Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

BACHELOR DEGREE THESIS

TITLE: Machine Learning to detect and prevent unpaid transactions

DEGREE: Grau en Enginyeria de Sistemes de Telecomunicació

AUTHOR: Judith Trujillo Martín

DIRECTOR: Cristina Barrado Maxí

DATE: February 7th 2019

Title: Machine Learning to detect and prevent unpaid transactions.

Author: Judith Trujillo Martín

Director: Cristina Barrado Maxí

Date: February 7th 2019

Overview

Financial entities are increasingly looking for new technologies in order to improve their services and increase their profits. One of the challenges raised by a well-known financial entity is to reduce doubtful credit from the customers' transactions (face-to-face purchases). When transactions are unpaid by customers, the financial entity has to afford it and the corresponding amount of money is added to the doubtful credit. These transactions are accepted due to incorrect validation during the authorization. When the financial entity has connectivity problems and it is not able to validate a transaction, it allows a payment processor entity to validate the transaction. The payment processor entity does not have the legal permission to check if the customer has enough credit to assume the transaction. Therefore, the payment processor entity sometimes validates transactions that the customer will not pay. This project proposes to implement an algorithm able to predict which transactions will be paid by the customer and which not. This algorithm has to be executed by the payment processor entity. This project proposes to implement an algorithm able to predict which transactions will be paid by the customer and which not. Predictions are made without information about the available credit that the customer has. The algorithm has to be executed by the payment processor entity. After testing several models, it is concluded that the Random Forest algorithm trained with balancing techniques can predict 57.6% of all the transactions that will not be paid by customers. All with a certainty of 97.3%.

Títol: Machine Learning aplicat a la detecció i la prevenció de les transaccions moroses.

Autor: Judith Trujillo Martín

Director: Cristina Barrado Maxí

Data: 7 de febrer de 2019

Resum

Les entitats financeres busquen, cada cop més, noves tecnologies que els hi permetin millorar els seus serveis i augmentar els seus beneficis. Un dels reptes que es planteja una coneguda entitat financera és reduir el crèdit dubtós provinent de les transaccions (compres presencials) que realitzen els seus clients. Les transaccions acceptades que els clients acaben no pagant són responsabilitat de l'entitat financera i s'afegeixen al crèdit morós que aquesta ha d'assumir. Les transaccions són acceptades degut a una validació errònia en el moment de l'autorització. Quan l'entitat financera no és capaç de validar una transacció per problemes de connexió, dona permís a una entitat processadora de pagaments per a que validi la transacció. L'entitat processadora de pagaments no té capacitat legal per consultar si el client disposa de crèdit o no per assumir l'operació que vol fer. Per tant, pot ser que validi transaccions que el client no pagarà. Aquest projecte proposa implementar un algoritme que s'executi des de l'entitat processadora de pagaments i sigui capaç de predir quines transaccions seran pagades pel client i quines no. Tot això, sense la necessitat de consultar el crèdit disponible del client. Després de proves amb diversos models, es conclou que l'algoritme *Random Forest* entrenat amb tècniques de balanceig pot arribar a predir el 57.6% de les transaccions que no seran pagades pels clients. Tot amb una certesa del 97.3%.

Contents

List of Figures

List of Tables

CHAPTER 1 INTRODUCTION

Unpaid credit in Spain supposed 6.3% of the total credit ceded by financial entities, reaching 75.762 million euros in August 2018, according to the *Banco de España* [1]. Although in the last year this amount has decreased, it is not enough for financial entities, which want to continue decreasing that percentage. The effects of unpaid credit are negative for financial entities and also for the country global economy. Credit is considered unpaid when it is not returned to the corresponding financial entity within the established period. Unpaid credit includes mortgages, loans, insurance and transactions (purchases) made by customers. If a customer cannot assume a transaction, the transaction has to be denied during its validation process. Sometimes it is not possible to know if a customer can assume a transaction. In order to understand why this happens, the validate process is explained below.

1.1. Payment validation circuits

When someone pays with a credit card, a validation circuit starts up. The validation circuit has to accept transactions which can be afforded by the user who made it. In addition, these transactions have to comply with the legal regulations from the *Banco de España*, in the case of Spain.

Face-to-face payments are very similar to Internet payments. However, this project is focused on face-to-face payments. Below, all the information related to a validation circuit of a face-to-face payment is presented.

The validation circuit involves five agents:

- Credit Card: The credit card is the physical or digital support that every user has in order to pay any service or product.

- Point of Sale Terminal: The Point of Sale Terminal, i.e. POS is the hardware that every trade has to establish communication with credit cards and with the financial entities.

- Broadcasting Entity: The Broadcasting Entity is a financial entity which gives services to users. It stores the user's money and transfers it on every payment he/she does.

- Receiving Entity: The Receiving Entity is a financial entity which gives services to trades. It stores the trade’s money and receives the incomes payments.

- Payment Processor Entity: The Payment Processor Entity is an intermediary between the Broadcasting Entity and the Receiving Entity. The Payment Processor Entity has two main purposes: To give support to both entities and to validate transactions when the Broadcasting Entity is not able to do it.

Every agent, except for the credit card, is able to accept or to deny transactions. In order to do that, agents have internal rules. These rules are requisites that every transaction has to comply with in order to be accepted. Depending on the validation circuit, rules are applied by one agent or by another.
In order to understand how transactions are validated, *Fig 1.1* illustrates every step of the validation circuit. The green tick represents that the agent applies rules.

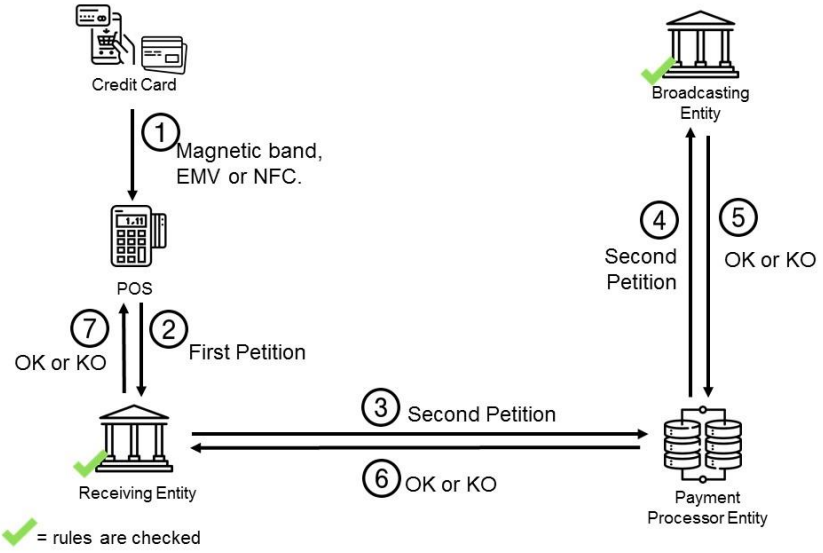

Fig 1.1 The complete validation circuit of a payment.

1) A Credit Card establishes a connection with the POS terminal. Different technologies can be used in order to connect the credit card and the POS terminal. For example, using the magnetic band, the EMV chip or the NFC chip.
2) The POS terminal generates a petition to the Receiving Entity.
3) The Receiving Entity checks its rules and validates or not the first petition. If the Receiving Entity validates the first petition, it generates a second petition which goes to the Payment Processor Entity. If not, a KO is returned to the POS terminal (7).

4) The Payment Processor Entity sends the second petition to the Broadcasting Entity.
5) The Broadcasting Entity validates the second petition using its rules. If the transaction is accepted, a positive answer goes back to the POS terminal (5), (6) and (7). If not, a negative answer goes back and the transaction is denied.

As green ticks in *Fig 1.1* shows, rules are applied by the Receiving Entity and by the Broadcasting Entity. The Receiving Entity verifies that the trade is able to receive the payment. The Broadcasting Entity verifies that the user has enough credit to afford the payment. Both entities verify that the transaction is legal.
A transaction can be denied by two main reasons.

- The transaction does not comply the rules at the Broadcasting Entity, for example, if the user's bank balance has not enough credit to pay the purchase, or if the Credit Card is expired.
- The connection between any of the five agents is not possible due to external reasons. Taking into account all the transactions done during a day, around the 9% suffer problems in the connection with the Receiving Entity or with the Broadcasting Entity.

In order to avoid recurrent denial, the Broadcasting Entity creates two alternatives circuits. One that allows offline validations to be done and another in which transactions are validated by the Payment Processor Entity.

1.1.1. Offline validation circuit

Some specific points of sale are located in places where the connectivity is poor or null. Therefore, it is not possible to establish a good connection between the POS terminal and the Receiving Entity. In this case, simple rules are included at the POS terminal in order to validate transactions. A green tick in *Fig 1.2* illustrated the POS terminal rules. This alternative is used only in specific situations, for example, during a flight. The real movement of the money is done by a batch process executed during the night. *Fig 1.2* illustrates the offline validation circuit.

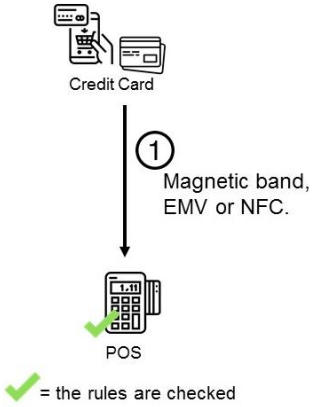

Fig 1.2 The offline validation circuit of a payment.

1.1.2. Supported validation circuit

The Payment Processor Entity is able to validate transactions in case the second petition cannot arrive at the Broadcasting Entity. To validate transactions, the Payment Processor Entity and the Broadcasting Entity has agreements to establish which rules the Payment Processor Entity will apply. Every transaction has to comply with the rules in order to be accepted. The accepted transactions which are validated by the Payment Processor Entity are called supported transactions. *Fig 1.3* represents the validation circuit supported by the Payment Processor Entity.

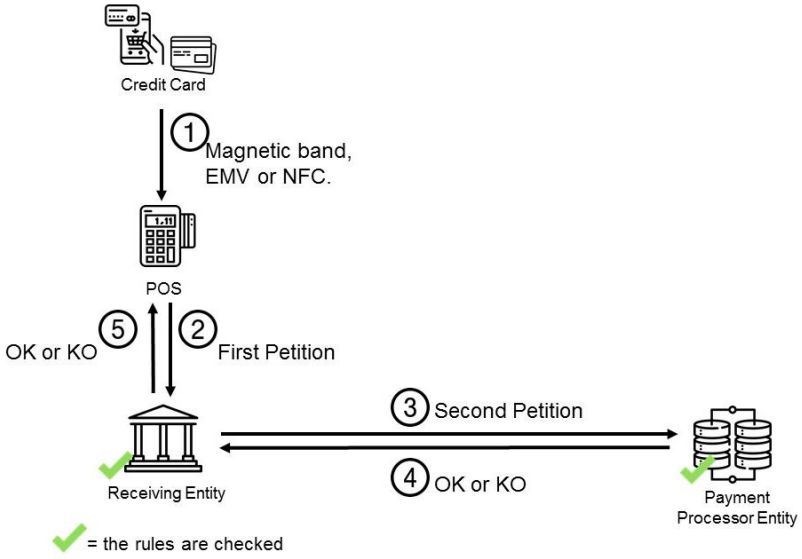

Fig 1.3 The supported validation circuit of a payment.

POS terminal rules and Payment Processor Entity rules are not able to check if the user has enough credit to afford the supported transaction. Information about the user's bank balance is only available at the Broadcasting Entity. Therefore, a supported transaction from a user with zero bank balance can be accepted. If the user cannot afford the supported transaction, the Broadcasting Entity has to pay it. The unpaid credit of the Broadcasting Entity increases and with it all related negative aspects.

A prestigious Broadcasting Entity wants a solution to decrease the number of unpaid supported transactions. In order to front facing the problem, this Broadcasting Entity extracts every day all the supported transactions into two types of dataset.

- Supported transactions dataset: Includes all the transactions supported by the Payment Processor Entity done during a day.

- Unpaid supported transactions dataset: Includes only the unpaid transactions that are supported by the Payment Processor Entity.

1.2. Problem formulation

The project is focused on reducing the number of unpaid supported transactions. To achieve it, an algorithm will be applied as an extra rule on the Payment Processor Entity. The algorithm has to be able to predict which supported transactions will not be paid by the user. Therefore, the algorithm has to act as a binary classifier. An important requirement that the algorithm must comply with is being able to predict without checking if the user has credit on his bank balanced. Finally, the algorithm has to deal with the unbalanced situation between paid and unpaid supported transactions. There is a small amount of unpaid supported transactions compared to all the supported transactions. Finding them is not an easy task.

Algorithm decisions determine whether the transaction is accepted or not. A supported transaction predicted as unpaid will be denied. A supported transaction predicted as paid will be accepted.

1.3. Project objectives

The main objective of this project is to search for a machine learning algorithm able to predict unpaid supported transaction from previous experiences. In order to achieve it, other specific objectives are defined:

- Study the available data and collect more information than the one obtained by first sight.
- Solve the imbalanced problem and fit the data with the algorithm by using different balancing techniques.
- Apply several classifier models.
- Evaluate the predictions with different metrics in order to find the best model.

More personal objectives are:

- Solve the difficulties and limitations that appear when a project is made in a big Broadcasting Entity.
- Learn about concepts that have not been studied during the bachelor degree, like machine learning and the payment validation circuit.
- Learn how to make a good project and write a good document in English.

1.4. Project outline

In order to get a better understand of the project, a summary of each chapter is explained bellow:

Chapter 2. Machine Learning

The first part of the chapter presents a general introduction to the meaning of machine learning and all the necessary concepts to implement a machine learning model. From how to split the input data until the performance measures to evaluate the results. Models, their algorithms, and the most common problems are also explained.

Chapter 3. Set up datasets and algorithms

This chapter presents how methods in chapter 2 are applied to datasets and the implementation of the algorithms.

Chapter 4. Results

Predictions from the applied models are evaluated with different metrics. Comments and comparisons are also added to reach a conclusion.

Chapter 5. Conclusions

In this last chapter, the best solution found is exposed. It is checked if all the objectives have been achieved and what could be the next steps.

CHAPTER 2 MACHINE LEARNING

2.1. Introduction to machine learning

Machine learning is the idea associated with all those methods that allow machines to improve their performance in a problem without being explicitly programmed. To achieve it, methods are based on mathematical statistics are used. They have two purposes: first, learn by training with previous examples and then solve the problem for new information. Machine learning is based on applying to machines the reasoning that humans use. This idea is called artificial intelligence.

Before the 1950s statistical methods were used. In the 1950s, machine learning algorithms began to be created until the 1970s, when the interest in the artificial intelligence decrease and also its investment. In the 1980s and 1990s, machine learning is once again popular due to the possibility of creating large amounts of data and thanks to some discoveries such as backpropagation. From then until now, new algorithms have been appearing. They have maintained interest and hopes in this field [2]. Nowadays, machine learning if very popular in many areas like computer vision, medical diagnoses, fraud detection, the use of credit cards, anti-spam, etc.

2.1.1. Types of learnings

Machine learning models and their algorithms are classified into three different types according to the learning methodology they use [3].

a) Supervised Learning

Supervised learning models learn from input and output data. Or in other words, from examples that are already solved. Some supervised models are decision trees, support vector machines, neural networks, etc.

b) Unsupervised Learning

Unsupervised learning models learn from input data. Therefore, the provided examples are not solved. Some unsupervised models are clustering, dimensionality reduction, recommender systems, deep learning, etc.

c) Reinforcement Learning

Reinforcement learning models learn by feedback from input data. A reward if they solve the problem correctly or a penalization if they do not. They use the try and failure technique, trying many times until they find the best solution. This type

of learning is used when the correct solution is known, but not the way to solve the problem.

2.1.2. Predictive modelling

Machine learning models are not able to solve all kind of problems. Most are based on predictive modelling, descriptive modelling or on association rules.

Models which act as a predictive modelling rely on predicting new outputs. If the output is categorical, they are a classifier. If the output is numerical, they apply a regression. Those which acts as a descriptive modelling are able to assign inputs into clusters, in order to find similarities from inputs. Finally, an association rules model provides new information, presented as rules, for finding specific relations between inputs.

All these models need a considerable amount of inputs. These inputs are called instances and are grouped in datasets. Every instance has its information distributed in different variables. These variables are called features. On predictive modelling, the predicted output is called the target feature.

2.2. Datasets and data partition

Predictive models need some data for training and other data to validate how they predict for instances. The most used method is to split the provided dataset into two parts: a training set and a validate set, in order to perform the two actions. Normally, a larger part is allocated for training and a smaller part for validation. 80% and 20% respectively is the most common ratio. However, some methods such as cross-validation are used to change the ratio between the two subsets. It is proved that these techniques improve the algorithm predictions, making them more accurate [4]. Once the training and the validation are done, the final verification has to be done. In order to do that, a different dataset, with new instances is used, it is the test set. *Fig 2.1* shows an example of the three described datasets involved in the procedure.

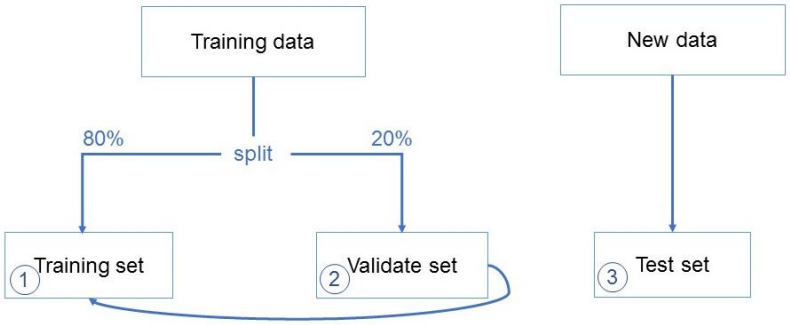

Fig 2.1 Scheme example of the datasets used on predictive models.

2.3. Data Exploration

The first step in every big data problem must be to know as much as possible about the information on the provided dataset. For instance, it is important to know the meaning of every variable and the statistical relationship between them. A good option could be to make an exploratory analysis, in order to obtain extra information that it is not possible to know at first sight.

2.3.1. Correlation Matrix

One of the most useful method is the correlation matrix. Each value of this matrix represents the *Pearson Correlation Coefficient* between two variables. This coefficient measures the linear relationship between them. It is a value from -1 to 1. Where 1 means a perfect positive correlation, -1 a perfect negative correlation and 0 means no correlation at all.

If a dataset has p variables, the size of its correlation matrix will be p x p. The steps to obtain the matrix are the following ones:

Each variable is introduced on two vectors with p [5] values each:

$$X_j = [X_1\ X_2\ \cdots\ X_j\ ] \quad X_k = [X_1\ X_2\ \cdots\ X_k\ ] \quad p = j = k \tag{2.1}$$

In order to calculate the correlation between two values, it is necessary to calculate the covariance S_{jk} and typical deviations from marginal distributions $S_j\ S_k$ [6].

$$corr(X_j, X_k) = \frac{S_{jk}}{S_j\ S_k} = \frac{\sum_{i=1}^{n}(X_{ij} - \bar{x}_j)(X_{ik} - \bar{x}_k)}{\sqrt{\sum_{i=1}^{n}(X_{ij} - \bar{x}_j)^2}\ \sqrt{\sum_{i=1}^{n}(X_{ik} - \bar{x}_k)^2}} \tag{2.2}$$

Adding values on its sites makes the correlation matrix. Notice that the diagonal is 1, the reason is because in these positions the two variables are the same.

$$Corr\ Matrx = \begin{bmatrix} 1 & corr(X_1, X_2) & \cdots & corr(X_1, X_p) \\ corr(X_2, X_1) & 1 & \dots & corr(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ corr(X_p, X_1) & \dots & \dots & 1 \end{bmatrix} \quad \textbf{(2.3)}$$

The matrix is usually represented as a distribution/ gradient of colours. Tree colours are used, one of them represents the vales near to 1, another the values near to -1 and the last colour the values near to 0. This last is always white. That permits to use darker tonalites for correlations nears to the ends (1 and -1), and lighter tonalities for those near to 0.

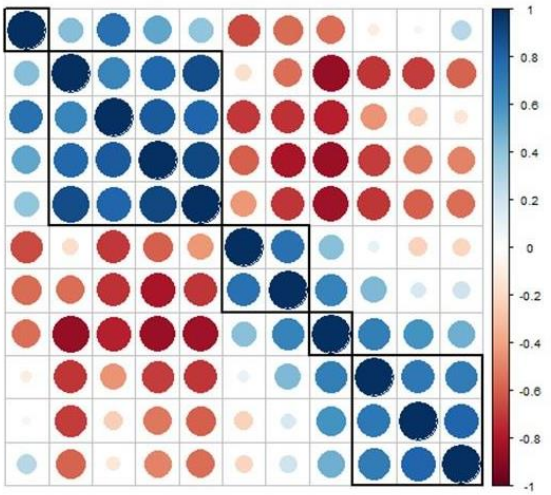

Fig 2.2 Correlation Matrix with gradient.

Source: *https://analyticsdataexploration.com/finding-patterns-in-predictor-variables/*

Another common way to represent it is using a two-dimension graphic for every coefficient. Each graphic dimension is a variable and each point represent an instance of the dataset. There are three possible graphics according to the types of correlation:

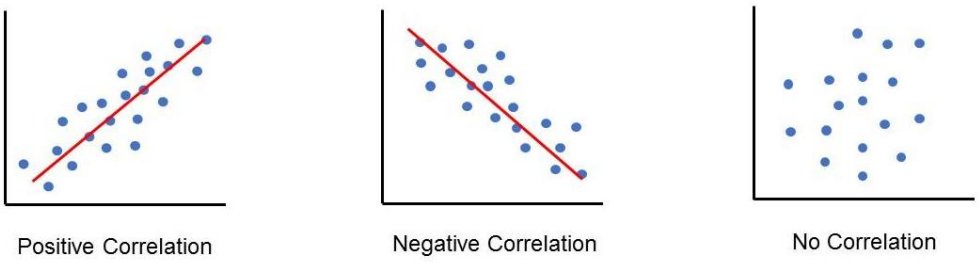

Fig 2.3 The three types of correlation.

A good understanding of the correlation matrix will allow us to make a better selection on the features. This will improve the efficiency of the predictive algorithm, which will not have to process values with no relevant information. In addition, it will also help to improve the interpretation of future algorithm results.

2.4. Balanced the dataset

Predict for a classification problem has some difficulties, the most relevant appears when the classification categories are not equally represented, classify instances into the minority class is not an easy task. The main problem is that a predictive algorithm could classify everything as the majority class and still be correct in a high percentage. Balance the data set is the most efficient solution to avoid this trouble. There are different methods of addressing it: the under-sampling, the over-sampling and the combination of both methods.

2.4.1. Under-sampling

Under-sampling is based on reducing the instances of the majority class, in order to obtain the same quantity as the minority class. *Fig 2.4* shows this idea. Every instance is represented as a point on a plane with x variables. Every variable is a feature; therefore, the image represents instances with 3 features.

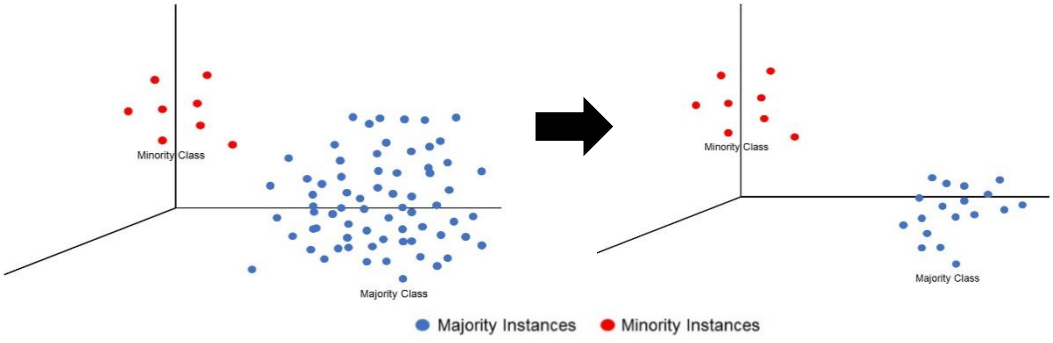

Fig 2.4 Illustration in 3 dimensions of the under-sampling method.

Different under-sampling techniques have been tested in different machine learning problems, for example, an under-sampling of instances considerate like outliers [7]. However, methods like this need a deeper data analysis and they do not solve the unbalanced problem better than selecting random samples. For that reason, select random samples of the majority class is the most used technique.

Even though a random selection balances the training set, it has a remarkable disadvantage; the shortage of data. An under-sampled training set could have not enough instances to train the predictive algorithm. For that reason, there are other methods, as over-sampling, which works the other way around fitting the minority class into the majority class.

2.4.2. Over-sampling

For some training sets, especially for those with a tiny quantity of minority instances, the over-sampling method could be better than under-sampling. Over-sampling is based on the creation of new minority class samples until obtaining the same number of instances as the majority class. An illustration of this idea is represented in *Fig 2.5*.

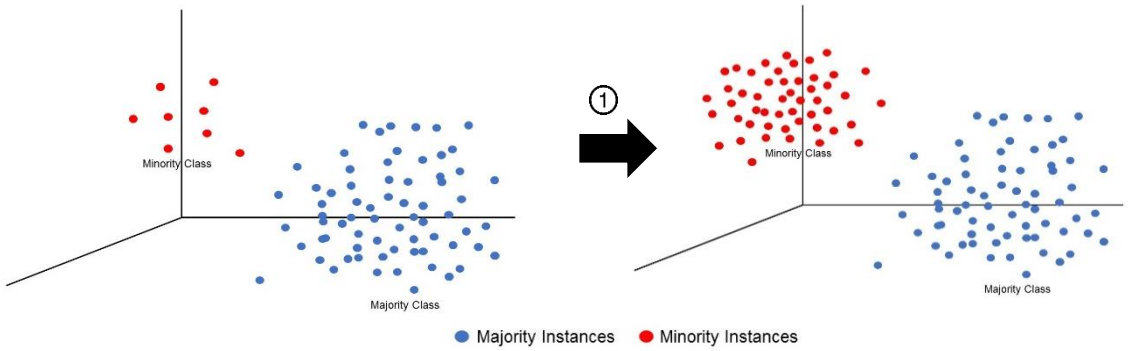

Fig 2.5 Illustration in 3 dimensions of the over-sampling method.

A very simple technique is to replicate minority instances, using the same values as the actual instances. However, this technique does not show any good result at the test set. The reason is that the model is learning the same values again and again. A relatively new technique, which seems to not have this kind of problem is the Synthetic Minority Over-sampling Technique, i.e. SMOTE.

2.4.2.1. Synthetic Minority Over-sampling Technique

SMOTE synthesizes new minority instances based on actual minority class instances. This technique starts tracing an imaginary line towards closest neighbours (other instances) of the same class. After that, it creates unreal instances on these lines. With this method, synthetic instances have similar

feature values as real instances. Following the same idea as *Fig 2.4* and *Fig 2.5*, *Fig 2.6* shows an illustration of the SMOTE concept.

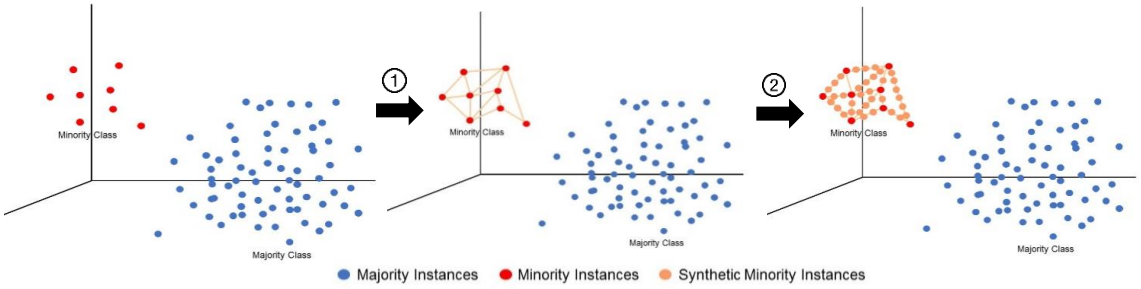

Fig 2.6 Illustration in 3 dimensions of the SMOTE technique.

Even though the SMOTE technique creates instances with different values from the actual instances, the same problem which appears using under-sampling can appear, obtaining bad predictions at the test set. This problem is called overfitting and it is introduced in the following sections. When this happens, there is another alternative to improve the predictions, using a combination of both methods: under-sampling and over-sampling.

2.4.3. Under-sampling with over-sampling

Use both techniques in the same training set could provide two benefits: avoid overfitting and improve directly the prediction for new instances [8]. The main idea is to first reduce the majority class and then create new minority instances. Therefore, in order to balance the dataset, less majority class instances have to be deleted and less minority class instances created. For the under-sampling method, the most common technique used in this case is the random selection of instances, step 1 in *Fig 2.7*. For over-sampling is the SMOTE technique, steps 2 and 3 in *Fig 2.7*.

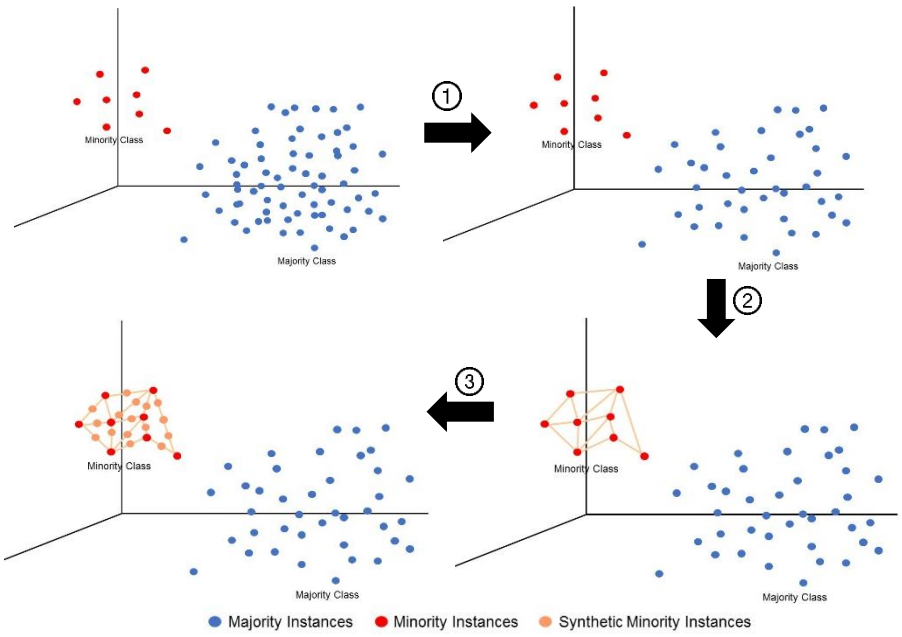

Fig 2.7 Illustration in 3 dimensions of the combination of under-sampling (random technique) and over-sampling (SMOTE technique) methods.

2.5. Underfitting and Overfitting

Under and Overfitting are two of the most common modelling situations in Machine Learning algorithms, they could affect negatively in the predictions. Underfitting could appear when the predictive model does not fit at all with the data. Overfitting, contrary, could appear when the model fits extremely well with the train and validate set, but not with the test set.

Fig 2.8 represents a distribution of instances that belong to two different classes: reds belong to the minority class and blues to the majority. A model is trained with these instances, creating a threshold in order to classify future instances, it is represented as a grey dashed line. Although not all the instances are classified correctly, the model follows the tendency of the distribution. Regardless of the error that the model may have, it is an example of a classifier that fits well with the training set.

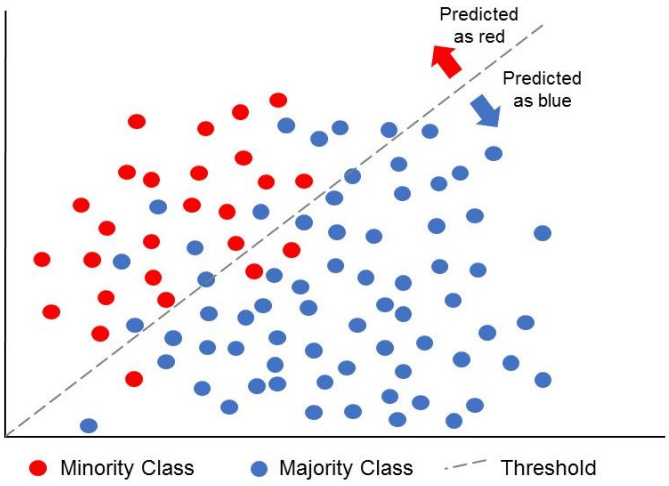

Fig 2.8 Illustration of a classifier that fits correctly with the data.

2.5.1. Underfitting

When the model is underfitted, it does not fit well with the training set, as a consequence, it is not able to classify correctly new instances. There are two main reasons of underfitting. One is the use of a too simple model when a more complex one is required and another one is to not have enough instances. In *Fig 2.9*, the grey dashed line represents the threshold which it does not follow the tendency of the training set.

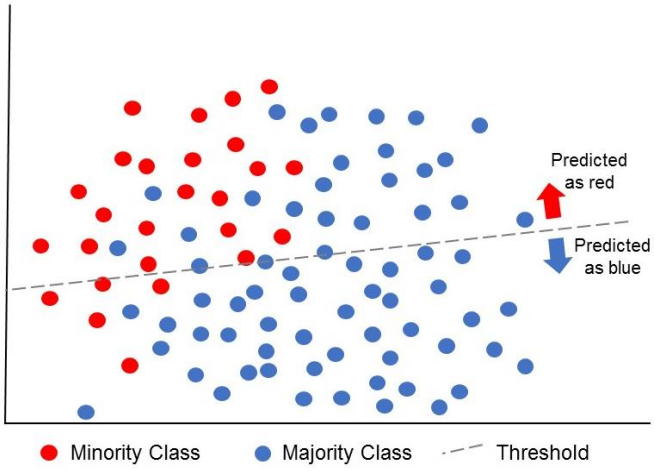

Fig 2.9 Illustration of an underfitted model.

2.5.2. Overfitting

Overfitting appears when the model is fitted extremely well to the training set. Some problems appear when the model takes into account outliers and is not being able to follow the tendency of the majority instances. In this situation, the results from the validate set are excellent. However, at the test set results become bad, the model is not able to classify correctly new instances. *Fig 2.10* shows an example of a threshold that conforms with outliers. All instances that appear are correctly classified, but when new ones need to be classified, a considerable error could appear due to the overfitting.

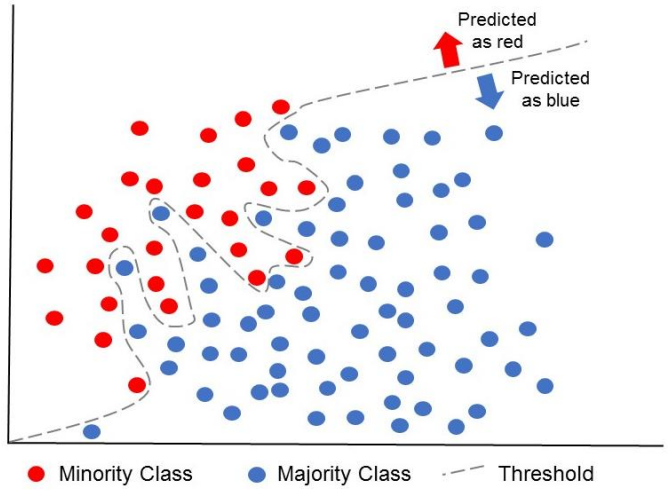

Fig 2.10 Illustration of an overfitted model.

2.5.3. Techniques to avoid fitting problems

Some solutions to avoid under and overfitting are used frequently. However, they do not always work, it depends on the trying training set. The most common ones are: to add new useful data, to reduce the number of features, and to change the predictive model.

2.5.3.1. Add new useful data

Add new instances in the training set could make the algorithm not take into account the outliers. That makes the algorithm able to follow the tendency of the majority ones and predict correctly the new cases. It is necessary to add a large number of new instances. In many cases, it is an inefficient task or there is not enough data to add.

2.5.3.2. Add or reduce features

A different way to have more information on the training set without adding extra instances is including new features. Taking into account the correlation matrix, it is possible to select different features that could be useful. On one hand, if the model is underfitted because it does not fit with the training set, this could be one of the best solutions to solve the problem. On the other hand, instances could be considerate as outliers due to a specific feature. If it is possible to delete this feature without losing relevant information, the reduction of features can avoid overfitting.

2.5.3.3. Use a different model

Sometimes, a complex model with a simple input could perform memorizing the data. That is something important to avoid because it causes overfitting. In this case, the solution could be to use a less complex model. The same happens the other way around, a model that is not complex enough could cause underfitting. It is important to find the model that best fits the data.

2.6. Boosting

Boosting is an algorithm used in supervised learning. Typically, it is combined with other algorithms in order to improve performance. Boosting is based on using different classifiers and adding weights on training sets. One of the most popular boosting algorithms is the *ADABoost*, formulated by Yoav Freund and Robert Schapire in 2003 [9]. To understand how boosting algorithms work, the main steps are explained below. In addition, *Fig 2.11* illustrate the procedure.

Step 1: An algorithm is trained with a training set in order to create a classifier.

Step 2: The same instances on the training data are predicted by the classifier.

Step 3: Different weights are added to the previous instances. Higher weights are associated to those instances which are predicted wrongly. Lower weights are for instances which are predicted correctly. A weighted dataset is created.

Step 4: A new classifier is trained with the weighted dataset. This classifier is focused on the instances with higher weights.

Step 5: steps 3 and 4 are repeated as many times as the as many times as the data scientist establishes.

Step 6: All the previous classifiers are combined in order to create a more robust one.

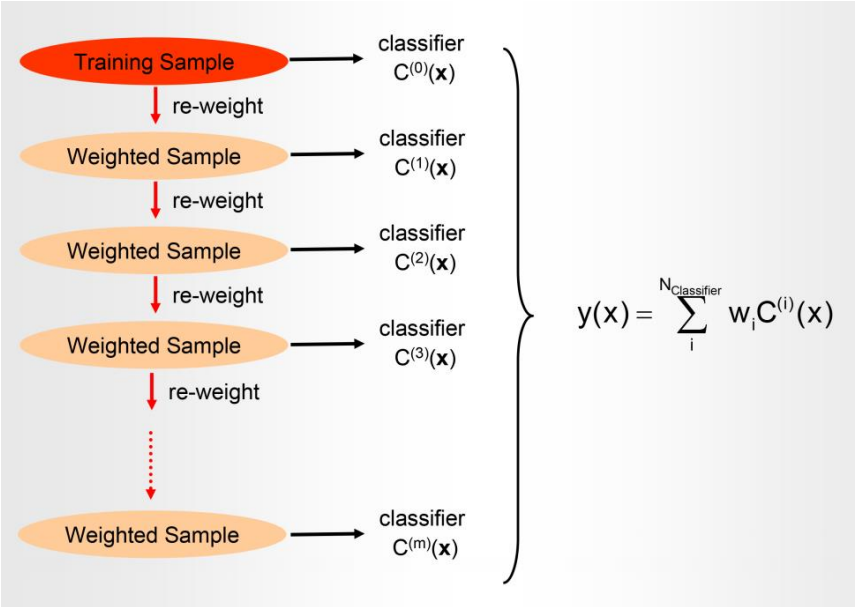

Fig 2.11 Diagram of a boosting algorithm [10].

Many models implement this algorithm as an extra method. In this project, the boosting is implemented in the Decision Tree model, which is explained in future sections.

2.7. Predictive models

Within the great variety of models that are used in machine learning, three have been chosen. The criterion for choosing them are:

- They have to be trained with labelled data, or in other words, using supervised learning.
- They have to be able to classify into two different classes, there is no need to apply a regression model.
- The implementation must be efficient.
- The way they perform have to be understood at least in a general way.
- Their results have to be easy to interpret.

In addition, it has been taken into account which are the most used models and algorithms in credit transactions problems, like fraud detection. C5.0 algorithm is one of those [11], therefore the Decision Tree model are chosen. Taking advantage that the Random Forest model works in a similar way, this model is also chosen. The third one is the Naïve Bayes model, a simple and very effective model in many real wold problems. Naïve Bayes model also meets all the above criteria.

2.7.1. Decision Tree

Decision Tree model follows a very intuitive procedure based on rules. The understanding of this procedure allows data scientists to get an easy interpretation of the results. Sometimes, the complexity of some models causes the black box effect, which means that data scientists are using an algorithm without knowing how it works. With a brief explanation of how Decision Trees are built, it is possible to avoid this situation. Another reason is that Decision Trees have an excellent ratio between simplicity and how well they predict. Because of that, they are one of the most used models in Machine Learning.

2.7.1.1. Make the Decision Tree model

In order to understand how a Decision Tree model is built, it is important to have in mind the parallelism between a tree and the diagram of the model. Each node of the Decision Tree corresponds to a feature, each branch to a decision or rule and each leaf corresponds to a class label of the target feature (the one that has to be predicted) which can be categorical or continuous value. A categorical target means that the Decision Tree is working as a classifier, a continuous value means that it is working as a regression model. In this section all the explanations are based on a classification Decision Tree because is the one that will be implemented. *Fig 2.12* shows an easy example of a classification Decision Tree.

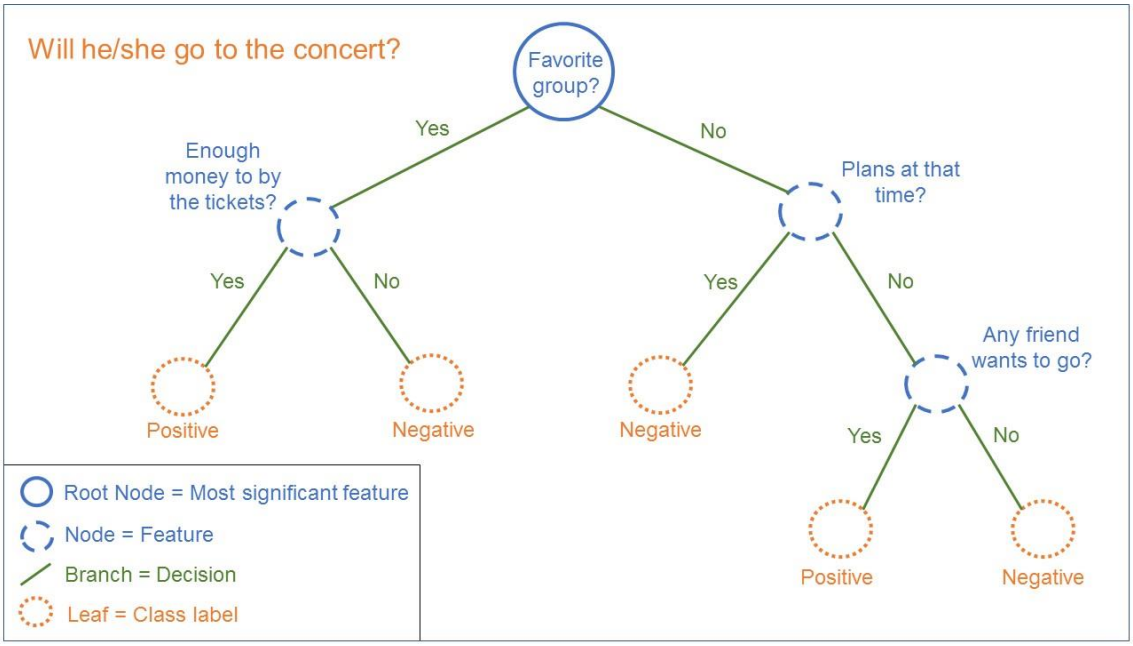

Fig 2.12 Example of a Decision Tree working as a classifier of two classes (positive and negative).

The first step that algorithms need in order to build a Decision Tree is to know which the target feature is, and if it is categorical or continuous. Once that is

known, the following step is to place other features at the nodes of the Decision Tree. On the root node goes the feature which best splits the data or, in other words, the one with more relevant information. Other features are sorted by how heterogeneous its information is. Data is spat into homogenous subsets and different metrics are implemented for every feature. Each algorithm has a metric for sorting features. For example, ID3 uses Information Gain, a metric based on the Entropy Function, both ideas are explained below. Other algorithms like CART uses the Gini Index. This project implements the C5.0 algorithm therefore, the Information Gain is explained below.

Entropy Function is defined in order to understand how the Information Gain is calculated. The Entropy Function was developed by Claude Shannon in 1948 [12]. The Entropy Function measures the uncertainty or disorder of an amount of information. For example, if the data is completely homogeneous the entropy is zero. However, if the different data values are equally divided, the Entropy Function is equally entropy to one. Equation (2.4) show how to calculate the Entropy Function.

$$H = -\sum_{i}^{J} P_i * log_2(P_i) \qquad \textbf{(2.4)}$$

J is the maximum number of classes an amount of data has. P is the probability of a specific class to appear.

Fig 2.13 shows three examples: $H_1 = 0$ is the Entropy of an amount of data that has a unique class, $H_2 = 0.964$ the Entropy when there are 11 and 7 values for each class of a total of 18. Finally, $H_3 = 1$ represents the Entropy when all the classes, two in this example, are equally represented.

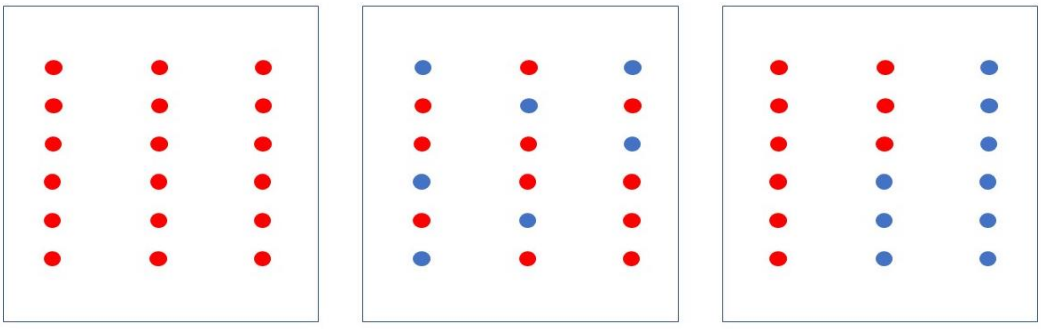

Fig 2.13 Entropy in three different examples: with one class, with two classes not equally represented, with two classes equally represented.

Being $X = (x_1, x_2, \dots, x_n)$ all the features at the dataset and y the target feature, Information Gain is the difference between the Entropy Function of the target feature and Entropy Function of the feature we want to evaluate, as an example the first feature x_1, according to the target. Equation (2.5) shows the idea:

$$IG(y, x_1) = H(y) - H(y|x_1) \quad \textbf{(2.5)}$$

Equation (2.6) shows how to calculate the first component of equation (2.5). Notice that in order to make calculations simpler, the target feature: y has two possible classes: positive and negative. $\mathrm{P}(+)$ is the probability of obtaining a positive class value and $P(-)$ the probability of obtaining a negative class value.

$$H(y) = H(+,-) = -1 * \Big(P(+) * log_2\big(P(+)\big) + P(-) * log_2\big(P(-)\big)\Big) \quad \textbf{(2.6)}$$

The second term of equation (2.5) is calculated on equation (2.7). It is considerate that feature x_1 has only two classes, that does not mean that all features need to be categorical either with two classes. The Entropy Function is now calculated according to the two classes form target feature: y.

$$H(y|x_1) = H(+,-|x_1) = P(+|x_1) * H(+|x_1) + P(-|x_1) * H(-|x_1) \quad \textbf{(2.7)}$$

The Information Gain is calculated for every feature in the Decision Tree, equation (2.7) is an example of the calculation for the first feature: x_1. The feature with the highest Information Gain value goes at the root node. The lower the Information Gain is, the further the feature is from the root node. Finally, the Decision Tree associates a target class to every leaf. Therefore, Decision Trees are built from the root until the leaf. All this procedure is done with the training set.

A Decision Tree with many leaves has a lower classification error that a one with less. However, an excess of leaves could cause overfitting. A challenge of many algorithms is to know the optimum value of leaves to build.

2.7.1.2. Predict new instances with the Decision Tree

When a new instance has to be classified, either from the validate set or from the test set, the Decision Tree acts like a group of rules. The first rule applied is the one that corresponds to the root node. The instance follows the Decision Tree structure until it arrives to a leaf, where it is classified as the corresponding class.

2.7.1.3. Why C5.0 algorithm?

There are many algorithms for implement a Decision Tree. The most popular are: The Iterative Dichotomiser 3, i.e. ID3, its successors: The C4.5 and the C5.0, the Classification and Regression Trees, i.e. CART and CHi-squared Automatic Interaction Detector, i.e. CHAID, among others. This project uses the C5.0 algorithm which descends from the C4.5. The C4.5 is considered for most data scientist the number one of the machine learning algorithms [13]. However, other papers show that the C5.0 is more precise than the C4.0 [14]. Both algorithms have they origins on the ID3. ID3 is a combination of different methods based on tree diagrams. It was developed by Ross Quinlan in the late 1970's. Years later, in the 1980s Quinlan joined together the methods and created the C4.5. He continued working on this algorithm until he developed the C5.0. However, it was not until 2013 when Max Kuhn and Kjell Johnson create a more complete version, the current C5.0 [14].

2.7.2. Random Forest

Random Forest is a predictive model, also based in decision trees but more complex than the Decision Tree model. Random Forest applies the *bagging* method, which it is based on using different learning models (different trees) and compare them in order to make a better decision.

The first Random Forest was developed by Tin Kam Ho in 1995 [15]. Ho introduced some of the main Random Forest characteristics, like the *bagging* method and the randomly selection of features, which is explained bellow. Years later in 2001, Leo Breiman and Adele Culter added the controlled variance [16], making possible to detect and avoid Decision Tree based on outliers.

2.7.2.1. Make the Random Forest model

The first step to make a Random Forest model is to create subsets from the original training set. When subsets are selected randomly with replacement, the behaviour of the model in each subset is similar [17]. That happens because there is no relation between subsets and features. A group of close features could act very different than another group.

After that, a Decision Tree is created for every subset. These Decision Trees are not exactly the same as the one explained previously. The following steps show how they are built:

Step 1: Some features are randomly selected as candidates to be at the root node. The Information Gain of candidates is calculated. The one with the high Information Gain is placed at the root node.

Step 2: For the second node, candidates are selected randomly without taking into account the root node feature. The Information Gain of candidates is calculated. The one with a high value is placed at the second node.

Step 3: Step 2 is repeated until the Decision Tree is completed.

The difference between a Decision Tree on the Decision Tree model and on the Random Forest model is that in the first model, all the features are candidates to be at a specific node, except those which are already chosen. However, on the Random Forest model, only those which have been randomly selected are candidates.

Random Forest models use between 100 to several thousand trees, the number is chosen depending on the type of data and the generated error. The most common method to calculate the error is the out-of-bag error, i.e. OOB.

In order to obtain the OOB error, every instance of the training set is predicted on those trees that did not have it in their subset (out of the bag). Therefore, they had not been trained with the instance. On each prediction, the error is calculated and the mean of all the errors is the OOB. The OOB decreases when new trees are added to the Random Forest model. However, a huge number of trees can make the model inefficient. The key is to choose an optimum number of trees. Generally, the Random Forest algorithm provides an output with the OOB error according to the number of trees generated. *Fig 2.14* is an example of this output.

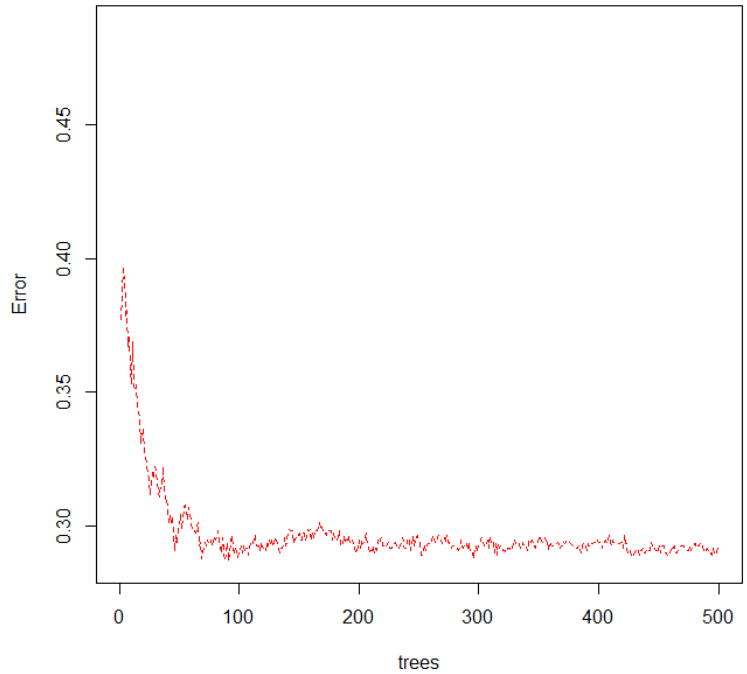

Fig 2.14 Example of a plot of the OOB Error against the number of trees on a Random Forest model.

Source: https://www.researchgate.net/figure/2-Plot-of-Out-of-Bag-Error-Against-Number-of-Trees_fig23_278784281

2.7.2.2. Predict new instances with the Random Forest

When a new instance has to be classified, it is introduced to every Decision Tree, which makes the decision according to its rules. When the instance is classified by all the Decision Trees, a new function counts how many times each class has been predicted. The Random Forest model classifies the instance into the most predicted class. For example, in *Fig 2.15* the instance introduced is classified at the positive class, because three of four trees have predicted as positive.

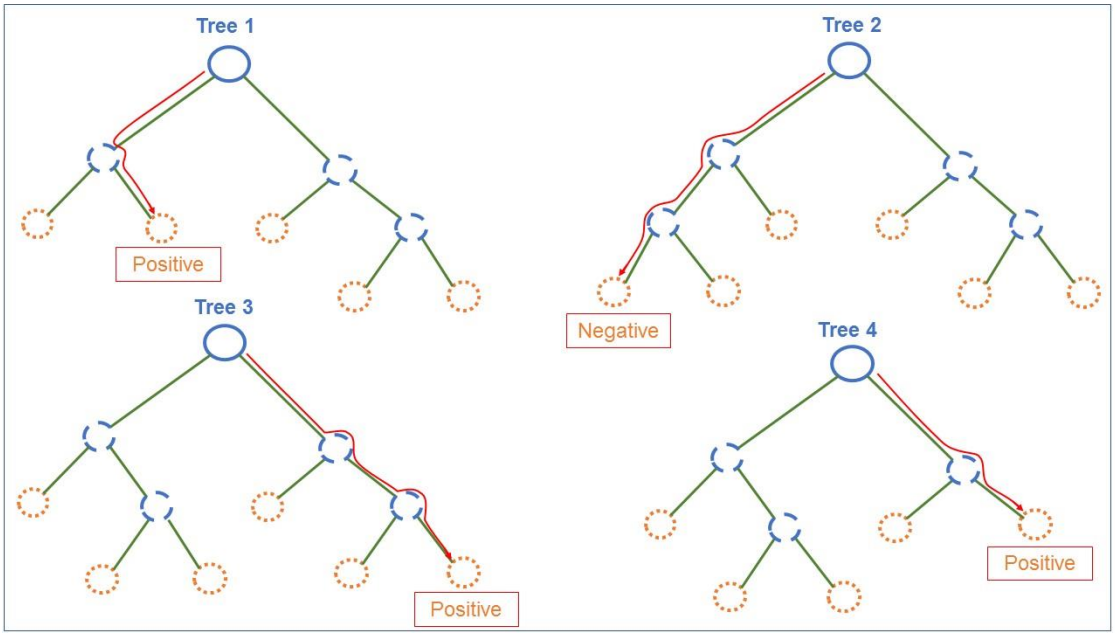

Fig 2.15 A Random Forest formed by four Decision Trees. An instance (in red) is classified at the positive class.

2.7.2.3. Why the Random Forest algorithm?

Unlike Decision Tree, Random Forest has one highly standardized algorithm to implement the model. It is called the Random Forest Algorithm.

2.7.3. Naïve Bayes

Naïve Bayes is a predictive model used for classification tasks, especially for high-dimensional datasets. For example, for document classification or for sentimental analysis. Naïve Bayes provides a better accuracy when features are normally distributed and relevant [18]. The word “naïve” refers to the assumption that all features have an equal effect on the target feature, every feature is independent from each other. Naïve Bayes model is based on the Bayes’ theorem, which was formulated for the first time by Thomas Bayes in 1763. Bayes’ theorem is based on knowing the probability of even A given B: $P(A|B)$ Equation 2.8 shows how to calculate it.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \textbf{(2.8)}$$

$P(B|A)$ is the probability of occurrence of event B given the event A is true. $P(A)$ and $P(B)$ are the probabilities of the occurrence of event A and B respectively. On the Bayes' theorem, every probability has a name, they are included in equation 2.9.

$$posterior = \frac{likelihood \times prior}{evidence} \quad \textbf{(2.9)}$$

2.7.3.1. Make the Naïve Bayes model

Previous equation applied to a machine learning problem [19] can be written as:

$$P(y_k|X) = \frac{P(X|y_k)P(y_k)}{P(X)} \quad \textbf{(2.10)}$$

For each k possible classes from the target feature y.

The nomenclature is the same used on the Decision Tree explanation, chapter 2.7.1. y represents the target feature, which can have k classes $y = (y_1, y_2, \dots, y_k)$. In order to simplify equations, only two classes $(k = 2)$ have been considerate for the target feature; positive class and negative class. X represents the n features that an instance has. $X = (x_1, x_2, \dots, x_n)$.

The first step is to calculate the probability of obtain each class: $\mathrm{P(y)}$, They are represented as: $\mathrm{P}(+)$ and $\mathrm{P}(-)$. Using the Bayes' Theorem nomenclature, these values the prior probability.

Evidence $P(X)$ is calculated multiplying the n features probability. It is illustrated in equation 2.11.

$$P(X) = P(x_1)P(x_2) \dots P(x_n) \quad \textbf{(2.11)}$$

Previous probabilities can be calculated in different ways, for example, dividing the number of instances from one class into all the instances in the training set. Another common technique is to assume that all the classes have the same probability. However, depending on the type of data, it is necessary to make an assumption on the feature's distributions. When the data is discrete, features are assumed to have a Multinomial or Bernoulli distribution. For continuous data, Gaussian distribution [20].

In order to obtain the likelihood probability, it is necessary to calculate the probability of occurrence of each event X given the class y. It is shown in equation (2.12).

$$P(X|y) = P(x_1|y)P(x_2|y) \dots P(x_n|y) \quad \textbf{(2.12)}$$

Finally, all the previous equations are included on the Bayes' Theorem equation in order to obtain the posterior probability.

$$P(y|x_1, x_2, \dots, x_3) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad \textbf{(2.13)}$$

This equation has to be formulated for all the target feature classes. At the end, every new instance has a posterior probability for each class. In this specific case, there will be two calculations: one for the positive class and another for the negative class.

2.7.3.2. Predict new instances with the Naïve Bayes

Once the probably model is done, a decision rule has to be applied in order to decide if a new instance will be classified as positive or as negative. The most common technique to use is the maximum a posteriori, i.e. MAP. This technique consists on classify the new instance on the class with the highest posterior probability. For example, an instance with a 0.8 posterior probability for the negative class, and 0.2 for the positive class, will be classified at the negative class.

2.7.3.3. Why the Gaussian Naïve Bayes?

The Gaussian Naive Bayes model makes the assumption that values are distributed according to a Gaussian distribution. It is used for continuous feature values. Some features on the datasets used in this project are continuous, for example, the amount in euros of each supported transaction. For that reason, the Gaussian Naive Bayes model can fits well with the dataset.

2.8. Performance Measures

In order to evaluate the performance of classification algorithms, different parameters are used. All of them measure how similar the predictions and the actual values are [3]. Depending on the input test/validate set, some types of measures are more suitable than others, for example unbalanced data require different metrics than a balanced.

2.8.1. Confusion Matrix

The confusion matrix is a measure focused on evaluating the performance of a binary classifier. The general idea is to illustrate the number of times instances of a class are classified as another class. Each row represents predicted instances, while each column represents actual instances. At the diagonal of the matrix are the values that have been predicted correctly.

Every value at the confusion matrix has a name based on its class and if the prediction is correct or not. An actual positive instance predicted as positive is called “True Positive” i.e. TP, predicted as negative is called “False Negative” i.e. FN. The same happens with a negative instance: predicted as a negative is a “True Negative” TN and predicted as a positive is a “False Positive” i.e. FP. This nomenclature is shown in *Table 2.1*.

Table 2.1 The confusion matrix of a binary classifier.

		Actual	
		Negative	Positive
Predicted	Negative	True Negative (TN)	False Negative (FN)
	Positive	False Positive (FP)	True Positive (TP)

One of the advantages of the matrix is that from its values can be calculated other metrics like accuracy, precision, recall, etc.

2.8.2. Accuracy

Accuracy is the ratio between correct predicted instances (TN, TP) and the total of the instances evaluated. Equation (2.8) represents this ratio:

$$accuracy = \frac{TN + TP}{TN + TP + FP + FN} \quad \textbf{(2.14)}$$

A high accuracy (near to 1) means that there are many instances predicted correctly. However, not always accuracy is useful to evaluate how good or bad an algorithm predicts. When false predictions have a very negative impact, accuracy is not suitable. For example, in fraud detection; 99% of the transaction are no-fraudulent while only 1% are fraudulent. An algorithm with an accuracy of 99% may seem appropriate. However, it probably could not detect any of the fraudulent instances, which are the most important ones to predict correctly. Therefore, it is necessary a higher accuracy on the minority class detection. Other parameters like precision and recall are able to measure this situation.

2.8.3. Precision and Recall

Precision is the ratio between correctly positive predictions and all the instances predicted as positive.

$$precision = \frac{TP}{TP + FP} \quad \textbf{(2.15)}$$

Recall is the accuracy of the positive instances. It is calculated making the proportion of positive instances that are correctly detected and all the actual positives ones. Recall is also called Sensitivity.

$$recall = sensitivity = \frac{TP}{TP + FN} \quad \textbf{(2.16)}$$

2.8.4. The F_1 score

The F_1 score is a metric that combines precision and recall. F_1 score goes from 0 to 1 and it is the best option for taking into account both metrics. Low precision or recall values have more impact than high ones. Therefore, in order to obtain a high F_1 score both recall and precision must be high values. Therefore, F_1 score is the harmonic mean of precision and recall, this is shown in equation (2.11).

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad \textbf{(2.17)}$$

It is not possible to obtain ideal values in both metrics; when the precision is increased the recall is reduced and vice versa. The challenge is to obtain the best balance between both metrics.

2.8.5. Specificity

Specificity is the accuracy of the negative instances. It is calculated making the proportion of negative instances that are correctly detected and all the actual negative ones.

$$specificity = \frac{TN}{TN + FP} \quad \textbf{(2.18)}$$

2.8.6. The ROC Curve

The Receiver Operating Characteristic Curve, i.e. ROC curve is a probability curve that plots the sensitivity or recall on y-axis and "1- specificity" on x-axis [3]. The area that falls below is called the Area Under the Curve. i.e. AUC. It goes from 0 to 1, and represents the separability between two classes. The ROC curve and the AUC are some of the most used metrics to evaluate binary classifiers. Although some studies show that the both parameters can also be used for multi-class problems, results are not as reliable as they are in binary problems [21].

Several illustrations of a balanced problem are provided in *Fig 2.16*, *Fig 2.17*, *Fig 2.18*, *Fig 2.19*. These illustrations include a representation of the probability distribution, in order to understand why the ROC curve has a particular shape.

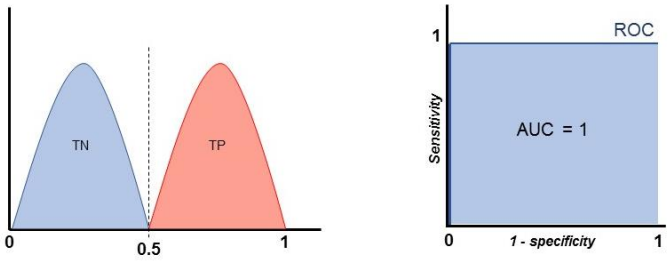

Fig 2.16 Probability distribution on the left and ROC Curve on the right for an AUC = 1.

Fig 2.16 shows an AUC value equal to 1, which means that the model evaluated classifies all the instances in their corresponding class, it is perfectly able to distinguish between both classes. Therefore, the challenge is to obtain an AUC near to 1. Notice that instances can be TN or TP, there is no false instance.

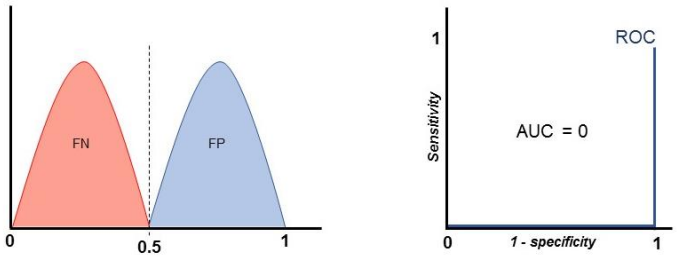

Fig 2.17 Probability distribution on the left and ROC Curve on the right for an AUC = 0.

Fig 2.17 shows the other way around; an AUC is equal to 0. In this case, the model classifies all the negative instances as positive, and the positives as negatives. Instances can be FN or FP, there is no true instance.

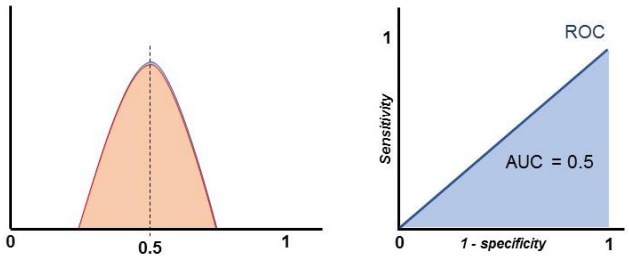

Fig 2.18 Probability distribution on the left and ROC Curve on the right for an AUC = 0.5.

Fig 2.18 represents an AUC equal to 0.5. It illustrates a critical case in which the model cannot distinguish between both classes.

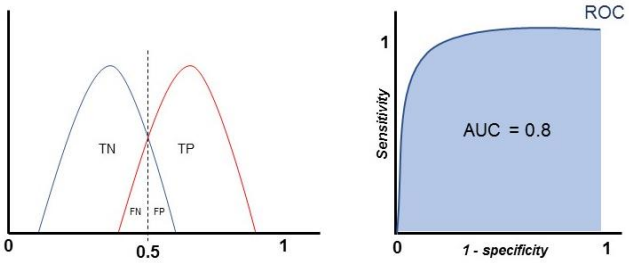

Fig 2.19 Probability distribution on the left and ROC Curve on the right for an AUC = 0.8.

Finally, when the AUC is not any of the previous values (1, 0, 0.5), the model is able to classify but with some error. The four different types of instances (TP, FP, TN and FN) appears. This is the most common satiation on classifiers. *Fig 2.19* shows an example.

CHAPTER 3 SET UP OF DATASETS AND ALGORITHMS

3.1. Group daily datasets

As it is said in Chapter 1.1, the financial entity extracts two daily datasets: one dataset called (a), with all the supported transaction (paid and unpaid). And other dataset (b) with only the unpaid supported transactions.

Every day, both datasets are extracted independently. Dataset (a) has around 1000 instances per day and 48 columns. Dataset (b) has on average 60 instances every day and 14 columns.

The first step is to group all instances from one month into a single dataset, both for (a) and for (b). The goal is to get enough supported transactions to train, validate and test the algorithms.

The daily datasets used for training and validating algorithms are from September 2018. The daily datasets to test algorithms are from November 2018. The reason is that when the project was started the most completed and recent data was from these two months.

3.2. Cross datasets

However, not only a big number of instances is needed. If we want to train algorithms based on supervised learning, the input data must be labelled. Therefore, the next step is to use dataset (b) to label dataset (a). The labelled dataset will be (c), which will have the same instances and columns as dataset (a) and an extra column for labeling those supported transaction which are unpaid.

The objective is to find in dataset (a) all the supported transactions which appear in dataset (b), and labelled as unpaid supported transaction. For every supported transaction in (b), the main features are compared with the ones in every instance in (a). Although in dataset (a) there are more features also relevant, only those which are in both datasets can be compared. Therefore, the chosen features are:

- Credit card identification
- The amount of the supported transaction in euros.
- The date and the time at which the supported transaction was made.
- The identification of the trade involved in the supported transaction.

Two problems appear when data has to be compared:

Problem 1: Each dataset has a different type of credit card identification. In financial entities it is usual to use different identifies in order to protect the customer's privacy. However, in this text, they are different because these datasets were defined for a different application. To require the financial entity a new extraction takes too many time and money. Therefore, the crossing between datasets has to be done without comparing the credit card identification.

Problem 2: Some supported transactions do not have the correct time value. When this happens, the time value is by default 00:00h. Therefore, the dataset contains a large number of supported transactions with the same time value, even though they were done in different times of the day.

Combination of both problems makes impossible to differentiate between some supported transactions, which have the same features values. Therefore, there is no way to label these instances, into paid or unpaid, with a 100% certainty.

Instead, every supported transaction is labelled with its probability to be unpaid on dataset (c). To do that, all the instances with the same feature values in (b) are summed and then divided into the sum of the ones in (a) that match. The described procedure is represented in equation (3.1):

$$unpaid\ probability\ (\%) = \frac{\sum matching\ instances\ in\ (b)}{\sum matching\ instances\ in\ (a)} * 100 \qquad \textbf{(3.1)}$$

At this point, there are three alternatives to aim the project:

- As a regression, because probabilities are continuous functions.
- As a multiclass classifier, by establishing margins in the probability.
- As a binary classifier, by establishing margins in the probability.

However, as it is said in Chapter 1.2, the solution must be an algorithm able to classify into two binary classes: paid or unpaid supported transactions. Therefore, the chosen solution is the third. A binary classifier.

In order to obtain a binary dataset, all the values calculated by equation (3.1) are turned into only two possible values: 0 or 1. Using the nomenclature in CHAPTER 2, 0s represent the negative class and 1s the positive class. Three different approximations are contemplated:

- Type 1 approximation: 0 if the unpaid probability is different to 100%. 1 if the unpaid probability is equal to 100%.
- Type 2 approximation: 0 if the unpaid probability is less than 50%. 1 if the unpaid probability is equal or higher than 50%.

- Type 3 approximation: 0 if the unpaid probability is 0%. 1 if the unpaid probability is different to 0%.

Type 2 approximation is the one with a lower approximation error. Therefore, the machine learning methods, the predictive models and the tests are applied to dataset (c) labelled with this approximation. Fig 3.1 represents the number of supported transactions at each class using the Type 2 approximation on dataset (c).

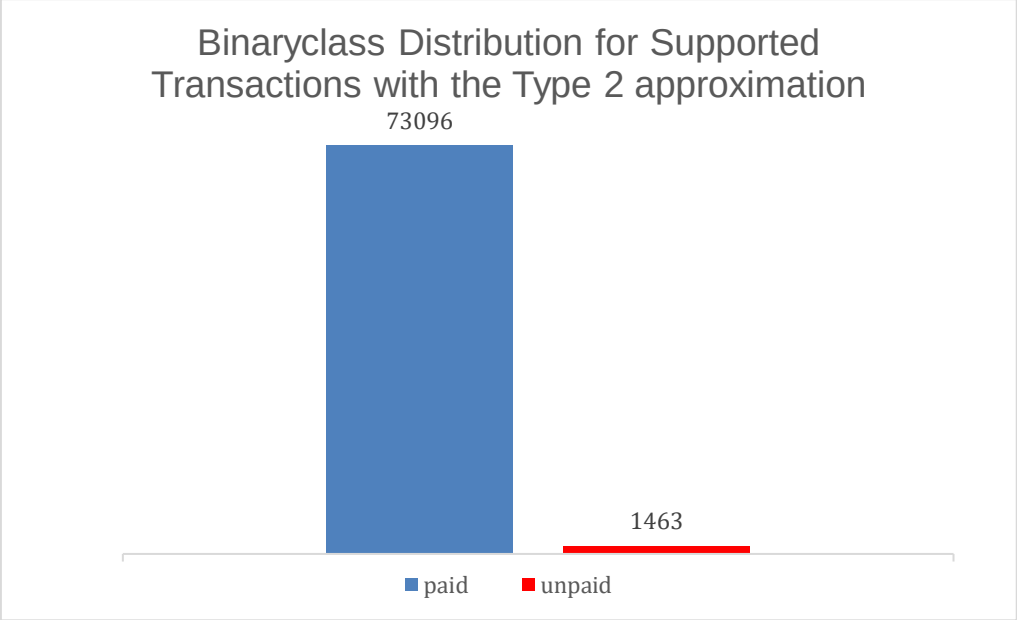

Fig 3.1 Bar plot of the binary class distribution for supported transactions using the type 2 approximation.

Fig 3.1 also shows how unbalanced is the distribution. Before training the algorithms, it will be necessary to apply techniques to balance the data.

3.3. Explore datasets

In order to train the machine learning algorithms with the best possible conditions, two procedures are needed. First to clean the data and then to select the most relevant features.

3.3.1. Clean the data

Every supported transaction on the cross dataset (c) has 48 features. Not all the features have relevant information, therefore, the first step is to delete all the useless features, like those which are 0 or NA values in all the supported transactions. Appling this idea, the features are reduced to 29.

Some other type of features that required extra treatment are those which contains relevant information but are not presented in the most appropriate way to train the algorithms. An easy example is the feature that contains the date and the time of a supported transaction. First of all, it is necessary to split the date into one feature and the time into another. Additionally, the same date will only appear in supported transactions of the same day, but not anymore. Although the date information is really important, it is not a good feature to train algorithm. Therefore, a solution could be to converted it into something more useful, for example, the day of the week.

3.3.2. Select features

Once the irrelevant features have been deleted and some other adjusted, the next step is to select those features which have more relevant information. In order to do that, an exploration of the data will be applied to the dataset (c). The correlation matrix explained in Chapter 2.3.1 is used. On a correlation matrix, the more relevant features are those that have more correlation with the target feature, the one that has to be predicted by the algorithm and less with other features.

From the 29 features of the dataset (c), 12 have been selected to implement the correlation matrix. These 12 features contain the most relevant information of each supported transaction. The other 17 features are related to technical values of the validation circuit, they are considered irrelevant to train algorithms.

Fig 3.2 show the correlation matrix of these 12 features. This correlation matrix is analysed without taking into account the meaning of each feature. The chosen features will be described later.

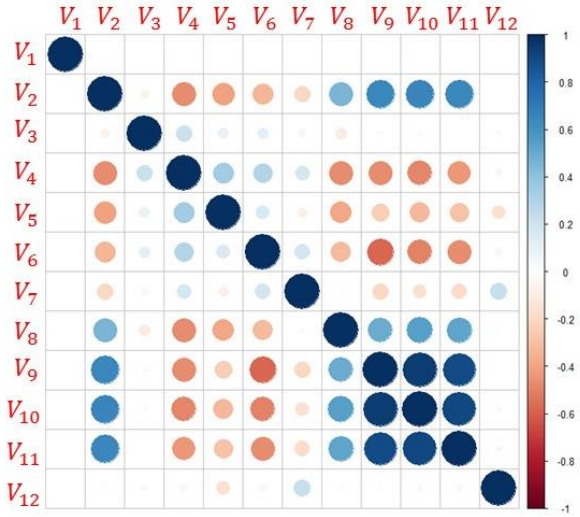

Fig 3.2 Correlation Matrix of the labelled dataset (c).

The correlation between the features: V_9 , V_{10} , V_{11} and also V_8 is very high, practically 1. This means that although they are different features, they represent the same information. Therefore, it will be enough to train the algorithm with only one of the three features. V_8 is the most appropriate because is the one with less correlation.

The most important idea that the matrix shows is the correlation between every feature and the target feature, V_{12} in this case. The most related features with V_{12} are V_5 and V_7. The most independent are V_1, V_2, V_3,V_4 and V_6. With all this information the chosen features to train the algorithms are shown in *Table 3.1*.

Table 3.1 The chosen features and the information contained in each one.

V_2	The amount in euros of the supported transaction.
V_3	The day of the week in which the supported transaction was made.
V_4	The time at which the supported transaction was made.
V_5	The identification of the trade involved in the supported transaction.
V_6	The national identification of the sector where the operation was made.
V_7	The international identification of the sector where the operation was made.
V_8	The identification of the receiving entity involved in the supported transaction.

Although V_1 is an independent value from V_{12} it is not chosen because is the credit card identification. As Chapter 1.2 presents, the objective of the project is to detect unpaid supported transactions without taking into account the card which performs it. V_1 has been introduced in the matrix simply to see if it has no correlation with other feature.

3.4. Balance and split datasets

Once the features are chosen, datasets have to be ready to apply the algorithms. The first step is to split the dataset (c) into a training set and a validation set, the chosen ratio is the one explained in Chapter 2.2 80% for the training set and 20% for the validate set. After that, techniques explained in Chapter 2.4 are applied in order to balance the data. Each technique provides a new dataset, which is also split. All of them and the test set are shown on the table 3.2 until table 3.6. As Chapter 3.2 explains all the methods and algorithms are applied to dataset (c) with the Type 2 approximation.

3.4.1. Dataset (c)

The dataset (c) with the Type 2 approximation has a 98% of paid supported transactions and only a 2% of unpaid. It is a very unbalanced dataset to train algorithms. *Table 3.2* show the distribution.

Table 3.2 Distribution of paid and unpaid supported transactions on dataset (c) with Type 2 approximation and its training and validate set.

	Paid	**Unpaid**	**Total**
Training data Type 2	73096 **98%**	1463 **2%**	74559
Training set	58397	1155	59552
Validate set	14699	308	15007

3.4.2. Under-sampled dataset

The technique based on selecting random samples of the majority class (paid supported transactions) has been applied in order to balance the original dataset. The number of samples selected is 1463, which is the same number that the minority class (unpaid supported transactions) has. With this technique the dataset is balanced with 50% supported transactions for each class. However, with this technique, very few supported transactions are available to train the algorithm (2329). It is shown in *Table 3.3*.

Table 3.3 Distribution of paid and unpaid supported transactions on the under-sampled dataset and its training and validate set.

	Paid	**Unpaid**	**Total**
Training data	1463 **50%**	1463 **50%**	1926
Training set	1168	1161	2329
Validate set	295	302	597

3.4.3. Over-sampled dataset

The SMOTE technique has been applied to the original dataset. The technique has created 70000 synthetic supported transactions from the minority class, following the procedure explained in Chapter 2.4.2.1. The over-sampled dataset is balanced (50.6% and 49.6%) and the number of supported transactions to train the algorithm is quite enough: 115844.

Table 3.4 Distribution of paid and unpaid supported transactions on the over-sampled dataset and its training and validate set.

	Paid	**Unpaid**	**Total**
Training data	73096 **50.6%**	71463 **49.4%**	144559
Training set	58601	57243	115844
Validate set	14495	14220	28715

3.4.4. Under and Over-sampled dataset

Both techniques are applied to the original dataset. First, 6000 paid supported transaction have been randomly selected (under-sampling). Being around the 6000 supported transactions of each class is enough to train the algorithm and obtain good predictions. After that, the SMOTE technique creates synthetic unpaid supported transactions (over-sampling). Appling these methods in this order, only 4389 supported transactions had to be created by the SMOTE.

Table 3.5 Distribution of paid and unpaid supported transactions on the under and over-sampled dataset and its training and validate set.

	Paid	Unpaid	Total
Training data	6000 **50.6%**	5852 **49.4%**	11852
Training set	4817	4636	9453
Validate set	1183	1216	2399

3.4.5. Test set

The test set includes supported transactions made in November 2018, while the original dataset includes from September 2018. The test set is used only to make a final evaluation of how good or bad the predictions are. Any balance technique and has to be applied to the test set. Due to external reasons from the project, in November there were quite fewer supported transactions than in September. 45459 and 74559. However, the unbalanced distribution remains. In order to follow the same logic as the training and validate sets, the Type 2 approximation is also applied to the test set.

Table 3.6 Distribution of paid and unpaid supported transactions on the test set with Type 2 approximation.

	Paid	Unpaid	Total
Test data Type2	44463 **97.8%**	996 **2.2%**	45459

3.5. Algorithm implementation

To get a good interpretation of the results, it is really useful to understand how algorithms create the models. This section presents the most significative characteristics of each model. The characteristics are obtained with the under and over-sampled dataset. It is a good example because it is the most thorough dataset. However, this does not mean that it is the one with the best predictions. That will be determined in CHAPTER 4.

Algorithms are developed in R, using RStudio. Each algorithm is implemented by the pertinent package. The Decision Tree model is implemented wit, the *C5.0* package, described as: Decision Trees and Rule-Based Models package. the

Random Forest model with the *randomForest* package. The Naive Bayes model with the *e1071* package.

3.5.1. Decision Tree

In order to create the Decision Tree model, three boosting interactions are made. Although with the under and over-sampled dataset it is possible to make more than three iterations, with the unbalanced dataset only three are possible because of the high computational capacity it requires. With three iterations, the estimated error on the training set is 0.166. *Table 3.7* presents the error and the tree size in each iteration. As Chapter 2.6 explains, in each iteration a tree is created. Therefore, the size parameter corresponds to the number of nodes at each tree. The final row corresponds to the definitive Decision Tree model on the under and over-sampled dataset.

Table 3.7 Size and errors on each Decision Tree iteration and at the final model.

Iteration	Size	Errors
0	157	1667 (17.6%)
1	49	2778 (29.4%)
2	96	2024 (21.4%)
boost	-	1566 (16.6%)

The Decision Tree algorithm can use the same feature on different nodes. Therefore, it provides which are the most used features and the least. The most used is the V_4, the time at which the supported transaction was made. Following by V_6, the national identification of the sector where the operation was made. The least used feature is the V_2, the amount in euros of the supported transaction. Finally, it is interesting to know that the root feature is also the V_4 feature.

3.5.2. Random Forest

The most significative characteristic of the Random Forest model is the number of generated trees. As Chapter 2.7.2 presents, the number of trees is chosen according to the OOB error. *Fig 3.3* is created by the Random Forest algorithm. The black line illustrates the OOB error against the number of trees. Red and green line represent the error on the paid and unpaid classes respectively. Notice that between 0 to 150 trees, the OOB decreases considerably. However, when the number of trees is larger, the OOB remains stable. Looking at *Fig 3.3* , it can be considered that with 500 trees the OOB is minimum. Therefore, with 500 trees

the paid class has an error of 0.174 and the unpaid class of 0.161. The OOB error is 0.168.

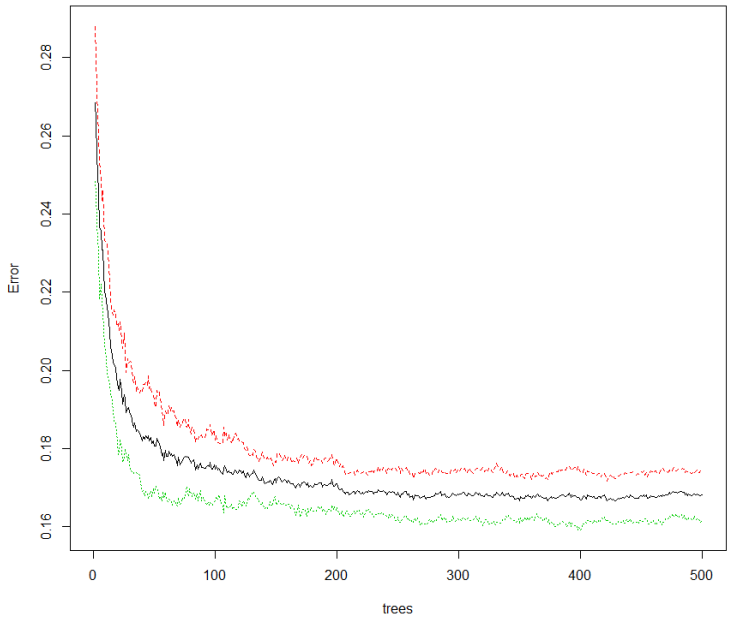

Fig 3.3 Plot of the OOB error (black) and each class error (red and green) in the Random Forest.

3.5.3. Naïve Bayes

The Gaussian distribution plot of the classes feature is one of the most used plots to understand how the Naive Bayes model performs. *Fig 3.4* shows an example. However, due to the distribution of the features, the plot obtained does not make sense, because of that it is not added in this chapter.

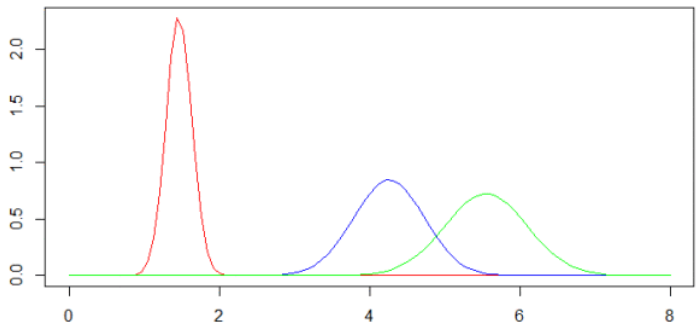

Fig 3.4 Example of a of the Gaussian distribution of a feature with three classes.

Source: https://bicorner.com/2015/07/16/naive-bayes-using-r/

CHAPTER 4 RESULTS

Results are presented in three sections. The first one shows the results from the validate dataset. This section is divided into four subsections. One for each balance technique with a comparison for every predictive model. At the end of this section, two of the best models are chosen in order to do the final test. Section two shows the results from the test set on the two best models. Finally, section three validates that the approximation Type 2 has not distorted the results.

Results are represented by the metrics explained in Chapter 2.8. First, a table showing the numeric metrics is provided. It adds the accuracy, precision, recall and the F_1 score. After that, a plot with the three ROC curves, one for each predictive model is also included. In order to not difficult the reading, all the confusion matrixes are collected on the Appendix 1 CONFUSSION MATRICES.

4.1. Validation results

4.1.1. Dataset (c)

The results obtained from dataset (c) with Type 2 approximation are as expected. Algorithms classify new instances with a tendency to the negative class due to the unbalanced between both classes. The ratio is 98% of paid supported transactions and 2% of unpaid, see *Table 3.2*.

As Chapter 2.8.2 explains, accuracy can be a useless metric dealing with unbalanced datasets. For example, accuracy values are high on Decision Tree model and Random Forest model, see *Table 4.4*. This happens because there is an important amount of supported transactions classified correctly. However, the majority are paid supported transactions, due to the unbalanced dataset. Accuracy do not reflect that there are many unpaid supported transactions predicted badly.

Although accuracy is an interesting metric to be taken into account, it is not decisive in this project. It is better to analyse the F_1 score.

The Decision Tree model makes quite a bad prediction, F_1 score is equal to 0.013, see *Table 4.1*. It only detects 2 unpaid supported transactions from a total of 308, see *Table 6.1* on the appendix. The Random Forest model improves the predictions, compared with the Decision Tree model. 21 unpaid supported transactions are predicted correctly from a total of 308, see *Table 6.2*. Its F_1 score

is 0.125, see *Table 4.1*. Anyway, 0.125 is not good a good result. On the other hand, the Naive Bayes model is able to predict unpaid instances, it has a high recall: 0.763. However, it predicts many paid supported transactions as unpaid, therefore the precision on the Naïve Bayes model is really low: 0.027, see *Table 4.1*. Low precision is something really important to avoid on the prediction of unpaid supported transactions. Every supported transaction predicted as unpaid is traduced as a denial. Therefore, a model with low precision causes many wrong denials. To conclude, the model which predicts better on the dataset (c) is the Random Forest. It has the higher F_1 score: 0.125.

Table 4.1 Accuracy, Precision, Recall and F_1values of each model on the original dataset.

	Decision Tree	Random Forest	Naïve Bayes
Accuracy	0.980	0.980	0.421
Precision	0.500	0.724	0.027
Recall	0.006	0.068	0.763
F_1	0.013	0.125	0.051

As *Fig 4.1* illustrates, the Naive Bayes has the best ROC curve (blue) and therefore the best AUC (0.589), due to the high recall. Decision Tree has an AUC of (0.503) and the Forest Random of (0.534). Although Naive Bayes seems to be the best model on the dataset (c). The fact that it predicts many paid supported transactions as unpaid, makes it not a suitable option.

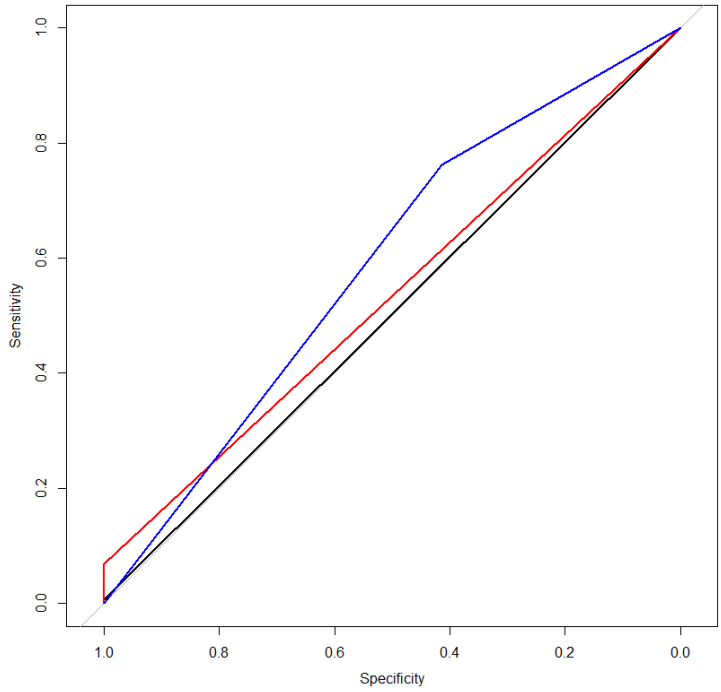

Fig 4.1 ROC curve of Decision Tree (black), Random Forest (red), Naïve Bayes (blue) on dataset (c) (validate set).

4.1.2. Under-sampling

Results on the under-sampled dataset improve quite a lot from the ones on the dataset (c). Models are able to distinguish between classes. The F_1 score of the three models is around 0.60 – 0.68, see *Table 4.2*. The reason of the improvement with respect dataset (c) is that the under-sampling dataset is now balanced, with a ratio of 50% and 50% of each class, see Chapter 3.4.2. In this case, best predictions using the under-sampling technique are obtained by the Random Forest model.

Table 4.2 Accuracy, Precision, Recall and F_1 values of each model on the under-sampled dataset.

	Decision Tree	Random Forest	Naïve Bayes
Accuracy	0.626	0.657	0.638
Precision	0.647	0.645	0.622
Recall	0.573	0.715	0.725
F_1	0.608	0.678	0.670

Fig 4.2 illustrates that the three models have similar ROC curves. However, the Random Forest has the best AUC value (0.656) followed by the Naive Bayes (0.637) and the Decision Tree (0.627).

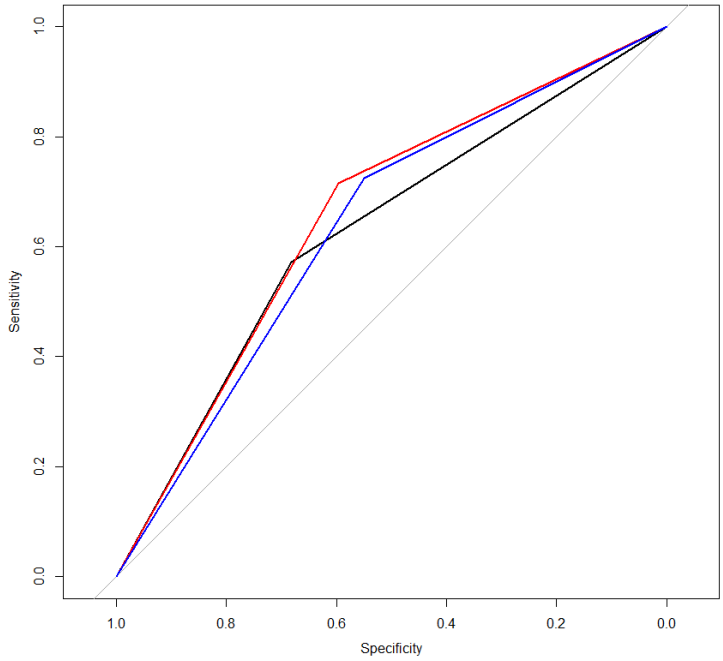

Fig 4.2 ROC curve of Decision Tree (black), Random Forest (red), Naïve Bayes (blue) on under-sampled dataset (validate set).

4.1.3. Over-sampling

Decision Tree and Random Forest predictions on the over-sampled dataset are so good that possibly both models are overfitted. As *Table 4.3* shows, F_1 scores are between 0.9 and 1. Naive Bayes model do not seem to be overfitted. Its F_1 score is 0.573. As results shows, Naïve Bayes model seem to not be modified so much by the over-sampling technique. The assumption that features are distributed by a Gaussian function can be the reason. To conclude, best predictions using the over-sampling technique are obtained by the Decision Tree model.

Table 4.3 Accuracy, Precision, Recall and F_1 values of each model on the over-sampled dataset.

	Decision Tree	Random Forest	Naïve Bayes
Accuracy	0.962	0.930	0.604
Precision	0.960	0.902	0.614
Recall	0.964	0.963	0.573
F_1	0.962	0.931	0.573

Fig 4.3 illustrates that Decision Tree and Random Forest models predict really well, lines black and red. The AUC values are (0.962) and (0.930) respectively. The Naive Bayes AUC is (0.603) much lower.

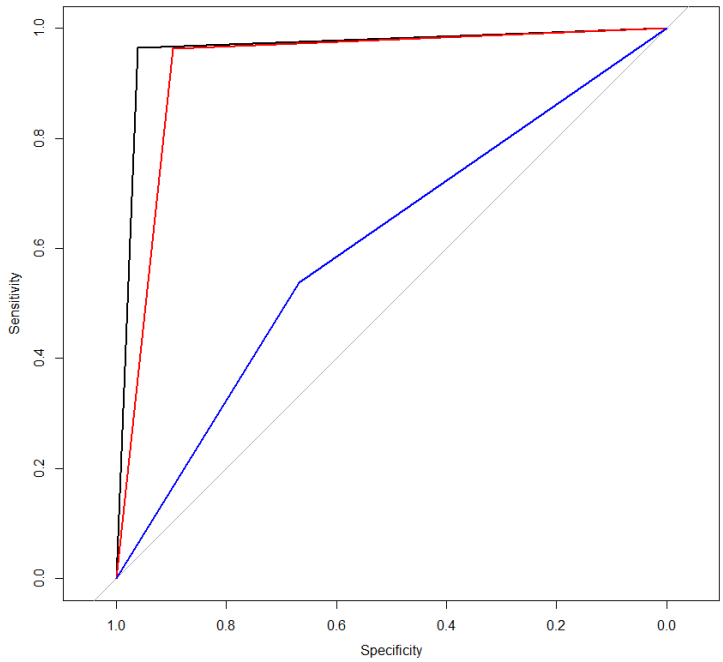

Fig 4.3 ROC curve of Decision Tree (black), Random Forest (red), Naïve Bayes (blue) on over-sampled dataset (validate set).

4.1.4. Over-sampling and under-sampling

When both methods are used on the same dataset, predictions continue being good enough. F_1 score is between 0.8 and 0.85 for the Decision Tree and the Random Forest model. It is not as higher as in the over-sampled dataset. However, that could be good when models are overfitted. The test set will determine if using both methods on the same dataset reduces the problem of overfitting. Naive Bayes predictions do not change from the over-sampled dataset. Notice that its F_1 score is around 0.57 - 0.58 in both cases. Using both balance techniques, Random Forest model has the best predictions.

Table 4.4 Accuracy, Precision, Recall and F_1 values of each model on the under and over-sampled dataset.

	Decision Tree	Random Forest	Naïve Bayes
Accuracy	0.808	0.841	0.632
Precision	0.838	0.839	0.683
Recall	0.770	0.850	0.512
F_1	0.802	0.844	0.586

The ROC curve at *Fig 4.4* illustrates the Random Forest as the best model with an AUC of (0.841). The Decision Tree is quite near, with an AUC of (0.808). Finally, the Naive Bayes with an AUC of (0.634).

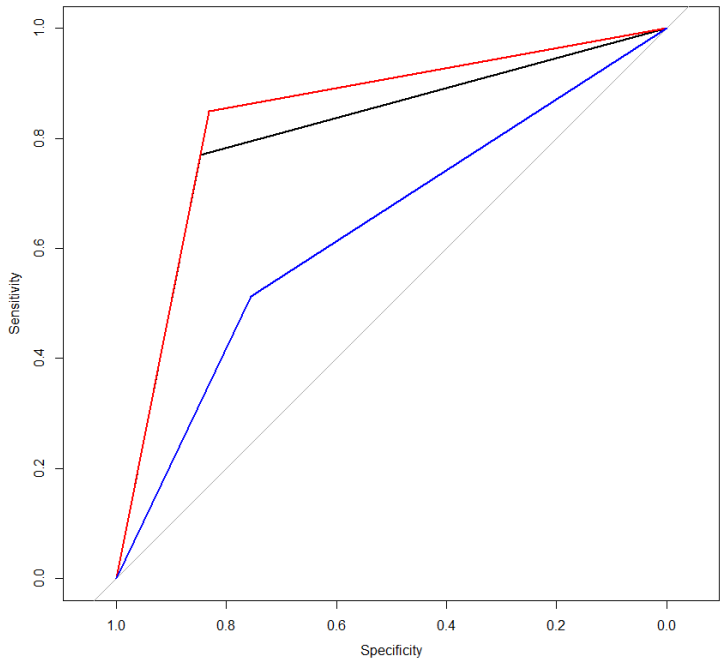

Fig 4.4 ROC curve of Decision Tree (black), Random Forest (red), Naïve Bayes (blue) on under and over-sampled dataset (validate set).

Taking into account the F_1 score, the best model of each dataset is included in *Table 4.5* in order to make a comparation.

Table 4.5 Accuracy, Precision, Recall and F_1 values of the best model at each dataset.

	Dataset (c)	Under-sampled dataset	Over-sampled dataset	Under and Over-sampled dataset
	Random Forest	Random Forest	Decision Tree	Random Forest
Accuracy	0.980	0.657	0.962	0.841
Precision	0.724	0.645	0.960	0.839
Recall	0.068	0.715	0.964	0.850
F_1	0.125	0.678	0.962	0.844

The Random Forest model has the best prediction in ¾ of the datasets. However, the Decision Tree has also good predictions in the over-sampled dataset. Both models will be tested with the test set in order to decide which is the best model.

4.2. Test results

The test set is applied to the two models chose previously: The Decision Tree model trained with the over-sampled dataset and the Random Forest model trained with the under and over-dataset. *Table 4.6* and *Fig 4.5* show that the Random Forest model, with the under and over-sampled technique, has the best predictions on the test set. The F_1 is 0.485. Another important conclusion obtained from test results is that the over-sampled technique causes overfitting. On the validate set, predictions are much better than the ones on the test set.

Table 4.6 Accuracy, Precision, Recall and F_1 values on the test set.

	Decision Tree model on Over-sampled test set	Random Forest model Over and under-sampled test set
Accuracy	0.785	0.973
Precision	0.035	0.418
Recall	0.336	0.576
F_1	0.064	0.485

In *Fig 4.5* the differences between both models are also shown. The Random Forest model on the under and over-sampled data set has an AUC equal to 0.779 while the Decision Tree model on the over-sampled dataset has an AUC equal to 0.565.

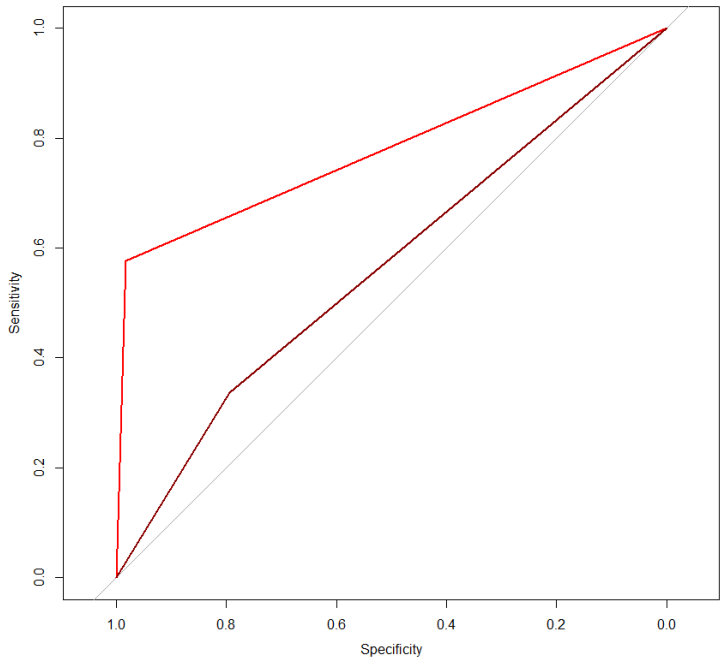

Fig 4.5 ROC curve of Decision Tree model (dark red) and Random Forest model (red).

4.3. Validate the approximation

As Chapter 3.2 explains, the results are made with the Type 2 approximation. In order to make sure that the approximation has not distorted the results, the Random Forest model on the over and under-sampled dataset is applied to the three different types of approximations.

The same steps used in CHAPTER 3 are applied to the dataset (c) with Type 1 approximation, and dataset (c) with Type 3 approximation. First, datasets are balanced with under-sampling and over-sampling techniques. Then, a Random Forest model is trained with each balanced dataset. Finally, tests set test their corresponding model: Test set with Type 1 approximation test the model trained with the dataset (c) with Type 1 approximation, and the same procedure to Type 2 and 3. The results on the tests set are shown at *Fig 4.6*. More details and previous results are in Appendix 2 VALIDATE THE APPROXIMATION.

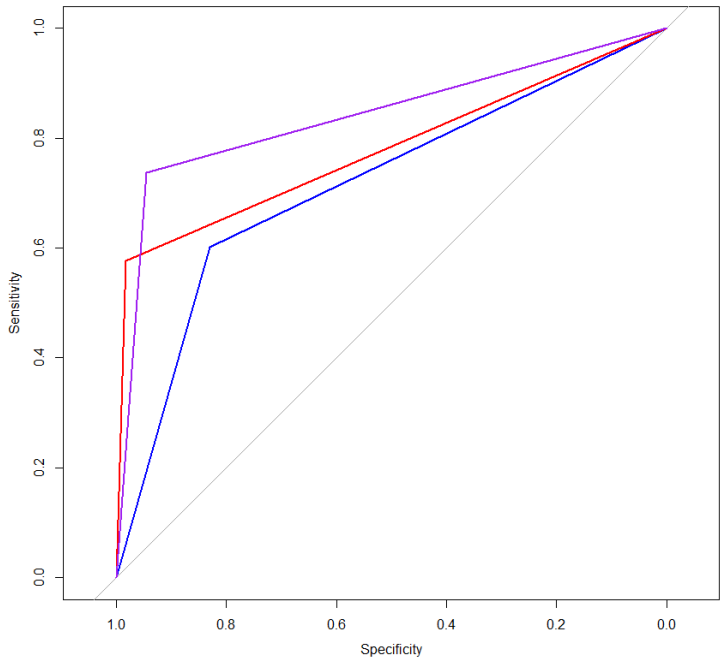

Fig 4.6 ROC curves of Random Forest model for different approximations. Type1: blue, Type2: red and Type3: purple.

ROC curves in *Fig 4.6* show that the approximation does not distort the results. Thee lines are similar. The red line, with an AUC of 0.779, corresponds to the Type 2 approximation. The blue line, with an AUC of 0.715. corresponds to the Type 1 approach. This AUC has the smallest value because the Type 1 approximation only recognizes the 2.1% of supported transactions as unpaid, see Appendix 2 VALIDATE THE APPROXIMATION. The purple line, with an AUC of 0.841, corresponds to the Type 3 approximation, which recognizes the biggest amount of unpaid supported transactions: a 4.8%, see Appendix 2 VALIDATE THE APPROXIMATION.

CHAPTER 5 CONCLUSIONS

All the objectives set at the beginning of the project have been satisfactorily fulfilled. Balancing techniques have greatly improved predictions. Although in some cases they have caused overfitting, the method of combining two opposite techniques (under-sampling and over-sampling) has allowed algorithms to fit the final test. This final test proves that algorithms are ready for new data.

The Random Forest is the model that best fits with supported transactions. The combination of the Random Forest model and the two opposite balancing techniques provides a predictive accuracy of 0.973 and a recall of 0.576. In other words, the financial entity, which is actually detecting 0% of unpaid supported transactions, will be able to detect the 57.6% of all the unpaid supported transactions. Being right in 97.3% of the times. Taking into account that the average amount of money per supported transaction is 25.6€, the financial entity can avoid around 14.000€ as unpaid credit per month. However, a lot of work has to be done before the implementation of the algorithm in the Payment Processor Entity. For example, to evaluate the Random Forest algorithm with metrics that measure different characteristics, like the speed of execution or the scalability.

Related to the other models, the Decision Tree is a model that also fits with supported transactions, but due to its simplicity, predictions are not as good as the ones obtained by the Random Forest model. Naïve Bayes model does not fit with the nature of the data. It performs well in situations where the classes of the target feature are equiprobable, but not in predicting supported transactions.

This project has shown that machine learning models are able to change the way financial entities solve their problems.

Bibliography

[1] Antonio Heredia, “La morosidad de la banca cae en agosto,” *El Mundo*, 2018.

[2] B. Marr, “A Short History of Machine Learning,” *Forbes*, 2016.

[3] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O’Reilly M. 2017.

[4] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” *Proc. 14th Int. Jt. Conf. Artif. Intell.*, vol. 2, no. 0, pp. 1137–1143, 1995.

[5] Charles Zaiontz, “Basic Concepts of Correlation,” 2015. .

[6] N. E. Helwig, “Data, Covariance, and Correlation Matrix,” pp. 1–40, 2017.

[7] N. Japkowicz, “The Class Imbalance Problem: Significance and Strategies,” *Proc. 2000 Int. Conf. Artif. Intell.*, pp. 111--117, 2000.

[8] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE : Synthetic Minority Over-sampling Technique,” vol. 16, pp. 321–357, 2002.

[9] R. E. Schapire, “Explaining adaboost,” *Empir. Inference Festschrift Honor Vladimir N. Vapnik*, pp. 37–52, 2013.

[10] D. J. Koskinen, “Multivariate Method - Boosted Decision Tree,” 2016.

[11] V. L. Clifton Phua, Damminda Alahakoon, “Minority report in fraud detection,” *SIGKDD ACM Spec. Interes. Gr. Knowl. Discov. Data*, vol. 6, no. 1, pp. 50–59, 2004.

[12] C. E. Shannon, “A Mathematical Theory of Communication,” vol. 27, no. 3, pp. 379–423, 1948.

[13] Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang, H. M. · G. J. M. · A. N. · B. L. · P. S. Y. ·, and Z.-H. Z. · M. S. · D. J. H. · D. Steinberg, “Top 10 algorithms in data mining,” *2nd Bienn. Int. Conf. Dyn. Des. 26th Int. Conf. Des. Theory Methodol.*, vol. 14, no. 1, pp. 1–37, 2008.

[14] S. PANG and J. GONG, “C5.0 Classification Algorithm and Application on Individual Credit Evaluation of Banks,” *Syst. Eng. - Theory Pract.*, vol. 29, no. 12, pp. 94–104, 2009.

[15] Tin Kam Ho, “Random Decision Forests,” *Third Int. Conf. Doc. Anal. Recognit.*, vol. 136, no. 1, pp. 4–5, 1995.

[16] Leo Breiman, “Random Forests,” *Springer*, vol. 45, no. 1, pp. 5–32, 2001.

[17] Tin Kam Ho, “The random subspace method for constructing decision

forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 832–844, 1998.

[18] M. A. H. Ian H. Witten, Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java*, Morgan Kau. California, USA, 1999.

[19] L. I. Kuncheva, "On the optimality of Naïve Bayes," *Florida Artif. Intell. Res. Soc. Conf.*, 2004.

[20] P. John, George H.; Langley, "Estimating Continuous Distribution in Bayesian Classifiers," *Uncertain. Artif. Intell.*, pp. 338–345, 1995.

[21] D. J. Hand and R. J. Till, "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems," *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001.

Appendix 1 CONFUSSION MATRICES

This section includes the confusion matrices of the validate and test process, done it in CHAPTER 4. Metrics like accuracy, precision, recall, and the F_1 score are calculated from these tables.

Table 6.1 Confusion matrix of the Decision Tree model on the original dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	14697	306
	Positive	2	2

Table 6.2 Confusion matrix of the Random Forest model on the original dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	14691	287
	Positive	8	21

Table 6.3 Confusion matrix of the Naïve Bayes model on the original dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	6088	73
	Positive	8611	235

Table 6.4 Confusion matrix of the Decision Tree model on the under-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	201	129
	Positive	94	173

Table 6.5 Confusion matrix of the Random Forest model on the under-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	176	86
	Positive	119	216

Table 6.6 Confusion matrix of the Naïve Bayes model on the under-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	162	83
	Positive	133	219

Table 6.7 Confusion matrix of the Decision Tree model on the over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	13920	510
	Positive	575	13710

Table 6.8 Confusion matrix of the Random Forest model on the over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	13003	524
	Positive	1492	13696

Table 6.9 Confusion matrix of the Naïve Bayes model on the over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	9687	6577
	Positive	4808	7643

Table 6.10 Confusion matrix of the Decision Tree model on the under and over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	1002	280
	Positive	181	936

Table 6.11 Confusion matrix of the Random Forest model on the under and over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	984	182
	Positive	199	1034

Table 6.12 Confusion matrix of the Naïve Bayes model on the under and over-sampled dataset (validate set).

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	894	593
	Positive	289	623

Table 6.13 Test set confusion matrix of the over-sampled dataset on a Decision Tree model.

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	35335	661
	Positive	9128	335

Table 6.14 Test set confusion matrix of the over and under-sampled dataset on a Random Forest model.

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	43665	422
	Positive	798	574

Appendix 2 VALIDATE THE APPROXIMATION

To validate that the Type 2 approximation do not distorted the results, the model with the best predictions, the Random Forest, is tested with different approximations. The final results are exposed in Chapter 4.3. Validate the approximation. However, in order to make the reading understandable not all the steps and results are in there. This Appendix 2 shows this complementary information.

7.1. Original datasets (c)

Each type of approximation is applied to dataset (c). *Table 7.1* shows the transactions in dataset (c) with the Type 1 approximation. With this approximation, the dataset has the most unbalanced distribution, only the 1.7% of the transactions are unpaid. *Table 7.2* shows the distribution on dataset (c) with the Type 2 approximation, this is the dataset with a lower error approximation. It is used in CHAPTER 3. Finally, *Table 7.3* represents the distribution on dataset (c) with the Type 3 approximation. This approximation causes the best distribution, the 4.1% of the transactions are unpaid. However, the dataset is unbalanced and balance techniques need to be applied.

Table 7.1 Distribution of paid and unpaid transactions on the dataset (c) using the Type 1 approximation.

	Paid	**Unpaid**	**Total**
Training data Type 2	73262 **98.3%**	1297 **1.7%**	74559

Table 7.2 Distribution of paid and unpaid transactions on the dataset (c) using the Type 2 approximation.

	Paid	**Unpaid**	**Total**
Training data Type 2	73096 **98%**	1463 **2%**	74559

Table 7.3 Distribution of paid and unpaid transactions on the dataset (c) using the Type 3 approximation.

	Paid	Unpaid	Total
Training data Type 2	71465 **95.9%**	3094 **4.1%**	74559

7.2. Under and over-sampled datasets

In order to validate the approximation, it is enough to validate it with the test set. Because of that the validate set is not provided. *Table 7.4, Table 7.5* and *Table 7.6* show the distribution of transaction at each type of approximation.

Table 7.4 Distribution of paid and unpaid transactions on the under and over-sampled dataset using the Type 1 approximation.

	Paid	Unpaid	Total
Training data Type 1	6000 **53.6%**	5188 **46.4%**	11188
Training set	4815	4157	8972

Table 7.5 Distribution of paid and unpaid transactions on the under and over-sampled dataset using the Type 2 approximation.

	Paid	Unpaid	Total
Training data Type 2	6000 **50.6%**	5852 **49.4%**	11852
Training set	4817	4636	9453

Table 7.6 Distribution of paid and unpaid transactions on the under and over-sampled dataset using the Type 3 approximation.

	Paid	Unpaid	Total
Training data Type 3	6000 **50.9%**	5868 **49.1%**	11768
Training set	4819	4692	9511

7.3. Test sets

Table 7.7, Table 7.8 and *Table 7.9* show the distribution of the test set using the different types of approximations.

Table 7.7 Distribution of paid and unpaid transactions on the test set, using the Type 1 approximation.

	Paid	Unpaid	Total
Test data Type 1	44519 **97.9%**	940 **2.1%**	45459

Table 7.8 Distribution of paid and unpaid transactions on the test set, using the Type 2 approximation.

	Paid	Unpaid	Total
Test Type 2	44463 **97.8%**	996 **2.2%**	45459

Table 7.9 Distribution of paid and unpaid transactions on the test set, using the Type 3 approximation.

	Paid	Unpaid	Total
Test Type 3	43289 **92.2%**	2170 **4.8%**	45459

7.4. Results

Table 7.10, Table 7.11 and *Table 7.12* represent the confusion matrices of each test.

Table 7.10 Confusion matrix of the test set with Type 1 approximation. Over and under-sampled technique on the Random Forest model.

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	36965	375
	Positive	7554	565

Table 7.11 Confusion matrix of the test set with Type 2 approximation. Over and under-sampled technique on the Random Forest model.

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	43665	422
	Positive	798	574

Table 7.12 Confusion matrix of the test set with Type 3 approximation. Over and under-sampled technique on the Random Forest model.

		Actual unpaid	
		Negative	Positive
Predicted unpaid	Negative	40916	569
	Positive	2373	1601

From the previous confusion matrices, Accuracy, Precision, Recall and F_1 values are obtained. *Table 7.13* shows them. Type 3 approximation has the best F_1 score, the reason is because there are more transactions labelled as unpaid. Type 1 approximation has the worst F_1 due to the low amount of transactions labelled as unpaid.

Table 7.13 Accuracy, Precision, Recall and F_1 values of each approximation type on the test set.

	Type 1	Type 2	Type 3
Accuracy	0.826	0.973	0.935
Precision	0.070	0.418	0.403
Recall	0.601	0.576	0.738
F_1	0.125	0.485	0.521

The ROC curve is added at Chapter 4.3. Validate the approximation, as *Fig 4.6*. The results illustrated in *Fig 4.6* shows the same idea seen in *Table 7.13*