Ecole National Supérieur d'Informatique et d'Analyse des Systèmes

Data Mining Project report

Credit Risk

Submitted by:
Salma DKIER

Submitted to :
Pr Hatim DERROUZ

Contents

Abstract

The management of credit risk is a key challenge for financial institutions seeking to minimize credit defaults and retain customers. Given the increasing number of credit providers and options available to borrowers, customer engagement and retention have become critical factors for maintaining competitiveness and profitability in the banking sector.

Effective credit risk management projects rely heavily on customer retention as it allows banks to reduce the number of customers who default on their credit obligations. By implementing measures to manage credit risk and prevent customer churn, banks can secure their profitability while simultaneously strengthening their competitive edge.

This project aims to evaluate the performance of various machine learning algorithms, including Naive Bayes, Random Forest, K-means, ANN, SVM, and Decision Tree, to identify the most suitable model with the highest accuracy and precision. The development of a machine learning model that can accurately predict and prevent credit risk in the banking sector would be a valuable tool for enhancing customer retention and reducing credit defaults.

Introduction

The advent of machine learning has brought about significant changes to many areas of human activity, including finance. Credit risk estimation is a crucial aspect of modern financial systems that rely on credit and trust. Accurately estimating credit risk is essential to the entire system, as any error in estimation could lead to systemic failures. Lenders, therefore, devote significant resources to predict the creditworthiness of consumers and companies to develop lending strategies that minimize their risks. Previously, statistical methods such as Linear Discriminant Analysis and Logistic Regression were used to estimate credit risk. However, these methods struggle with large datasets.

Recent advances in computing power and the availability of large credit datasets have paved the way for AI-driven credit risk estimation algorithms. Machine learning techniques, including k-Nearest Neighbor, Random Forest, and Support Vector Machines, have proven to be more effective and flexible than statistical methods. Additionally, the application of deep learning techniques to large credit risk datasets has further improved the accuracy and efficiency of credit risk estimation.

Through machine learning analysis of customer behavior and identification of patterns that indicate a customer is at risk of leaving, banks can proactively engage with customers to address their concerns and offer personalized solutions. Machine learning models can also be used to optimize product offerings and personalize customer experience, leading to improved customer retention and increased profitability.

Despite the potential benefits, implementing machine learning models in the banking sector requires addressing challenges such as explainability and transparency, proper data governance and privacy measures, and compliance with regulatory requirements. Nonetheless, banks that successfully implement

these technologies stand to gain a competitive edge in the market and prevent credit risk.

Chapter 1

Context

1.1 Introduction

The following chapter outlines our project on Credit Risk Prediction in the banking industry through the use of machine learning. Initially, we will provide an overview of the project's general context, followed by a detailed presentation of the problem and objectives. The core objective of a credit risk project is to assess the creditworthiness of potential borrowers and predict the possibility of defaulting on loans or credit obligations. To accomplish this objective, we will employ various machine learning techniques. Ultimately, we will outline the essential stages of the machine learning lifecycle, starting from data collection to Model selection and Training, to ensure the successful execution of our project.

1.2 Definition

Credit risk is an integral part of lending activities that involves the potential financial loss that a lender may experience if a borrower fails to meet their repayment obligations. To mitigate this risk, lenders assess the creditworthiness of borrowers by analyzing their financial information, credit history, and other relevant factors. By doing so, lenders can make informed decisions about the terms and conditions of credit agreements.

Credit risk can vary depending on a range of factors, including market conditions, economic trends, and the borrower's creditworthiness. For this reason, ongoing monitoring and evaluation of credit risk are crucial to ensure the financial stability of lenders. Credit risk is present in various types of credit arrangements, such as personal loans, business loans, mortgages, and credit cards, and can change over time due to shifts in market dynamics and the borrower's financial situation.

Figure 1.1: Risk management

1.3 Machine Learning to predict the credit risk

Machine learning is a subset of artificial intelligence that enables computers to learn and improve from experience without being explicitly programmed. It has gained significant attention in the financial industry, particularly in credit risk prediction, due to its ability to process vast amounts of data and identify complex patterns. Credit risk prediction is a crucial aspect of lending decisions,

and the use of machine learning models can enhance the accuracy and efficiency of this process.

Machine learning models can be trained on historical credit data to predict the likelihood of a borrower defaulting on their credit obligations. These models analyze various factors, including credit history, financial information, and borrower behavior, to generate a credit score or risk assessment. By using machine learning algorithms, financial institutions can make better lending decisions, optimize credit limits, and reduce the risk of credit defaults.

Machine learning models offer several advantages over traditional statistical methods for credit risk prediction. They can handle vast amounts of data, detect non-linear relationships between variables, and provide real-time credit risk assessments. Furthermore, machine learning models can be continuously improved through the inclusion of new data and the optimization of algorithms. This can help financial institutions stay up-to-date with changing market conditions and borrower behavior, enabling them to make more informed credit decisions. And to achieve, that we can use a variety of machine learning techniques, such as

- **Naive Bayes** .
- **Desicion tree** .
- **Random forest** .
- **K-Nearest Neighbors (KNN)** .
- **Support Vector Machines (SVM)** .
- **Artificial Neural Network (ANN)** .
- **K-Means (K-M)**

1.4 Machine Learning lifecycle

Machine learning for credit risk prediction typically involves several key stages. The first step is data collection and exploration, which involves gathering relevant datasets and analyzing them to identify key features, relationships, and

potential issues. This stage is crucial in developing an effective model that accurately predicts credit risk.

After data collection, the next stage is data preprocessing, where the data is prepared for modeling. This involves handling missing values, outliers, skewness, normalization, and other data cleaning tasks. Data preprocessing ensures that the model is trained on accurate and consistent data, leading to better performance.

The third stage is model training and evaluation. This involves developing machine learning models, optimizing them through techniques such as cross-validation, and evaluating their performance using metrics such as accuracy, precision, recall, and F1-score. The model's performance is critical in selecting the best model(s) for credit risk prediction. Once the model(s) have been selected, they can be used to make predictions and recommendations to the bank on how to improve customer retention.

.

1.5 Conclusion

This chapter provided an overview of our project on Credit Risk Prediction in the banking sector using machine learning. The primary goal of our project is to evaluate the creditworthiness of potential borrowers and determine the likelihood of default or non-payment on loans or credit obligations. We discussed the importance of credit risk management in the lending industry and how it is essential to ensure the soundness and stability of financial institutions.

With the increasing availability of data and advancements in machine learning, banks now have the opportunity to leverage data-driven technologies to improve their credit risk management and enhance their decision-making processes. Our project aims to use machine learning algorithms to accurately predict credit risk and assist banks in making informed decisions about lending terms, interest rates, and credit limits.

In the upcoming chapters, we will delve into the details of the machine learning algorithms we will be using and the specific steps of the machine

learning lifecycle, from data collection to model selection and training. We aim to provide a comprehensive understanding of our project and the methodologies we will be employing to ensure its success.

Chapter 2

Data Preparation

2.1 Introduction

Data preparation is a crucial step in machine learning. Before data can be used to train models, it must be formatted in a specific way that algorithms can understand. This often involves handling missing or invalid data and shaping the data to extract relevant features. The quality of the prepared data can have a significant impact on the accuracy and usefulness of the resulting model

2.2 Naive Bayes

Naive Bayes is a machine learning algorithm used for classification tasks. It is based on Bayes' theorem, which describes the probability of a hypothesis based on prior knowledge and evidence. The "naive" in Naive Bayes refers to the assumption that all features in the data are independent of each other, which simplifies the computation of probabilities. The algorithm is relatively simple and efficient, making it popular for large-scale text classification tasks. However, the independence assumption may not always hold in real-world data, which can affect the accuracy of the model. Nevertheless, Naive Bayes remains a useful and widely used classification algorithm.

2.2.1 Use case

Naive Bayes is commonly used for text classification tasks, such as spam filtering, sentiment analysis, and document categorization. It works by calculating the probability of a document belonging to a certain category based on the frequency of words in the document and the frequency of words in each category. detection.

2.2.2 Advantages and disadvantages

Advantages :
-Naive Bayes is simple and easy to understand, making it an ideal algorithm for beginners.
-It requires a small amount of training data to make predictions, making it computationally efficient and fast.
-It can handle a large number of features relative to the amount of data.
Disadvantages :
-The assumption of independence between features is not always true in real-world datasets.
-Naive Bayes may produce suboptimal results compared to other more sophisticated algorithms such as decision trees or random forests.
-It is sensitive to the quality of the input data, especially outliers and irrelevant features.

2.3 SVM

SVMs are among the machine learning algorithms that solve both classification and regression or anomaly detection problems. They are known for their strong theoretical guarantees, their great flexibility and their ease of use even without a great knowledge of data mining. Their goal is to separate the data into classes using a border that is as "simple" as possible, so that the distance between different groups of data and the border between them is maximum. This distance is also called “margin” and SVMs are thus called “wide margin separators”, the “support vectors” being the data closest to the border.

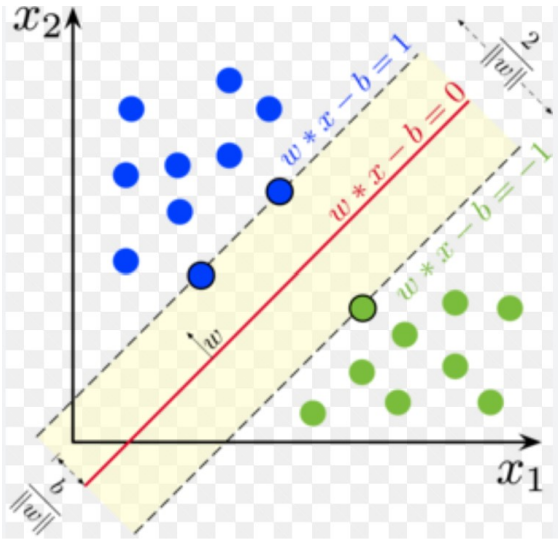


Figure 2.1: SVM

The figure above shows the separation lines produced by the SVM method. The first line is close to the training points, ie it is not optimal, on the contrary the line in the middle is optimal and classifies the blue and red points well. Right in the middle place the border as far as possible from the blue circles, but also as far as possible from the red circles.

2.3.1 Use case

SVM is used for text classification issues such as category assignment, spam detection, and sentiment analysis. They are also commonly used for image recognition problems, particularly in shape recognition and color classification. SVM also plays a vital role in many areas of handwritten symbol recognition, such as postal automation services.

2.3.2 Advantages and Diasdvantages

Advantages :
- It is Easy to use.
- It has a regularization parameter, which encourages the user to avoid overfit-

ting.
- It has high prediction accuracy on small datasets.
Disadvantages :
- It is not suitable for larger datasets, as training time with SVMs can be long.
- It is less efficient on datasets containing noise and a lot of outliers.
- It can artificially introduce a class imbalance in the construction of individual models, if the scores are poorly calibrated, the comparisons of the output of the decision functions are not fair.

2.4 Random Forest

The "random forests" algorithm is a classification algorithm that reduces the variance of single decision tree forecasts, thereby improving their performance. To do this, it combines many decision trees in a bagging-type approach. A random forest is made up of a set of independent decision trees. Each tree has a fragmented view of the problem due to a double random selection: by
- a random draw with replacement on the observations (the rows in your database). This process is called tree bagging.
- a random draw on the variables (the columns of your database).
This process is called feature sampling. The Random Forest Classifier works in a voting way, that is, after the generation of thousands of trees, it takes the observation predicted and it goes through all the decision trees, each tree gives a prediction (class 0 or 1), then it calculates the sum of the predictions by class and the most voted class to consider as the answer.

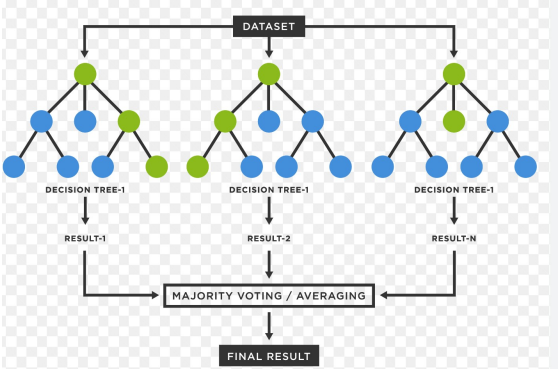


Figure 2.2: Random forest

2.4.1 Advantages and Disadvantages

The decision tree algorithm has different advantages and disadvantages, among these advantages and disadvantages we list:
Advantages :
-They are easy to understand.
- They allow you to choose the most appropriate option among several.
- They are resistant to noise and missing values.
- They allow a quick classification.
Disadvantages:
- Decision tree learning can lead to very complex trees which poorly generalise the learning set. - They require a large number of individuals.
- The performance deteriorates when the number of classes increases.

2.5 KNN

kNN stands for k-Nearest Neighbours. It is a supervised learning algorithm. This means that we train it under supervision. We train it using the labelled data already available to us. Given a labelled dataset consisting of observations (x,y), we would like to capture the relationship between x — the data and y — the label. More formally, we want to learn a function g : X→Y so that given an unseen observation X, g(x) can confidently predict the corresponding output Y. KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones with the shortest distances to it. As a result, it's often referred to as a distance-based algorithm.

In order to correctly classify the results, we must first determine the value of K (Number of Nearest Neighbours).

In the following diagram, the value of K is 5. Since there are four cats and just one dog in the proximity of the five closest neighbours, the algorithm would predict that it is a cat based on the proximity of the five closest neighbors in the red circle's boundaries.

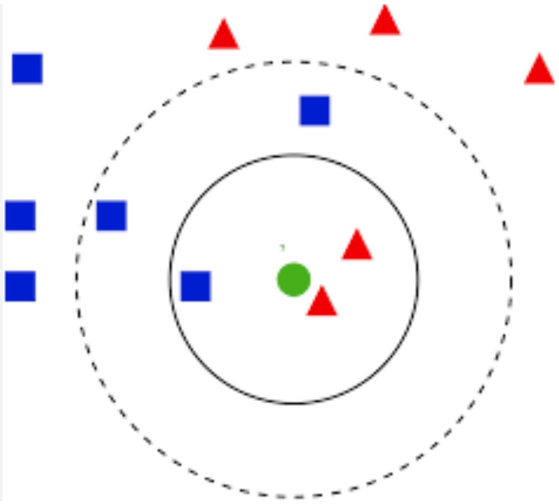


Figure 2.3: kNN

2.5.1 Advantages and Disadvantages

The kNN algorithm has different advantages and disadvantages, among these advantages and disadvantages we list:
Advantages :
-No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period.
- Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.
- KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.).
Disadvantages:
- Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.
- Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

2.6 Decision tree

Decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the dataset into subsets based on the values of its attributes, and then assigning labels to the resulting subsets. The algorithm constructs a tree-like structure where each internal node represents a test on a particular attribute, and each leaf node represents a class label or a numeric value.

Decision trees are particularly useful for solving problems that involve a large number of variables and a complex decision-making process. They are simple to understand, interpret, and visualize, making them a popular choice for data exploration and analysis. Decision trees are also non-parametric, which means they do not make any assumptions about the underlying distribution of the data.

One of the key advantages of decision trees is their ability to handle both categorical and continuous variables. They can also handle missing data by either ignoring it or imputing values based on the available data. However, decision trees can suffer from overfitting, especially when the tree is too complex or when the dataset is noisy. In addition, decision trees can be sensitive to small changes in the data, which can result in different tree structures and classification outcomes.

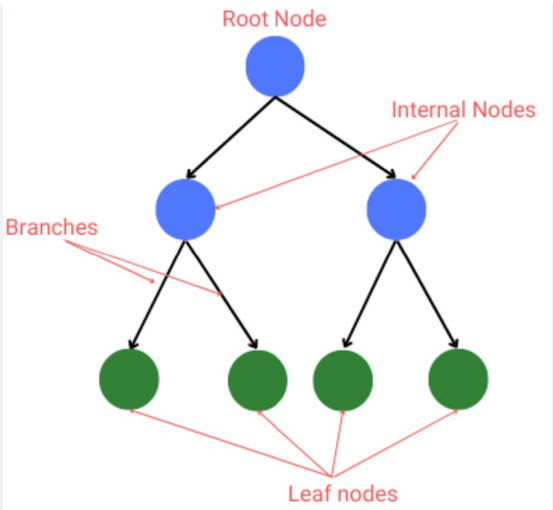


Figure 2.4: Decision Tree

2.6.1 Advantages and Disadvantages

Advantages: Easy to understand and interpret: Decision trees provide a clear and easy-to-understand representation of the decision-making process. The tree structure is intuitive and can be visualized graphically, making it easy to communicate the decision process to stakeholders.

Applicable to various types of data: Decision trees can be used for both categorical and continuous variables, making them a versatile tool for a wide range of problems.

Non-parametric: Decision trees make no assumptions about the distribution of the data, making them robust to outliers and data that do not conform to a specific distribution.
Disadvantages:
Overfitting: Decision trees can be prone to overfitting the training data, especially if the tree is allowed to grow too deep. This can result in poor performance on new, unseen data.

Instability: Small changes in the training data can lead to large changes in the resulting tree, making decision trees unstable and difficult to replicate.

Bias: Decision trees can be biased towards features that have many levels or categories, as these features tend to create more splits in the tree.

2.7 K-Means

K-means is a popular clustering algorithm used in machine learning and data mining. It is an unsupervised learning algorithm that aims to partition a given dataset into k clusters, where k is a user-defined parameter.

The algorithm starts by randomly selecting k centroids (points) as cluster centers. It then assigns each data point to the cluster whose centroid is closest to it, using a distance metric such as Euclidean distance. After all the data points have been assigned to clusters, the centroids are recalculated as the mean of all the data points in each cluster. This process is repeated until the centroids no longer change, or a maximum number of iterations is reached.

The output of the k-means algorithm is k clusters, each represented by a centroid. The algorithm aims to minimize the sum of squared distances between each data point and its assigned cluster centroid. K-means can be used for various applications, such as image segmentation, customer segmentation, and anomaly detection.

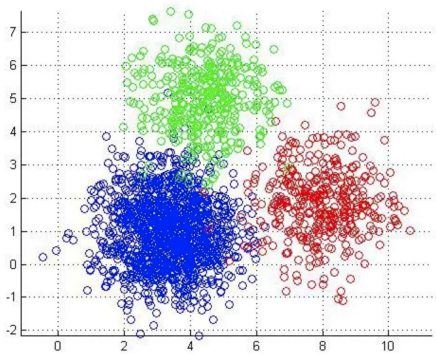


Figure 2.5: K-Means

2.7.1 Advantages and disadvantages of Neural Network usage:

Advantages:

Simple and easy to understand: K-means is one of the simplest and easiest to understand clustering algorithms, making it an excellent choice for beginners.

Scalable and efficient: K-means clustering can be scaled to handle large datasets and is computationally efficient, making it a popular choice for many applications.

Versatile: K-means can be used for a wide range of clustering problems, including image segmentation, customer segmentation, and anomaly detection.

Disadvantages:

Sensitivity to initial conditions: K-means clustering is sensitive to initial conditions, which can lead to different results for different runs of the algorithm.

Difficulty handling categorical data: K-means is designed to work with continuous numerical data, making it difficult to handle categorical data without pre-processing.

Requires predefined number of clusters: K-means requires the number of clusters to be predefined, which can be a challenge in situations where the optimal number of clusters is not known beforehand.

2.8 Neural Networks

The basic idea behind a neural network is to simulate (copy in a simplified but reasonably faithful way) lots of densely interconnected brain cells inside a computer so you can get it to learn things, recognize patterns, and make decisions in a humanlike way. The amazing thing about a neural network is that you don't have to program it to learn explicitly: it learns all by itself, just like a brain!
A typical neural network has anything from a few dozen to hundreds, thousands, or even millions of artificial neurons called units arranged in a series of layers, each of which connects to the layers on either side. Some of them, known as input units, are designed to receive various forms of information from the outside world that the network will attempt to learn about, recognize, or otherwise process. Other units sit on the opposite side of the network and signal how it responds to the information it's learned; those are known as output units. In between the input units and output units are one or more layers of hidden units. Information flows through a neural network in two ways. When it's learning (being trained) or operating normally (after being trained), patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units. This common design is called a feedforward network.

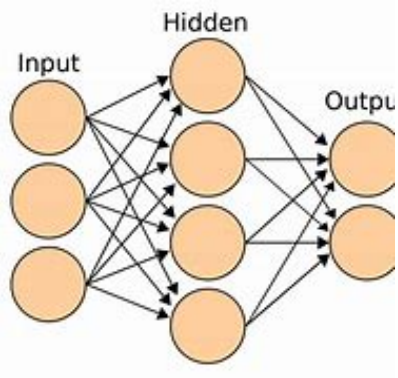


Figure 2.6: Neural Network

2.8.1 Advantages and disadvantages of Neural Network usage:

Advantages:

- Neural networks give a better result when they gather all the data and information whereas traditional machines learning algorithms will reach a level, where more data doesn't improve the performance.
- Ability to work with incomplete information
- Fault tolerance
- Neural networks are good to model with nonlinear data with a large number of inputs
- Artificial neural networks have numerical strength that can perform more than one job at the same time.

Disadvantages:

- One of the most distinguishing disadvantages of the neural network is their ''black box" nature. It means that we don't know how and why the neural network came up with a certain output.
- Neural networks require much more data than any other traditional machine learning algorithms, as in at least thousands if not millions of samples

- Neural networks are computationally expensive than any other traditional algorithms.
- There is no specific rule for determining the structure of a neural network. The appropriate network structure is achieved through experience and trial and error.
- The duration of the network is unknown

2.9 Conclusion

In this chapter we presented the different characteristics, advantages and disadvantages of each machine learning model. Now that we have an idea about all of them, we will proceed to transform our data so that it would be optimal for the training and fitting of our models.

Chapter 3

Data Pre-prossessing

3.1 Introduction

To effectively apply machine learning algorithms to a specific challenge, it's crucial to prepare and transform the data into an appropriate format. This involves identifying and removing irrelevant or redundant data, handling noisy or untrustworthy data, and selecting important features for the model. Data preparation and filtering can be time-consuming, but it's essential for achieving accurate and useful results.

In this project, the dataset has already been cleaned, so the focus will be on feature selection, data transformation, scaling, and addressing imbalanced data. Feature selection involves identifying the most important features for the model, while data transformation involves converting data into a suitable form for the algorithm. Scaling is necessary to ensure that all features are weighted equally, and resolving imbalanced data is crucial to avoid biased model outcomes.

By properly preparing and transforming the data, we can ensure that the machine learning algorithms are trained on relevant, high-quality data and produce accurate predictions or recommendations.

3.2 Data Visualisation

Data visualisation refers to the graphic representation of information and data using visual elements such as graphs and maps. Data visualisation makes it possible to explore, see and understand trends or unusual values in the data in a very accessible way. In Machine Learning, data visualisation tools and technologies are essential for analysing huge volumes of information and making decisions based on the data.

3.3 Feature Selection:

Feature selection is the process of identifying critical or influential variable from the target variable in the existing features set.
In this particular set we don't have a lot of features so we proceeded to only eliminate thee ones who based on intuition wouldn't help with our classification problem.
The resulting features are the ones in the following figure:

3.4 Conclusion:

In this chapter we went over the basics of preprocessing our data and preparing it to fit it into the models. We visualized, selected it's features, transformed it and sampled it. At this stage our dataset is ready to be fed into a Machine Learning model.

Chapter 4

Analysis of results

4.1 Introduction

This chapter will showcase the models used in our project and their respective performances after addressing the issue of data imbalance. We will conduct a comparative analysis to determine which algorithm outperforms the others.

4.2 Tools used

4.2.1 Python

Python is a versatile programming language widely used in various fields such as machine learning, big data, and data science. Its simple syntax and powerful libraries make it a popular choice for data analysis and modeling.

Figure 4.1: Python logo

4.2.2 Kaggle Notebook

Kaggle Notebook is a powerful tool for data scientists and machine learning practitioners to collaborate and explore code in a cloud computational environment. The Notebooks, also known as kernels, provide a vast repository of public and open-sourced codes that are reproducible and help in discovering new machine learning projects. With Kaggle Notebook, users can explore and run machine learning codes while being able to easily collaborate with others. It is a convenient platform that enables reproducible and collaborative analysis, making it a valuable asset in the data science and machine learning community.

Figure 4.2: Kaggle logo

4.3 Classification after data imbalance processing

4.3.1 Decision Tree

After applying the decision tree model we obtained the following results:

	precision	recall	f1-score	support
0	0.47	0.55	0.51	86
1	0.81	0.76	0.78	214
accuracy			0.70	300
macro avg	0.64	0.65	0.64	300
weighted avg	0.71	0.70	0.70	300

Figure 4.3: decision tree classification report

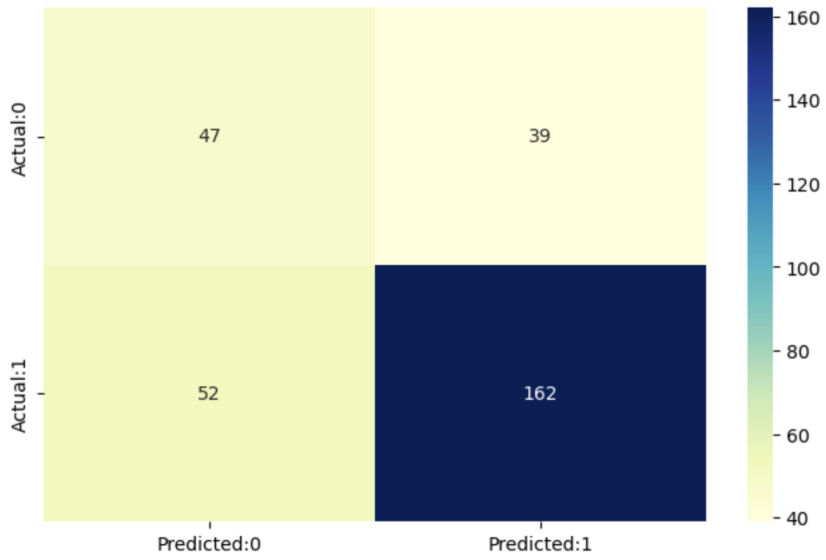
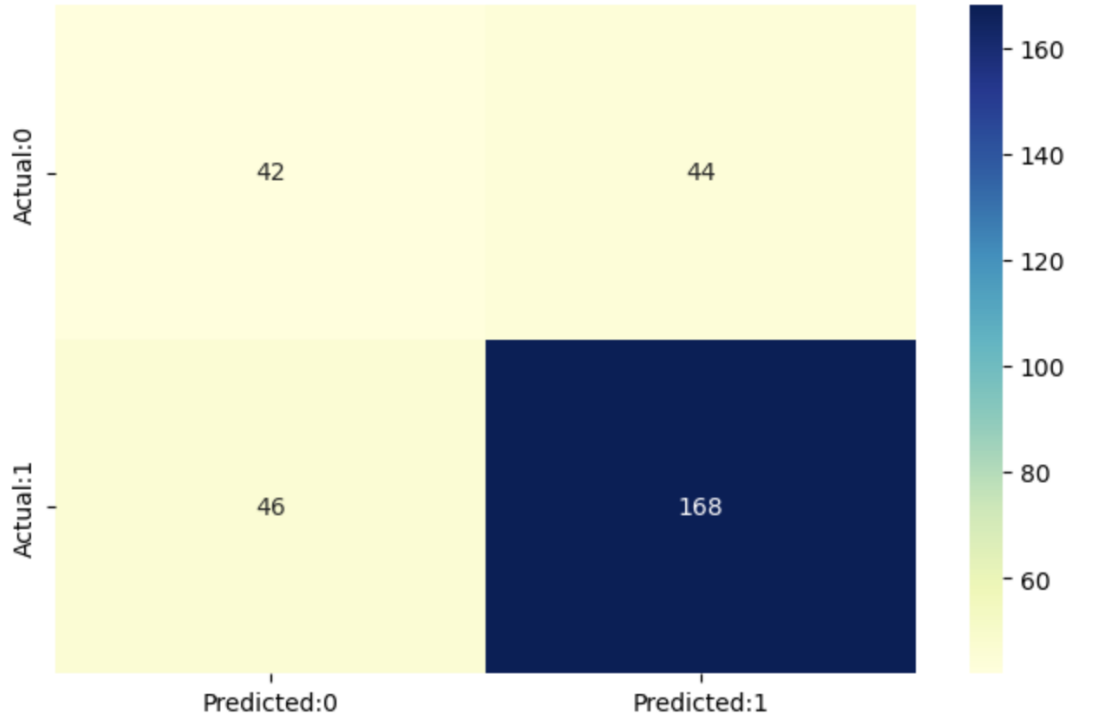


Figure 4.4: decision tree confusion matrix

4.3.2 Neural network

After applying the NN model we obtained the following results:



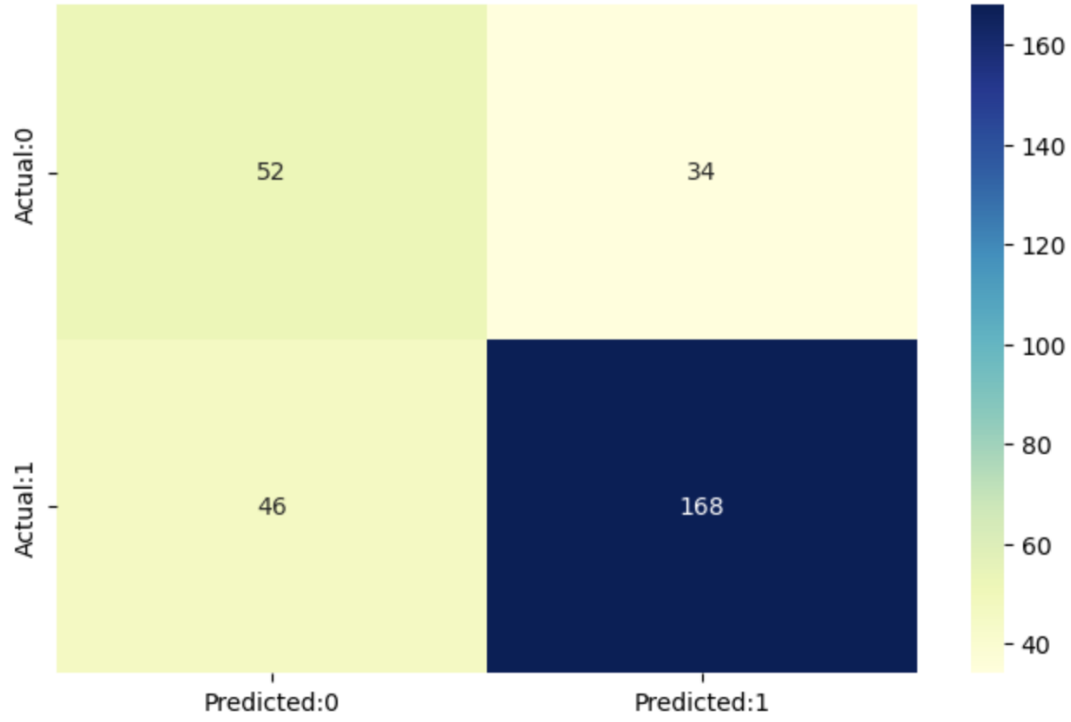

```
3]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.48      0.49      0.48        86
           1       0.79      0.79      0.79       214

    accuracy                           0.70       300
   macro avg       0.63      0.64      0.64       300
weighted avg       0.70      0.70      0.70       300
```

Figure 4.5: Neural network details

4.3.3 SVM

The SVM model gave us the following results :



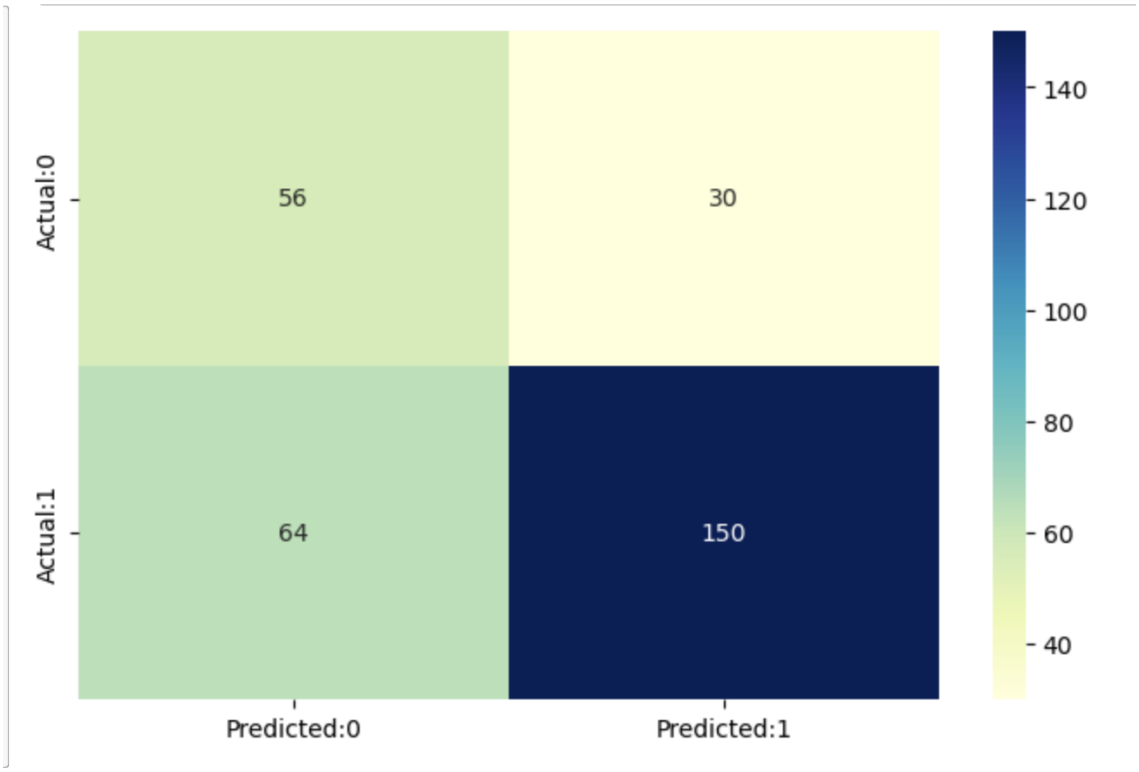

```
]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.53	0.60	0.57	86
1	0.83	0.79	0.81	214
accuracy			0.73	300
macro avg	0.68	0.69	0.69	300
weighted avg	0.75	0.73	0.74	300

Figure 4.6: SVM details

4.3.4 Random Forest

As the higher number of trees gives better performace, we used 100 trees.



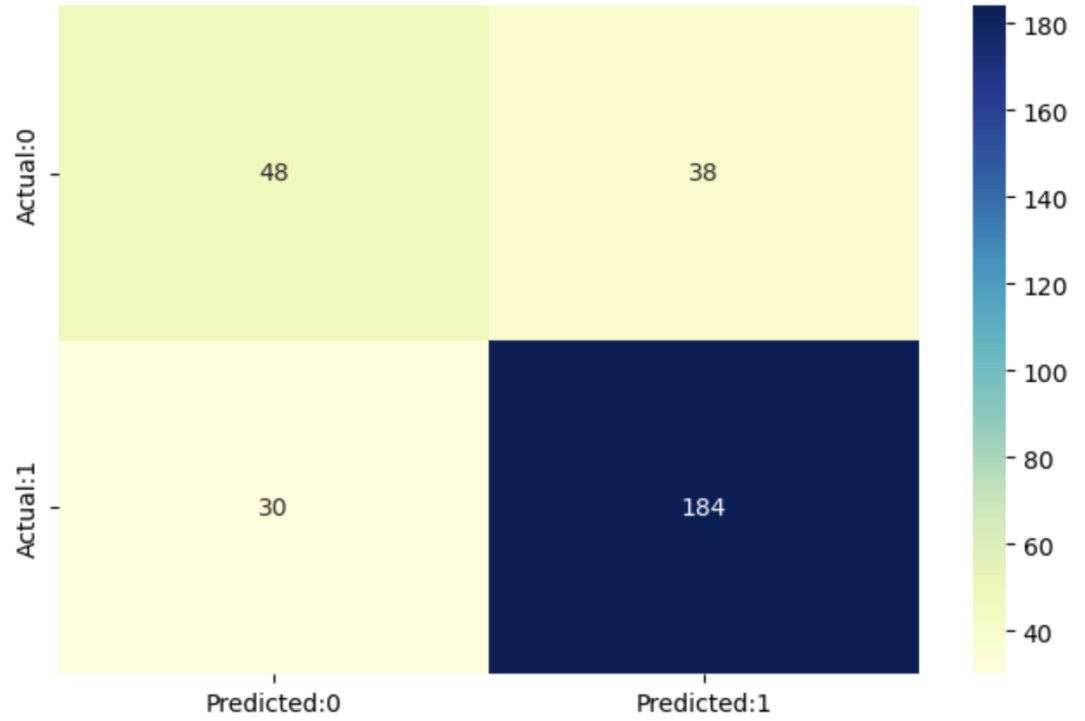

```
]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.47	0.65	0.54	86
1	0.83	0.70	0.76	214
accuracy			0.69	300
macro avg	0.65	0.68	0.65	300
weighted avg	0.73	0.69	0.70	300

Figure 4.7: Random forest

4.3.5 KNN

We chose a K value of 2550001 as the more neighbors we use, the more reliable the results. However, an exaggerated number of neighbors may cause an overfitting problem.



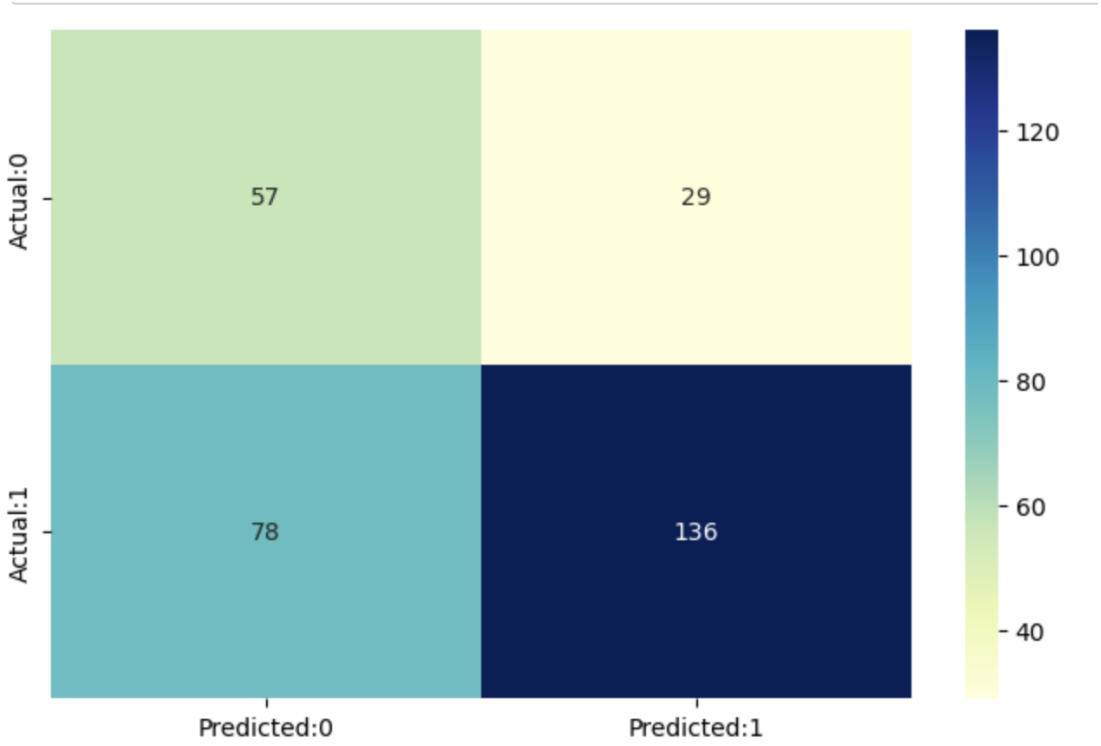

```
]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.62	0.56	0.59	86
1	0.83	0.86	0.84	214
accuracy			0.77	300
macro avg	0.72	0.71	0.71	300
weighted avg	0.77	0.77	0.77	300

Figure 4.8: KNN

4.3.6 K-Means

After applying the K-Means model we obtained the following results:



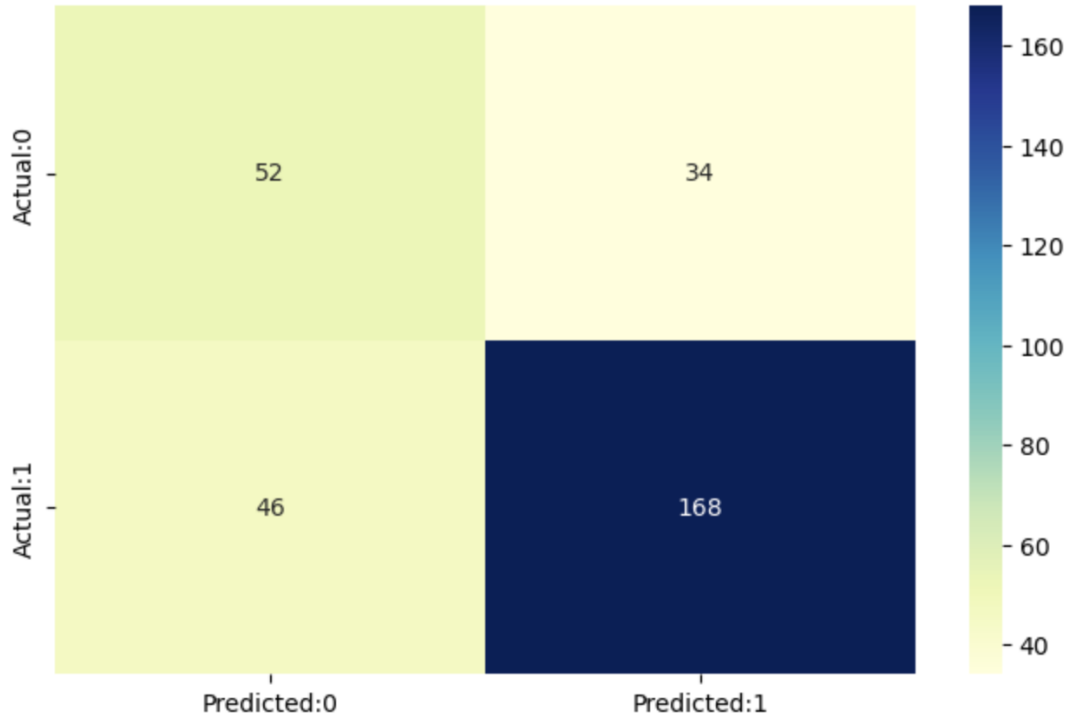

```
7]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.42	0.66	0.52	86
1	0.82	0.64	0.72	214
accuracy			0.64	300
macro avg	0.62	0.65	0.62	300
weighted avg	0.71	0.64	0.66	300

Figure 4.9: K-Means

4.3.7 Naive Bayes

After applying the naive bayes model we obtained the following results:




```
]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.53	0.60	0.57	86
1	0.83	0.79	0.81	214
accuracy			0.73	300
macro avg	0.68	0.69	0.69	300
weighted avg	0.75	0.73	0.74	300

Figure 4.10: Naive bayes

4.3.8 Comparison of the algorithms

-Random Forest: achieved the highest accuracy of 0.756, making it the most accurate model among the ones listed.
-Decision Tree: achieved an accuracy of 0.63, which is relatively lower compared other models.
-Naive Bayes: achieved an accuracy of 0.71, which is decent but not as high as Random Forest.

-Neural Network: achieved an accuracy of 0.71, which is comparable to Naive Bayes, but still lower than Random Forest.
-K-Means: achieved an accuracy of 0.69, which is lower than both Random Forest and Naive Bayes.
-KNN: achieved an accuracy of 0.68, which is relatively low compared to other models.
-SVM: achieved an accuracy of 0.69, which is comparable to K-Means, but still lower than Random Forest.
Based on these accuracy scores, Random Forest seems to be the best-performing model among these ones. However, it's important to note that accuracy alone may not be the only criteria for selecting a machine learning model. Other factors such as model interpretability, speed, and scalability may also need to be considered depending on the specific use case.

4.4 Conclusion

The chapter discussed the programming languages and frameworks utilized for building the credit risk model. Additionally, a comparison was made between the various algorithms implemented in the project.

Conclusion

Looking at our database we can see that we have a lot of columns but a very limited amount of data, when we look at our data we can see that we have categorical variables and numeric variables, which makes our work easier is that we don't have null values. When we look at our correlation we can see that we do not have a very large correlation between the variables, when we look at the exploratory analysis we can see that we have some predominance in our data such as male and single people with their own home and clients who have already applied for credit previously, when we look at our continuous variables we can see that it is usually people in their 30s who apply for credit and the older the person, the smaller the amount of data, when we look at our bivariate analysis some variables caught my attention, like checkingstatus where people who have a value <0 are more likely to be people with a higher credit risk, another variable that also caught my attention was the variable savingstatus, people who have a lower amount of money saved are also more likely to be people with higher credit risk, Another variable that really caught my attention was the Duration variable, which when it has higher values are more likely to have a higher risk profile, the same thing for the creditamount variable but a little lighter. Talking of data modeling, I tested both LabelEncoder and OneHotLabel Encoder for our categorical variables and got a better result with OneHot, looking at our Machine Learning models I could prove what I already had in mind, it was necessary to balance the our target class because we have many more customers with a "good" profile than with a "bad" profile, as we have little data and our target class is well balanced, we can't get an excellent result in our Machine Learning models, most models manages to learn well customers with a "good" risk result but generally does not manage to do very well in models with a "negative" result, in my view the best model we have is the Random Forest model which obtained 75 accuracy, it was very good at predicting customers with a "good" risk profile and it was not so bad at predicting the "Bad" risk customers. When we look at the

most important variables for defining the model, we can see that the variables I predicted in the exploratory analysis were generally used, Duration, Credit Amount and checkingstatus.