Mohammed VI Polytechnic University

Data Science and Decision Support

END OF YEAR PROJECT MEMORY

Applied Machine learning's Clasification Models in Bank Loans

---

ABOUHANE Mehdi El ALAOUI Omar ABDENOURI Khaoula

Supervisor: Professor IDRI Ali

July 27, 2020

Acknowledgment

Firstly, we would like to address our sincere thanks to Professor IDRI Ali who kindly supervised during the process and supported us with the necessary knowledge to go along with the study. We are increasingly grateful for his help as without him this work wouldn't be possible.

Also a big thanks to the teaching staff of the AL KHWARIZMI department at the Mohammed VI Polytechnic University for their generosity and their sacrifices during this hard time due to the COVID-19 Pandemic.

Finally, we thank all those who have contributed directly or indirectly to the success of this work. We hope that this work is within the horizon of your expectations as well as the level of the efforts provided.

Abstract

With the magnification of the danger of financial crisis, it has became mandatory to be extreamly careful while making any kind of decision inside financial institution. In this study we will be proposing a solution to better decide when it comes to assigning lawns. This solution is fundamentally built on Machine learning in the sense that we will test various algorithms on a data available on-line to analyse and understand what is the best way to handle the data and how to effectively build a predictive model.

In the following will give an elaborated explanation of how we are going to preprocess the data, a deep idea about the algorithms we will be implementing and the process of evaluating the results.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

ANN	Artificial Neural Network
AUC	Area Under the Curve
AUC	Area Under the Curve
CMV	Continuous Missing Values
FN	False Negative
FP	False Positive
GFC	Global Financial Crisis
KNN	K Nearest Neighbours
ML	Machine Learning
QMV	Quantitative Missing Values
ReLU	Rectified Linear Unit
ROC	Receiver Operator Characteristic
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

General Intoduction

0.1 Contexte and Motivations

Nowadays, one of the most significant sources of income for financial institutions are loans. Whether it be a commercial bank, a state agency or just a private investor, their biggest concern is to ensure the acquisition of their credits. In fact, Bank loans have been widely recognized as the key source of financing in general. Thus, medium and small businesses are constantly facing the same issue of getting the money that will permit them to grow up their company and enable them to better access the capital market. And it happens, that this type of companies are the vital source of dynamism and economic growth in emerging countries[1].

Generally, banking foundations tend to have the cheapest interest rate and provide flexibility to costumers in terms of giving them freedom to use the money as they see fits. That is to attract a large number of lenders and thus give more loans in order to maximize their profit.

However, defaults on loan repayments can conduct a baking crisis, additionally to a potential fall in credit score and increases in interest rates. Banks try to avoid these situations by guaranteeing a proper selecting based on qualifications of the most judicious customers. Yet, this method has proven to be extremely deficient and not reliable especially after the Financial crisis of 2007–08 also known under the name of the Global Financial Crisis (GFC)[2].

Eventually, banks were forecast to hit Billions and Billions of Dollars all over the world. The most legitimate idea to have after such a tragedy will be to reinforce the decision making when reviewing the clients loan application [3]. In the following we will give a breif presentation of the proposed solution and an overview of this report's structure.

0.2 Problematic and Objectives

Over the recent years, Machine Learning has tremendously made upgraded human decisions in various fields.It is a hot topic in research and industry, with new methodologies developed all the time. The speed and complexity of the field makes keeping up with new techniques difficult even for experts and potentially overwhelming for beginners[4]. So, what is exactly Machine learning?

Machine Learning is an artificial intelligence technology that allows computers to learn without having been explicitly programmed for this purpose. To learn and develop, however,

computers need data to analyze and train on. In fact, Big Data is the essence of Machine Learning, and it is the technology that allows to fully exploit the potential of Big Data.

Machine learning allows us to discover patterns and make predictions from data based on statistics, on data mining, on pattern recognition and predictive analytic. This can go by different ways depending on the type of problem. We can between two main types:

- **Supervised Learning** is used to learn models from labeled training data. So it's necessary that we have known data for the output we are trying to predict. That means we try to build a model that makes predictions based on evidence in the presence of uncertainty. To train the model we use a known set of inputs data and known responses to the data. To evaluate the model, we use just a known set of inputs then we compare the new data (outputs) with the known responses.
- **Unsupervised Learning** in this technique, the data has no labels. The machine just looks for whatever patterns it can find. The machine tries to structure and sort the data entered (inputs) according to certain characteristics. This is like letting a dog smell tons of different objects and sorting them into groups with similar smells.

Since we are interested in building a model to help finance experts decide whether it is safe or not to offer a loan to a certain client. We are then going to use a Supervised Learning Algorithm. And as we go deeper into how will Supervised Learning help us reach our aim, it is important to emphasize the difference between a classification problem and a regression problem.

- **Classification** we use classification techniques when we want to predict discrete responses, for example: whether, a consumer is loyal or not, or whether an email is genuine or spam.
- **Regression** we use regression techniques if the nature of our response (outputs) is a real number, such as temperature or the price of a house

As we want the outcome of our leraning model to be if we should garantee the offer of a lender or deny it. We are clearly be in need of a classification model. In the upcomming section we will give more details about the structure that we will be following in this study to successfully build a model that will minimise or prevent the risk of a bank crisis.

0.3 Study Structure

In this study we will be proposing a solution to better decide when it comes to assigning lawns. This solution is fundamentally built on Machine learning in the sense that we will test various algorithms on a data available on-line to analyse and understand what is the best way to handle the data and how to effectively build a predictive model.

In the following will give an elaborated explanation of how we are going to preprocess the data, a deep idea about the algorithms we will be implementing and the process of evaluating the results.

Chapter 1

Background and Related Work

1.1 Classification Models and Optimisation Techniques

1.1.1 Classification Models

i KNN Algorithm

The K-Nearest Neighbours algorithm assumes that similar things exist in the closest proximity. In other words, to make a prediction, the K-NN algorithm will fetch for the K most similar observations to the instance to be determined based on the previously recorded observations present on our dataset.

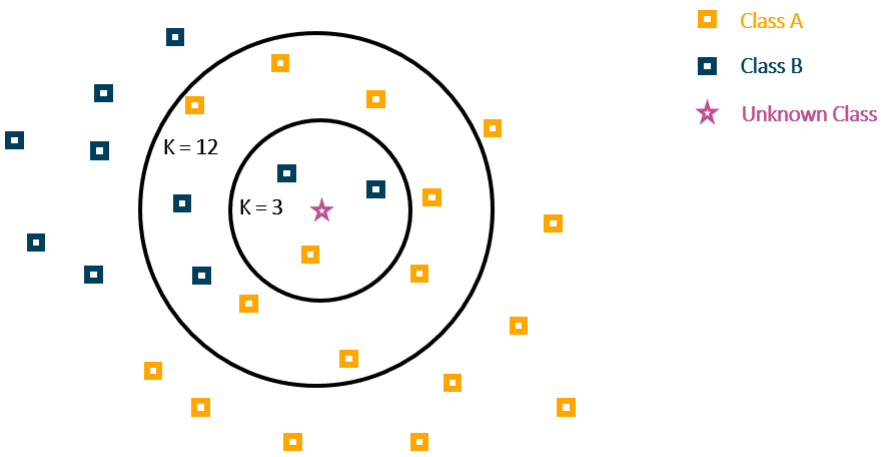

Figure 1.1: Concept of the KNN Algorithm

Naturally, our prediction will only depend on the instances we gathered from the K neighbours. Either using the mean (or median) if the problem we are facing is a regression problem, or if we are dealing with a classification problem then the prediction will be made using the mode.

Algorithm 1 KNN Algorithm

Requirments: Dataset D; Distance Function d; Int K; Observation to be predicted X;

Let L be an array list
for all observation X_i in the dataset D **do**
  Calculate $d(X_i, X)$
  Append L with distance value
end for
Sort the L list
Take the K observations equivalent to the K first distances of L
calculate the modal class of the K observation and return it as the predicted value

As shown in Algorithm 1, in order to find the most resembling observation in our dataset to the one we intend to predict, we need to define the number of neighbours we are taking into consideration and a distancing norm. There are several distancing norms applicable in this algorithm among which we are using: the Euclidean distance, the Manhattan distance and the Minkowski distance defined as follow.

For $X = (x_1, ..., x_n) \in R^n$ and $Y = (y_1, ..., y_n) \in R^n$

- The Euclidean distance

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- The Manhattan distance

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|}$$

- The Minkowski distance of order p

$$d(X, Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

ii Decision Tree Algorithm

The Decision Tree Algorithm creates a training model that predicts the class or value of the target variable by proceeding simple decision rules inferred from the prior data. Its structure is frequently compared to a flowchart structure where to each node we assign a test on a variable, with that, the branches will be equivalent to the test results of the previous node and the leafs will be the class to be attributed. In the other hand, the construction is done by choosing at each node the variable on which the test will be perform.

This choice is done so that the tree generated would be the smallest possible with respect to a list of attribute.

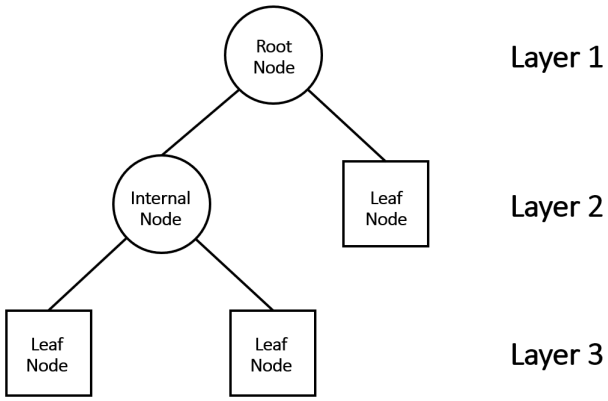

Figure 1.2: The Architecture of a Decision Tree Model

Algorithm 2 Decision Tree Algorithm

Requirments: Dataset D; List of features F; Targeted feature C; Criterion

if K is empty or D is empty **then**
  return "Empty inputs" failure statement
end if
Create a node N
if F is empty **then**
  return N as a leaf node labeled with the majority class in C
end if
if all the observations in D are all of the same class **then**
  return N as a leaf node labeled with the one and only class in C
end if
root = the feature from F that has the maximum Gain according to the Criterion
Set the branches to the various attributes of the feature in the root
for all b in the sets of branches **do**
  updte F by substructing the feature used as a root
  if F is empty **then**
    return the tree
  else
    D' = the observations of D that fales in the condition of the brance b
    apply the decition tree on the D' data to go ferther in the branche
  end if
end for

If we want to compute a Decision Tree Algorithm it is important to specify a maximum length for our Tree and a criterion. Whether it is the Gini Impurity or the Entropy, both criterion have the same task, that is to calculate the purity of a split in a Decision Tree.

Given a sample of a subset, let p_+ the percentage of positive values and p_- the percentage of negative values

- Entropy

$$H = -p_+ log(p_+) - p_- log(p_-)$$

- Gini Impurity

$$GI = 1 - (p_+ + p_-)$$

iii SVM Algorithm

The Support Vector Machine's goal is to define the hyper-plans in an n-dimensional space that split the observations in our data into categories representing the existing classes. Then by maximising the distance between the hyperplane and the nearest observation of each class, we can distinguish the one that splits our data the best. In the further predictions we simply find the category where our observation falls in the n-dimensional space and assign to it the equivalent class. Note that even thought it is widely used in classification challenges, it is also useful in regression problems.

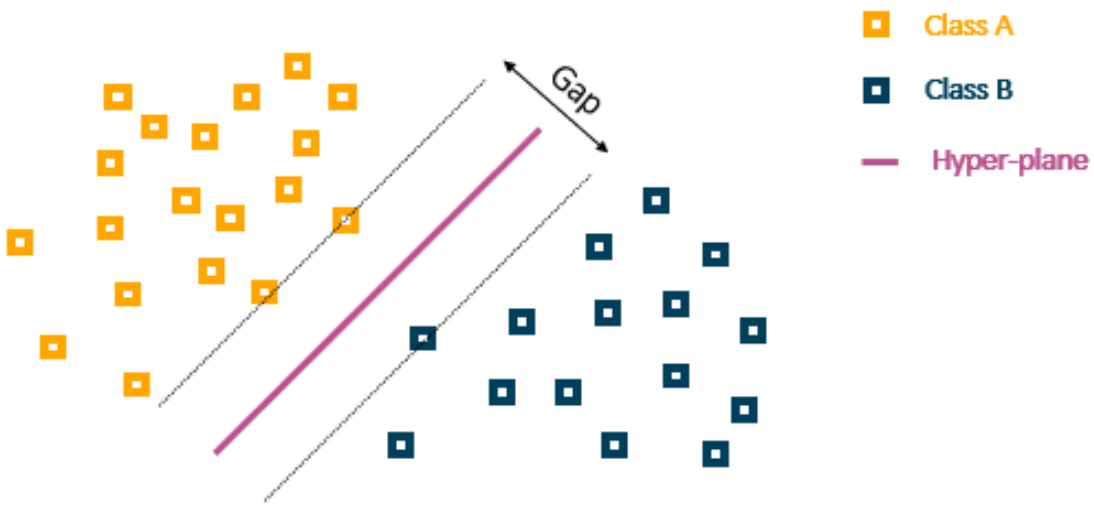

Figure 1.3: SVM's Hyper-plane Visualisation

Algorithm 3 SVM Algorithm

Requirments: Dataset D;

while there is improvment in the optimization **do**
  Select two multipliers w_1 and w_2
  Construct the hyperplane
  Optimize the hyperplane using only the two multipliers
end while

Optimizing SVM's hyper plan has to do with two important variables:

- Gamma : which describes how far the influence of a single training example reaches.
- Cost : that controls trade-off between a smooth decision boundary and classifying training points correctly.

iv Ensemble learning (XGboost):

Ensemble methods combines several trees based algorithms to construct better predictive performance than a single tree based algorithm. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner, thus increasing the accuracy of the model. When we try to predict the target variable using any machine learning technique, the main causes of difference in actual and predicted values are noise, variance, and bias. Ensemble helps to reduce these factors. Ensemble methods are divided into two groups:

- **Sequential Methods:** Base learners are generated sequentially. They exploit dependence between base learners.(Examples: Adaboost, XGboost)
- **Parallel Methods:** Base learners are generated in parallel. They exploit independence between base learners. (Examples: Random Forest, Bagged Decision Trees, Extra Trees)

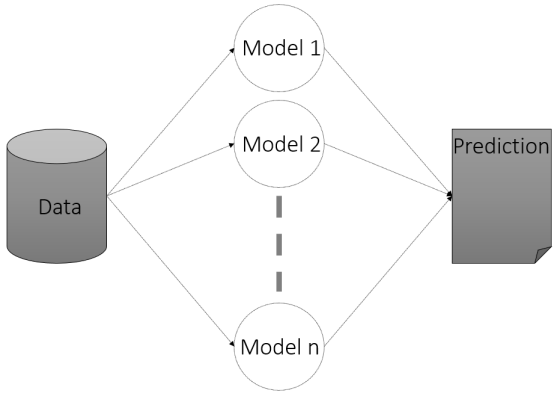

Figure 1.4: Ensemble Learning Technique

Now if we were to talk about XGBoost as an example of Ensemble Sequential methods, XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

That goes by specifying the number of boosting rounds which we often refer to as the number of gradient boosted trees as well as the maximum depth of the trees.

Figure 1.5: XGBoost Structure

v Neural Network Algorithm

Also known as the Artificial Neural Network (ANN), the Neural Network Algorithm is the result of a straight forward inspiration of the human brain. This Algorithm particularly stand out because it flexible as it the information goes throughout the model. That goes by the adjusting the weight of each connection in order to have a more accurate prediction with respect to a certain learning rate.

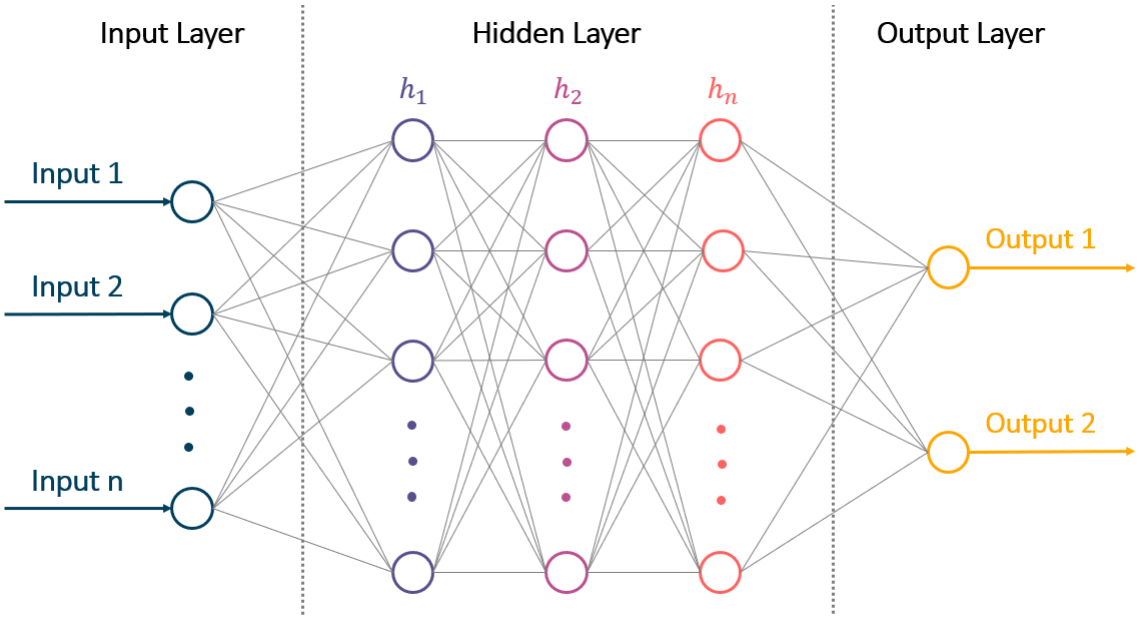

Figure 1.6: The Artificial Neural Network's Architecture

Before starting the algorithm it is important to point out that the Artificial Neural Network performance depends on the batch size we are using and the optimization technique whether it be an Adam approach (adaptive moment estimation) or an Rmsprop approach (root mean square propagation)

Algorithm 4 ANN Algorithm

Requirments: Dataset D;Learning rate L

Construct the network
assign a weight for each input in the network
Sum all the weights assigned
Give the sum to go through the activation function

1.1.2 Pre-processing Techniques

It is also mandatory to think about preprocessing a certain data before applying any of the previously emphasized models. The main reason is gross data contains various anomalies that may be the reason of a poorly performing learning processes if not a defective one. In the following we will be reviewing the main tools to successfully pre-process data sets.

i Feature Selection

Having a model that learns how to extract knowledge based on numerus features, the features must be chosen wisely. If irrelevant features are included in our study, we risk losing in accuracy and even produce an overfitted predictive model, and obviously working with many variables increases algorithm complexity and thus the model will be time consuming. In order to avoid that we can use the following filters.

- **Basic filter: Constant features**
  Features that shows mostly one value over all the observations supply our model with no information. It is recommended to drop such features.

- **Basic filter: Duplicated features**
  As they give a duplicated information to the models, there is no mean in keeping both copies.

- **Correlation filter**
  Correlation describes the linear relationship between two entities. A high correlation between two features means that we can constructively predict one from the other. They then provide the same redundant understanding in reference to the target. Again, one of them should be dropped from the data.

- **Univariate ROC-AUC filter**
  ROC-AUC filter is used to measure the dependencies of two features. It proceeds first by building a Decision Tree to combine features together with the target. For each combination we construct a Receiver Operator Characteristic graph (ROC) which is simply a

probability curve. We could then compare the combination based on the Area Under the ROC Curve (AUC) and only select the features with the highest ROC-AUC score.

ii Discretization

When dealing with continuous variable, it is hard to extract a meaningful interpretation out of the feature due to its infinite value and essentially because of outliers. We often remedy to this problem by discretizing the variable. Note that discretization can be done in different ways.

- **Equal-Width / Equal-Frequency**
  As its name clearly state, in these methods we will discretize either according to intervals of values with a constant width or according to values that falls in the same range of frequency.

- **K-Means**
  K-Means is initially a clustering algorithm that partition the impute values into k different non-overlapping groups with respect that the values in each group must be as similar as possible and the mean value of each group as far as possible.

iii Missing Values Imputation

Missing values are one of the most frequently encountered problems in the preprocessing stage. We find data scientist puzzled between deleting the missing values and work with a short but accurate data set or imputing them to have a large data set but less accurate. The highly used imputation techniques are using the mode, mean or median for continuous variables and the mode for categorical variables.

iv Balancing the Data

If we are working on a classification problem, it is very essential to make sure there is no risk to build a bias model that performs poorly at predicting a specific class. The key to avoid this problem is balancing or sampling the given data in other words each class must have the same number of samples. There are two main way to do that,

- **Smote**
  Synthetic Minority Over Sampling technique relies on creating artificial samples for the minority classed based on the already existing ones so as to increase its number to match the number of samples of the majority class.

- **Tomek Links**
  In the other way around, Tomek Links is an under-sampling technique that gets rid of samples from the majority class until its number matches the number of samples in the minority class. That is done by dropping the samples that are the very similar to a certain sample from the minority class.

v Feature Scaling

It is always better to work with standardized features with a fixed range of variation. This allows us to avoid working with highly varying values that threaten to bias our model. There are 3 main ways to do that. The Z-Score normalization serves as an output a data that has a 0 mean and a standard deviation of 1, following this formula:

$$Z = \frac{X - mean(X)}{stdev(X)}$$

The Min-Max normalization re-scale the values of a feature to fit in an interval between 0 and 1 by using this formula:

$$Z = \frac{X - min(X)}{max(X) - min(X)}$$

Lastly the Logistic normalization also re-scale the values of a feature to fit in [0,1] but using a logistic function. This is simply done using this transformation:

$$Z = \frac{1}{1 + exp(-X)}$$

vi Feature Encoding

One characteristic of machine learning algorithm is that can only read numerical data. So we should think about encoding our categorical features into numerical values. We will explicitly explain two ways to do this:

- **Label Encoding**
  If we want to apply a label encoder on a binary class feature, the result will still be a similarly structured feature but labeling the first class as 0 and the second class as 1. But if we are working on a multi class feature, the label encoder creates a different value for each class going from 0 to n (with n is the number of classes for a given feature). In this case we notice that a natural ordering is created among each feature, which is the reason why we avoid using label encoding for multi class features.

- **One Hot Encoding**
  In the other hand, One Hot Encoding appears to be very effectively used with multi class features since it creates a new feature for each class in the variable and it assigns to the feature the value 1 if it is present in a the observation and the value 0 if not. Note that the returning value of one hot encoding is a numpy array and not a dataframe.

1.1.3 Tools

1. **Python in Jupiter and PyCharm**
   Being one of the simplest programming languages, Python gained popularity among the Data Science community for the ecosystem of libraries and visualising tools that enables users to focus on more important issues rather than wast time on some technical code. It is often used along with the web application Jupiter that allows programmers to create notebooks to make the code easier to read with included markdown. We will also be using The integrated development environment PyCharm that contains a graphical debugger and allows the management of unit tests.

2. **Google Cloud Plateform**
   (GCP) is a cloud computing platform that offering hosting on the same infrastructure that Google uses internally for products such as its search engine. Cloud Platform provides developers with products to build a range of programs from simple websites to complex applications. It also offers modular cloud-based services, such as information storage, calculation, translation and forecasting applications.

3. **Git and Github**
   Git is a decentralized version management software. And linked to it is a web hosting and software development management service called GitHub. These tools are simply used for collaboration purposes.

1.2 Literature review on Bank Loans Classification

In recent years, the retrospective assessment of credit risk has strongly testified for the importance of this issue in the banking as well as the financial institutions. For that offering a loan is not in any mean a light decision to take. Since credit risk is strongly related to the solvency of a borrower, or in other words the probability for a creditor to get his money back, the first tool used as support for their judgment was standard analytics along with statistics methods. Yet the tremendous amount of data generated in this at such a high speed has favorized using Machine learning for this problem.

To be more precise we are facing a classification problem. And over the previous studies on our data, among the models built to efficiently predict if a client is more likely going to fail to reimburse a certain loan, the most straight forward simply consists of imputing the missing values with the mode if the variable is categorical or using the mean for nominal variables to finally apply the Logistic Regression to train the model. The efficiency of the model was measured using the accuracy only with a score of 80%.[12]

However in a different approach, for simplicity's sake, a dimensionality reduction was done followed by the application of two training models: the Random forest Model and the Logistic Regression. At last, using the accuracy parameter, comparing the two models led us to the conclusion that the logistic regression is a better fit to this type of problems but this time with an accuracy level of 82%.[13]

Similarly, an additional study applied 5 different models in the purpose of evaluating the most suitable model. Respectively the Logistic Regression, the K-Nearest Neighbors Classification, the Support Vector Machine, the Naïve Bayes and the Random Forest Classification has scored the following accuracy rates 81%, 79%, 41%, 80% and 81%. Proving another time that indeed the Logistic Regression.[14]

Accordingly, we will try to conduct a work so that we can extract the best training algorithm to perform on this data so that we score the maximum of precision while predicting the loan state of each borrower. But unlike the previous studies we will also try to alternate between various preprocessing technics as well as evaluating the data according to various criteria. Our view of the problem will then be wider, and our understanding of the solution will be clearer. Only then will we be able to conclude the best use of each model.

Chapter 2

Experimental Design

2.1 Loan Data Description

2.1.1 Context

This dataset, as imported from Kaggle Website[11], displays details about previous loaning experiences including customers personal information and whether they have been able to reimburse their loan or not. In this frame of reference, banker's and creditor's ultimate goal is to minimize the risk of loan payment failure and ensure that their customers eventually pay off all their debts by the end of the terms specified. So, based on this dataset we will attempt build a model that will predict if a certain borrower will be able to successfully clear up his debts in time.

This dataset initially consists of 100514 observations and 17 features.

2.1.2 Features description

Current Loan Amount: Amount of money requested by the borrower.

Term: Granted time in which the loan should be reimbursed (short term or long term).

Credit Score: Numerical expression of the total credit amount.

Annual Income: Annual Income of the borrower.

Years in Current Job: Number of years the borrower spent in his current job.

Home Ownership: If the borrower is owning the house, renting it or having a mortgage on it.

Purpose: Purpose of the loan request (Medical Bills, Wedding, Educational, ...).

Monthly Debt: The borrower's total monthly debts including credit cards expenses, loans and mortgages.

Years of Credit History: Records of a borrower's ability to repay his debts.

Months Since Last Delinquent: Date of last payment: Credit accounts that were paid over 30 late days from the due date.

Number of Open Accounts: Number of accounts the borrower has opened.

Number of Credit Problems: Number of loan problems encountered by the borrower.

Current Credit Balance: The amount of money that the borrower owes to his credit card company.

Maximum Open Credit: Maximum limit allowed to be accessed repeatedly by the borrower.

Bankruptcies: Number of bankruptcies experienced by the borrower.

Tax Lien: Legal claims against the borrower when failing to pay his taxes.

Loan Status: The dependent variable that specifies if the borrower did pay off his debt or not (Fully Paid and Charged Off).

2.1.3 Statistical Description and overview of some features

Before getting started with our model, it is always a good idea to take a look at the statistical discription of the numerical features. The dataset contains 12 numerical variables and the table 2.1 shows basic statistic description of each one of them.

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
Current Loan Amount	100000	11760450	31783940	10802	179652	312246	5249420	100000000
Credit Score	80846	1076.4	1475.4	585	705	724	741	7510
Annual Income	80846	1378277	1081360	76627	848844	1174162	1650663	165557400
Monthly Debt	100000	18472.4	12174.9	0	10214.1	16220.3	24012.05	435843.2
Years of Credit History	100000	18.1	7.01	3.6	13.5	16.9	21.7	70.5
Months Since Last Delinquent	46859	34.9	21.9	0	16	32	51	176
Number of Open Accounts	100000	11.1	5	0	8	10	14	76
Number of Credit Problems	100000	0.16	0.48	0	0	0	0	15
Current Credit Balance	100000	294637	376170	0	112670	209817	367958	32878970
Maximum Open Credit	99998	760798	8384503	0	273438	467874	782958	1539738000
Bankruptcies	99796	0.11	0.35	0	0	0	0	7
Tax Lien	99990	0.02	0.25	0	0	0	0	15

Table 2.1: Numerical variables description

And to give a better insight of the content of the categorical features in our data, we will go through the following plot figures of each one of them.

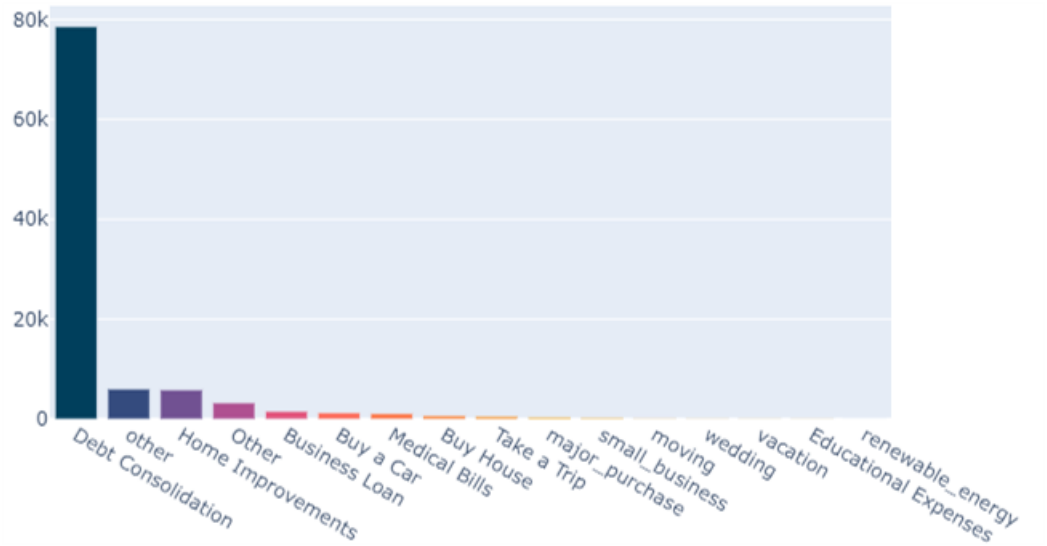

Figure 2.1: Values Occurrence in the "Purpose" feature

The figure above 2.1 show us the distribution of categorical features. We can notice that certain values do not occur as much as the others. It is the case for the "Purpose" feature and other features in our data.

It is even more dominant in the 'Tax Liens' feature where looking at the figure 2.2 we can even consider taking the variable as a constant and with that dropping it would be a good decision to take.

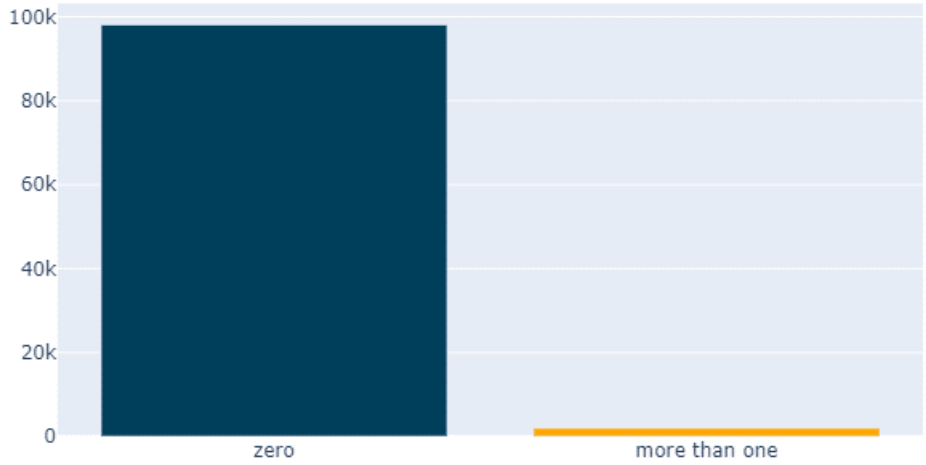

Figure 2.2: Values Occurrence in the "Tax Liens" feature

In the following we will focus on how to solve these problems and get our data ready to be affected the some learning algorithms.

2.2 Experimental Process

Now that we are familiar with our data, let's dive in some hands-on manipulations to extract the knowledge from the abundant information we have. In Machine Learning, whenever we come across a similar task we tend to immediately start thinking "What model may be the best fit for this data? What is the best way to preprocess my data and preserve the integrity of the information?". In the following we will try to perform various combinations of techniques that will get us from a raw data to a predictive model.

2.2.1 Preprocessing phase

In the sense of getting our data ready to go through a learning algorithm, we will display in this step for each issue to be solved the different solutions we are going to work on.

(i) Feature selection

Originally we have 17 features in our data which is relatively a lot of features. If we decide to carry on with all the variables we may end up supplying our model with redundant information.So, assuming we have related features in the raw data, we will be able to identify the ones we don't need in the learning.
To measure how much redundancy we could find between two variables we could use the Correlation parameter and by that dropping the feature that is highly correlated with some other feature.

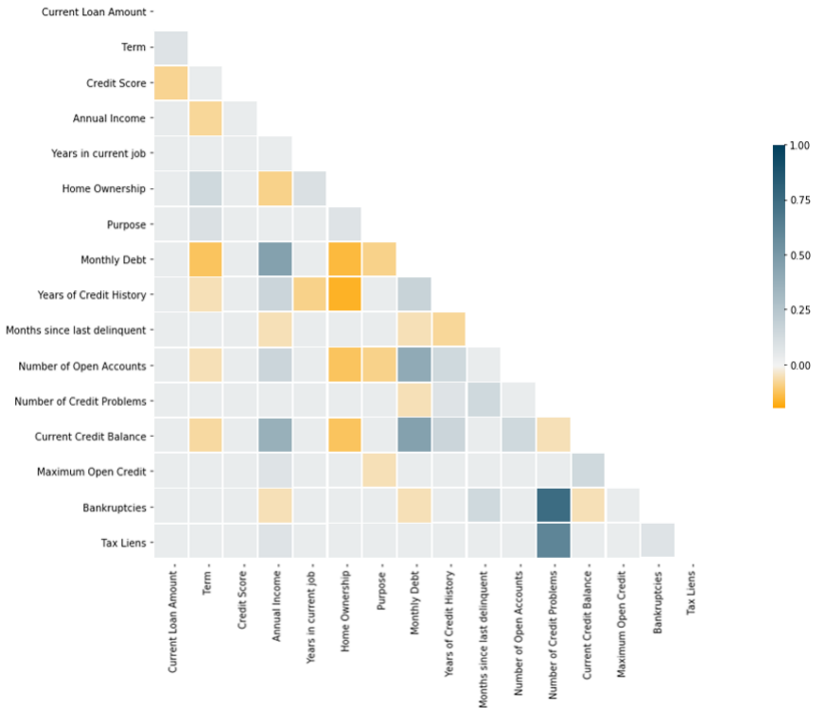

Figure 2.3: Correlation heat-map

Or we could use the Univariate ROC-AUC filter, in other means we will get rid of the features that has high ROC-AUC mean score. The table 2.2 displays the score of each feature.

Feature	**Mean ROC-AUC score**
Credit Score	0.694301
Term	0.571231
Current Loan Amount	0.568701
Home Ownership	0.534777
Annual Income	0.525802
Monthly Debt	0.524146
Maximum Open Credit	0.522025
Years of Credit History	0.520035
Current Credit Balance	0.518029
Years in current job	0.514153
Purpose	0.506554
Number of Open Accounts	0.505533
Bankruptcies	0.501616
Number of Credit Problems	0.496330
Tax Liens	0.481126
Months since last delinquent	0.360871

Table 2.2: ROC-AUC Mean score for each feature

In both cases, we will have to apply more filters to fetch for constant features as they supply us with no additional understanding of the problem.
After applying the feature selection, we will have 2 versions of the original data, one that suggests that Correlation filter is the right filter to choose and the other that Univariate ROC-AUC filter performs better in this case.

(ii) Discretization of some quantitative features

As we previously noticed, some of the features appear to have an excessive number of classes. We will take the "Purpose" (2.1) feature as an example and we will perform similarly on features that shows the same problem.

Values	**Occurrences**	**Percentage**
Debt Consolidation	78552	78.55%
other	6037	6.03%
Home Improvements	5839	5.83%
Other	3250	3.25%
Business Loan	1569	1.56%
Buy a Car	1265	1.26%
Medical Bills	1127	1.13%
Buy House	678	0.68%
Take a Trip	573	0.57%
major-purchase	352	0.35%
small-business	283	0.28%
moving	150	0.15%
wedding	115	0.11%
vacation	101	0.10%
Educational Expenses	99	0.09%
renewable-energy	10	0.01%

Table 2.3: The raw "Purpose" feature values count

Thus to consolidate our information and to avoid miss leading our algorithm, it is recommended to discretize our features to have fewer classes.

Values	Occurrences	Percentage
Debt Consolidation	78552	78.55%
Home Improvements	5839	5.83%
Other	15609	15.61%

Table 2.4: The discretized "Purpose" feature values count

(iii) Missing values imputation

The first thing that catches our eyes on the table 2.5 bellow is that most features contain exactly 514 missing values. A simple reflection can lead us to the conclusion that our data contain empty rows that needs to be dropped.

Feature	Missing values	Percentage
Loan Status	514	0.5%
Current Loan Amount	514	0.5%
Term	514	0.5%
Credit Score	19668	19.6%
Annual Income	19668	19.6%
Years in Current Job	4736	4.7%
Home Ownership	514	0.5%
Purpose	514	0.5%
Monthly Debt	514	0.5%
Years of Credit History	514	0.5%
Months Since Last Delinquent	53655	53.4%
Number of Open Accounts	514	0.5%
Number of Credit Problems	514	0.5%
Current Credit Balance	514	0.5%
Maximum Open Credit	516	0.5%
Bankruptcies	718	0.7%

Table 2.5: Missing Values Description

Besides, the "Credit Score", "Annual Income" and "Years in Current Job" features contain a reasonable amount of missing values unlike the "Mounts Since Last Delinquent" which we will drop.
Now we need to impute the missing values still present in our data. for that we must first distinguish between the quantitative and continuous features as we may not deal with the missing values similarly for both.

- Qualitative missing values imputation:
  Taking all the sets of data that we have previously generated (two sets) we will proceed with the imputation of the qualitative missing values with two different methods: either we delete the missing values or impute them with the mode. Once again we generate two versions of the already existing sets to end up with four sets in total.

- Continuous missing values imputation:
  This time with the continuous features, we will either delete their missing values or impute them with the median or the mean. And with these three solutions and the four sets we gathered previously we now have twelve sets as outcome.

(vi) Feature Encoding

Given that this data has to be interpreted by the computer and as the computer have no clues regarding how to manipulate strings, we will have to encode our features And we will be doing that using One Hot Encoding.

(v) Balancing

Taking a look at our target feature we notice that the distribution of classes is not well balanced. As represented in figure (2.4) the "Charged Off" is only covering 22.63% of the instances in the data.

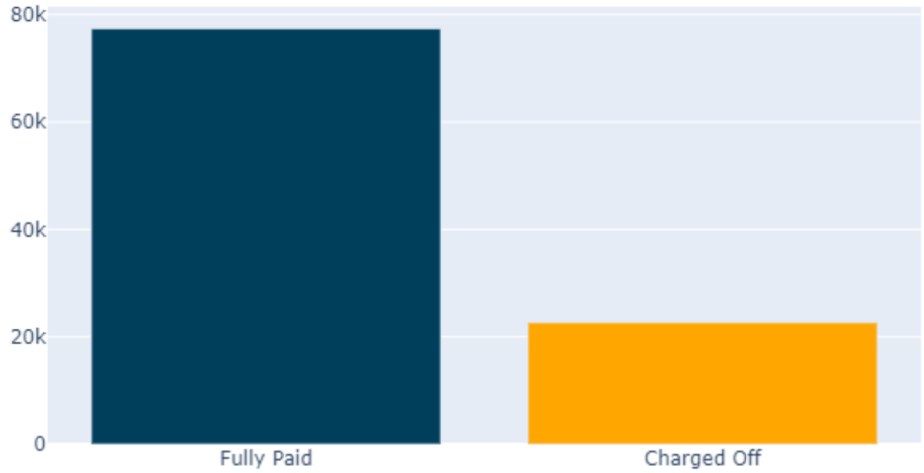

Figure 2.4: Values Occurrence in the "Loan Status" variable

To overcome this situation we will apply the Smote over sampling techniques and the Tomek Links under Sampling techniques. And with that multiplying the number of sets we had by two.

(vi) Normalization

Having our numerical variables scaled is important if we want our model not to be bias. We will proceed to the normalization consequently using the Z-Score, the Min-Max and the Logistic normalization. But as there is some algorithms that are not affected by the lack of normalization in some variables as in the Decision tree and XGBoost algorithms, we could as well not normalize the data at all.

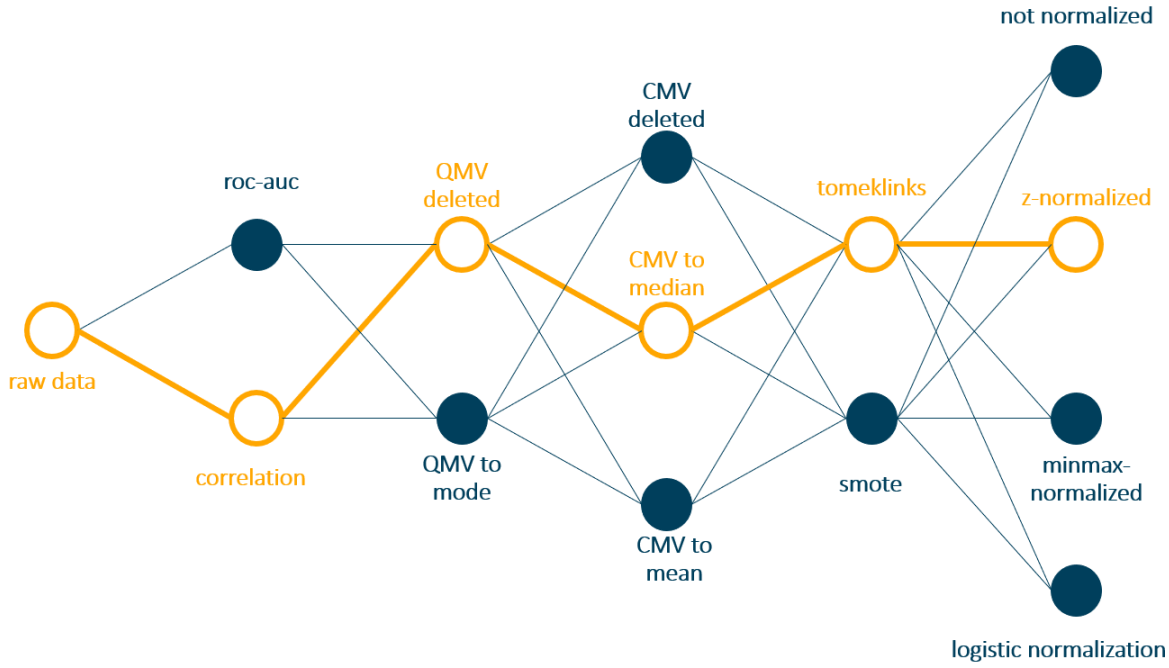

Figure 2.5: Summary diagram of the preprocessing phase

To summarize all the configurations we have until now we could use the diagram in the figure (2.5) below that contains 96 configurations in total. To better see what a configuration looks like in this neural network like diagram, we highlighted the path taken to get a preprocessed data by using the correlation filter to select the features than deleting the quantitative missing values and replacing the contains missing values by the median, applying a tomeklinks under sampling and finally z-normalizing the features.

2.2.2 Learning phase

Now that we got our initial data preprocessed into different ways. We can now think of the learning algorithm we are going to apply.

- **KNN**

The K nearest neighbours algorithm is very sensitive to the normalisation. For that we will only go for this algorithm if the data is normalized. The parameters to be tuned that are the numbers of neighbours and the metric will take values respectively from $[2, 8]$ and $\{euclidean, manhattan, minkowski\}$.

- **SVM**

The support vector machine also require a normalized data. this algorithm too will not be applied to the not normalized set of data. And again the cost parameter will take values from $\{0.1, 1, 10, 100\}$ and the gamma parameter from $\{1, 0.1, 0.01, 0.001\}$.

- **ANN**

The artificial neural network is another algorithm that needs to be performed on a normalized data. We will apply it using a batch size in $\{128, 256, 512, 800\}$ and an optimization technique from $\{adam, rmsprop\}$ knowing that the network will be constructed of one hidden layer. the input layer will contain twenty neurons and the hidden one ten neurons and will both have Relu as the activation function. Meanwhile the output layer will contain one neuron since we are working on a binary classification and will have the sigmoid function as the activation function. We will also set the epochs to two-hundred with an early stopping if the model is not making a significant improvement to avoid overfitting.

- **XGboost**

As it is not affected by weather the features are normalized, the XGBoost can be applied to the not normalized data by setting the number of gradient boosted trees in $\{100, 300, 500, 800\}$ and the maximum depth in $[5, 14]$

- **Decision Tree**

Joining the row of algorithms that do not necessarily need the data to be normalized, the decision tree will be applied with a criterion in $\{gini, entropy\}$ and the maximum depth between 4 and 17 or setting it as "None".

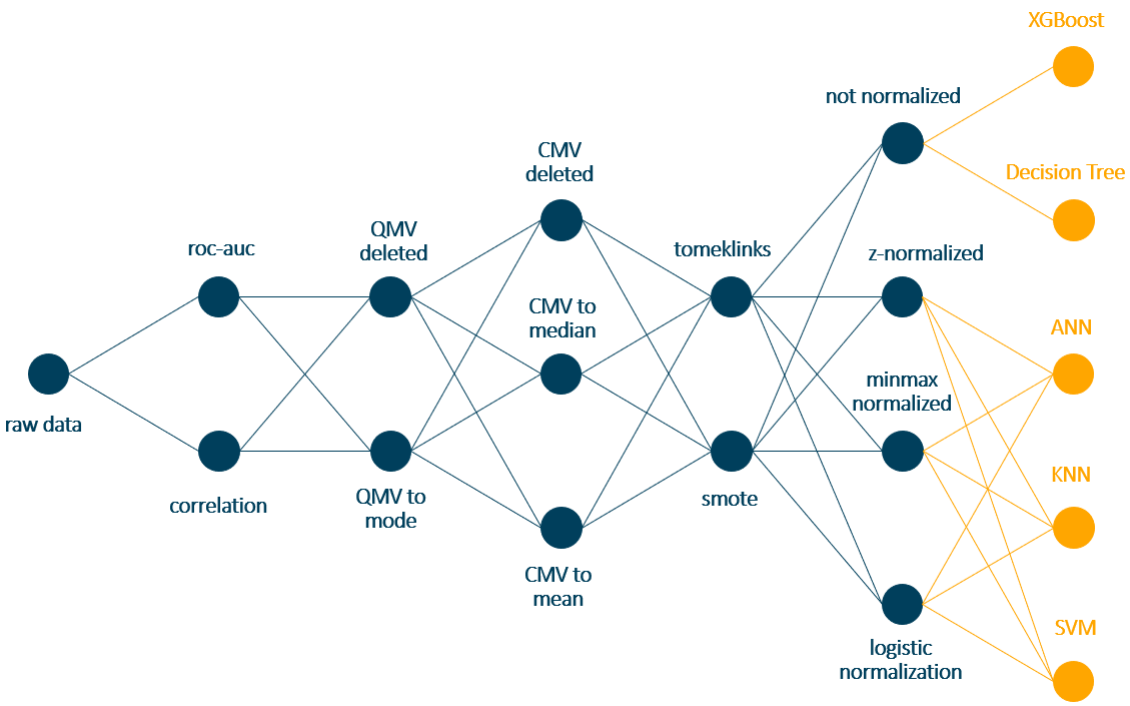

Figure 2.6: Summary diagram of the experimental process

2.3 Performance Criteria

After executing these massive computations we expect to get relatively massive outcomes as we will collect the tuning result for each of the two hundred sixty-four (264) configurations. Eventually we will have to choose among the configurations the one that presents the best results and with which hyperparameters. In the upcoming section we will discover how to select the best configuration.

2.3.1 Grid search

In our path to choose the best way to extract knowledge of our data, we will first try to find for each of the two hundred sixty-four configuration the best hyperparameters to select. We could, for example, optimise our hyperparameters manually, but we are dealing with a two dimensional problem and manual selection a value for each parameter at a time may not work. Alternatively, we can choose a set of possible values for both variables. Then we will create a nested for loop to try all the combinations for the parameters.

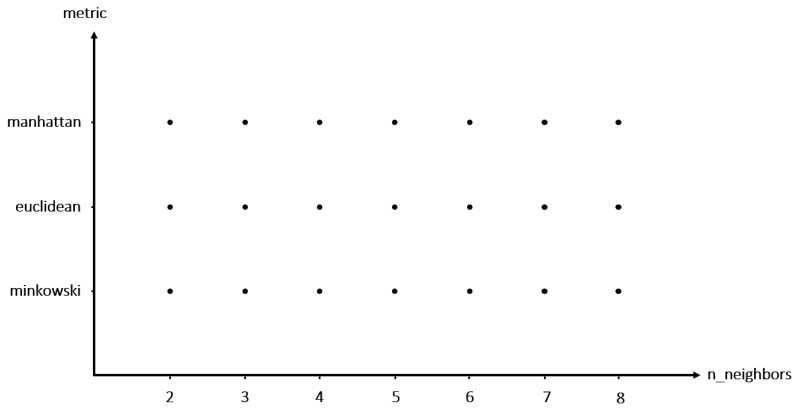

Figure 2.7: Grid of the KNN hyperparameters combinations

This method is called Grid search as we go through all the combinations in the figure (2.7) to search for the best combination
The question to be asked now is how are we going to decide which of the combinations is better? based on what scale of measurement?

2.3.2 Performance measure

To assess how well our model is working we can evaluate its performance on our testing split of the data. That goes by enumerating the number of "Charged off" loans that were indeed predicted to be "Charged off" (True Positive) and the ones that were not (False negative), the number of "Fully paid" loans that were predicted to be "Fully paid" (True negative) and the ones that were not (False positive). We can organize these numbers in a matrix that we call a

confusion matrix.

	Predicted as "Charged off"	Predicted as "Fully paid"
Truly "Charged off"	**TP**	**FN**
Truly "Fully paid"	**FP**	**TN**

Table 2.6: Confusion matrix structure

Among the useful scores we could measure based on the confusion matrix are the following:

- **Precision:** the probability that the prediction is correct when the programme clams that a loan is Charged off.
$$Precision = \frac{TP}{TP + FP}$$
- **Recall:** the probability to correctly predict the state of a loan if it is actually charged off.
$$Recall = \frac{TP}{TP + FN}$$
- **Accuracy:** the probability to correctly predict the state of any given loan.
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Borda Count

Note that when evaluating a model, or comparing it to another one we can notice that according to the Precision score the model is doing a good job on the testing set. But if we take a look at the Accuracy we will conclude that it is rather performing poorly. It is then unfair for us to judge a model according to just one of these three scores.

One way to solve this problem is using a voting system called **Borda Count** that provides a ranking according to different criteria each with a certain degree of importance. To better understand how it works we will apply it in our context.

$$Borda\_count\_score = \frac{(c_1 * Precision) + (c_2 * Recall) + (c_3 * Accuracy)}{c_1 + c_2 + c_3}$$

But since we have no preference or whatsoever to distinguish between the parameters we will just set the $c_1 = c_2 = c_3 = 1$

Cross Validation

Another thing we should take into consideration is that our judgement also depends on the way we split our test and training data. We may get lucky and have a very good score for a certain model when it will not perform that good in general.
So the right way to do it is instead of relying on our luck we would split the data into N equal splits and then selecting each time one of the N splits to be the testing set. And with that the algorithm will be performed on the data N times with N different scores. We will then rely on the mean score to rate the performance of the model.

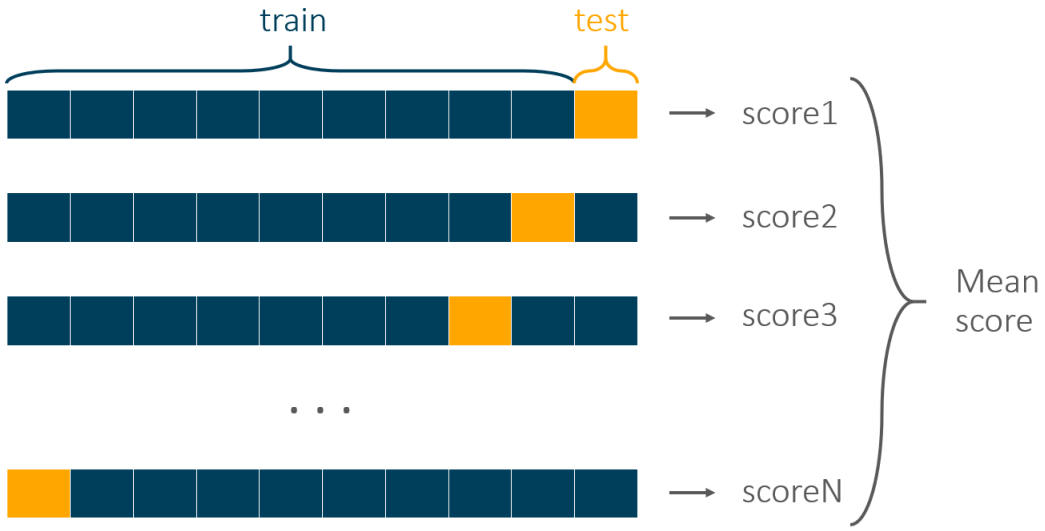

Figure 2.8: K-fold Cross Validation method

Chapter 3

Implementation Results

We have smoothly described both the data we are working on and the model we are intending to use in the previous chapter. In the following we will be presenting the results of our implementation and further discuss the outcomes and hopefully derive a significant conclusion from the study.

3.1 Results Presentation

We will start by mainly unveiling the outcome of the experimental process going form an algorithm to another to find the best combination that suited each.

3.1.1 Neural Network Algorithm

You may recall from figure (2.6) that the ANN algorithm will be processing up to 96 versions of our data. But after the hyperparameter tunning we will have more than seven hundred results to show, which is a lot. That is why we chose to only present the results concerning the data that have shown the best results. In this case, the data where we used the ROC-AUC filter, deleted all the missing values and over-sampled our data before using the Z-normalisation on our feature is the best for the ANN algorithm.

[Batch size, optimizer]	Accuracy	Precision	Recall	Borda Count	Rank
[128, adam]	0.837	0.852	0.837	0.842	5
[128, rmsprop]	0.838	0.850	0.838	0.842	7
[256, adam]	0.838	0.852	0.838	0.843	2
[256, rmsprop]	0.838	0.852	0.838	0.843	3
[512, adam]	0.838	0.852	0.838	0.843	4
[512, rmsprop]	0.838	0.850	0.838	0.842	6
[800, adam]	0.839	0.849	0.838	0.842	8
[800, rmsprop]	0.838	0.853	0.838	0.843	1

Table 3.1: Results on the parameter tuning of the ANN algorithm

Taking a look at the parameter tuning on this learning algorithm, even though the grid search result here doesn't vary a lot, we see that we should initialize the batch size to 800 and use the "rmsprop" optimizer.

3.1.2 KNN Algorithm

The KNN algorithm has proven that the best data to use for the training is again the one where we used ROC-AUC filter, deleted all the missing values, undersampled the data and normalized it with the Z-normalisation.

metric, N neighbors	Accuracy	Precision	Recall	Borda Count	Rank
[minkowski, 2]	0.756	0.772	0.756	0.761	20
[minkowski, 3]	0.806	0.806	0.806	0.806	11
[minkowski, 4]	0.790	0.792	0.790	0.791	17
[minkowski, 5]	0.815	0.816	0.815	0.815	5
[minkowski, 6]	0.803	0.803	0.803	0.803	14
[minkowski, 7]	0.820	0.822	0.820	0.821	2
[minkowski, 8]	0.811	0.811	0.811	0.811	8
[euclidean, 2]	0.756	0.772	0.756	0.761	20
[euclidean, 3]	0.806	0.806	0.806	0.806	11
[euclidean, 4]	0.790	0.792	0.790	0.791	17
[euclidean, 5]	0.815	0.816	0.815	0.815	5
[euclidean, 6]	0.803	0.803	0.803	0.803	14
[euclidean, 7]	0.820	0.822	0.820	0.821	2
[euclidean, 8]	0.811	0.811	0.811	0.811	8
[manhattan, 2]	0.759	0.776	0.759	0.765	19
[manhattan, 3]	0.809	0.809	0.809	0.809	10
[manhattan, 4]	0.791	0.794	0.791	0.792	16
[manhattan, 5]	0.817	0.818	0.817	0.817	4
[manhattan, 6]	0.805	0.805	0.805	0.805	13
[manhattan, 7]	0.821	0.823	0.821	0.822	1
[manhattan, 8]	0.812	0.812	0.812	0.812	7

Table 3.2: Results on the parameter tuning of the KNN algorithm

We notice in the table 3.2 that the Minkowski and the Euclidean metrics are doing equally the same quality of prediction on all the tuned parameters, which is not really surprising seeing that their equations are pretty similar. In the other hand the Manhattan metric is slightly doing better. we can also see that the best hyperparameters we could use are the Manhattan metric with 7 as the number of neighbours.

3.1.3 SVM Algorithm

Going with the same procedure with the SVM Algorithm, after precessing all our data we found that once again using the ROC-AUC filter, deleting all the missing values, oversampling then Using the Z-normalisation is the best we could do to have the best results.

[C, gamma]	Accuracy	Precision	Recall	Borda Count	Rank
[0.1, 1]	0.820	0.840	0.820	0.827	14
[0.1, 0.1]	0.831	0.851	0.831	0.838	11
[0.1, 0.01]	0.833	0.852	0.833	0.840	8
[0.1, 0.001]	0.753	0.780	0.753	0.762	16
[1, 1]	0.834	0.845	0.834	0.838	12
[1, 0.1]	0.834	0.852	0.834	0.840	5
[1, 0.01]	0.833	0.852	0.833	0.839	9
[1, 0.001]	0.834	0.852	0.833	0.840	3
[10, 1]	0.826	0.830	0.826	0.828	13
[10, 0.1]	0.835	0.849	0.835	0.840	6
[10, 0.01]	0.834	0.852	0.834	0.840	2
[10, 0.001]	0.833	0.852	0.833	0.839	10
[100, 1]	0.812	0.813	0.812	0.813	15
[100, 0.1]	0.836	0.847	0.836	0.840	4
[100, 0.01]	0.834	0.852	0.834	0.840	1
[100, 0.01]	0.834	0.852	0.834	0.840	7

Table 3.3: Results on the parameter tuning of the SVM algorithm

The results in the Table (3.3) represent the parameter tuning result of the C and the gamma constant. And the best parameters to train this model according to out grid search will be 100 and 0.01 for C and gamma respectively.

3.1.4 Decision Tree Algorithm

Now for the Decision Tree algorithm, as part of the algorithms that doesn't require a normalisation, we found that the best way to preprocess the data before training our model is as expected applying the ROC-AUC filter followed by deleting our missing values and undersampling the data.

[criterion, maximum depth]	Accuracy	Precision	Recall	Borda Count	Rank
[gini, 4]	0.756	0.772	0.756	0.761	30
[gini, 5]	0.760	0.780	0.760	0.766	27
[gini, 6]	0.764	0.786	0.764	0.771	25
[gini, 7]	0.766	0.789	0.766	0.774	24
[gini, 8]	0.777	0.779	0.777	0.777	21
[gini, 9]	0.788	0.788	0.788	0.788	19
[gini, 10]	0.800	0.800	0.800	0.800	15
[gini, 11]	0.812	0.813	0.812	0.813	13
[gini, 12]	0.820	0.823	0.820	0.821	12
[gini, 13]	0.826	0.831	0.826	0.828	7
[gini, 14]	0.829	0.834	0.829	0.830	4
[gini, 15]	0.828	0.831	0.828	0.829	6
[gini, 16]	0.826	0.829	0.826	0.827	8
[gini, 17]	0.824	0.826	0.824	0.824	10
[gini, None]	0.796	0.796	0.795	0.796	18
[entropy, 4]	0.756	0.772	0.756	0.761	29
[entropy, 5]	0.760	0.780	0.760	0.766	28
[entropy, 6]	0.764	0.786	0.764	0.771	26
[entropy, 7]	0.767	0.789	0.767	0.775	23
[entropy, 8]	0.775	0.777	0.775	0.776	22
[entropy, 9]	0.786	0.786	0.786	0.786	20
[entropy, 10]	0.799	0.800	0.800	0.800	16
[entropy, 11]	0.812	0.813	0.812	0.813	14
[entropy, 12]	0.820	0.823	0.820	0.821	11
[entropy, 13]	0.827	0.833	0.827	0.829	5
[entropy, 14]	0.831	0.837	0.831	0.833	1
[entropy, 15]	0.830	0.834	0.830	0.831	2
[entropy, 16]	0.830	0.833	0.830	0.831	3
[entropy, 17]	0.826	0.828	0.826	0.827	9
[entropy, None]	0.799	0.800	0.799	0.799	17

Table 3.4: Results on the parameter tuning of the DT algorithm

The two criterion are showing results that are very close. Thus, the best hyperparameters we could choose for the Decision Tree algorithm are the "gini" criterion and a maximum depth set to 14.

3.1.5 XGboost Algorithm

Finally the XGboost algorithm joining the list of algorithms that doesn't require a normalization performs best the the data where the feature selection was based on the ROC-AUC filter, the missing values were deleted and the data was Z-normalized.

[Maximum depth, Number of estimators]	Accuracy	Precision	Recall	Borda Count	Rank
[5, 100]	0.875	0.897	0.875	0.882	2
[5, 300]	0.875	0.894	0.875	0.881	4
[5, 500]	0.874	0.891	0.874	0.880	8
[5, 800]	0.873	0.888	0.873	0.878	15
[6, 100]	0.875	0.897	0.875	0.882	3
[6, 300]	0.874	0.892	0.874	0.880	9
[6, 500]	0.872	0.888	0.873	0.878	17
[6, 800]	0.871	0.885	0.871	0.876	31
[7, 100]	0.875	0.897	0.875	0.882	1
[7, 300]	0.874	0.890	0.874	0.879	10
[7, 500]	0.872	0.886	0.872	0.877	19
[7, 800]	0.871	0.883	0.871	0.875	32
[8, 100]	0.874	0.894	0.874	0.881	5
[8, 300]	0.873	0.888	0.873	0.878	13
[8, 500]	0.872	0.885	0.872	0.877	24
[8, 800]	0.871	0.882	0.871	0.875	34
[9, 100]	0.874	0.893	0.875	0.881	1
[9, 300]	0.873	0.887	0.873	0.877	18
[9, 500]	0.872	0.884	0.872	0.876	26
[9, 800]	0.870	0.881	0.870	0.874	40
[10, 100]	0.874	0.892	0.874	0.880	7
[10, 300]	0.872	0.886	0.872	0.877	20
[10, 500]	0.871	0.883	0.871	0.875	33
[10, 800]	0.871	0.882	0.871	0.874	35
[11, 100]	0.873	0.890	0.873	0.879	11
[11, 300]	0.872	0.885	0.872	0.877	23
[11, 500]	0.872	0.884	0.872	0.876	27
[11, 800]	0.871	0.881	0.871	0.874	38
[12, 100]	0.873	0.889	0.873	0.879	12
[12, 300]	0.872	0.884	0.872	0.876	25
[12, 500]	0.872	0.883	0.872	0.876	29
[12, 800]	0.871	0.881	0.871	0.874	36
[13, 100]	0.873	0.887	0.873	0.878	16
[13, 300]	0.873	0.884	0.873	0.877	22
[13, 500]	0.872	0.883	0.872	0.876	30
[13, 800]	0.871	0.881	0.871	0.874	37
[14, 100]	0.873	0.887	0.874	0.878	14
[14, 300]	0.873	0.884	0.873	0.877	21
[14, 500]	0.872	0.883	0.872	0.876	28
[14, 800]	0.871	0.881	0.871	0.874	39

Table 3.5: Results on the parameter tuning of the XGboost algorithm

According again to our computations, the hyperparameters that works the best with the previously stated data are a maximum depth and a number of estimators set respectively to 7 and 100.

3.2 Results Discussion

To further discuss the results presented, we will enumerate once again the algorithms we used and for each one analyse it's result and conduct a examination.

3.2.1 Neural Network Algorithm

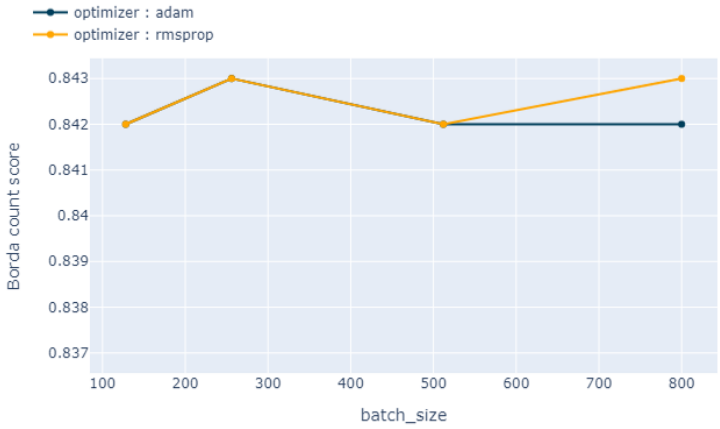

Figure 3.1: Parameter tuning of the ANN algorithm summary

Looking back at the results on the ANN Algorithm we can notice that the batch size doesn't affect the quality of our prediction seeing that the borda count values are always in [0.842, 0.843]. And the same for the optimizer, seeing that the "adam" and "rmsprop" optimizers almost overlap. Consequently we can say that in this case the grid search performed on this Algorithm didn't advantage us a lot in term of advancing the Borda count score.

3.2.2 KNN Algorithm

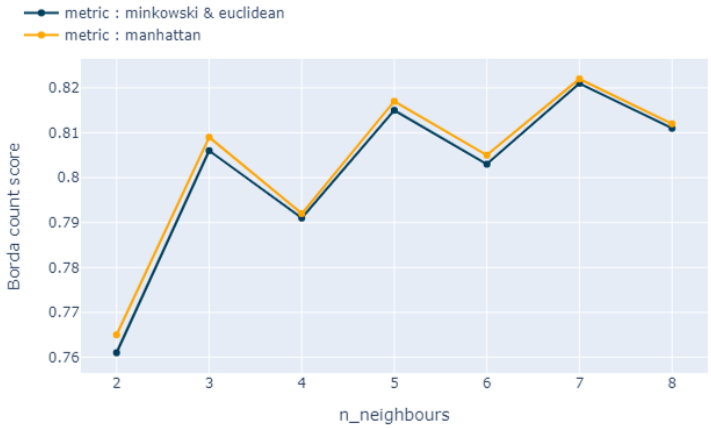

Figure 3.2: Parameter tuning of the KNN algorithm summary

Now about the KNN Algorithm, the figure above show us how the Borda count score slowly improves with some frequent variations when we increase the number o neighbours. It also gives us the intuition that if we try even greater number of neighbours we may be able to improve our score but not significantly as the slope decreasing with the increase of the parameter K. In contrast the hyper parameter search on the metric variable didn't provide us with much improvements, as we see once again the scores are almost overlapping.

3.2.3 SVM Algorithm

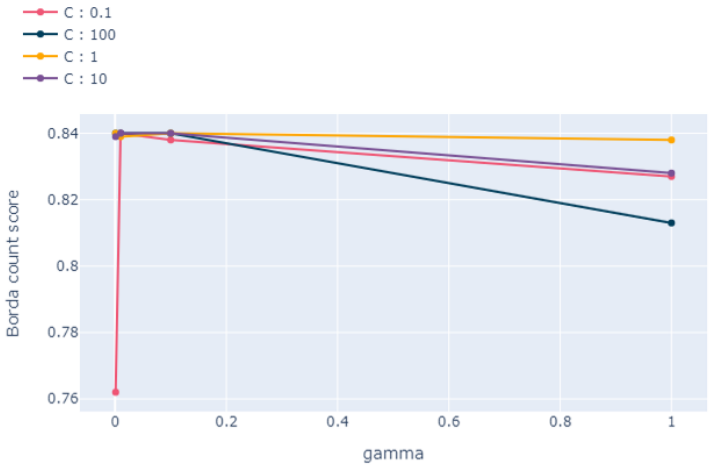

Figure 3.3: Parameter tuning of the SVM algorithm summary

For the SVM model we can see that the results are all high at law values of gamma except for when $C = 1$, that performs very poorly for a low value of gamma. And we also notice that the model is badly affected when we increase gamma. Even thought $C = 100$ performs the best in our case, it is also the one that is the most affected by the increase of gamma.

3.2.4 Decision Tree Algorithm

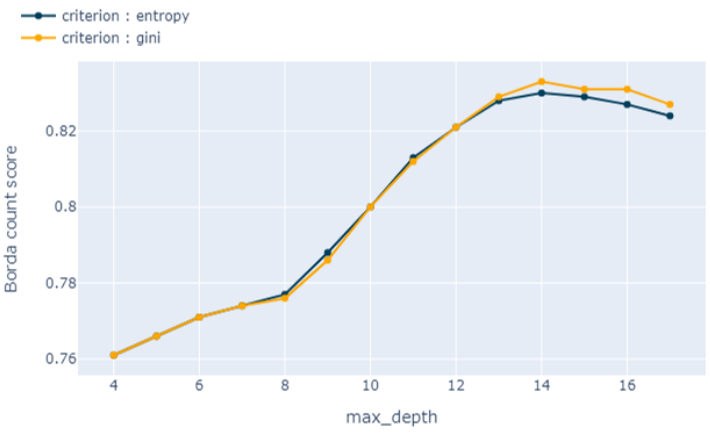

Figure 3.4: Parameter tuning of the DT algorithm summary

Another time we can point out that the scores in the DT models are also relatively overlapping when performing a parameter search on the criterion parameter. We also note that the curve apears to have a maximum. We could suppose that even if we try higher values for the maximum depth variable we are less lucky to perform better on our testing set.

3.2.5 XGboost Algorithm

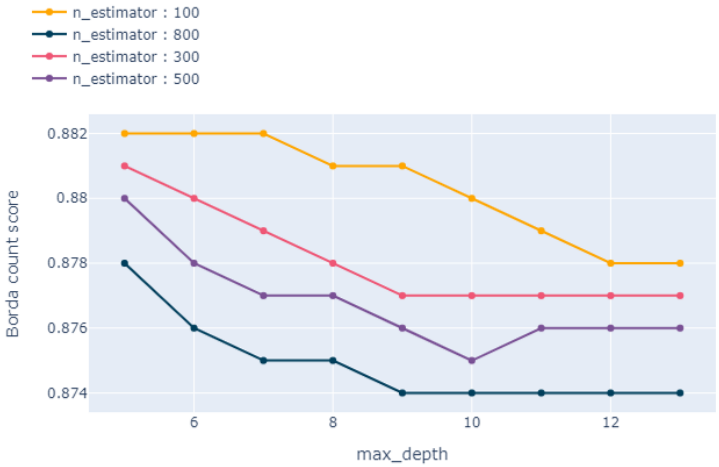

Figure 3.5: Parameter tuning of the XGboost algorithm summary

Lastly, the XGboost show a big difference of performance when shifting the "n-estimator" variables with the best performance at the value $n_estimator = 100$. In general all the curves act the same way: the performance tend to decrease in quality when increasing the "max-depth" hyperparameter. Thus, previous literature has shown that the XGBoost model doesn't perform as well as our study has shown. And the reason behind is that when performing this model, the "max-depth" hyperparameter was set at a very high value and grid search was performed on the model. Otherwise it would have proven to be better performing.

3.2.6 Results summary

To wrap up our results, we want to display the best performances in each Algorithm and compare between them.

Features	Missing Values	Balancing	Normalisation	Algorithm	Best Parameters	Borda Count
ROC-AUC	Deleted	SMOT	Not Normalized	DT	["criterion":"entropy", "max-depth":14]	0.833
ROC-AUC	Deleted	SMOT	Not Normalized	XGBoost	["max-depth":7, "n-estimator":100]	0.882
ROC-AUC	Deleted	SMOT	Z-score	KNN	["criterion":"entropy", "max-depth":12]	0.822
ROC-AUC	Deleted	SMOT	Z-score	SVM	["n-neighbors":7, "metric":"manhattan"]	0.840
ROC-AUC	Deleted	SMOT	Z-score	ANN	["batch-size":256, "optimizer":"rmsprop"]	0.844

Table 3.6: Results on the parameter tuning of the ANN algorithm

We conclude that with the right parameters XGBoost out performs greatly the other models. And It also out performs the results in previous studies on the same data. That is simply because

in previous literature the grid search step was missed and the parameters were chosen by mere intuition.

3.3 Study Limitation

The results provided by applying the XGBoost model are pretty satisfactory. However, the results are not ready to be generalized as there are several data sets concerning lawns available on-line. It is then recommended to train this model on the maximum of data that could be gathered before actually generalizing the predictive model. It is important to keep in mind that our study only gives an idea of what works better for this type of data and to what extend we can successfully predict the loan status of a certain customer. Also, we cannot guarantee that XGBoost is ultimatly the best model in terms of performance in terms of quality of performance as there are still some algorithms to test. Moreover, the avenue of more data could reveal some trends we were not aware of with just a limited data.

Chapter 4

Conclusion and Future Work

In a few word, this work proposes a solution to support the decision making of banking institutions in general when it comes to grant a lawn to a certain client. It also tackles the problem of choosing the right way to preprocess the data, the right algorithm to build the model and the right parameters to use on a certain model. The proposed solution is pretty iterative and consisted of building different versions of the same data but pre processed differently before getting ultimately splitted using cross validation.

Consequently, each version will have to be fed to a learning algorithm with respect to a grid search that will help define the hyperparameters. To building the predictive model, we tested the performance of diverse learning algorithms namely: the K-Nearest Neighbours Algorithm; the Decision Tree Algorithm; the Support Vector Machine Algorithm; the XGBoost Algorithm and the Artificial Neural Network Algorithm.

Eventually after gathering all the results we compared the different performances based on the Accuracy, the Precision and the Recall scores simultaneity buy unifying them in one score using the Borda count. This final scored allowed us to rank the models and as a result the XGBoost performance was buy far the best among the models we tested with Borda count score of 88.2%.

Moreover, further work on this problematic will consist of testing more learning algorithm such as the logistical regression following the same work flow . Trying to expand the data volume would also be very effective to better predict the lawn status of clients. It is also recommended to develop an interactive application to facilitate the use of the built in predictive model.

Bibliography

[1] Kashyap, A., K., Rajan, R., Stein, J.C., Banks as Liquidity Providers, An Explanation for the Coexistence of Lending and Deposit- Taking, The Journal Of Finance, Vol. LVII, No. 1, pp 33-73 Feb. 2002

[2] Warwick J McKibbin, Andrew Stoeckel, The Global Financial Crisis: Causes and Consequences, working papers in international economics, November 2009, No. 2.09

[3] Peltonen, T. A. (2006). Are Emerging Market Currency Crises Predictable? A Test. Working Paper Series 571. European Central Bank.

[4] Atiya AF (2001) Bankruptcy prediction for credit risk using neural networks: a survey and new results. IEEE Trans Neural Netw 12(4) 929–935

[5] Lior Rokach and Oded Maimon. "Data Mining With Decision Trees Theory and applications". In: (2015).

[6] John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Microsoft Research 1998

[7] J. A. Freeman and D. M. Skapura. Neural Networks, Algorithms, Applications and Programming Techniques. Addison-Wesley, New York, 2nd edition, 1992.

[8] N. Bhatia et al, Survey of Nearest Neighbor Techniques. International Journal of Computer Science and Information Security, Vol. 8, No. 2, 2010.

[9] Friedman JH (2001). "Greedy function approximation: a gradient boosting machine." Annals of Statistics, pp. 1189-1232.

[10] Peter Flach, Performance Evaluation in Machine Learning: The Good, The Bad, The Ugly and The Way Forward, Intelligent Systems Laboratory, University of Bristol, UK.

[11] https://www.kaggle.com/zaurbegiev/my-dataset

[12] https://www.kaggle.com/aakashsinghrawat/bank-loan-dataset

[13] https://www.kaggle.com/rahu7292/ml-from-scratch-with-loan-status-prediction

[14] https://www.kaggle.com/panamby/bank-loan-status-dataset