values of 0, 5 and 10% are statistically worse than the highest values for this parameter. Therefore, considering these different arguments, the selected value for this parameter is 15%. The boxplots of these experiments are available in Figure B.6 in the Appendix.

2.7 Shapley Error Model

The Shapley Error Model is a custom model derived from RUS and the Error Model. Indeed, this model follows the same operation as the Error Model and adds an additional loop at the end of it. The idea is that when reintegrating the most representative samples of the majority set, some of them might be redundant with each other or with those already present in the undersampled set. The aim of this model is therefore to try to remove the redundant samples and replace them with samples that would be less redundant for an overall improved efficiency of the model.

To do this, we will use the SHAP (SHapley Additive exPlanations) method [16], a method to explain individual predictions. SHAP is based on the game theoretically optimal Shapley values [17]. Here is a quick explanation of how this model works.

SHapley Additive exPlanations (SHAP) quick explanation

The goal of SHAP is to explain the prediction of an instance x by computing the contribution of each feature to the prediction, these are called Shapley values. Shapley values tell us how to fairly distribute the prediction among the features, i.e. what is the contribution of each feature in the final predition. These Shapley values are denoted $\phi_j \in \Re$ for each feature j. SHAP specifies the expanation as :

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

where g is the explanation model, $z' \in \{0,1\}^M$ is the coalition vector and M is the maximum coalition size. In the coalition vector, an entry of 1 means that the corresponding feature value is "present" and 0 that it is "absent".

The objective of this model is therefore to find the ϕ_j's. For this, the method will compare the prediction of the model with and without some features (by setting their corresponding z_j to 0). This comparison will allow to see what is the contribution of this feature to the final solution, or in other words what is the ϕ_j corresponding to this feature.

Finally, for x, the instance of interest, the coalition vector x' is a vector of all 1's, i.e. all feature values are "present". The formula simplifies to:

$$g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j$$

and the corresponding Shapley values are the ϕ_j's. It is using to this vector of Shapley values that we will try to remove redundant samples. A more in-depth explanation of the theoretical basis behind Shapley values and the SHAP model is available in [17][18].

Back to the explanation of the Shapley Error model

In order to understand in depth how this model works, we will now look at it step by step with an example. In Figure 2.3, you can see a schematic diagram of how

this model works. We will now go through these steps.

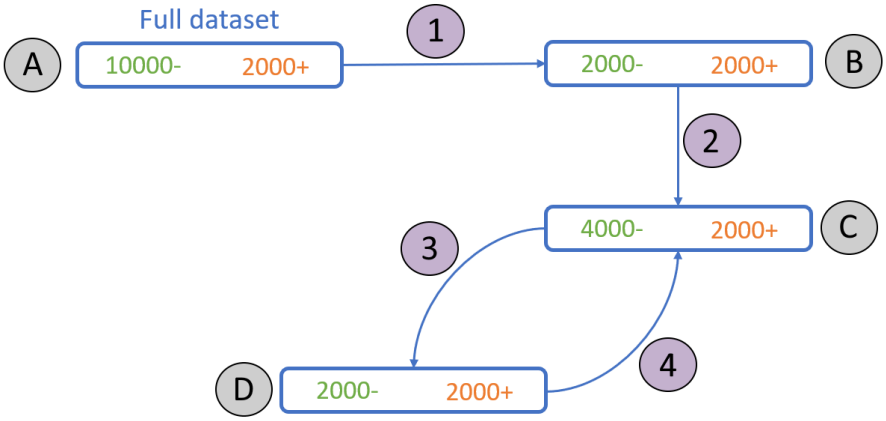


Figure 2.3: Illustration of how the Shapley Error model works

- **Steps A to C :** These steps are exactly the same as for the Error Model, see Section 2.6 for explainations.
- **Step 3 :** The Shapley values of all instances are calculated from the trained model using SHAP. These values are therefore in the form of a vector of the size of the set of features.

  These vectors are compared 2 by 2 in order to find those which are the closest in a geometrical sense (Euclidean distance). The most redundant samples are removed until the sample size is the same for the majority class as for the minority class.
- **Steps D & 4 :** These steps are identical to steps B & 2 explained above.

The loop consisting of steps C to 4 can be repeated several times. Indeed, it is possible that steps D and 4 reintroduce some redundant samples. The final performance of the model may vary depending on the number of times this loop is executed, so in the rest of this section we will analyze the effect of the parameter "number of loop executions" on the efficiency of the final model.

Hyper-parameter tuning

As explained earlier, the parameter on which this analysis will be performed is the number of times the end loop is executed. This parameter will be varied from 0 to 2.

One could ask why not also play on the same parameter as with RUS (see Section 2.1) or with Error Model (see Section 2.6). Indeed, since RUS and Error Model are an integral part of the Shapley Error Model, these parameters also play a role in the latter. However, we hypothesized that since these models are very similar, the parameters selected for RUS and Error Model, i.e. 1.0 and 15% repectively, are best suited for maximum efficiency in the Shapley Error Model.

In Figure A.7 in Appendix you can observe that the Nemenyi tests are inconclusive for all metrics. Indeed, all the values of the parameter are in a range smaller than a critical difference (CD).

In this work, we are primarily interested in effective prediction of the minority class, and this hyper-parameter does not seem to have an effect on this. Indeed, the most important metric, the balanced accuracy score (BAS), does not vary significantly when the studied parameter varies. Therefore, in order to select a value for this parameter, we will look at two things. First, the goal of this custom model is not to do exactly the same thing as the Error Model as we want to compare these models. We will then not select the value 0 for this parameter. Secondly, each new iteration of the loop adds a consequent time to the execution of this method. Indeed, the loop itself has a time complexity in $\mathcal{O}(n^2)$. Therefore, in view of these arguments, the parameter selected for this method is $n = 1$ iteration of the loop in order not to unnecessarily increase the computation time while keeping a model fundamentally different from the error model. The boxplots of these experiments are available in Figure B.7 in the Appendix.

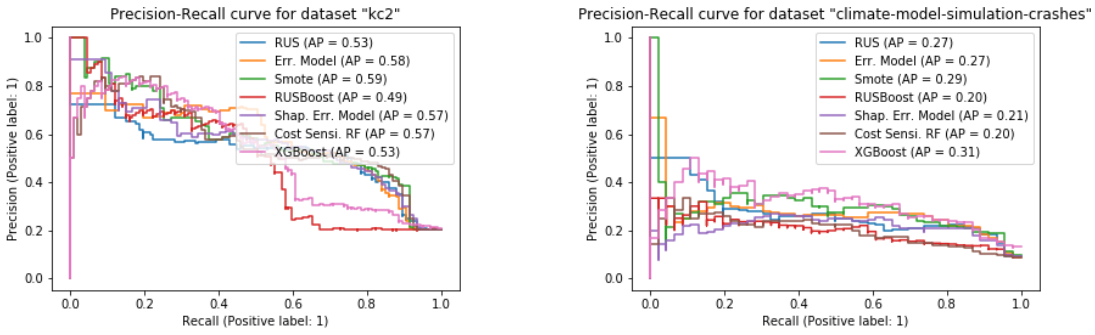
Chapter 3

Results

In this chapter, we will look at the results of the different models presented in Chapter 2. These results will be analyzed through the 5 metrics presented in Table 1.1 and through precision recall graphs.

3.1 Precision-Recall curves

Figures 3.1a to 3.1k below show the precision-recall graphs for the 11 datasets studied.



(a) (b)

Figure 3.1: Precision-Recall graphs for the 11 studied datasets.

First of all, it is interesting to see that the results of the implemented models are very dependent on the dataset. Indeed, the profile of the Precision-Recall curve is different for each dataset and takes very variable average precision (AP) values, of the order of 0.25 for the least efficiently predicted dataset while of the order of 0.85 for the most efficiently predicted dataset.

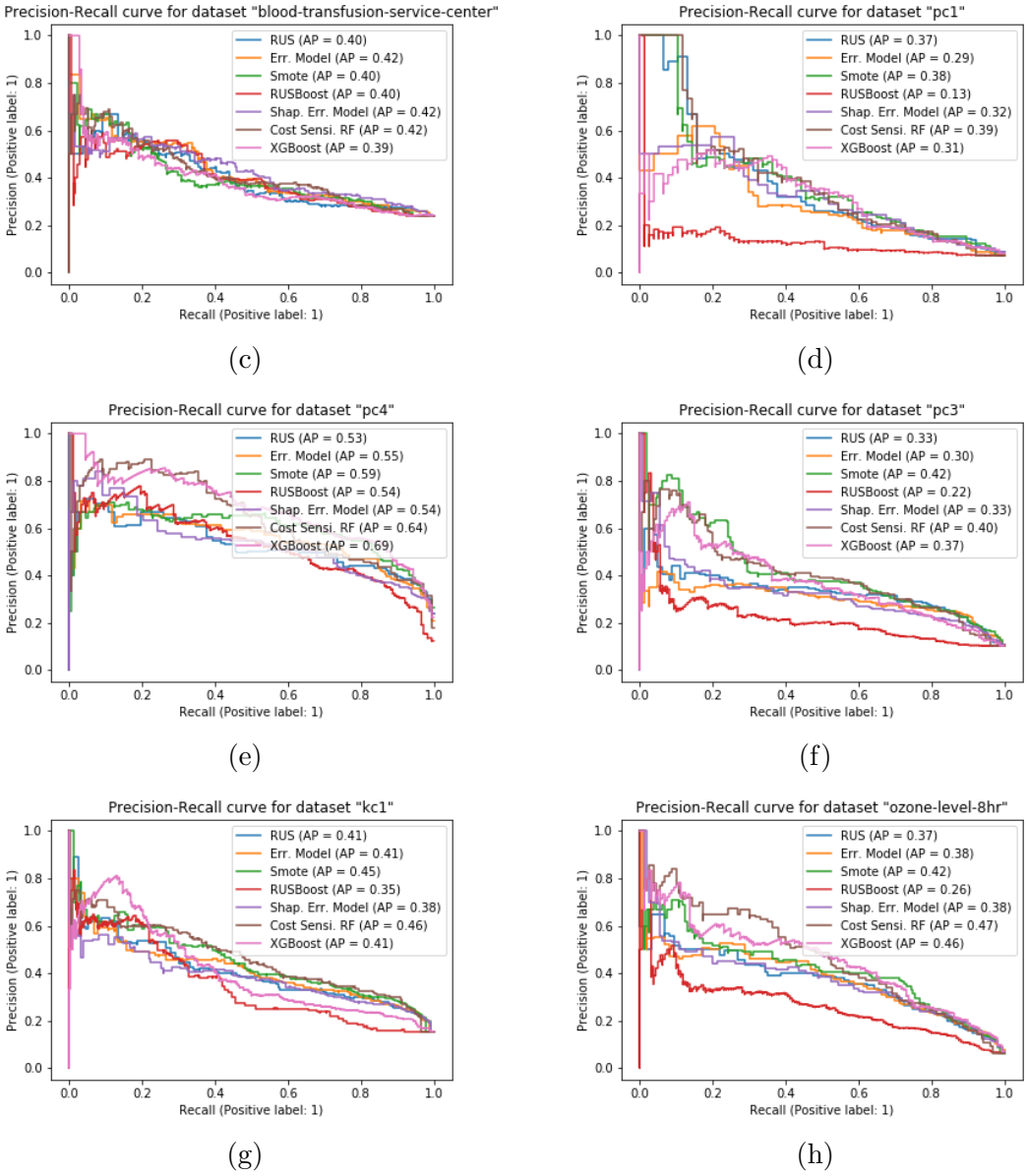


Figure 3.1: Precision-Recall graphs for the 11 studied datasets.

As for the models themselves, it is easy to notice that one model in particular stands out badly. Indeed, the curve corresponding to the RUS-Boost model is almost always well below the others. This result is quite unclear to understand considering that the RUS model gives significantly better results.

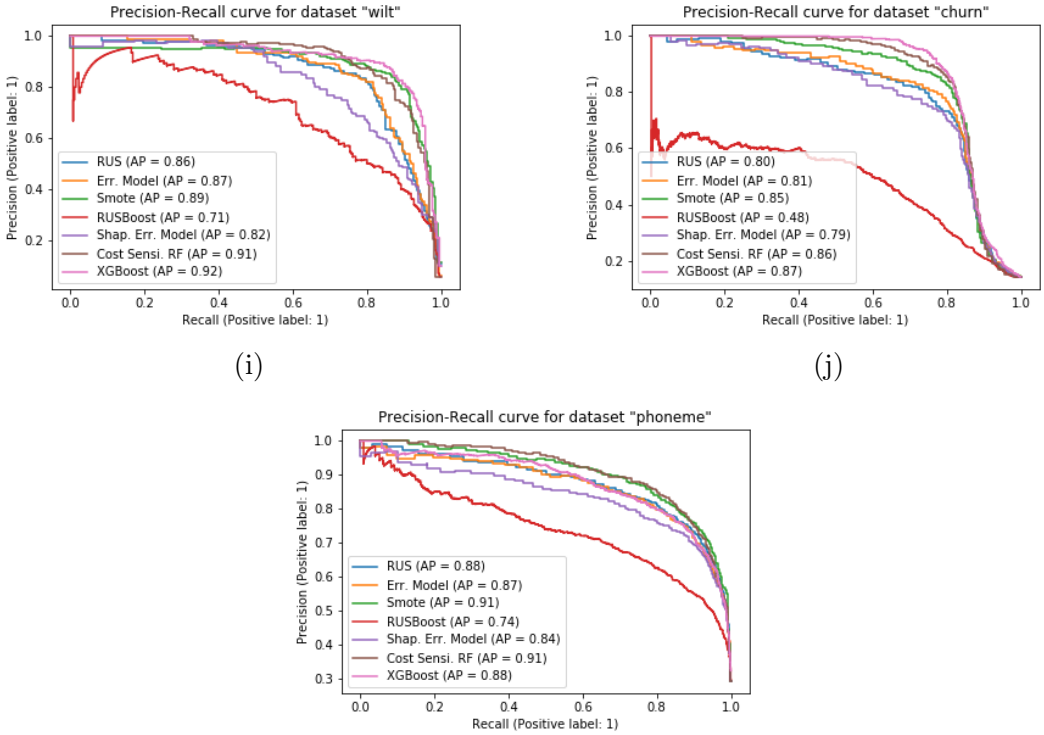


Figure 3.1: Precision-Recall graphs for the 11 studied datasets.

3.2 Statistical comparison between the different models

In the rest of this chapter, we will compare the different models with each other according to 5 metrics: average precision, ROCAUC score, F1-score, accuracy and balanced accuracy score. This comparison will be done using the Nemenyi statistical test to see if some models are statistically better than the others.

Average precision

On Figure 3.2, you can see the result of a Nemenyi test performed on the average precision of these 7 models.

We can conclude from this test that the RUS-Boost model is statistically less efficient than the XGBoost, SMOTE and Cost-Sensitive RF models. This result corroborates the observations made above about the low efficiency of the RUS-Boost model in terms of average precision. For the rest of the models, it is not

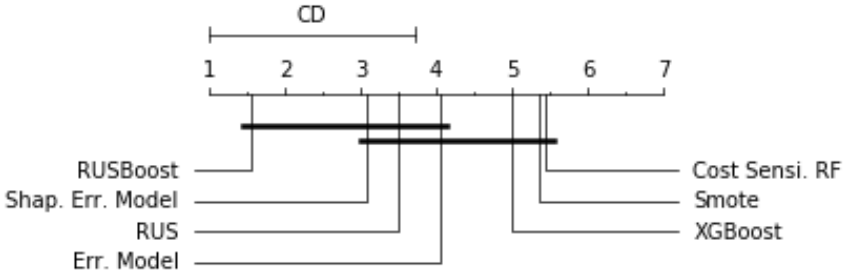


Figure 3.2: Nemenyi test on the average precision of the 7 models.

possible to draw any conclusions.

ROCAUC score

On Figure 3.3, you can see the result of a Nemenyi test performed on the ROCAUC score of the 7 models studied in this work.

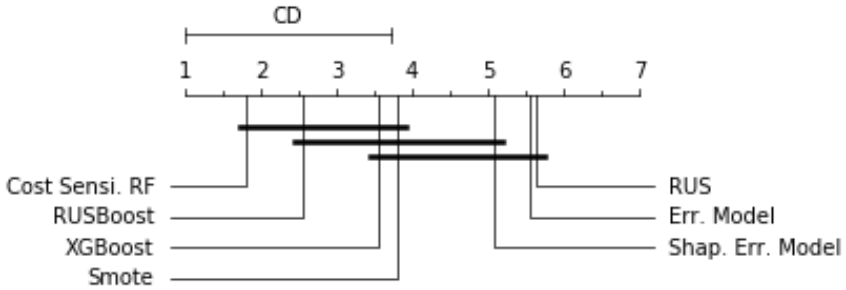


Figure 3.3: Nemenyi test on the ROCAUC score of the 7 models.

Several conclusions can be made about this test:

- The RUS and Error Model are statistically better than the Cost-Sensitive RF and RUS-Boost model.
- The Shapley Error Model is statistically better than the Cost-Sensitive RF.

As for the XGBoost and SMOTE models, it is not possible to differentiate them significantly from the other models according to this metric.

F1-score

On Figure 3.4, you can see the result of a Nemenyi test performed on the F1-score of the 7 models studied in this work.

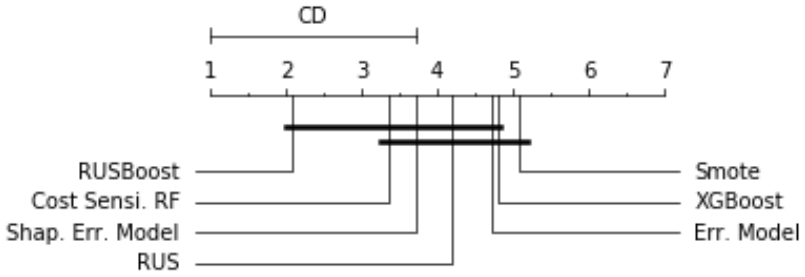


Figure 3.4: Nemenyi test on the F1-score of the 7 models.

Only one conclusion can be made from this test, and that is that the RUS-Boost model is statistically less effective than the SMOTE model in terms of the F1-score metric. As for the other models, it is not possible to differentiate them significantly.

Accuracy score

On Figure 3.5, you can see the result of a Nemenyi test performed on the accuracy of the 7 models studied in this work.

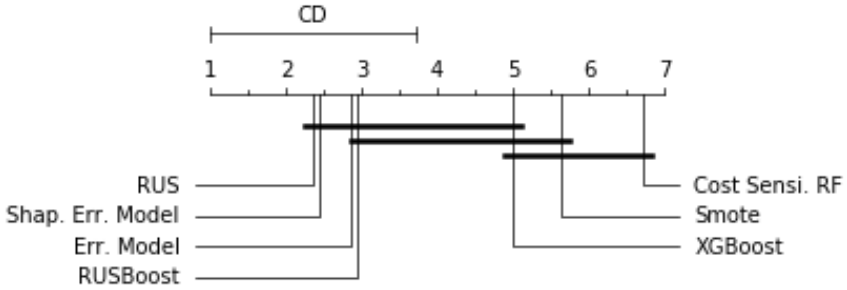


Figure 3.5: Nemenyi test on the accuracy of the 7 models.

Several conclusions concerning the accuracy of the different models can be made from this test:

- The Cost-Sensitive RF model is statistically better than the RUS, Error Model, Shapley Error Model and RUS-Boost models.

- The SMOTE model is statistically better than the RUS and Shapley Error Model.

As for the XGBoost model, it is not possible to differentiate it sufficiently from the other models concerning this metric.

Balanced accuracy score

On Figure 3.6, you can see the result of a Nemenyi test performed on the balanced accuracy score of the 7 models studied in this work.

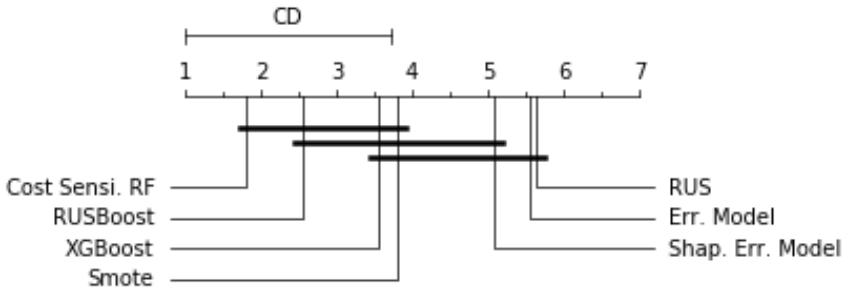


Figure 3.6: Nemenyi test on the balanced accuracy score of the 7 models.

Several conclusions concerning the balanced accuracy score of the different models can be made from this test:

- The RUS and Error Model models are statistically better than the RUS-Boost and Cost-Sensitive RF models.
- The SEM model is statistically better than the Cost-Sensitive RF model.

As for the XGBoost and SMOTE models, it is not possible to differentiate them sufficiently from the other models concerning this metric.

3.3 Lessons from these results

Some conclusions can be drawn from these results. First, we notice that some of these models favor an efficient prediction of the majority class while others favor the minority class. For example, the Cost-Sensitive RF and RUS-Boost models have an overall accuracy score in the highest of all the models considered but a balanced accuracy score in the lowest, a sign that these models primarily favor the majority class. Indeed, this metric is mainly dependent on the efficiency of

the model on the majority class.

Conversely, models such as RUS, Error Model or Shapley Error Model have an overall accuracy score in the lowest of all the models considered but a balanced accuracy score in the highest, which indicates that these models primarily favor the minority class. It is this type of model that interests us most in this work because we are primarily interested in predicting the minority class efficiently.

Finally, the XGBoost and SMOTE models appear to give average values for all metrics, a sign that these models are effective on both the minority and majority class, though without achieving as much effectiveness as other models on the minority class of most interest.

Concerning the two custom models derived from the RUS model, the Error Model and the Shapley Error Model, we notice that they are hardly differentiable from the RUS model whatever the metric used.

For the Error Model, this non-differentiation is the most marked. However, we observe that the accuracy, the average precision and the F1-score linked to this model remain slightly higher than that for RUS. This is due to the fact that we add samples from the majority class after the undersampling, inflating these metrics which are quite dependent on the majority class.

For the Shapley Error Model, this similarity is less marked than for the Error Model. Indeed, it seems that the selection based on the Shapley values of the samples added after the RUS does not bear its fruits and has for only effect to decrease the general efficiency of the model.

Moreover, the Shapley Error Model and SMOTE model share a big problem compared to some other models. Indeed, these two models as they are now are very expensive in terms of computation time, and are therefore difficult to use on large datasets such as those of banking transactions. The time complexity of the Shapley Error Model is $\mathcal{O}(n^2)$ while for smote, an oversampling of the minority class is performed until reaching the same number of samples as in the majority class, having the effect of significantly increasing the number of samples. However, the implementation has a lot to do with it and can be enhanced. For example, TreeSHAP is 100x faster than KernelSHAP. This is why no time comparison was performed in this work.

Discussion

The purpose of this discussion is to take a step back from the information presented earlier in this work and to take a critical look at the limitations of our models as well as some of the choices we have made during the implementation. This will allow the reader to have an overview of the main limitations and points of discussion regarding this work.

First of all, we notice through the results presented in Chapter 3 that the efficiency of the implemented models is very dependent on the dataset. Indeed, although they have similar characteristics, the 11 datasets on which we were able to work are fundamentally different and the models can therefore sometimes give quite different results on them. Therefore, it is impossible to predict the results of these models on a real dataset of banking transactions. Even if a small number of anonymized bank transaction datasets are available, we have chosen to restrict ourselves to datasets available on OpenML in order to have a large number of datasets at our disposal and remain generic.

Secondly, some of the implemented models are not realistically usable on large datasets. Indeed, the Shapley Error Model and SMOTE model are temporally very expensive and it is therefore not conceivable to deploy them on a real banking transaction dataset. On the one hand, the Shapley Error Model has a time complexity in $\mathcal{O}(n^2)$ while on the other hand the SMOTE model almost doubles the number of samples when generating samples of the minority class. Nevertheless, it was interesting to study the efficiency of these models in order to compare them with other known and less expensive models.

Finally, it could be interesting to further study the Error Model by changing the error function used by it to select the samples to be reintroduced in the sample. Indeed, this error function is arbitrary and we could imagine other ways to express it. However, this model remains above all a variant of RUS and one should not expect a very different efficiency from it. Even so, since the purpose of this model is to detect fraudulent bank transactions, any improvement in its efficiency, even minor, could result in significant gains in money.

Conclusion

Machine learning is an indispensable tool in the fight against fraud. Although already widely used by banks, some steps of the fraud detection process are not yet fully developed and require human intervention. One of the main difficulties for machine learning models is to deal with class imabalance. Through this work, we explored several models in order to find those that give the best results on an imbalanced dataset.

First, we looked at the characteristics of a bank transaction dataset in order to select datasets with similar characteristics to work on. Then, we defined the different models implemented in this work and performed an analysis of their optimal parameters. Finally, we compared these 7 models in order to draw conclusions on which of them are the most interesting for further research in this field.

Following this analysis, it is possible to conclude that Undersampling models are the most interesting models to use in the context of fraudulent transaction detection. Indeed, the Random Undersampling model, Error Model and Shapley Error Model allow the best prediction of the minority class among all the implemented models. Moreover, except for the custom Shapley Error Model, the computation time of these models is less than the others because a large proportion of the samples of the majority class are eliminated.

Of the two custom models Error Model and Shapley Error Model, only the Error Model is interesting to keep. Indeed, the Shapley Error Model is very expensive in terms of calculation and we did not succeed in proving that it brings a real improvement when compared to the Error Model. As far as the latter is concerned, it can be interesting to keep because it allows to increase the efficiency of the model on the majority class without however lowering too much the efficiency on the minority class.

As for the other models implemented, none of them obtains an equivalent efficiency on the minority class. Indeed, these models tend to predict the majority class better, to the detriment of the minority class that interests us most. More-

over, some models such as Synthetic Minority Oversampling Technique are very expensive to implement in terms of computing power.

Finally, it is important to specify that the results obtained in this work are very dataset dependent. Although the selected datasets have similar characteristics to real bank transaction datasets, it is impossible to say if the implemented models will have similar results on them.

Bibliography

[1] "Nilson report, issued november 2019," https://nilsonreport.com/upload/content_promo/The_Nilson_Report_Issue_1164.pdf, (accessed Apr 14, 2022).

[2] I. Sadgali, N. Sael, and F. Benabbou, "Detection of credit card fraud: state of art," *Journal of computer science and network security*, no. 18, pp. 76–83, 2018.

[3] "Credit card fraud detection system," https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_2_Background/FDS.html, (accessed May 20, 2022).

[4] M. Krivko, "A hybrid model for plastic card fraud detection systems," *Expert Systems with Applications*, vol. 37, no. 8, pp. 6070–6076, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2010.02.119

[5] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, "Effective detection of sophisticated online banking fraud on extremely imbalanced data," *World Wide Web*, vol. 16, no. 4, pp. 449–475, 2013.

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2016.12.035

[7] D. Devi, S. K. Biswas, and B. Purkayastha, "A Review on Solution to Class Imbalance Problem: Undersampling Approaches," *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, pp. 626–631, 2020.

[8] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 4, pp. 687–719, 2009.

[9] Y.-A. Le Borgne, W. Siblini, B. Lebichot, and G. Bontempi, *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.*

Université Libre de Bruxelles, 2022. [Online]. Available: https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook

[10] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, 2016.

[11] J. Brownlee, "A gentle introduction to imbalanced classification," https://machinelearningmastery.com/what-is-imbalanced-classification/#:~:text=Imbalanced%20classification%20is%20the%20problem,and%20may%20require%20specialized%20techniques., (accessed May 4, 2022).

[12] "Recency, frequency, monetary model with python — and how sephora uses it to optimize their google and facebook ads," https://towardsdatascience.com/recency-frequency-monetary-model-with-python-and-how-sephora-uses-it-to-optimize-their-go (accessed May 29, 2022).

[13] "Openml," https://www.openml.org/, (accessed Oct 15, 2021).

[14] J. Demsar, ""Statistical Comparisons of Classifiers over Multiple Data Sets"," *Journal of Machine Learning Research 7 (2006) 1–30*, 2006.

[15] "Xgboost documentation," https://xgboost.readthedocs.io/en/stable/, (accessed Mar 21, 2022).

[16] S. M. Lundberg and S.-I. Lee, ""A unified approach to interpreting model predictions."," *Advances in Neural Information Processing Systems*, 2017.

[17] "Shapley values," https://christophm.github.io/interpretable-ml-book/shapley.html, (accessed Mar 20, 2022).

[18] "Shap model," https://christophm.github.io/interpretable-ml-book/shap.html, (accessed Mar 20, 2022).

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

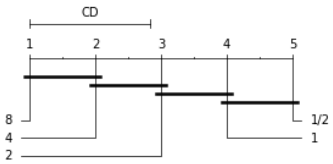
[20] S. Mishra, ""Handling Imbalanced Data: SMOTE vs. Random Undersampling"," *International Research Journal of Engineering and Technology (IRJET), vol. 04, no. 8*, 2017.

Appendices

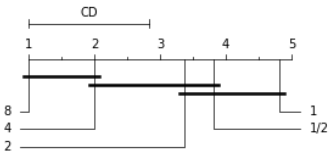
Appendix A

Nemenyi tests for hyper-parameter selection

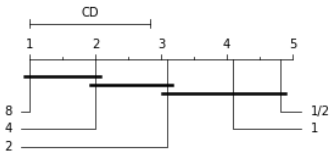
RUS



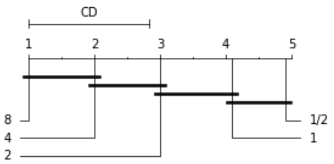
(a) Accuracy score



(b) Balanced accuracy score



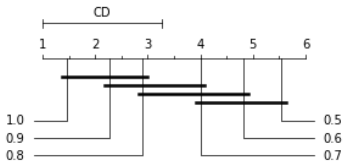
(c) F1-score



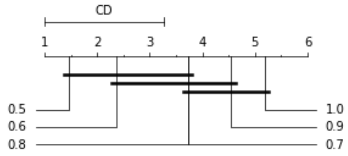
(d) ROCAUC score

Figure A.1: Nemenyi tests of the model regarding 4 metrics for RUS.

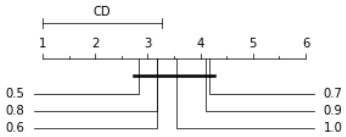
SMOTE



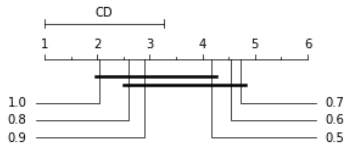
(a) Accuracy score



(b) Balanced accuracy score



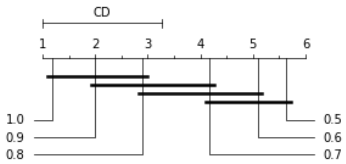
(c) F1-score



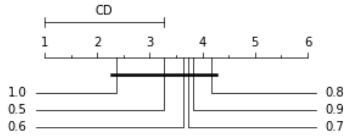
(d) ROCAUC score

Figure A.2: Nemenyi tests of the model regarding 4 metrics for SMOTE.

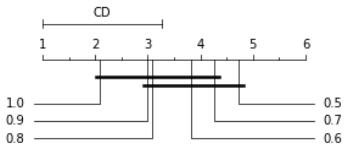
RUS-Boost



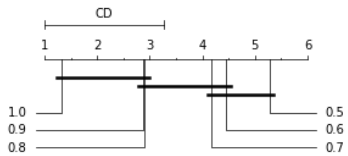
(a) Accuracy score



(b) Balanced accuracy score



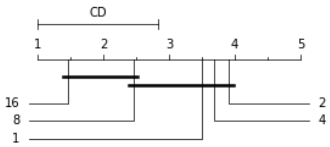
(c) F1-score



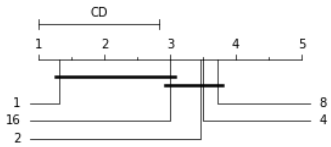
(d) ROCAUC score

Figure A.3: Nemenyi tests of the model regarding 4 metrics for RUS-Boost.

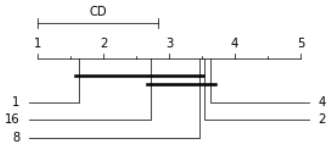
Cost-Sensitive Random Forest



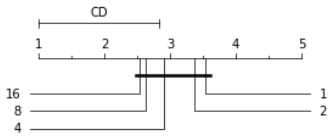
(a) Accuracy score



(b) Balanced accuracy score



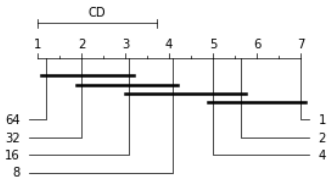
(c) F1-score



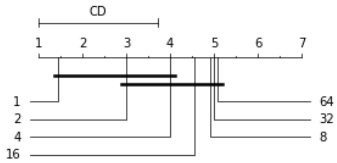
(d) ROCAUC score

Figure A.4: Nemenyi tests of the model regarding 4 metrics for Cost-Sensitive RF.

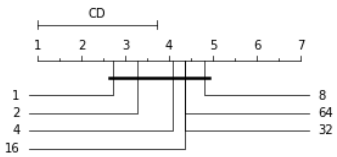
XGBoost



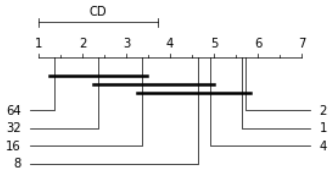
(a) Accuracy score



(b) Balanced accuracy score



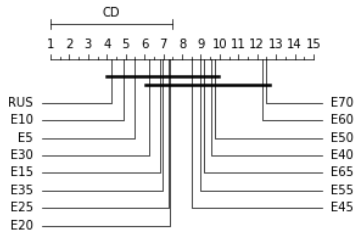
(c) F1-score



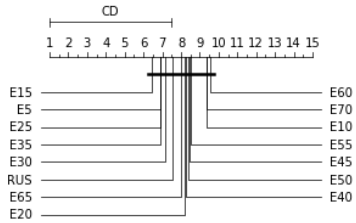
(d) ROCAUC score

Figure A.5: Nemenyi tests of the model regarding 4 metrics for XGBoost.

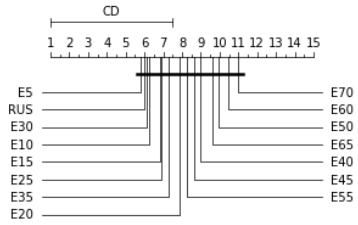
Error Model



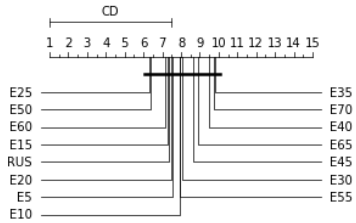
(a) Accuracy score



(b) Balanced accuracy score



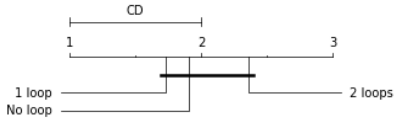
(c) F1-score



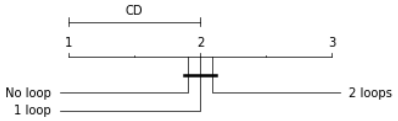
(d) ROCAUC score

Figure A.6: Nemenyi tests of the model regarding 4 metrics for Error Model.

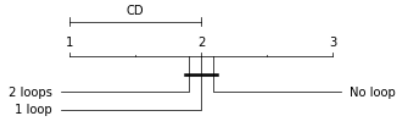
Shapley Error Model



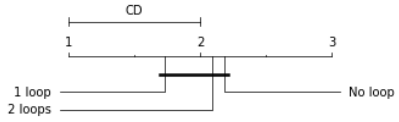
(a) Accuracy score



(b) Balanced accuracy score



(c) F1-score



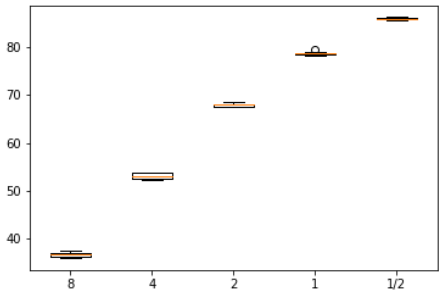
(d) ROCAUC score

Figure A.7: Nemenyi tests of the model regarding 4 metrics for Shapley Error Model.

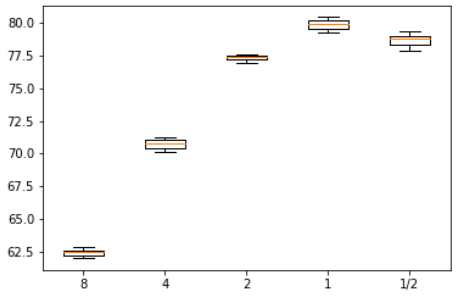
Appendix B

Boxplots for hyper-parameter selection

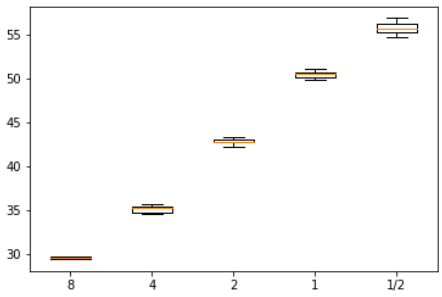
RUS



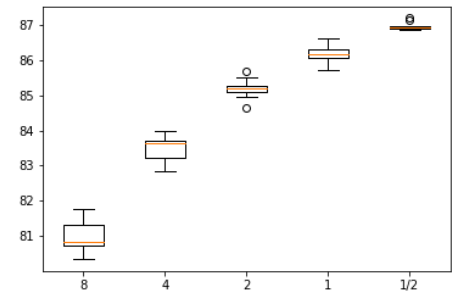
(a) Accuracy score



(b) Balanced accuracy score



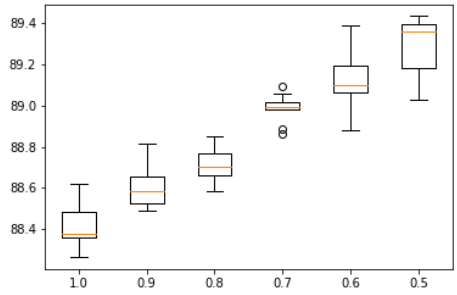
(c) F1-score



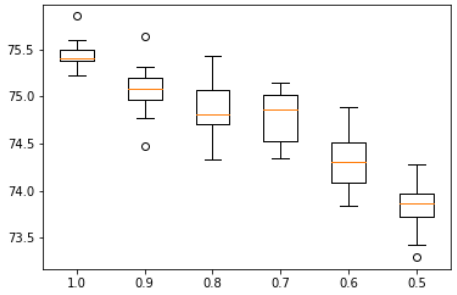
(d) ROCAUC score

Figure B.1: Four metrics as a function of the ratio of the number of samples in the minority class over the number of samples in the majority class after undersampling.

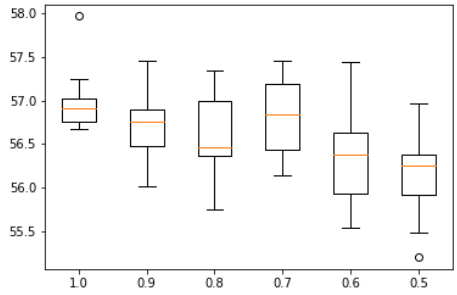
SMOTE



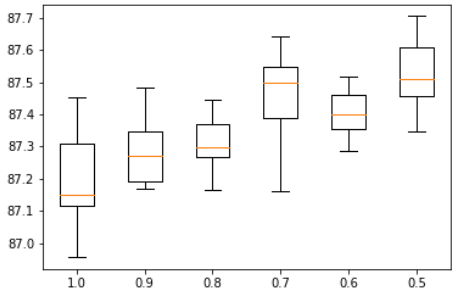
(a) Accuracy score



(b) Balanced accuracy score



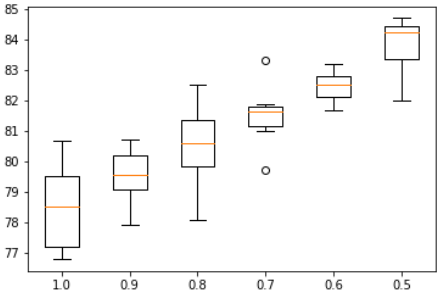
(c) F1-score



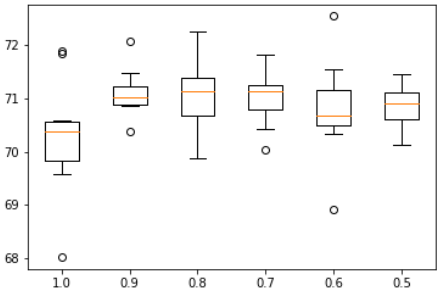
(d) ROCAUC score

Figure B.2: Four metrics as a function of the ratio of the number of samples in the minority class over the number of samples in the majority class after resampling.

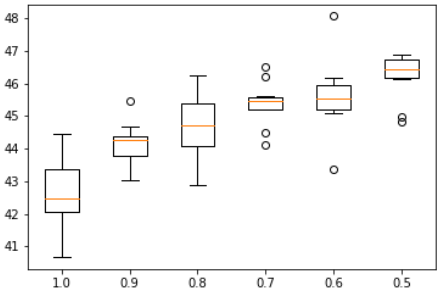
RUS-Boost



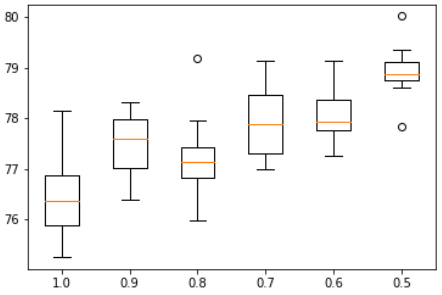
(a) Accuracy score



(b) Balanced accuracy score



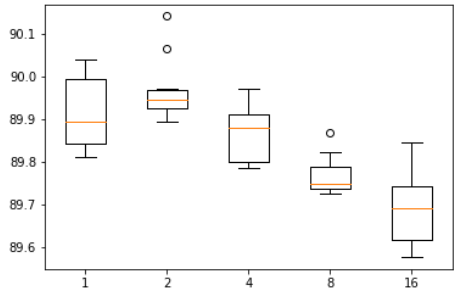
(c) F1-score



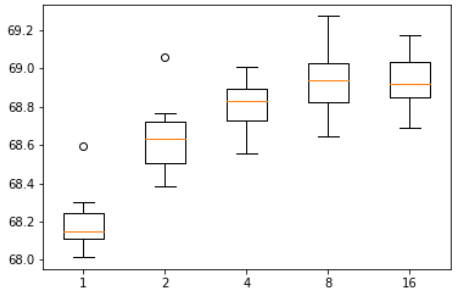
(d) ROCAUC score

Figure B.3: Four metrics as a function of the ratio of the number of samples in the minority class over the number of samples in the majority class after resampling.

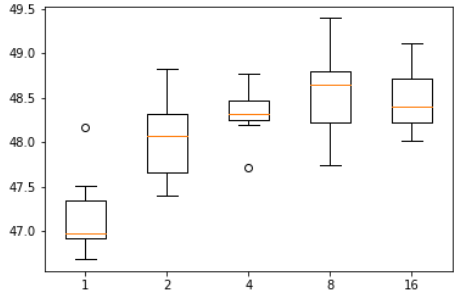
Cost-Sensitive Random Forest



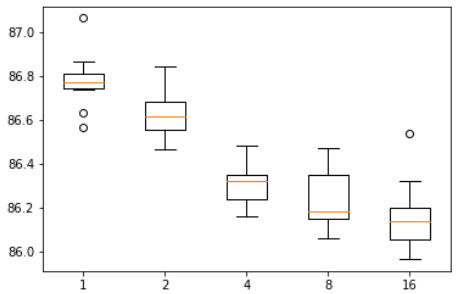
(a) Accuracy score



(b) Balanced accuracy score

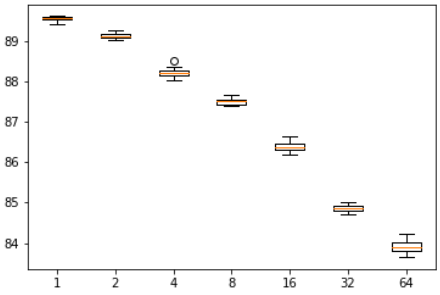


(c) F1-score

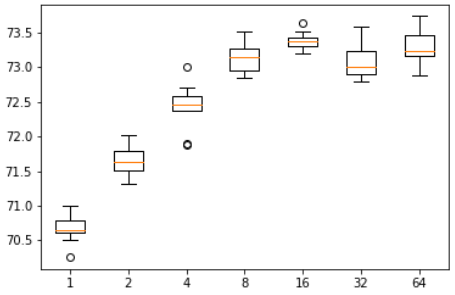


(d) ROCAUC score

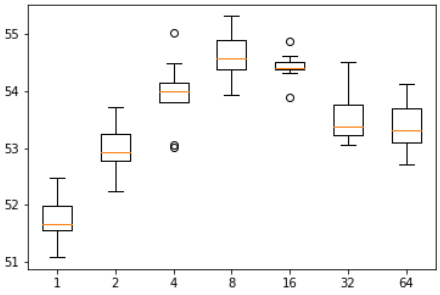
Figure B.4: Four metrics as a function of the weight associated to the minority class.



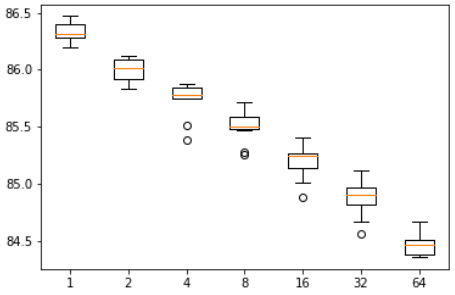
(a) Accuracy score



(b) Balanced accuracy score



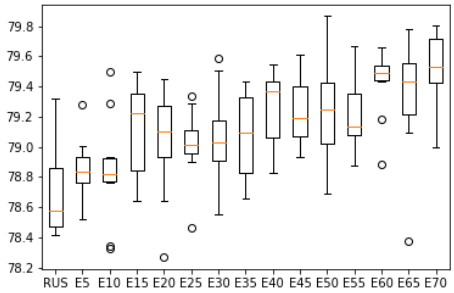
(c) F1-score



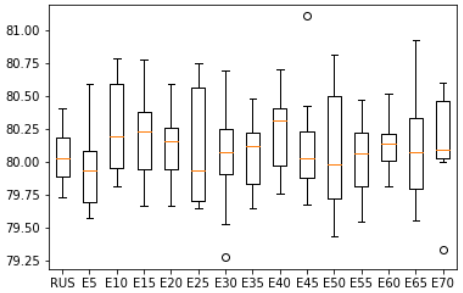
(d) ROCAUC score

Figure B.5: Four metrics as a function of the ratio of the weights associated to each class.

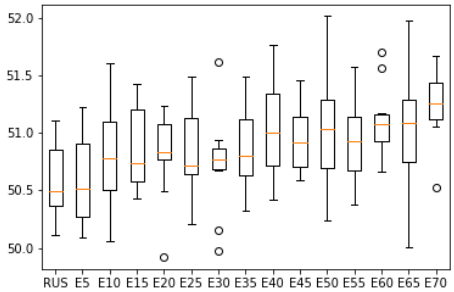
Error Model



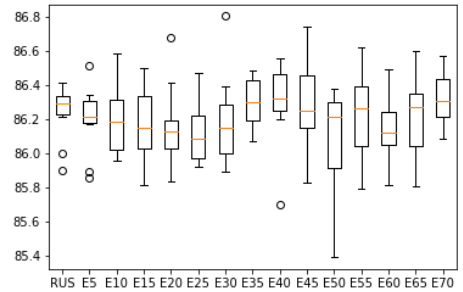
(a) Accuracy score



(b) Balanced accuracy score



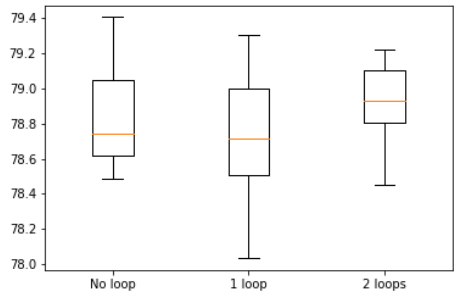
(c) F1-score



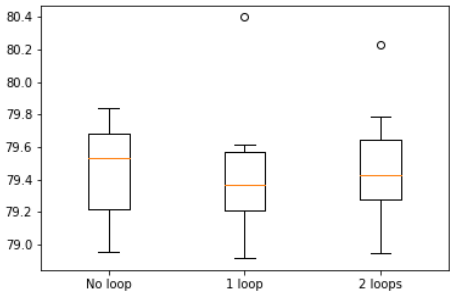
(d) ROCAUC score

Figure B.6: Four metrics as a function of the fraction of the samples removed by RUS that are restored in the sample.

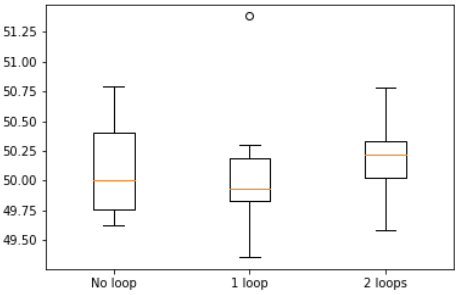
Shapley Error Model



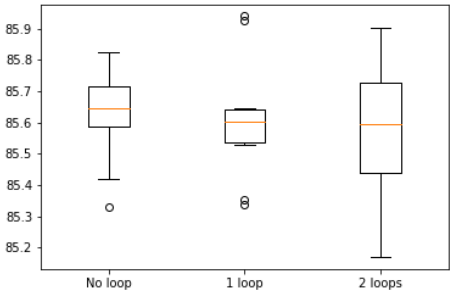
(a) Accuracy score



(b) Balanced accuracy score



(c) F1-score



(d) ROCAUC score

Figure B.7: Four metrics as a function of the number of times we remove redondant samples using their shapley values.

Abstract :

Payment card fraud is a significant problem for business owners, card issuers, and transactional services firms, resulting in significant financial losses each year. With the growing volume of data created by payment card transactions, it has become impossible for a human analyst to identify fraudulent patterns in transaction datasets, which are frequently characterized by a huge number of samples and multiple dimensions. As a result, during the last decade, the development of payment card fraud detection systems has shifted to approaches based on machine learning techniques, which automate the process of detecting fraudulent patterns from vast amounts of data.

However, not all steps can be automated. Some conclusions cannot yet be made efficiently by machine learning. More specifically, ML algorithms have a lot of difficulty dealing with class imbalance. Indeed, transaction data contains far more genuine transactions than fraudulent ones: in a real-world dataset, the percentage of fraudulent transactions is often considerably under 1%. Learning from imbalanced data is difficult since most learning algorithms struggle with big class disparities. Class imbalance necessitates the adoption of additional learning procedures such as sampling or weighting.

In this work, we will try to improve the efficiency of existing machine learning models as well as to create new ones with the goal of pushing for the total automation of this process, removing the need for human operators. Moreover, this study aims to try to draw conclusions on the avenues to be explored in future research concerning the detection of fraud in transactions.

Résumé :

La fraude à la carte de paiement est un problème important pour les propriétaires d'entreprises, les émetteurs de cartes et les sociétés de services transactionnels, entraînant des pertes financières considérables chaque année. Avec le volume croissant de données créées par les transactions, souvent caractérisées par un nombre énorme d'échantillons et de multiples dimensions, il est devenu impossible pour un analyste humain d'identifier efficacement les fraudes. Par conséquent, au cours de la dernière décennie, le développement des systèmes de détection de la fraude s'est orienté vers des approches basées sur des techniques d'apprentissage automatique, qui automatisent le processus de détection des schémas frauduleux à partir de grandes quantités de données.

Cependant, toutes les étapes ne peuvent pas être automatisées. Certaines conclusions ne peuvent pas encore être tirées efficacement par l'apprentissage automatique. Plus précisément, les algorithmes d'apprentissage automatique ont beaucoup de mal à gérer le déséquilibre des classes. En effet, les données de transaction contiennent beaucoup plus de transactions authentiques que de transactions frauduleuses : dans un ensemble de données du monde réel, le pourcentage de transactions frauduleuses est souvent nettement inférieur à 1%. L'apprentissage à partir de données déséquilibrées est difficile, car la plupart des algorithmes d'apprentissage ont du mal à gérer de grandes disparités entre les classes. Le déséquilibre des classes nécessite l'adoption de procédures d'apprentissage supplémentaires telles que l'échantillonnage ou la pondération.

Dans ce travail, nous tenterons d'améliorer l'efficacité des modèles d'apprentissage automatique existants ainsi que d'en créer de nouveaux dans le but de pousser à l'automatisation totale de ce processus, en supprimant le besoin d'opérateurs humains. De plus, cette étude vise à essayer de tirer des conclusions sur les pistes à explorer dans les recherches futures concernant la détection de la fraude dans les transactions.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management

Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique

www.uclouvain.be/lsm