# Quantitative Financial Risk Management

Assignments IV & V (2021): Credit Scoring Model

Kho, Kevin
(2713279)
Romero León, Pedro Carlos
(2710821)

January 19, 2022

## 1 Introduction

The following report compiles an overview, and practical application, of a credit scoring model by deploying a classification technique, namely Logistic Regression, and its comparison with other existing machine learning techniques. Hence, within the fields of Quantitative Financial Risk Management, this report focuses on credit risk, or counter party risk. The model is based on credit information gathered from a P2P lending platform (available at:https://www.prosper.com).

The decision to construct the model based on this information lies in the fact that Peer-to-peer and the FinTech lending space have become a hot topic in the financial service space. FinTech companies are providing the space for borrowers and investors to meet in digital environments and satisfy their capital needs, away from the traditional commercial banking industry. This enables agents to smooth consumption across time without relying on the services of incumbents. Although this enables market participants with a wide variety of projects to develop to acquire funding that would not have been possible otherwise, it is certainly a risky business. It is in the interest of the platforms and the proper investors to have effective methods to assess the credit quality of the borrowers. This is, to analyse their probabilities of defaults, as this would reduce information asymmetry and adjust the pricing of these investments to the outstanding probability of default.

To assist the platform in achieving this, we decided to look at the loan data available and try to predict the likelihood of loans to default and the probability of each borrower participating of the platform to default. The main method for predicting the default of a certain loan member is as follows:

1. Initially, for each loan in the loan dataframe it is predicted whether or not the loan has defaulted. Logistic regression is used for these predictions.
2. By matching the loan ID's (along with its prediction of default) with the member ID's, the probability of default (PD) for each loan member is computed. This is done this way since each loan member is allowed to hold multiple loans and thus the PD of each member is taken to be the mean of the prediction of default of the loan.

This document will focus on the first point as this step of the method is the main prediction step for which LR and ML models can be deployed. The second step is simply a deterministic computation step based on the prediction made in step 1.

## 2 Logistic Regression

### 2.1 Main Intuition

Being part of the limited dependent variables models family, the Logistic regression presents itself as a good candidate to deal with binary dependent variables, as it overcomes most of the limitations of the linear probability model, in that the traditional LTPM, depending on the problem at hand, might deliver negative estimated probabilities or even probabilities above 1.

Logistic regression overcomes this by transforming the regression problem by means of the sigmoid function. This transformation bounds the fitted values between (0,1). Consider the sigmoid function as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

Visually, the fitted regression model would acquire an S-shape, due to the transformation induced.

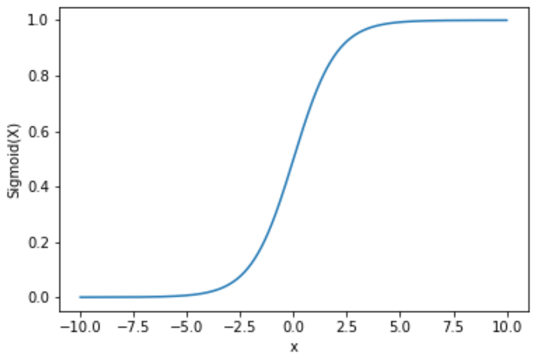


**Figure 1:** Sigmoid function representation

### 2.2 Mathematical Foundations

Consider the following formulation:

$$P_i = E[y_i = 1|x_i] = \frac{1}{1+e^{-z_i}} = \frac{1}{1+e^(-\beta_0 + \beta_1 x_i + .. + u_i)} \tag{2}$$

Setting the residual $(u_i)$ to its expected value $E[u_i] = 0$, so that we have:

$$P_i = E[y_i = 1|x_i] = \frac{1}{1+e^(-\beta_0 + \beta_1 x_i + .. + \beta_n x_n)} \tag{3}$$

It is easy to demonstrate that as $Z_i$ ranges from $(-\infty, \infty)$, $P_i$ will range from (0,1) asymptotically. However, the relationship will no longer be linear, and thus, traditional estimation through OLS is no longer possible. We also need to account for the probability of $y_i \neq 1$, or in other words $y_i = 0$. Taking this into consideration, the likelihood function would look as:

$$L_i = (\frac{1}{1+e^{-z_i}})^{y_i}(\frac{1}{1+e^{z_i}})^(1-y_i) \tag{4}$$

The function needed will be based on the joint likelihood of all N observations rather than on a single observation. Assuming that each observation is independent, the joint likelihood will be described as:

$$L_i = (\theta|x_{2i}, x_{3i}, x_{4i}, ..., x_{ki}; i = 1, 2..., N) \tag{5}$$

Thus, the likelihood of the set of parameters $(\beta_0 + \beta_1 + .. + \beta_k)$ would be given by:

$$L_(\theta) = \prod_{i=1}^{N}(\frac{1}{1+e^{-z_i}})^{y_i}(\frac{1}{1+e^{z_i}})^(1-y_i) \tag{6}$$

However, for computational reasons, it turns out to be easier to maximise an additive function than a multiplicative function, hence, by taking logarithms we transform the function into the Log-Likelihood Function:

$$LLF_(\theta) = \sum_{i=1}^{N}(y_i ln(1+e^{-z_i}) + (1-y_i)ln(1+e^{z_i})) \tag{7}$$

## 3 Data Visualizations & Pre-processing

The dataset provided comes twofold. One file named *prosper_data.csv* contains the actual loan information consisting of all members and all loans yielding 82 features and 10000 data points. The second file named *prosper_definitions.csv* contains the feature description of all the features that are either related to loan members or related to the loans. This file consists of 3 features and 54 rows where each row represents a feature that relates to either two. The features in this dataset are *Variable*, *Category*, and *Description* where the first contains the feature names, the second categorizes each feature in *member* or *loan*, and the third one provides a description for each feature.

From these two files it can be deduced that from the 82 features included in the data file, only 54 have relevance towards the loans or the loan members. This means that as a first feature selection step, we can already decide to reduce the amount of features from 82 to 54 and thus only keep the features that are described in the definitions file.

The goal is to predict for each loan whether the loan will default or not and to estimate for each member their probability of default (PD). Therefore, for further exploration we decide to split the data into 2: we create a dataframe containing all the features that are loan-related and another dataframe containing all the featurs that are member-related. This enables us to analyze loan characteristics and member characteristics separately. The dataframe *df_loan* consists of 16 features and 10,000 data points and the dataframe *df_members* consists of 36 features and 10,000 data points.

### 3.1 Loans

The loans are categorized into 11 categories which can then be further divided up into the 2 main categories: *Current* which consists of the loans that fall under the *Current* loan status which are currently still open (and thus have not yet been labeled as defaulted or non-defaulted) and

*Non-current* which consists of all loans that do not fall under *Current* (these include *Completed*, *Defaulted*, *Chargedoff* etc.) which are already labeled as defaulted or non-defaulted.

The non-defaulted loans under *Non-current* are the loans that are categorized as *Completed* and *FinalPaymentInProgress* since these loans have successfully been paid-off or are in the process of doing so. The loans that fall under the other categories are labeled as defaulted. A distribution in absolute numbers for each loan-type is tabulated in Tab. 1 and the distribution of their relative amounts are visualized in Fig. 2.

**Table 1:** Count of the different Loan Status categories in the dataset (absolute terms)

| Loan Status | Number of credits |
|---|---|
| Current | 5021 |
| Completed | 3263 |
| Chargedoff | 1084 |
| Defaulted | 449 |
| Past Due (1-15 days) | 68 |
| Past Due (31-60 days) | 28 |
| Past Due (61-90 days) | 28 |
| Past Due (91-120 days) | 24 |
| FinalPaymentInProgress | 19 |
| Past Due (>120 days) | 2 |
| Total | 10,000 |

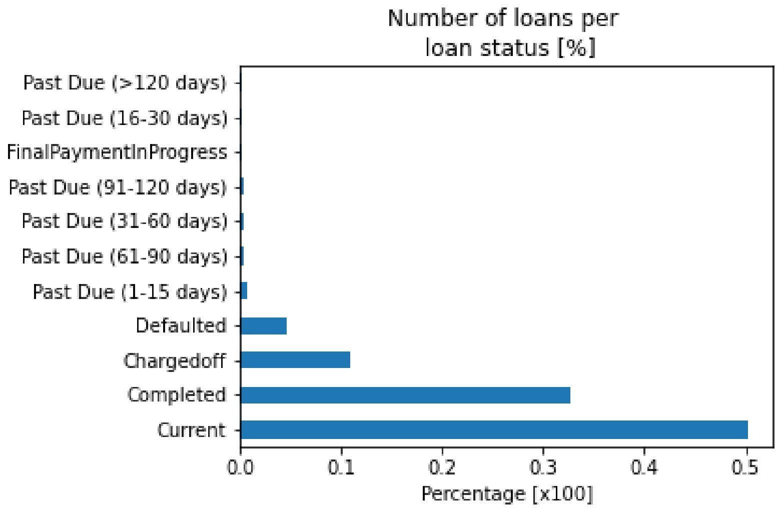


**Figure 2:** Number of loans for each loan status type

From this we can see that current outstanding loans form approximately 50.21% of all loans and the aggregated amounts of the other loan-types form the other 49.79%. This implies that the *Non-current* category consists of 4979 data points with 15 features (excluding our target feature *LoanStatus*). This set of data will serve as our target series with binary values yielding:

- 0 of the *LoanStatus* either has the value *Completed* or *FinalPaymentInProgress* (i.e. has not defaulted)
- 1 otherwise (i.e. has defaulted)

Figure 3 offers a first glance representation of both classes´ distributions. It is self-evident that this is an unbalanced situation, where the majority of the observations pertain to the non-defaulted class (0.0). It is important to bare this mind when splitting the input data into training and testing data. One needs to ensure that, within the splits, the same proportion of defaulted and non-defaulted are present as as in the original set, regardless of the overall size of the split. There are several techniques like over-sampling or down-sampling to face this situation. However, we decided to allow for the stratify parameter in our Sklearn splitting class.

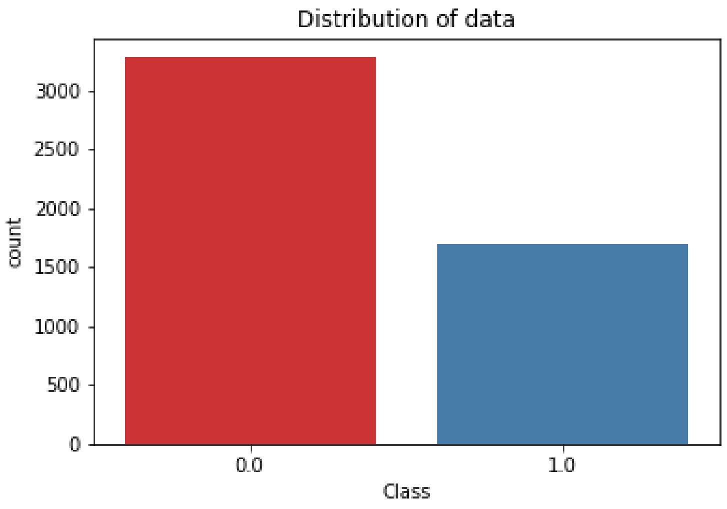


**Figure 3:** Binary distributions

**Missing Values** Next, we analysed the amount of missing values in this new dataframe. We could observe that at least four of our features contained a high number of missing values.

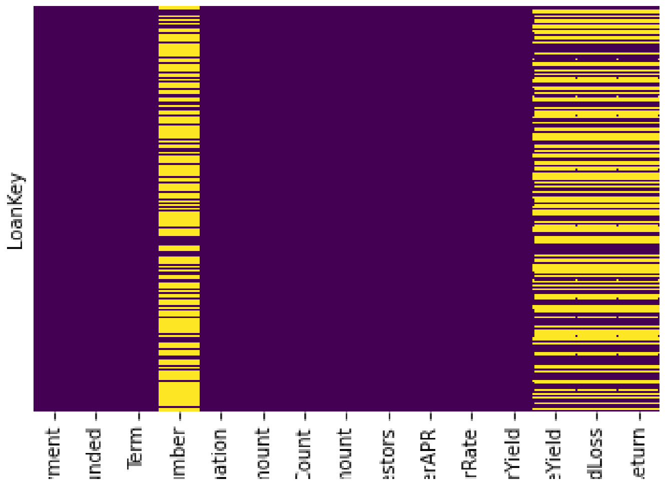


**Figure 4:** Distribution of missing values per feature

In order to quantify the total number of missing observations and to make a decision on whether or not to drop them from our study, we present below the number of missing values, in absolute and relative terms, for each of the variables considered.

**Table 2:** Missing values in the dataset

| Features | absolute | as (%) |
|---|---|---|
| MonthlyLoanPayment | 0 | 0 % |
| PercentFunded | 0 | 0% |
| Term | 0 | 0% |
| LoanFirstDefaultedCycleNumber | 3446 | 69,2% |
| LoanMonthsSinceOrigination | 0 | 0% |
| LoanOriginalAmount | 0 | 0% |
| InvestmentFromFriendsCount | 0 | 0% |
| InvestmentFromFriendsAmount | 0 | 0% |
| Investors | 0 | 0% |
| BorrowerAPR | 0 | 0% |
| BorrowerRate | 0 | 0% |
| LenderYield | 0 | 0% |
| EstimatedEffectiveYield | 2564 | 51.4% |
| EstimatedLoss | 2564 | 51.4% |
| EstimatedReturn | 2564 | 51. 4% |

**Feature Selection** Four of the features contain at least 50% of missing values. Although we could proceed to the imputation of nan´s using techniques such as a KNN, RidgesBayes or more simplified approaches, like replacing the values with the feature´s overall median, mean or most frequent value, we decided to drop these columns from the dataset. The reason behind such decision is that we considered the amount of missing values to be too high.

The first step undertaken in our exploratory data analysis was to obtain the descriptive statistics of the variables under consideration.

**Table 3:** Descriptive statistics of features in dataset

| | Statistics | | | | |
|---|---|---|---|---|---|
| Variable | Count | mean | std | min | max |
| MonthlyLoanPayment | 4979 | 223.72 | 181.17 | 0.00 | 1563.21 |
| PercentFunded | 4979 | 0.99 | 0.02 | 0.70 | 1.00 |
| Term | 4979 | 37.02 | 7.57 | 12.00 | 60.00 |
| LoanMonthsSinceOrigination | 4979 | 54.51 | 26.81 | 1.00 | 97.00 |
| LoanOriginalAmount | 4979 | 6255.90 | 5145.50 | 1000.00 | 30000.00 |
| InvestmentFromFriendsCount | 4979 | 0.04 | 0.25 | 0.00 | 6.00 |
| InvestmentFromFriendsAmount | 4979 | 26.37 | 365.40 | 0.00 | 15000.00 |
| Investors | 4979 | 103.17 | 108.22 | 1.00 | 1035.00 |
| BorrowerAPR | 4979 | 0.22 | 0.09 | 0.01 | 0.41 |
| BorrowerRate | 4979 | 0.20 | 0.08 | 0.00 | 0.36 |
| LenderYield | 4979 | 0.19 | 0.10 | -0.01 | 0.35 |

From the data presented above, the presence of outliers in some of the variables can be deduced. This type of isolated observation could influence the performance of the models presented. In order not to alter the results of the following steps in the analysis, we reserve the treatment of outliers until the selection of variables to be included in the model has been completed.

As part of the exploratory analysis of the data, in Annex 7.1 we provide an overview of the individual, and pairwise, distributions of the data in a matrix graphic format (See figure 14). This figure suggests the existence of a possible problem of exact multicollinearity between some of the variables under consideration. Prior to constructing the models, it is important to ensure that highly correlated variables are omitted in order to avoid spurious results.

Although a formal correlation analysis will be conducted to confirm this relationship, a strong linear relationship between some of the variables is already apparent at first glance. By pairs, the variables most suspected are: Lender Yield & Borrower Rate, LoanOriginalAmount & Monthly-LoanPayment; Borrower APR & Lender Yield; Borrower APR & Borrower Rate. In addition, two other variables, Investors & MonthlyLoanPayment, also showed signs of the possible correlation between them, albeit to a lesser degree than the above.

In addition to using field specific domain knowledge to determine whether these results are meaningful, for example, it becomes self-evident that the interest rate at which the borrower receives the principal (Borrower Rate) will be exactly the same as the rate at which the lender makes a return on his investment, in the form of a loan (LenderYield), we decided to use a filtering method, namely correlation analysis. Our computations are based on the Pearson's´ Correlation Coefficient. In the literature, variables are considered to be highly correlated if they display a coefficient ranging in between $70 - 90\%$, hence, to ensure accurate results of the models, we decided to establish our dropping threshold at $\pm 70\%$. The results of the analysis are visually presented in Annex 7.2 Figure 15.

At a glance, the graph confirms a high correlation between the pairs of variables mentioned above. As to decide which variables to drop, we decided to run separate classification tasks to see how the accuracy of the models got affected by dropping individual variables, and different combinations of them (top-down approach). After several trials we came to conclude that the highest attainable performance comes from dropping the variables: *BorrowerAPR*; *LoanOriginalAmount*;*LenderYield*. After dropping the variables we obtained an accuracy where the accuracy of the model was boosted to 67.269%. However, after having finalised the feature selection, we proceeded with winsorizing at the 1% and 99% level the following variables: *MonthlyLoanPayment*; *InvestmentFromFriendsAmount*; *Investors*. This slightly improved the accuracy of the model, from an initial 67.26 % to 67.36%. Please see Section 4 for the presentation and discussion of the results.

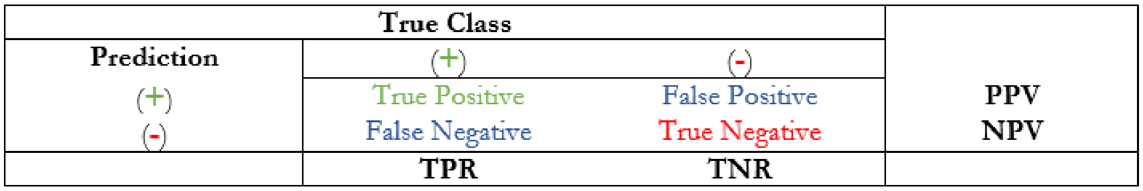
After having cleaned the set from missing observations and highly correlated variables, we ended up with a features dataset with 4979 different instances and 8 different dimensions that would be employed to fit our models. To conclude the feature selection and preprocessing section, it is worth mentioning the standardisation of the variables, subtracting the mean and dividing by their standard deviations (seeking to ensure an equitable contribution of the variables in the estimations), and the partition made between training and testing data to avoid overfitting the models. Thus, our models would be trained using 70% of the data and tested with the remainder 30%.

## 4 Evaluation Metrics

In this section, we present a brief explanation of the evaluation measures used to determine the appropriateness of the models proposed. In this section, we present a brief explanation of the evaluation measures used to determine the appropriateness of the models proposed. For the case at hand, we focus on the evaluation of imbalanced binary situations. By imbalanced we mean that both types of label distributions are imbalanced. Often the lower class is of the greatest importance than the higher class. By evaluating these kinds of classifiers with accuracy measures, like standard classification errors or loss functions, we incur the risk of the accuracy paradox, as the binary imbalanced situation might be extremely misleading.If we could establish the costs of the imbalances, we could rather perform this task more accurately. There are alternative metrics for binary systems.

**Confusion Matrix** Tabulates the predictions against actual classes, basically, how many correct and incorrect values have been predicted compared to the actual values. We divided it into positive and negative classes. Actually, the fact that we call it positive it has nothing to do with its sign. We split the entries in the confusion matrix in:

**Figure 5:** Confusion Matrix



|  | **True Class** |  |  |
|---|---|---|---|
| **Prediction** | (+) | (-) |  |
| (+) | True Positive | False Positive | **PPV** |
| (-) | False Negative | True Negative | **NPV** |
|  | **TPR** | **TNR** |  |

Where we can identify several factors:

- True Positives or positive class. For the case at hand this class comprises Non-Defaulted loans.
- False Positives. Observations wrongly predicted as Non-Defaulted by the model, when they actually were defaulted.
- True Negatives. Correctly predicted as defaulted.
- False Negatives. Predicted as defaulted when in reality were non-defaulted

**Receiver Operating Characteristics (ROC)** The basics ROC metrics are defined by the entries of the confusion matrix. We basically calculate several ratios that are informative about the scores of each of the entries. For instance:

$$PositivePredictiveValue(PPPV) = \frac{TP}{TP + FP} \tag{8}$$

$$TrueNegativeRate(TRR) = \frac{TN}{TN + FP} \tag{9}$$

$$NegativePredictiveValue(NPV) = \frac{TN}{TN + FN} \tag{10}$$

$$NegativePredictiveValue(NPV) = \frac{TN}{TN + FN} \tag{11}$$

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} \tag{12}$$

These ratios answer the questions:

- PPV. If we predict a 1, how likely is it to be a real 1?

- TRR. How many of the values did we predict as 0?
- NPV. If we predict a 0, how likely is it to be a real 0?
- TPR. How many of the values did we predict as 1?

**Accuracy** Counts the number of correct classifications and express it as a ratio.

**Precision** The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

**Recall** The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

**F1-Score** F1 score for each class. The F1 score is the harmonic mean of the precision and recall. The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. The F-score is also used in machine learning. However, the F-measures do not take true negatives into account, hence measures such as the Matthews correlation coefficient, Informedness or Cohen's kappa may be preferred to assess the performance of a binary classifier.

**Support** The support is the number of occurrences of each class in $y_t rue$.

## 5 Model Fitting Results

In the following section, one may find our main results from fitting the Logistic Regression to the data, excluding those variables that showed higher correlations and did boost the model's performance once dropped. Its performance is evaluated by considering the following diagnostics: Odds ratio (OR), confusion matrix, and ROC curve. Moreover, the relevance of the features used will be evaluated by feature importance plots, information value diagnostics, and SHAP.

### 5.1 Logistic Regression Performance

**Parameter estimation** For fitting the LR function, the loan data is split into 70% training data and 30% test data where the data is stratified to maintain the same data distribution as in the original dataset. This results in the following parameters and their corresponding odds ratios (OR):

**Table 4:** Logit Regression Results

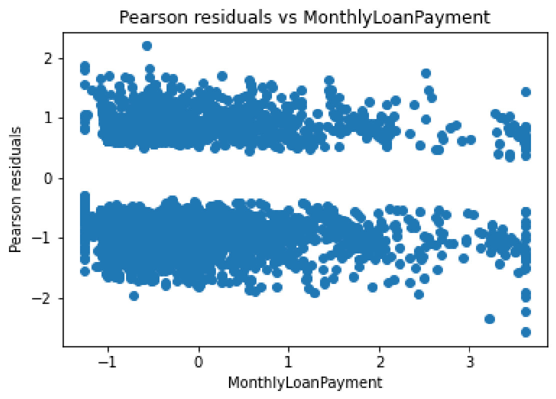
| | **coef** | **Odds Ratio** | **std err** | **z** | **P> \|z\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|---|
| **MonthlyLoanPayment** | 0.2747 | 1.3161 | 0.049 | 5.647 | 0.000 | 0.179 | 0.370 |
| **PercentFunded** | -0.0069 | 0.9932 | 0.036 | -0.193 | 0.847 | -0.077 | 0.063 |
| **Term** | 0.2227 | 1.2495 | 0.038 | 5.807 | 0.000 | 0.148 | 0.298 |
| **LoanMonthsSinceOrigination** | 0.3116 | 1.3656 | 0.039 | 7.989 | 0.000 | 0.235 | 0.388 |
| **InvestmentFromFriendsCount** | -0.0396 | 0.9612 | 0.056 | -0.710 | 0.477 | -0.149 | 0.070 |
| **InvestmentFromFriendsAmount** | 0.0009 | 1.0009 | 0.056 | 0.015 | 0.988 | -0.109 | 0.110 |
| **Investors** | -0.1546 | 0.8568 | 0.051 | -3.042 | 0.002 | -0.254 | -0.055 |
| **BorrowerRate** | 0.5139 | 1.6718 | 0.041 | 12.684 | 0.000 | 0.434 | 0.593 |

Note that the coefficients indicate the increase/decrease of the log-odds of each feature rather than of the odds themselves. Therefore, by taking the exponential of the coefficients we obtain the OR which do indicate the effect of the features on the increase/decrease on the odds. For example, the odds of defaulting increases by 1.3161 with a unit increase in *MonthlyLoanPayment.*

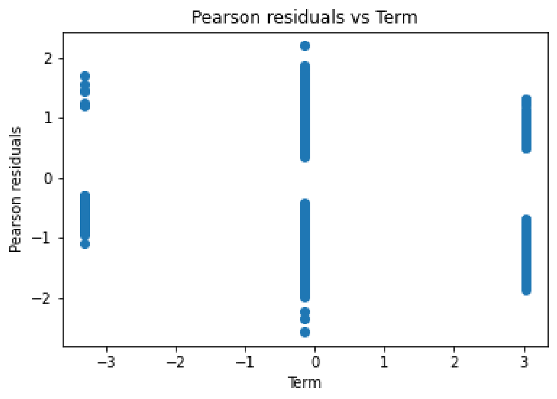
**Residual Analysis** The residuals indicate the difference between the fitted and observed values. In particular it indicates the difference between the one-step ahead prediction output and the measured output from the dataset. Essentially, residuals represent the portion of the validation data which is not explained by the fitted model. For logistic regression fitted values several residual diagnostic measures exist. Here we will investigate the Pearson residuals of the model. The model is fitted well whenever the trend line shows no correlation between the residuals and the attribute against which it is plotted, i.e. there exists a trend line around the 0 axis. The Pearson residual is computed according to

$$p_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i(n_i - \hat{\mu}_i/n_i}}$$

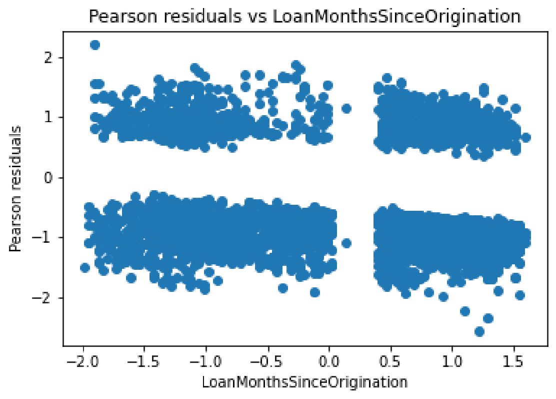
where $p_i$ is the residual, $y_i$ is the observed value, and $\hat{\mu}_i$ denotes the fitted value. Fig. 6 visualizes the plots for the Pearson residuals for the dataset plotted against each of the significant attributes.



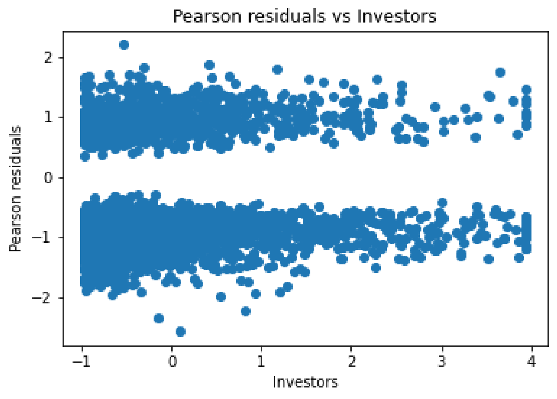
(a)



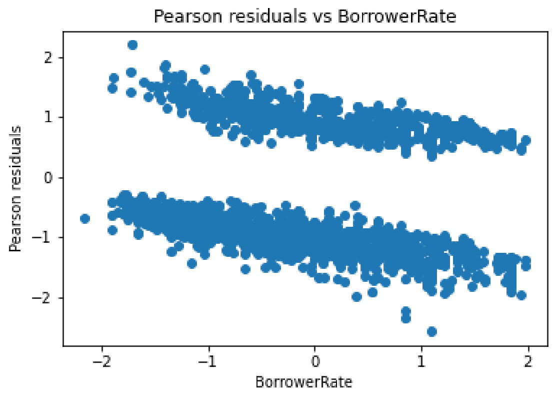
(b)



(c)



(d)



(e)

**Figure 6:** Pearson residuals as function of each attribute

These residual plots shows that the model is a good fit for all attributes since the trend for all attributes are linear. This implies that all of the validation data is explained by these 5 attributes, i.e. there is no other latent feature that carries a portion of information of the validation data.

**Feature selection** From these results we observe that the *BorrowerRate*, i.e. the rate against which the borrower intitiates the loan is seen as the most relevant feature for predicting default since the estimated coefficient for this specific feature is highest. We can also conclude that the features *PercentFunded*, *InvestmentFromFriendsCount*, and *InvestmentFromFriendsAmount* are statistically insignificant for the predictive power of the model and thus these can be disregarded. Among the statistically significant features we see that the *Investors* carries the least predictive power.

**Odds Ratio** Regarding OR, their values can be interpreted as follows:

- OR $>$ 1: positive relationship, i.e. as the feature increases in value, the probability value of the output also increases
- OR $<$ 1: negative relationship, i.e. as the feature increases in value, the probability value of the output decreases
- OR $=$ 1: no association, i.e. the exposure of the output to the feature does not affect the odds of the outcome

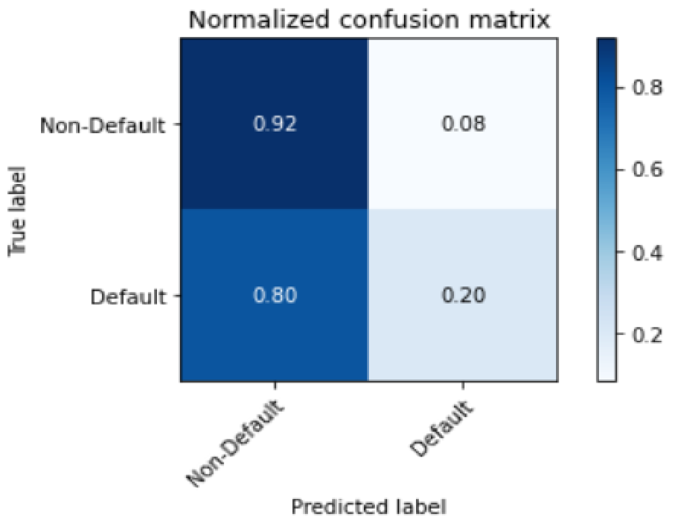
This implies that *MonthlyLoanPayment*, *Term*, *LoanMonthsSinceOrigination*, and *BorroweRate* have a significant positive effect on the output whereas *Investors* has a negative effect on the output.

**Table 5:** Classification report for the fitted model-Logistic

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.660122 | 0.411765 | 67.36% | 0.535943 | 0.575508 |
| recall | 0.989848 | 0.013752 | 67.36% | 0.501800 | 0.657296 |
| f1-score | 0.792039 | 0.026616 | 67.36% | 0.409327 | 0.531262 |
| support | 1313 | 679 | 1992 | 1992 | 1992 |

In the confusion matrix we can see that LR performs very well in terms of predicting non-defaults but performs rather poorly in terms of correctly predicting defaults.

**Figure 7:** Confusion Matrix-Logit



**ROC Curve** Finally, the data has been split into 5 folds. Over each fold the AUC has been computed and the ROC curve as visualized in Fig. 8. We can see that the model significantly outperforms the purely random performance with a mean AUC value of $0.68 \pm 0.02$.

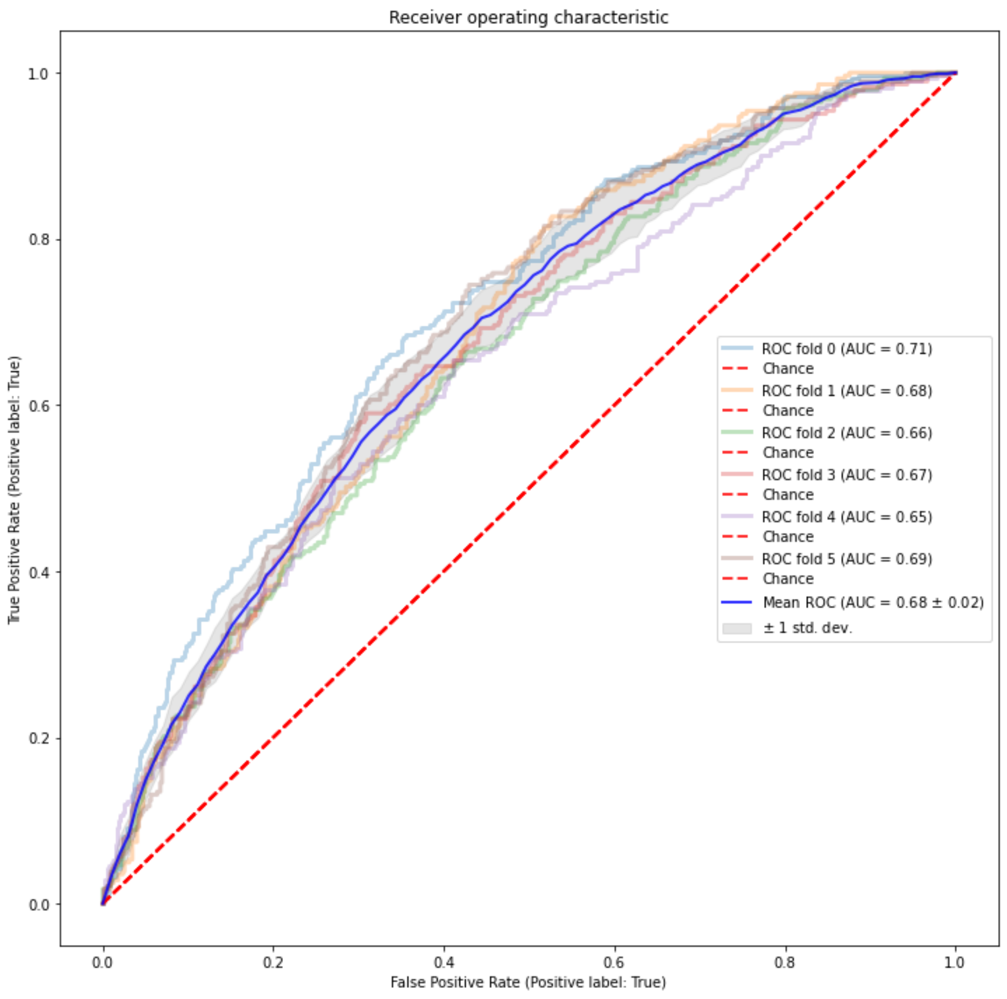


**Figure 8:** ROC curve for logistic regression fit

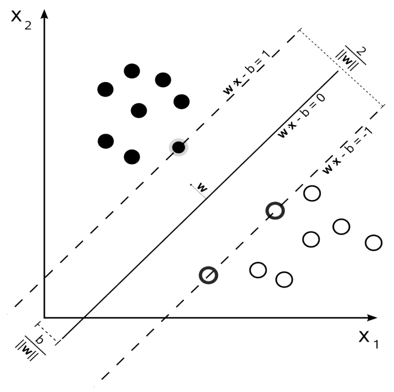
## 5.2 Machine Learning Models

In this subsection one may find the robustness checks on the Logistic Classification algorithm. In order to test the performance of the model several other machine learning algorithms were employed.

### 5.2.1 Support Vector Machine Algorithm (SVM)

Within the family of supervised machine learning algorithms, SVM is mostly employed for classification purposes. Separation is performed through a hyperplane and two margin lines. The latter are built upon two other hyperplanes, parallel to the main one. Margins pass through at least one of the nearest data points, dividing the observations into positive and negative. These data points are defined as support vectors.

**Figure 9:** 2-D classification problem. Straight line hyper plane and margins



By employing this technique, the aim is to maximize the margins distances to ensure the model is accurate enough when classifying each of the data points according to the different classes. Whatever data observation falls within the margins will be classified as positive if they are within the positive margin and vice versa. In short, whenever the hyperplane is created, it should be done in such a way that the margins are "wide" enough to achieve a much more generalized model.

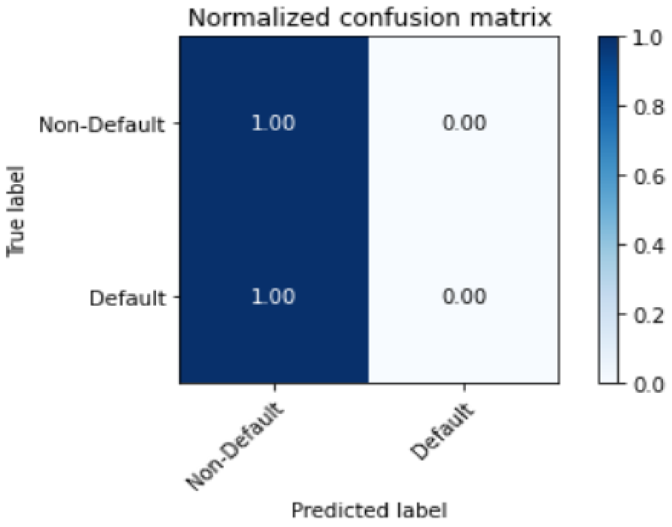
In two dimension problems, binary classifications, the margins acquire the form of straight lines. However, there might be the case when the data at hand is not easily separable in the 2-dimensional space by a straight line, since the existing relationship between the variables under consideration is non-linear. Fortunately enough, this algorithm can also be programmed to convert the plain into a higher dimension´ one by the deployment of a different kernel.

**Fitted Model** Please find below the report of the results on different evaluation metrics (Table6) and a graphical representation of the confusion matrix (Figure 10) for this model. We can see from the confusion matrix that the model does not perform well in terms of confusion matrix diagnostics. The model correctly predicts all non-defaults but also misclassifies all defaults, i.e. the model classifies all loans as a non-default. This is useless for a money lending firm and thus this model can be disregarded.

**Table 6:** Classification report for the fitted model

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.66 | 0.00 | 65.81% | 0.33 | 0.43 |
| recall | 1.0 | 0.00 | 65.81% | 0.50 | 0.66 |
| f1-score | 0.79 | 0.00 | 65.81% | 0.40 | 0.52 |
| support | 1313 | 679 | 1992 | 1992 | 1992 |

**Figure 10:** Confusion Matrix-SVM



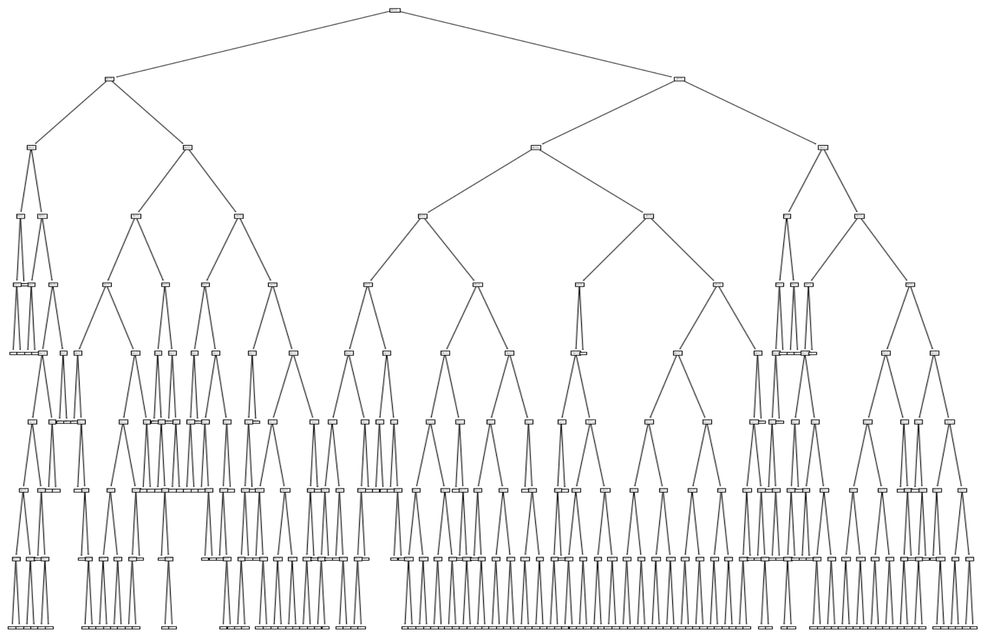
### 5.2.2 Decision Trees

Within the decision trees algorithms one may find:

- ID3
- C5.0
- C4.5
- CART (Classification and Regression Trees)

For the case at hand, we will consider the CART decision tree. Roughly speaking, this algorithm starts by adding a root node to the tree. This node receives the entire training set, whereas the rest of the nodes receive a list of rows as input. Each of the nodes will ask a true/false question to each of the features. In response to this question, the data get further split into two subsets. These subsets then become the inputs for two new "child" nodes that we add to the tree. Figure 11 depicts the structure of the tree built for the binary classification problem at hand. After cross-validating the results of fitting different trees we came to conclude that, in our case, the optimal tree has a maximum depth of 9 and 180 different leaves.

**Figure 11:** Decision Tree visualization



The goal of each question is to unmix the labels as we proceed down. To do that, we need to quantify somehow, how much does a question help to unmix the labels. We could quantify the amount of uncertainty using a specific metric called the Gini impurity. On the other hand, we could quantify how much a question reduces the uncertainty using a concept called information Gain. By using these, we will build the tree recursively until there are no more questions to ask, at which point we will add a "leave".

To do so, we first need to understand what type of questions we should ask the data and how to decide which question to ask when. Each node takes a list of rows as inputs and to generate the questions we will iterate over every value for every feature that appears in those rows. The best questions are the ones that reduce the uncertainty the most. Gini impurity enables us to quantify how much uncertainty there is at a node. This measurement ranges between 0 and 1. Where lower values entail less uncertainty or impurity, and higher readings establish a higher uncertainty. In other words, it quantifies our chances of being incorrect if you randomly assign a label to an example in the same set.

On the other hand, the information gain measurement Will enable us to find the question that reduces the uncertainty the most. It is just a number that describes how each formulated question helps to unmix the labels at a node.We begin by calculating the uncertainty of the starting set. We then compute the uncertainty of each of the child nodes once the question is answered. We calculate the weighted average of their uncertainty, because we care more about a large set with low uncertainty than a small set with high uncertainty. Then we subtract this from our starting uncertainty and that will account for the information gain. As we keep going, we will keep track of the questions that produce the most gain and that will be the best one to add at each node.

**Fitted Model** The diagnostics of the fitted model are summarized below. We see that, in terms of accuracy, the decision tree performs slightly worse compared to SVM and LR. However, the confusion matrix shows that in terms of misclassifying a default, i.e. predicting non-default whereas the true label is a default, is the lowest for this model. Note that missing a default prediction would hurt a lender more than misclassifying a non-default as a default. On the other hand, the TPR of the model scores 0.10 below the logistic regression model.

**Table 7:** Classification report for the fitted model

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.70 | 0.48 | 65.21% | 0.59 | 0.63 |
| recall | 0.82 | 0.32 | 65.21% | 0.57 | 0.65 |
| f1-score | 0.76 | 0.39 | 65.21% | 0.57 | 0.63 |
| support | 1313 | 679 | 1992 | 1992 | 1992 |

**Figure 12:** Confusion Matrix-Dec Tree

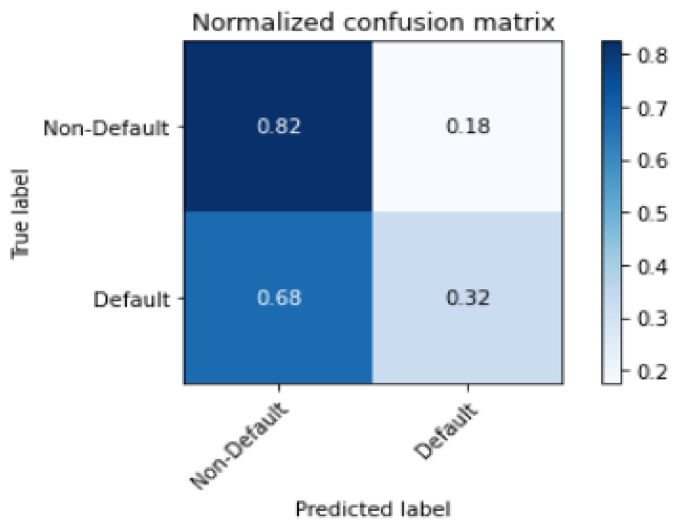
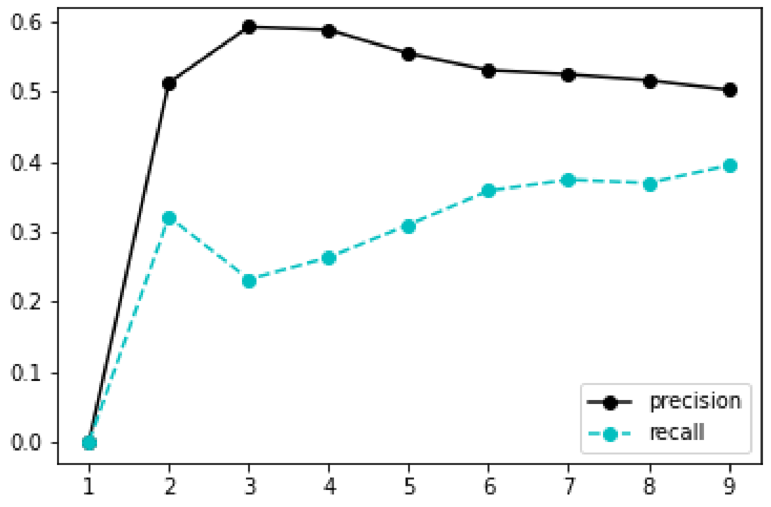


**Figure 13:** Results of the cross-validation. Determining the optimal tree-depth



## 6 Conclusions

For a credit lending firm it is important to be able to predict defaults of loans as accurate as possible. This implies that misclassifying a loan as defaulted whereas it has not defaulted weighs is considered to be less of an impactful error compared to misclassifying a loan as non-defaulted whereas it has defaulted.

In this article, three models have been deployed to test their accuracy on predicting defaults. The LR model provides an accuracy of around 68% on the test data, the highest score among the three models. However, the confusion matrix of the fitted model reveals that the model wrongly classifies a relative large amount of true defaults as non-defaults.

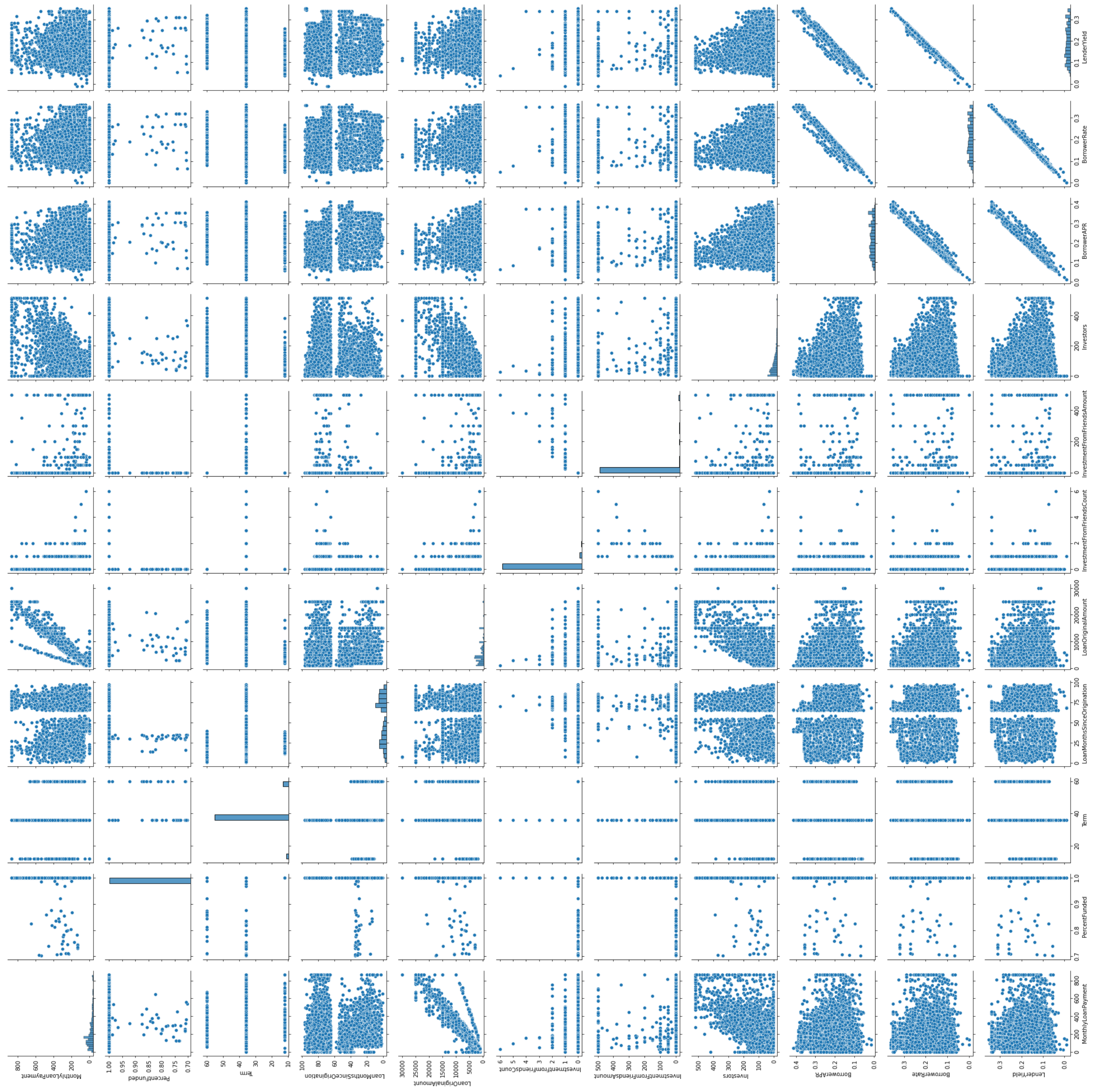
The decision tree scores slightly lower in terms of overall accuracy, with a score of around 66%. In terms of false negatives the model scores significantly better compared to both models but on the other hand performs worse on predicting true positives. However, taking into account the importance of correctly classifying defaults, it is just to trade a decreased accuracy in correctly predicting non-defaults for an increased accuracy in correctly predicting defaults.

The SVM performs the worst of the 3 models and provides no significant results as, according to its confusion matrix, the model classifies all loans as non-defaulted.

## 7 Annex

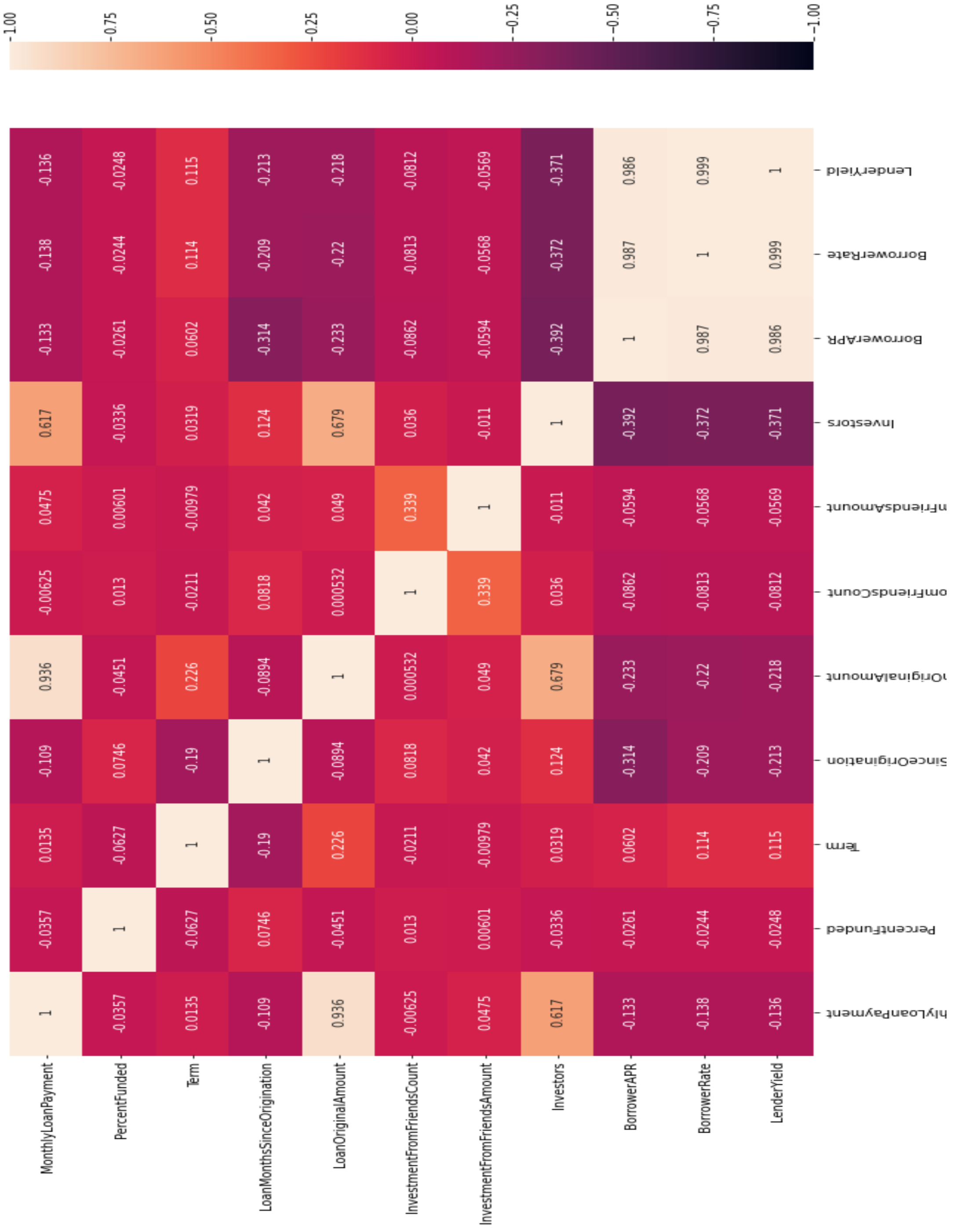
**Annex 1. Feature´s Individual & Pair-Wise Distributions (Horizontal representation)** Figure 14 is a matrix representation of the individual distributions, presented in the diagonal, whereas the different pair-wise scatter plots in the upper and lower triangles of the matrix enable grasping an idea of the existing (or non-existing) relationship between the variables.

**Figure 14:** Features´ individual, and pair-wise distributions



**Annex 2. Features Correlation Heatmap representation, based on Pearson´s Correlation Coefficient** Figure 15 presents the linear correlation coefficients of the different pairs considered in the dataset in order to proceed with the features´ selection as inputs for the models considered.

**Figure 15:** Features correlation´s heat map


MonthlyLoanPayment
PercentFunded
Term
LoanMonthsSinceOrigination
LoanOriginalAmount
InvestmentFromFriendsCount
InvestmentFromFriendsAmount
Investors
BorrowerAPR
BorrowerRate
LenderYield