# 1. Introduction

## 1.1 Overview

Lending Clubis the one of the world’s first and largest P2P lending platforms that enabled customers to apply for personal loans. Given its feasibility and convenience, the company has experienced a rapid development in recent years. Lending Club Statistic reported that $15.98 billion in loans had been made through its platform up to December 31, 2015 [1].

As P2P lending offers lower access thresholds than traditional banking, their clients are mostly individual or small business owners, and borrowers with low income who had been rejected by traditional banks. Given the characteristics of these customer groups, the applicability of the traditional personal credit evaluation methods becomes highly limited [2]*.* Since 2016, Lending Club, like many other P2P platforms, has been facing various difficulties related to loan default and been struggled to attract big investors. As of December 2020, Lending Club no longer operated as a P2P lender due to its rising financial loss and dropping share price [3]. It is thus crucial for P2P lending to create a safe business space for lenders to get back their invested money from borrowers.

Due to the COVID-19 pandemic, digital financial services have been accelerating significantly. Along with that, the online lending platforms are more likely to rise again. However, there is an urgent need for them to restructure their business strategies and risk assessment models to guarantee a secure business. According to Frank Gerhard [4], lending companies should collect vast amount of data to study customers’ behaviour and build their own real-time decision-making engine to replace humans upon making critical decisions in determining loan defaulters. This approach would be useful to create low-risk business models, attract and maintain investors as well as to guarantee timely approval processes and transactions for customers. It would also enable avoidance of biased decisions that might put investors at risk.

As such, in this project, I will resort to a Machine Learning (ML) approach to analysing behavioural patterns of loan defaulters based on financial information, in order to gain insights about their defaulting behaviour. Moreover, ML models will be built to classify defaulters and non-defaulters.

## 1.2 Research questions

The project research is guided by the following questions

**Q1**. *Is there a significant relationship between clients’ financial information and their defaulting behaviour, and what are the instrumental factors of repayment failures?*

A thorough exploratory data analysis will be performed on a loan dataset to identify behaviours associated with loan defaults.

**Q2**. *Can ML algorithms be utilised to accurately distinguish loan defaulters* *from non-defaulters, and which one is best for this purpose?*

This project will systematically explore and compare a wide range of important ML algorithms, namely, Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naïve Bayer (NB), Decision Tree (DT), Random Forest (RF), and Extreme Gradient Boosting (XGBoost), to obtain a well-performed ML model for predicting repayment failure based on financial features.

**Q3**. *Can we improve the performance of ML models (for default prediction) through performing hyper-parameters tuning to find a set of optimal hyper-parameter values?*

We will systematically carry out hyper-parameters tuning for all the ML classification algorithms being used in this project. Given the various success of this approach for enhancing performance of ML algorithms [5][6], one might expect to gain some additional improvement for the performance metrics.

## 1.3 Aim and objectives

The main **aim** of this project is to develop a ML model that is capable of predicting the chance of loan default based on clients’ financial information, in order to provide suitable support for lending decision making.

This aim will be accomplished through the following **objectives**:

1. Finding research gap: To study existing peer-reviewed publications related to the project topic to identify research gaps;
2. Data collection: To select a suitable secondary dataset which contains customers’ financial data from a lending institute;
3. Exploratory Data Analysis: To carry out data exploration such as anomaly detection, data cleaning and correlation between features in the dataset;
4. Data pre-processing: To perform data transformation and feature selection;
5. Computational method selection: To examine different existing ML classification algorithms for the aim of the research;
6. Model training: To train the model using the training subset (the dataset is divided into training and testing subsets);
7. Model testing: To test the obtained models using the testing subset (containing data not being used in the training process);
8. Evaluation of results: To compare different models based on performance metrics to choose the best one, and compare the project results with similar studies.

4. Results and Analysis

## 4.1 Data exploration

In this section, the dataset is explored. First, all necessary libraries and packages are imported. Function *read\_csv()* is used to import the dataset. Then *info(), describe(), is\_null()* and *dubplicated()* functions are usedtoobtain all the information available in the dataset. Table 2 shows a sample of the dataset.

Table 2. Sample of the dataset "loan\_data.csv"

Table

Description automatically generated

The shape and attributes’ type of the dataset is shown in Figure 8. There are 9578 rows and 14 columns, with “purpose” being the only categorical attribute. The remaining ones are numerical. The class attribute is “not\_fully\_paid”. There are no null values or duplicated values in the dataset which simplify the data cleaning process. For further details, see Figures B1 and B2, Appendix B.

Table

Description automatically generated

Figure 8. Information about attributes’ type and missing data

Below the analysis focuses on the class attribute, categorical attribute, then numerical attributes and their relationships with the target variable.

### 4.1.1 Class attribute

Figure 9 shows the distribution of the class attribute *not\_fully\_paid*. The percentage of fully paid clients is 83.99% and that of not fully paid clients is 16.01%. Clearly, the dataset is highly imbalanced (for more details, see Figure B3, Appendix B).

Chart, bar chart

Description automatically generated

Figure 9. Distribution of target variable “not\_fully\_paid”

### 4.1.2 Categorical attribute

The only categorical attribute is *purpose*. Figure 10 shows its different values and their quantities. “Debt\_consolidation” and “credit\_card” are the most popular purposes of borrowing money.

Chart

Description automatically generated

Figure 10. Count of loans by purpose of borrowing

Next, the relationship between *purpose* and the class variable is explored by plotting a stacked bar chart percentage of fully\_paid/not\_fully\_paid by purpose (Figure 11). It can be seen that “small\_business” has the highest rate of not fully paid loans, suggesting that small businesses usually face greater financial challenges compared to others. Thus, making profits in their business is typically not possible in short amount of time and they end up being a loan defaulter. People who borrowed money for “major\_purchase” or pay back “credit\_card” purpose have greater chance to repay their loan than others. This findings can be useful for loan approval decision-making.

Chart, bar chart

Description automatically generated

Figure 11. Percentage of not\_fully\_paid/ fully\_paid by purpose of borrowing.

### 4.2.3 Numeric attributes

Correlation heatmap was plotted to find the relationship between numeric variables (Figure 12). There are not many strong correlations between attributes. The strongest correlation is between *fico* and *int\_rate* (corr = -0.71), which is expected given that someone with a lower fico score will have a higher interest rate.

Chart, treemap chart

Description automatically generated

Figure 12. Correlation of numeric variables

Figure 13 shows correlations of the class attribute with all other attributes, which are not particularly strong. *Credit policy, int\_rate, fico, inq\_last\_6mths* and *revol\_util* have the highest correlations with *not\_fully\_paid*.

Chart, bar chart

Description automatically generated

Figure 13. Correlation with class attribute comparison

Next, histograms are built to see the distribution of the numerical variables (Figure 14). It can be seen that *log\_annual\_income* (borrower’s income that was down scaled using log transformation) is the only feature with a normal distribution, while others are highly skewed. As such, logarithmic transformation will be used to reduce skewness of other attributes. It is also clear that most distributions of the two target groups, fully paid and not fully paid customers, are very similar. This raises a concern that there might not be a clear pattern that ML algorithms can pick up and differentiate between the two groups.

Graphical user interface, website

Description automatically generated

Figure 14. Distribution of numeric variables

Now, the numeric attributes that have high correlations with the class attribute, namely *credit\_policy, fico, int\_rate, inq\_last\_6mths* and *revol\_util* are explored in detail. Box plots are created to display distributions of these attributes by groups of fully paid and not fully paid customers.

***credit\_policy****:* Figure 15 shows that clients that do not meet credit criteria had higher chance of becoming a loan defaulter. However, there is approximately 13.15% of people satisfying the credit criteria who still end up being defaulters. That raises an issue about the credit accepting process. The lenders might need to reconsider their credit policy to guarantee that people satisfying credit policy will repay their loan.

|  |  |
| --- | --- |
| Chart, bar chart  Description automatically generated | Chart, bar chart  Description automatically generated |

*Figure 15. Counts of clients according to credit criteria (left) and percentage of fully paid/not fully paid clients in each type of credit policy (right).*

***fico****:* It can be seen from Figure 16 that fico score of defaulters are generally smaller than fico of good borrowers.

Chart, box and whisker chart

Description automatically generated

Figure 16. Box plot of fico score by fully paid and not fully paid borrowers

***int\_rate****:* Figure 17 shows that not fully paid loan usually have a higher interest rate than fully paid loan.

Chart, box and whisker chart

Description automatically generated

Figure 17. Box plot of interest rate by fully paid and not fully paid borrowers

***inq\_last\_6mths***: Figure 18 shows that defaulters usually have a higher number of enquiries in the last 6 months. There is 25.58% of customers with more than 2 enquires (third quantile of inq\_last\_6mth attribute) in the last 6 months ending up being a defaulter. For more details, see Figure B8, Appendix B.

Chart, box and whisker chart

Description automatically generated

Figure 18. Box plot of inq\_last\_6mths by fully paid and not fully paid borrowers

Figure 19 depicts the relationship between the interest rate and fico with target groups of customers (fully paid and not fully paid), showing that customers with high fico and low interest rate are less likely to become loan defaulters.

Chart, scatter chart

Description automatically generated

Figure 19. Scatterplot of interest rate and fico, grouped by loan paid

Table 3 explores borrowers that pass/not pass credit policy and have interest rate smaller than 0.1 (first quantile of int\_rate) and fico greater than 737 (third quantile of fico). It can be seen that only **5.79%** of such borrowers are loan defaulters while **28.59 %** clients who do not meet credit policy with interest rate greater than 0.1 and fico lesser 737 wouldn’t repay their loans. More details are given in Figure B9, Appendix B.

Table 3. Relationship between credit policy, interest rate, fico and class attribute.

|  |  |  |  |
| --- | --- | --- | --- |
| credit\_policy | fico | int\_rate | percentage of defaulters |
| not pass(0) | < 737 | > 0.1 | 28.59 % |
| >=737 | <=0.1 | 11.11 % |
| pass (1) | < 737 | > 0.1 | 15.72 % |
| >=737 | <=0.1 | 5.79 % |

In conclusion, the findings from our thorough data exploration provide important and useful insights into clients’ behaviour. Namely, it can be seen that loans not fully paid records usually do not meet credit policy, have higher interest rate and low fico. They tend to have a larger number of enquiries in the last 6 months. Also, by purpose of borrowing, the proportion of defaulters in “small\_business” is significantly higher than in other groups of customers. In the following, we will combine EDA observations with a feature selection algorithm to develop an effective defaulter prediction model.

## 4.3 Data pre-processing

This process includes the following steps: Logarithmic transformation, One hot encoding, Train test split, SMOTE, Feature Scaling and Feature Selection.

### 4.3.1 Logarithmic transformation

Logarithmic transformation is performed to reduce skewness of some numeric attributes that are highly correlated to the class attribute, namely instalment, fico, day with credit line. No log transformation is done for other attributes because they do not follow a log-normal distribution. According to Feng et al., applying log transformation on right skewed data can make the distribution more skewed than the original one [50]. Figure 20 shows the distribution of numeric attributes after log transformation. Comparing Figures 13 and 20, it can be seen that the features’ skewness is slightly reduced.

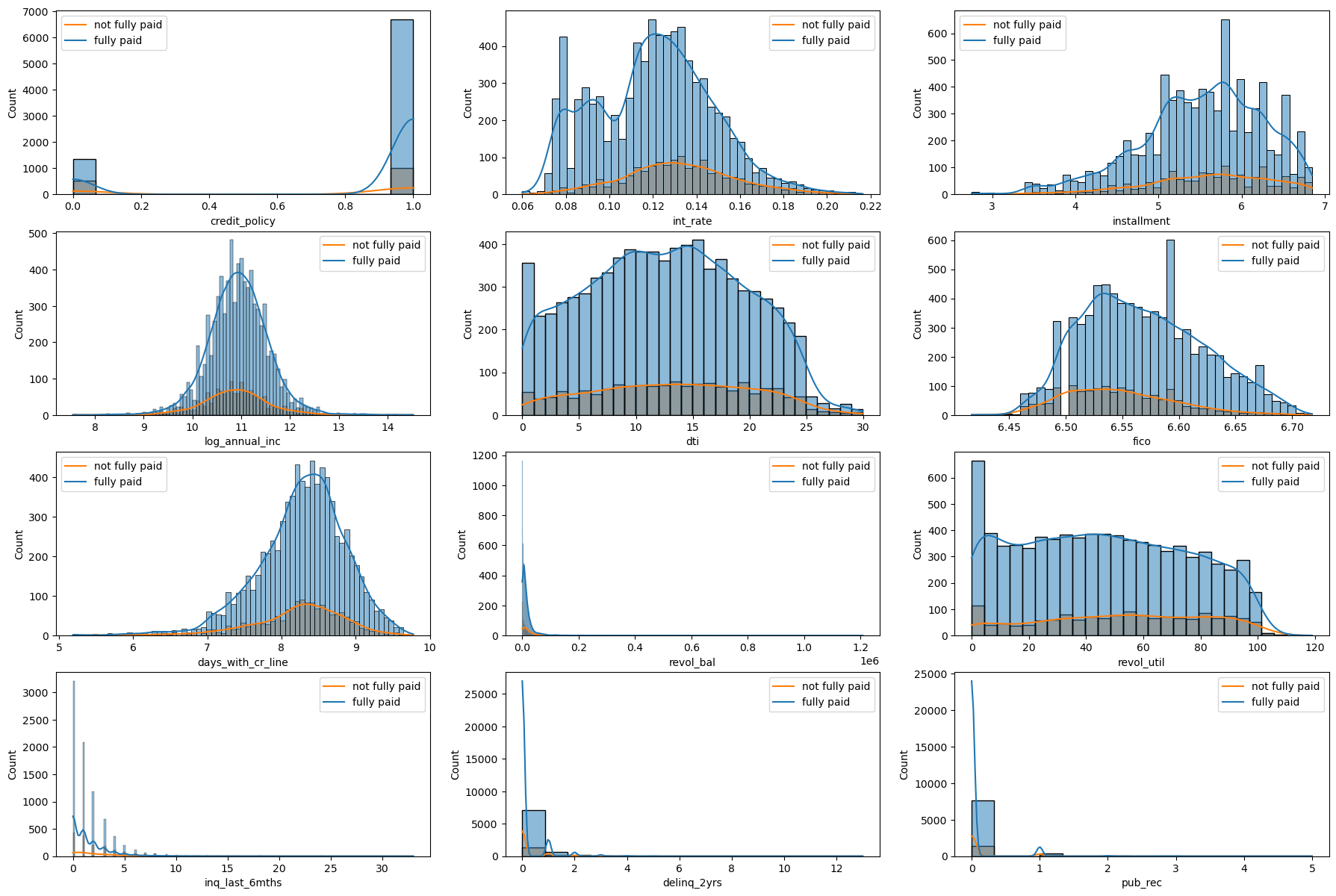


Figure 20. Distribution of numeric attribute after log transformation

### 4.3.2 One hot encoding

First, a matrix of independent variables and dependent variable vector is created from the dataset. Then One hot encoding is performed to transform categorical feature *purpose* into a dummy variable to improve performance of ML algorithms [28]. Figure 21 shows a sample of independent variables after One hot encoding.

Graphical user interface, text, application, email

Description automatically generated

Figure 21. Sample of independent variables after One hot encoding

### 4.3.3 Train-test split

The dataset is split into training and testing sets using stratify split (with test size = 0.2) to have the same proportion of defaulters/non-defaulters in these sets. Figure 22 shows the total numbers of defaulters and non-defaulters in training set (left) and testing set (right).

Chart, bar chart, waterfall chart

Description automatically generated

Figure 22. Count of defaulters/non-defaulters in training set (left) and testing set (right) after train test split

### 4.3.4 Oversampling data by SMOTE

Because the dataset is highly imbalanced, with a proportion of defaulters/non-defaulters approximately 19%, SMOTE is used to oversample the minority class (defaulters) in the train dataset. It is necessary because using a highly imbalanced dataset would lead to ML models biased toward majority class [31]. It is not necessary to perform SMOTE on test data to avoid data leakage. Figure 23 shows the number of defaulters (1) and non-defaulters (0) in the training set after SMOTE.

Chart, bar chart, treemap chart

Description automatically generated

Figure 23. Counts of defaulters (1) and non-defaulters (0) after SMOTE

### 4.3.5 Feature scaling

Feature scaling converts all features to the same scale so that one feature does not dominate others: if one feature is over-dominant, others will be neglected by ML models. Feature scaling is performed using standardization because it is more robust to outliers [12].

### 4.3.6 Feature Selection

In this project, RFE is used with LR as the base algorithm to perform feature selection. Based on EDA, the number of features is set to 12.

As outcomes of RFE, the selected features are: *encoded purposes, credit\_policy, inq\_last\_6mths, revol\_ball, fico, log\_annual\_income and installment*. This result is similar to the result obtained from EDA. In EDA, a correlation exists between the *int\_rate* and the class attribute, but the correlation coefficient between *int\_rate* and *fico* is relatively high (0.71). It might be the reason why *int\_rate* is not chosen by RFE.

## 4.4 Result of different ML algorithms

### 4.4.1 Logistic Regression

First, the ML model is trained with LR on the training dataset, using *LogisticRegression library*(Scikit learn). The built model then is tested, with results described in Table 4.

Table 4. Confusion matrix and classification report of LR model

Table

Description automatically generated

It can be seen that LR model’s performance is relatively low, with 0.64 accuracy and 0.62 macro average recall. LR predicts 59% of loan defaults correctly, but out of all loans that model predicted to be defaulted, only 24% actually was (precision of class 1 is 0.24).

To improve the algorithm’s performance, ***parameters tunning*** is performed, using three parameters, solver, penalty and C. GridSearchCV() is used to search for the best parameters values, giving 'C': 0.01, 'penalty': 'l2', 'solver': 'saga' (see details in Figure D2, Appendix D). After that, the model was trained and tested by LR using these values. The outcome is presented in Table 5.

Table 5. Confusion matrix and classification report of LR model after parameter tuning

Table

Description automatically generated

It can be seen that no significant improvement was achieved through parameters tunning. Accuracy and recall scores remain the same as 0.64 and 0.62, respectively.

### 4.4.2 K-Nearest Neighbors

The ML model is trained by KNN on the training dataset using *KNeighborsClassifier library*(Scikit learn). The built model is then tested, with results in Table 6.

Table 6. Confusion matrix and classification report of KNN model

Table

Description automatically generated

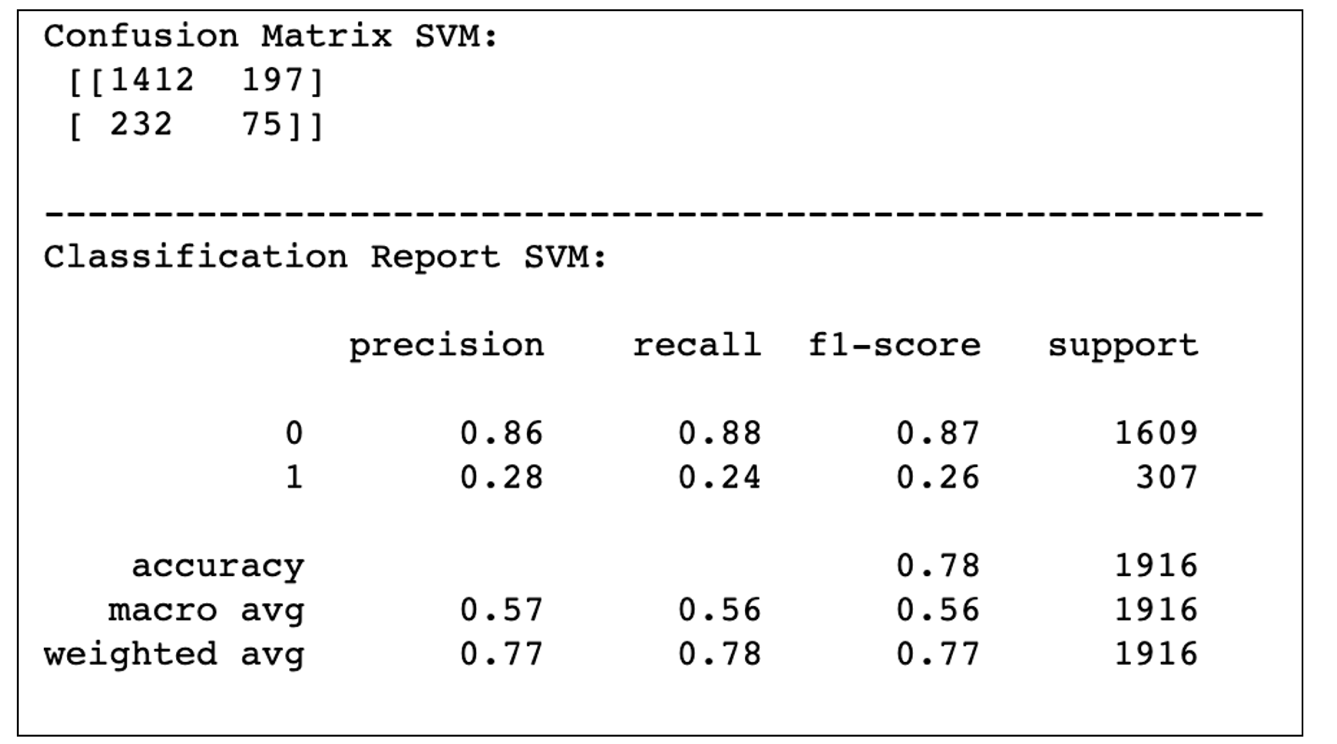
The performance is relatively low: accuracy = 0.67, recall macro average = 0.54. KNN performs better than LR when identifying good lenders, with precision = 0.85, recall = 0.73, f1-score = 0.79, while these scores for finding defaulters (class 1) are quite low.

KNN classifies new data based on its distance from neighbours, but as can be seen from EDA, the distribution of two classes in the target variable are similar, which might be the reason behind the poor performance [52].

### 4.4 .3 Support Vector Machine

The ML model is trained by kernel SVM on the training dataset using *SVC library*(Scikit learn). The built model is then tested, with results given in Table 8.

Table 8. Confusion matrix and classification report of SVM model



The performance obtained showed a higher accuracy (0.78) compared to LR and KNN. However, recall is lower than that of LR. SVM is also not good in identifying defaulters, with recall, precision, f1\_score all below 0.3. However, SVM performs well in predicting non-defaulters: 88% non-defaulters are correctly predicted.

### 4.4.4 Naïve Bayes

The ML model is trained by NB using *GausianNB library*(Scikit learn). The obtained model is then tested, with results shown in Table 10.

Table 10. Confusion matrix and classification report of NB algorithm

Table

Description automatically generated

NB performance is quite similar to SVM, with 0.71 accuracy and 0.58 recall macro avg. NB performs well on predicting good loans (precision for class 0 equal 0.87).

### 4.4.5 Decision Tree

The ML model is trained by Decision Tree using *DecisionTreeClassifier library*(Scikit learn). The built model is then tested, with results given in Table 12.

Table 12. Confusion matrix and classification report of DT classifier

Table

Description automatically generated

An improved accuracy (0.73) was achieved compared to other algorithms, but recall is lower (0.53). DT performs better at identifying non-defaulters, achieving for class 0, 0.83 recall and 0.85 precision. However, it performs ineffectively in finding defaulters, obtaining for class 1, 0.23 recall and 0.21precision.

### 4.4.6 Random Forest

The ML model is trained by RF using *RandomForestClassifier*. The trained model is then tested, where the results are shown in Table 13.

Table 13. Confusion matrix and classification report of RF algorithm

Table

Description automatically generated

There is a significant improvement in accuracy compared to other algorithm (0.82). It can be explained by the ability of RF to combine the predictions of different trees to generate the final prediction for each input. Indeed, RF performs well in identifying non-defaulters with 0.97 recall and 0.85 precision for class 0. However, it performs very poorly in finding defaulters, with 0.07 recall for class 1.

### 4.4.7 XGBoost

The ML model is trained by XGBoost using *XGBClassifier library*. The built model is then tested. See results in Table 14.

Table 14. Confusion matrix and classification report of XGBoost classifier

Table

Description automatically generated

The best accuracy is obtained compared to other algorithms (0.83), with a relatively low recall (0.53). XGBoost also performs well in identifying non-defaulters, with 97% of non-defaulters correctly predicted. However, it performs poorly in finding defaulters, with 0.09 recall for class 1.

## 4.5 Models comparison, results analysis and further improvement

This section analyses the performance of all the algorithms, to choose the best model for the given dataset. Other techniques will then be applied to further optimise its performance.

From the confusion matrices and classification reports, it appears that all models perform better for the majority class (non-defaulters) than the minority one (defaulters). The tree-based models, i.e. DT, RF and XGBoost, perform well on identifying non-defaulters, but poorly on identifying defaulters, which are the main target of this project.

As mentioned in Chapter 3 (Section 3.5), since our aim is to detect loan defaulters, the essential metric for comparing different ML algorithms’ performances is **recall**. Table 16 compares recall (macro average) and accuracy of all models (in percentage), sorting from highest to smallest by recall values. We observe that LR performs best, followed by NB and KNN. It outperforms other algorithms, even complex ones, such as XGBoost and RF, in predicting defaulters. Our results once more prove the strength and usefulness of LR for various binary classification problems [53]. Our systematic experiments also show that little to no significant improvement was made through parameters tunning.

Additionally, a comparison of the algorithms’ accuracy shows that XGB has the best accuracy (82.93%), followed by RF (82.41%), then DT (73.17) (see also Figure 18, Appendix D)

Table 16. Comparison of recall and accuracy of all models

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Given the high recall of LR, we explore if other ways exist to further improve its recall for the minority class. Indeed, by default LR classifies a data point by its probability of being True with a threshold of 0.5 (e.g. if the probability is greater than 0.5, it is classified as defaulter, and as non-defaulter otherwise). Thus, to improve the prediction in class 1 (defaulters), one might consider lowering the decision threshold, especially if it is not too detrimental for precision [54]. To that end, in Figure 24 precision and recall scores are presented for varying the threshold value.

Chart, line chart

Description automatically generated

Figure 24. Plots of precision and recall scores for varying threshold

Interestingly, it can be seen that the recall increases much faster than precision when reducing the threshold. For example, when the threshold is reduced from 0.5 to 0.4, recall increases significantly from 0.58 to 0.83 (i.e. by 0.25) while precision only reduces slightly from 0.24 to 0.21 (i.e. by 0.03). For additional details, a model with 0.4 threshold is trained and tested (using the same training and testing sets), with results shown Table 17. Note that, the implication of sacrificing precision for recall means many good customers could be refused because the model would classify them as defaulters and wouldn’t accept their application, so one might not want to reduce too much the threshold value.

Table 17. Confusion matrix and Classification report of LR model with threshold equal 0.4

Table

Description automatically generated

**From 179 to 254**

## 4.6 Critical analysis of the study

### 4.6.1 Strengths of the study

As a result of our systematic data exploration, the financial characteristics of non-repayment loans are identified. It can be seen that records of repayment failure usually do not meet the credit policy, have a higher interest rate with a lower fico score and have a larger number of enquiries in the last 6 months.

Moreover, other useful findings from our analysis that can be considered when deciding whether to approve or reject a loan application are:

* A high risk is associated with customers that borrowed money for a small business.
* The chance of a client to become a defaulter significantly increases if the interest rate is greater than 0.1 and fico is smaller than 737 (28.59% of these clients wouldn’t pay their loans while only 5.79% of clients with interest rates smaller than 0.1 and fico greater than 737 are likely to become loan defaulters).
* There is a considerable number of people who met the credit policy criteria but their loan is not fully repaid, so the lending institute should tighten their credit policy criteria.

In short, we were able to prove our hypothesis that with clients’ financial information only, lenders can predict the chance of non-full repayment from clients. ML models were successfully trained to make that prediction. In terms of accuracy, XGBoost and RF have shown the best performance. However, LR clearly outperformed all others for predicting defaulters. By adjusting its decision threshold to 0.4, LR was able to predict 83% of defaulters correctly (with minor impact on its precision). As such, we confirm that lending organisations can benefit from using our model as a supporting mechanism for predicting loan defaulters.

### 4.6.2 Limitations of the study

At the current state, the model’s performance is slightly less than satisfactory, without labelling a significant number of loans as defaulters when they probably would be repaid. Therefore, it should not be used as an automated loan approvals engine. Rather, it should be used to provide supplementary insights for decision makers for loan approvals.