Assignment 1

Machine Learning I, MBD

Álvaro Ezquerro Pérez
María Calvo de Mora Román
Celia Quiles Alemañ

27 de octubre de 2023

Contents

The objective of this assignment is to compare different classification algorithms with a real dataset. However, before training the different classification models, we must carry out an exploratory analysis of the different variables before making any assumptions.

1 EDA

EDA allows us to identify obvious errors, as well as better understand the patterns in the data, detect outliers or events. anomalous and find interesting relationships between the variables.

1.1 Import dataset

This assignment will analyze `FICO_Dataset.csv` the dataset.

The data contains information about Home Equity Line of Credit applications made by real homeowners. A HELOC is a line of credit typically offered by a bank as a percentage of home equity (the difference between the current market value of a home and its purchase price). The fundamental task is to use the information about the applicant in their credit report to predict whether they will repay their HELOC account within 2 years. Source.

We start by importing the dataset and seeing how many variables and rows it consists of.

It is a dataframe with 7442 observations and 11 variables which come from anonymized credit bureau data.

1.2 Check out the missing values

Secondly, we continue with the command `.info()` which informs us about the type of each variable and about how many non-null values do they have.

Since the number of observations is equal to 7442, some columns do have missing values. In total we have 2320 missing values:

- RiskPerformance: 2197 NAs
- MSinceOldestTradeOpen: 27 NAs
- PercentTradesWBalance: 56 NAs
- NumSatisfactoryTrades: 17 NAs
- NumTotalTrades: 23 NAs

Actually, most of the missing values come from the `RiskPerformance` output variable. That is, we have information on 2197 clients of whom we do not know whether or not they would pay their line of credit. Consequently, these observations will not be useful either to train the model (since we need input and output pairs), nor to evaluate its predictive capacity in the test set. Thus, since they do not serve our objectives, we should dispense with these 2197 observations. So, from now on we will work with the remaining 5245. As for the remaining 123 missing values, since these only represent 2% of the data we would use, we can eliminate these records, as it does not represent a significant loss.

1.3 Plot the data and check for outliners

Thirdly, we proceed to use the `.describe()` function to know which values each of our variables moves between, as well as other measures of centrality and dispersion, in order to identify outlier values. However, we will not be able to affirm the existence of outlier values until the training of future models. First of all, we realize that there are variables that contain negative values, specifically values between -9 and -7. According to the metadata we have from the data set, these numbers correspond to the following errors:

(-9): refers to null values, without registration;

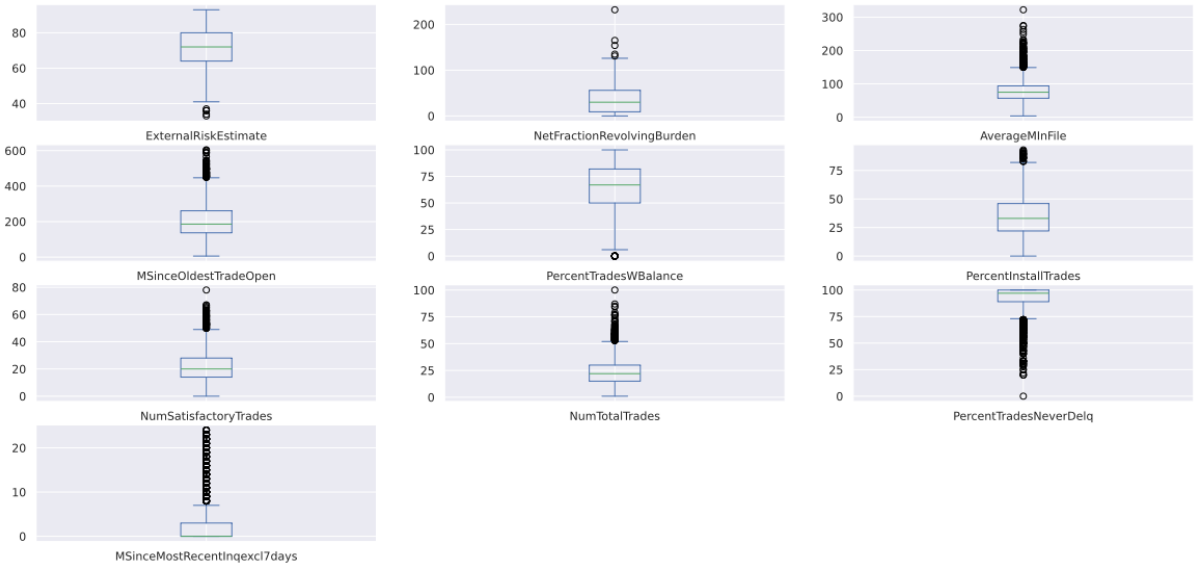
(-8): without usable or valid information, that is, it implies an absent value as well;

(-7): indicates that the corresponding condition is not met, it only appears in `MSinceMostRecentInqexcl7days`, so it means that the user has not made any query. Since the values -9 and -8 refer to null values, we do the same as in the previous section, we eliminate them from the data set.

As for the -7 values in `MSinceMostRecentInqexcl7days`, as they are only 50 observations of the dataset, and we can trick the model by assigning them the mean or median value, we decide to eliminate those rows as well. Once we have that issue covered, we can continue with the outliers identification. We repeat the `.describe()` procedure, and we'll obtain the histogram and the boxplot of each and every variable in the dataset.

Thanks to the boxplots, we realize that almost every variable has multiple outliers. The first three of them have only outliers with significant low values. While the others have only outliers with significant high values.

Figure 1.1: Input BoxPlots



However, even though the graphs identify significantly large or small values, we cannot say with these boxplots alone that they are outlier values, we will only say that they are possible outlier values. It will not be until the training of the models that we will be able to affirm such a theory. Consequently, in the first instance we do not make any drop in these values.

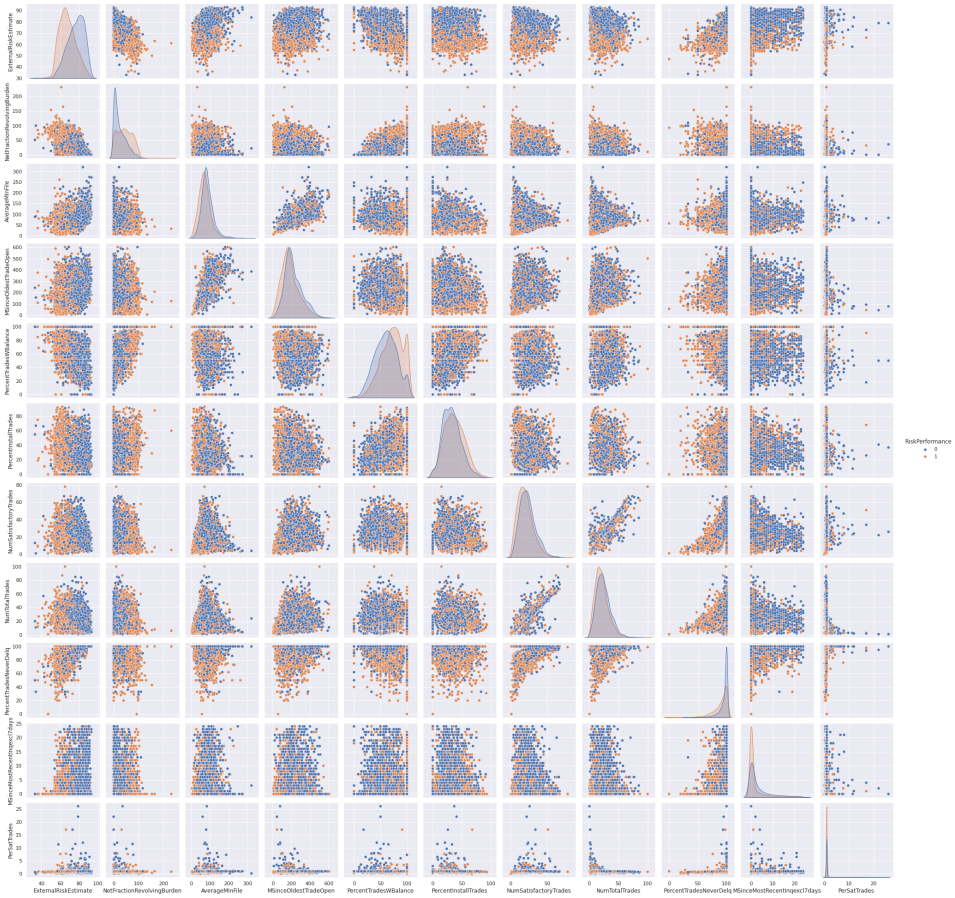
1.4 Encode categorical variables

Once we have identified the variables of the data set, and we know what their behavior is in general terms, we have to convert those variables that are categorical into factors. In our case, only the output variable is categorical, that is, `RiskPerformance`.

1.5 Exploratory analysis

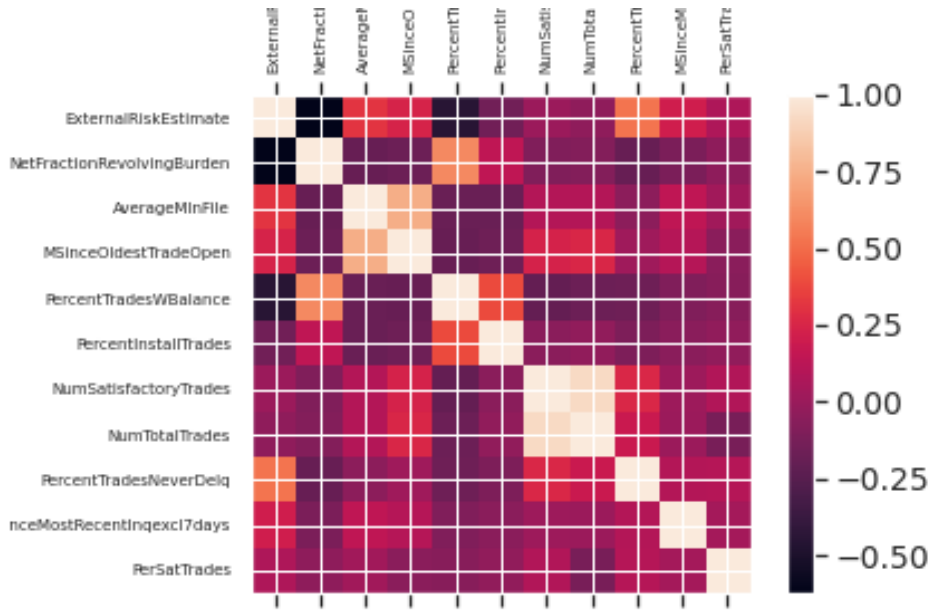
In this section, we are going to study the correlation plot between the numeric input variables. The reason for this is that, if two variables have a correlation close to 1, it implies that both provide the same information and, consequently, it would be optimal to get rid of one of them. In order to figure out an initial impression of how the variables are related, we have 'made a pairplot (Figure 1.2). However, the amount of variables with which we are working is huge, so interpeting the pairplot is confusing and tedious.

Figure 1.2: PairPlot



Therefore, we have proceed to obtain graphically a matrix of correlations (Figure 1.3) which is easier to interpretate.

Figure 1.3: Correlation Matrix



From the graph above we can prelude the correlation between each pair of variables. For example, in this terms: We observe how `AverageMInFile` vs `NetFractionRevolving Burden` or `NumSatisfacotryTrades` vs. `ExternalRiskEstimate` have low correlation, implying that they do not have a linear relationship , and it would be necessary to add all variables to the model as explanatory variables, since they provide different non-repetitive information.

On the other hand, we observe how `NumTotalTrades` vs. `NumSatisfactoryTrades` have a correlation around 0.9, which would imply that they have a high correlation, meaning that they would not provide additional information to the model, being able to dispense with one of them as an explanatory variable. Specifically, and in this case, in order to obtain exact the correlation between both variables, we proceed to calculate it explicitly and numerically.

Indeed, we obtain that both variables have a correlation of 0.93, implying that they provide practically the same information and one of them could be omitted. However, since we want to keep the information on how many satisfactory operations we proceed to calculate the percentage of satisfactory operations over total operations, adding the result as a new explanatory variable `PerSatTrades`. In this way, we will keep the information related to successful transactions but avoid having a variable practically the same as `NumTotalTrades`.

1.6 Check out for class imbalances

In the analysis prior to model training, it is also very important to identify if there is any type of imbalance in the categorical variables, because some models do not behave correctly if this imbalance exists. In our case, we can only parse it into the output variable.

When analyzing the imbalance between classes of the output variable, we observed a difference of around 100 between both classes. Considering that we are working with

a large data set, it could be considered a small difference, without having a significant consequence. Consequently, we do not have to apply any balancing techniques.

1.7 Splitting the data into train and test

Finally, to prepare for the training of the models, we divide the data set into a train set (to train the model and achieve the optimal values of the parameters and hyperparameters), and a test set to evaluate the predictive capacity of the model in question. Specifically, we are going to use a random set of 20% of the total observations as a test set.

Once we have the data set already analyzed and preprocessed, we can move on to the model training part. Specifically, as the variable to be predicted is dichotomous categorical, the models that we are going to train and compare are the following:

- Logistic regression
- k nearest neighbors
- Decision tree
- SVM
- Multi-layer Perceptron

With all these models we will carry out the adjustment and training of the model (if there are hyperparameters we will carry out a Cross-Validation for tuning), and we will analyze the results of each model. For this analysis we will obtain, on the one hand, the average accuracy obtained in the CV, as well as the confusion matrices and error measures both on the train (it will inform us about the goodness of the fit), and on the test (predictive capacity).

2 Logistic Regression

We start with the logistic regression model. Logistic Regression is a classification technique used in machine learning. It uses a logistic function to model the dependent variable. The dependent variable is dichotomous in nature, i.e. there could only be two possible classes. As a result, this technique is used while dealing with binary data [1]. First of all, we start by training the model with all the variables that we have left in the data set as inputs.

2.1 All variables as lineal inputs

We have started by training the model with RiskPerformance as the output variable, and the rest as linear inputs. This model lacks hyperparameters, so we have not needed to perform tuning.

Directly, we have obtained error measurements to know the quality of the model. Specifically, the average accuracy of the cross-validation with 5 folds is obtained, the accuracy in the training set (it tells us about the goodness of the fit) and the accuracy

[1] more information

in the test set (it tells us about the predictive capacity of the model against new input data).

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.7305	0.73	Data 0.74

The accuracy obtained can be greatly improved.

Figure 2.1: Confusion Matrix and Summary of the Logistic Regression

```
----- TEST CONFUSION MATRIX-----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 352 135
         1 127 381

                                                 Signif
Accuracy: 0.74                  Intercept
No Information Rate: 0.5        ExternalRiskEstimate          ***
P-Value [Acc > NIR]: 0.0        NetFractionRevolvingBurden    ***
Kappa: 0.47                     AverageMInFile                ***
Mcnemar's Test P-Value: 0.67    MSinceOldestTradeOpen
Sensitivity: 0.75               PercentTradesWBalance
Specificity: 0.72               PercentInstallTrades           **
Precision: 0.73                 PerSatTrades                    .
Recall: 0.72                    NumTotalTrades                ***
Prevalence: 0.51                PercentTradesNeverDelq          *
Detection Rate: 0.38            MSinceMostRecentInqexcl7days  ***
Detection prevalence: 0.52      ---
Balanced accuracy: 0.74         Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
F1 Score: 0.73
Positive label: 0
```

Given that the last two are practically the same, we can conclude that we have not fallen into overfitting. Furthermore, it was obtained that the accuracy if we considered all the obs. 0 or 1 would be 0.5 (`No Information Rate`), as our accuracy is much higher, we also conclude that it is better to launch this model than to assign all observations a 0 or 1. In parallel, we also obtain a table with the p-values of each of the explanatory variables.

Not all variables appear as significant:

- `MSinceOldestTradeOpen` and `PercentTradesWBalance` are not statistically significant.
- `ExternalRiskEstimate`, `NetFractionRevolvingBurden`, `AverageMInFile` and `MSinceMostRecentInqexcl7days` are the most significant variables.

Given that 3 variables did not seem significant in our model, we can either repeat the adjustment but excluding these variables and check if our predictive capacity improves, and or launch a model that considers non-linear relationships on non significant variables.

2.2 Only significant variables as lineal inputs

We repeat the same procedure but only considering as inputs the variables that in 2.1. appeared as significant in the model.

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.7292	0.73	Data 0.74

We obtain practically the same accuracy values. Consequently, no improvement is observed from the elimination of 2 of the variables. We can try the second proposal.

2.3 Only significant variables as lineal inputs and non significant ones as squared inputs

In this case, we reintroduce the significant variables as linear variables, but in order to convert those that were not significant into significant ones, we square them to try to capture hitherto unidentified relationships.

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.7262	0.73	Data 0.74

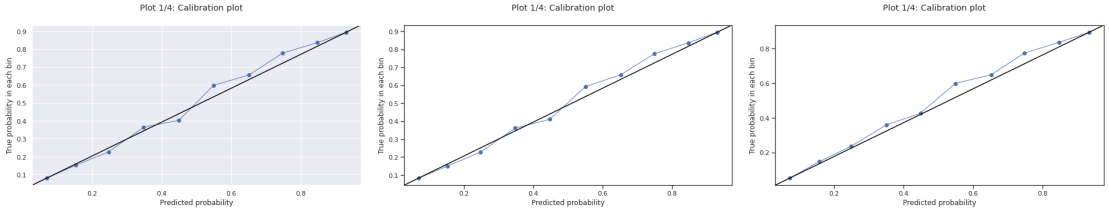
Once again, practically the same precision values are achieved. In fact, even as for the one corresponding to Cross-Validation, this is somewhat worse. Furthermore, when looking for which variables are significant in this model, we once again have those that were already significant in the previous models, and of the squared ones, we only find that `MSinceOldestTradeOpen` it does become significant.

Figure 2.2: Summary of the Logistic Regression with squared variables

```
                                          Pr(>|t|) Signif
ExternalRiskEstimate                     0.000000e+00    ***
NetFractionRevolvingBurden               6.375611e-11    ***
AverageMInFile                           6.591911e-04    ***
PercentInstallTrades                     1.290108e-03     **
PerSatTrades                             4.809831e-02      *
NumTotalTrades                           1.322832e-07    ***
PercentTradesNeverDelq                   3.404490e-03     **
MSinceMostRecentInqexcl7days             3.304024e-13    ***
1                                        2.663386e-01
MSinceOldestTradeOpen                    2.959539e-01
PercentTradesWBalance                    6.744455e-01
MSinceOldestTradeOpen^2                  2.746510e-03     **
MSinceOldestTradeOpen PercentTradesWBalance 1.170642e-01
PercentTradesWBalance^2                  3.201259e-01
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

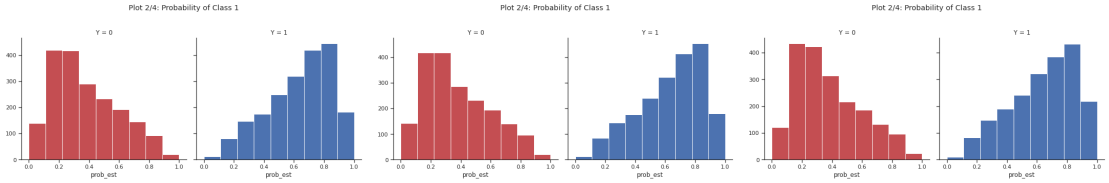
To conclude, the calibration plots, the probability histogram plots and the area under the ROC curve plots of all 3 logistic regression models will be plotted and compared. The plots of the 3 models are practically identical, there are no significant differences between them, in line with what happened previously when comparing the accuracy values. Regarding the calibration plots (Figure 2.3), they all adjust almost a straight line, though they tend to give more probability than the real and in the higher values we have more failure. The probability histograms (Figure 2.4) show that the 3 models work really well in the intermediate probability values, but not n the extremes. And, finally, regarding the ROC curves (Figure 2.5) do not tell nothing more special, they all leave an area under the ROC curve of approximately 0.79.

Figure 2.3: LR: Calibration plots



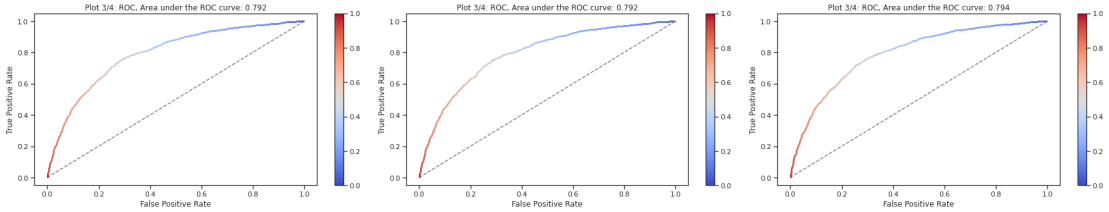
(a) LR I (b) LR II (c) LR III

Figure 2.4: LR: Histogram plots



(a) LR I (b) LR II (c) LR III

Figure 2.5: LR: ROC curve plots



(a) LR I (b) LR II (c) LR III

Consequently, we can conclude that there is no gain either by eliminating the 2 non-significant variables, or by squaring them in the logistic regression.

3 k Nearest Neighbors

Next, we proceed to train a KNN model. The kNN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples. Thus, for each unlabeled record in the test dataset, k-NN identifies k records in the training data that are the 'nearest' in similarity [2].

As with the logistic regression, we start training the model with all the variables that we have left in the data set as inputs after the EDA.

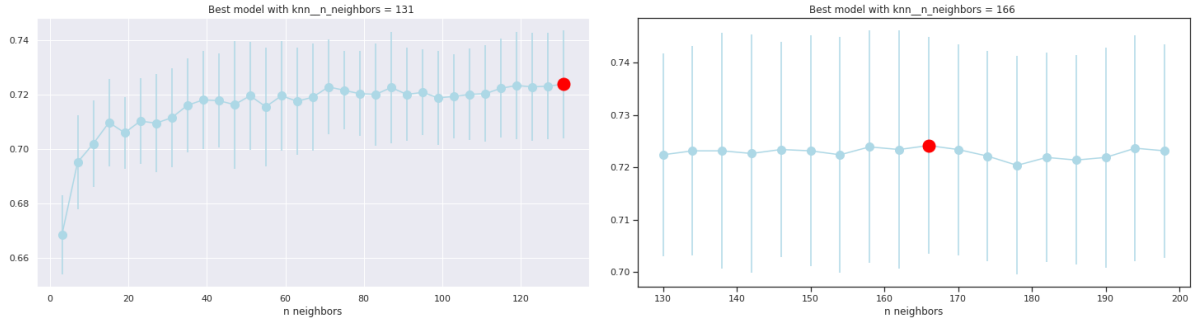
3.1 All variables as lineal inputs

First, we started by training the model with `RiskPerformance` as the output variable, and the rest as linear inputs. The KNN has a single adjustable hyperparameter K, which k is a variable term implying that any number of nearest neighbors could be used. Therefore, it is necessary to determine the optimum parameter for our model. This is achieved by making a sweep of the possible values of K and 10 Cross-Validation for each of them, selecting as optimum the one with the highest accuracy value. Specifically, the sweep of the possible values of K is specified from 3 to 135 with jumps of 4 by 4.

After that, the grid plot of the hyperparameters is obtained (Figure 3.1 (a)). In this graph we observe that the optimal hyperparameter appears at one end of the graph. Therefore, to make sure that it is the local maximum (optimal hyperparameter), we enlarge the graph on the left. In this way we obtain Figure 3.1 (b), in which we observe the local maximum and thus obtain an optimal K of 166.

[2](Prof. Antonio Muñoz, Prof. Eugenio Sánchez & Prof. José Portela, Machine Learning - Chapter 2: Classification III kNN Validation, Universidad Pontificia Comillas)

Figure 3.1: KNN: Hyperparameter GridSearchCV plots



(a) Initial (b) Final

Then, as we did for the logistic regression, we have obtained error measurements to know the quality of the model. Specifically, the average accuracy of the cross-validation with 5 folds is obtained, the accuracy in the training set (it tells us about the goodness of the fit) and the accuracy in the test set (it tells us about the predictive capacity of the model against new input data).

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.7224	0.73	Data 0.74

Again, as with logistic regression, the accuracy obtained can be improved considerably.Since the accuracy of the training and the test are practically equal, we can conclude that we have not fallen into overfitting and that the test seems to be representative of the data set. Furthermore, we have obtained that the accuracy if we would consider all obs. 0 or 1 would be 0.5 ('No Information Rate'), as our accuracy is much higher, we also conclude that it makes more sense to launch this model than to assign all observations a 0 or 1.Furthermore, a good value of precision and recall is obtained, which are also very similar.

Figure 3.2: KNN train and test Confussion Matrices

```
Confusion Matrix and Statistics
          Prediction
 Reference    0    1
         0 1424  523
         1  570 1460

Accuracy: 0.73
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.45
Mcnemar's Test P-Value: 0.16
Sensitivity: 0.72
Specificity: 0.73
Precision: 0.71
Recall: 0.73
Prevalence: 0.51
Detection Rate: 0.37
Detection prevalence: 0.5
Balanced accuracy: 0.73
F1 Score: 0.72
Positive label: 0
```

(a) Train

```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 365 122
         1 136 372

Accuracy: 0.74
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.48
Mcnemar's Test P-Value: 0.42
Sensitivity: 0.73
Specificity: 0.75
Precision: 0.73
Recall: 0.75
Prevalence: 0.51
Detection Rate: 0.37
Detection prevalence: 0.5
Balanced accuracy: 0.74
F1 Score: 0.74
Positive label: 0
```

(b) Test

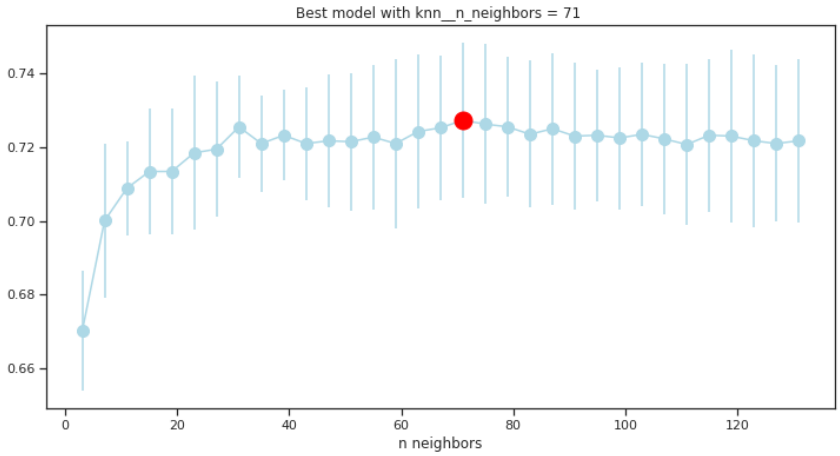
Given that the results can be considerably improved, we will take into account what

was discussed in the logistic regression. That is, to launch a model that excludes the non-significant variables and considers the nonlinear relationships on those variables found in the logistic regression model, where only MSinceOldestTradeOpen squared was concluded to become significant.

3.2 Only significant variables as lineal inputs and non significant ones as squared inputs

As with the logistic regression, the significant variables are reintroduced as linear variables, but the only variable found to be significant in the logistic regression model (Section 2.3) is squared in order to capture hitherto unidentified relationships. Again, the optimal hyperparameter is searched for in the same way as in the previous case (Section 3.1), obtaining the following grid plot of the hyperparameters:

Figure 3.3: KNN2: Hyperparameter GridSearchCV plot



In this case, it can be observed that the optimum K is at the center of the graph, so we can conclude that K=71 is indeed the optimum in this case since it is a local maximum. In this case, the error measurements obtained to know the quality of the model are:

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.72014	0.73	Data 0.74

Once again, practically the same precision values are achieved. In fact, even as for the one corresponding to Cross-Validation, this is somewhat worse. Consequently, no improvement is observed from the consideration of non-linear variables found to be significant in terms of accuracy.

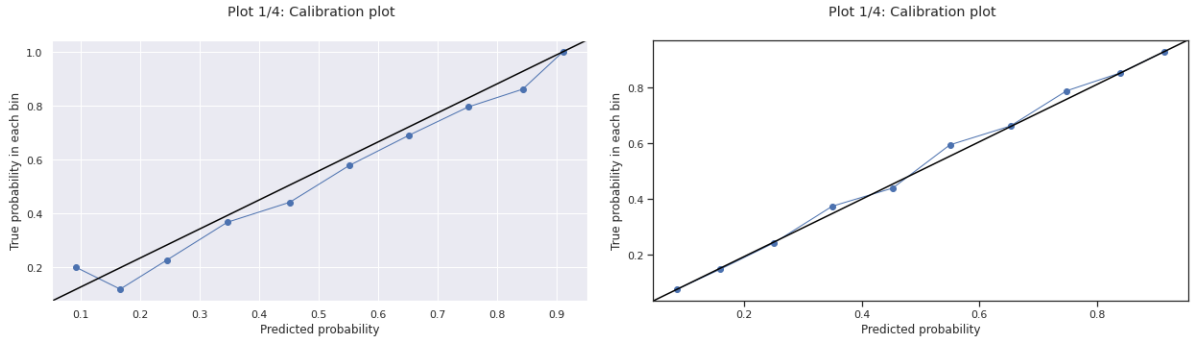
3.3 KNN model comparison

After having trained the two KNN models by modifying the input variables and after having seen the accuracy values obtained in the two cases, we could say that practically no improvement is observed after the consideration of nonlinear variables. However, if we look at the graphs obtained for each model this idea changes a little.

First, regarding the calibration plots, in the first trained model (the one that considered all variables as linear inputs), it can be seen that the model is more or less well calibrated (Figure 3.4 (a)), since a practically diagonal graph is obtained implying that the predicted probabilities correspond quite closely to the actual probabilities, although shifted downwards.

On the other hand, regarding the KNN model trained with only significant variables as lineal inputs and non significant ones as squared inputs (Figure 3.4 (b)), its calibration plot improves slightly compared to the previous case in which only linear variables were considered. It can be seen that in this case the calibration graph fits more closely to the diagonal than when we had all the variables as inputs.

Figure 3.4: Calibration curves

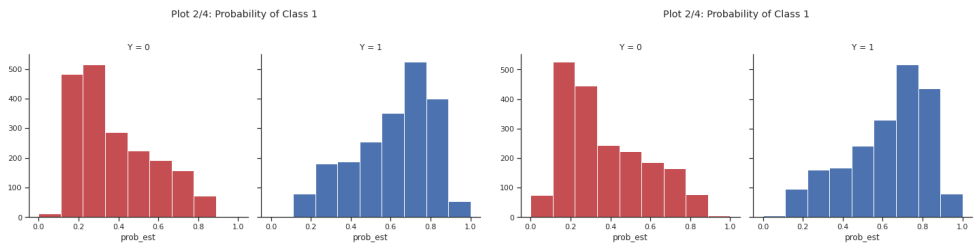


(a) All inputs

(b) No lineal inputs

The probability histogram in the first KNN model (Figure 3.5 (a))shows an asymmetric trend, implying that it is indeed giving a probability as expected (for example when Y=0 the probability estimated for Y=1 is very low). However, it is observed that the probability when Y=0 is very small at the left end (similarly if the probability histogram for Y=1 is observed) implying that at those extremes the predicted values are not good enough. On the other hand, the probability histogram for the second KNN model (Figure 3.5 (b)) improves slightly at the extremes compared to the previous case. However, these small improvements are so subtle that they cannot be considered extremely significant. Also, since the predictions in terms of accuracy have worsened, it does not make sense to carry out this new model.

Figure 3.5: KNN: Probability histograms



(a) All inputs

(b) No lineal inputs

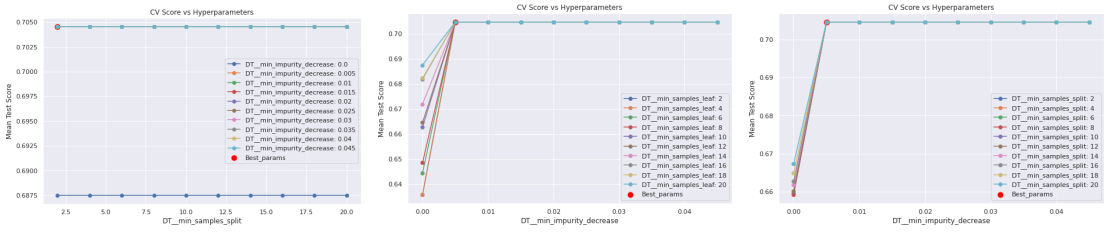
4 Decision tree

The fourth model trained has been a decision tree. The main basis of the decision tree models consists in splitting the input space recursively by minimizing the impurity of the nodes until the terminal nodes are pure enough, assessing this purity in terms of the Gini index [3]. As with the previous models , we start training the model with all the variables that we have left in the data set as inputs after the EDA. First of all, it is necessary to take into account that the decision tree has several hyperparameters. Specifically, the optimal value is to be determined in the same way as before, that is, by making a sweep of the possible values of each hyperparameter and 10 Cross-Validation for each of them, selecting as optimum the one with the lowest error value. The hyperparameters thus determined were:

- `min_impurity_decrease` (determines the minimum purity value allowed for each individual partition. Thus, the higher this value is, the more demanding the model is, the less it is allowed to grow). Specifically, the sweep of the possible values of `min_impurity_decrease` is specified from 0 to 0.05 with jumps of 0.005 by 0.005.
- `min_samples_split` (minimum number of observations in a node to continue making cuts) Specifically, the sweep of the possible values of `min_samples_split` is specified from 2 to21 with jumps of 2 by 2.
- `min_samples_leaf` (determines the number of samples to have at the terminal node). Specifically, the sweep of the possible values of `min_samples_leaf` is specified from 2 to 21 with jumps of 2 by 2.

After that, the grid plot of the hyperparameters is obtained (Figure 4.1). In each of them, the optimal hyperparameter obtained can be observed. In this case, the optimal hyperparameters are as follows `min_impurity_decrease = 0.005` , `min_samples_split = 2` and `min_samples_leaf = 2` .

Figure 4.1: Hyperparameters GridSearchCV plots



When obtaining the error measurements to know the quality of the model we obtain:

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.70179	0.7	0.7

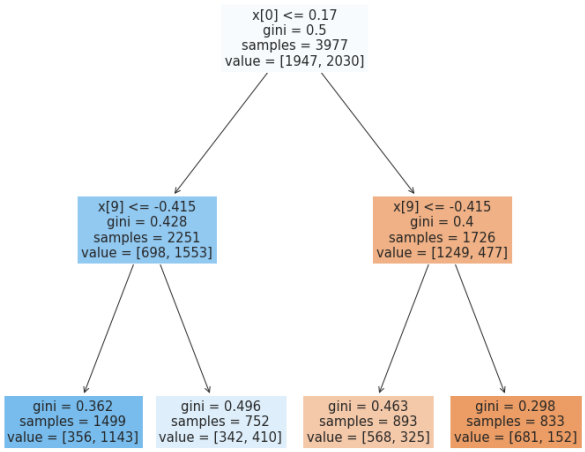
Once again, there is clearly room for improvement. Since the accuracy of the training and the test are equal, we can conclude that we have not fallen into overfitting and that

[3](Prof. Antonio Muñoz, Prof. Eugenio Sánchez & Prof. José Portela, Machine Learning - Chapter 2: Classification IV Decision Trees & SVM, Universidad Pontificia Comillas)

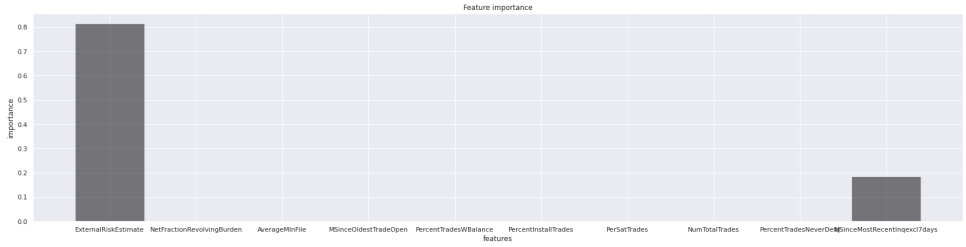
the test seems to be representative of the data set. Furthermore, we have obtained that the accuracy if we would consider all obs. 0 or 1 would be 0.5 ('No Information Rate'), as our accuracy is much higher, we also conclude that it makes more sense to launch this model than to assign all observations a 0 or 1.

However, although looking only at the accuracy it seems that the model has done well, we observe that in the graph of the significant variables (Figure 4.2 (b)) only the variables `ExternalRiskEstimate` and `MSinceMostRecentInqexcl7days` are obtained as characteristics. As a consequence, the tree has only 3 levels (Figure 4.2 (a)). In addition, the values of the optimal hyperparameters obtained are not very high and therefore not very restrictive, so they cannot be the cause of this weak tree growth. Therefore, it can be determined that this model is not the most optimal for the proposed problem, since by taking into account only these two variables and not improving enough to continue classifying, information from the rest of the variables is lost along the way, not providing an optimal classification.

Figure 4.2: Tree and features plots



(a) Tree



(b) Features importance

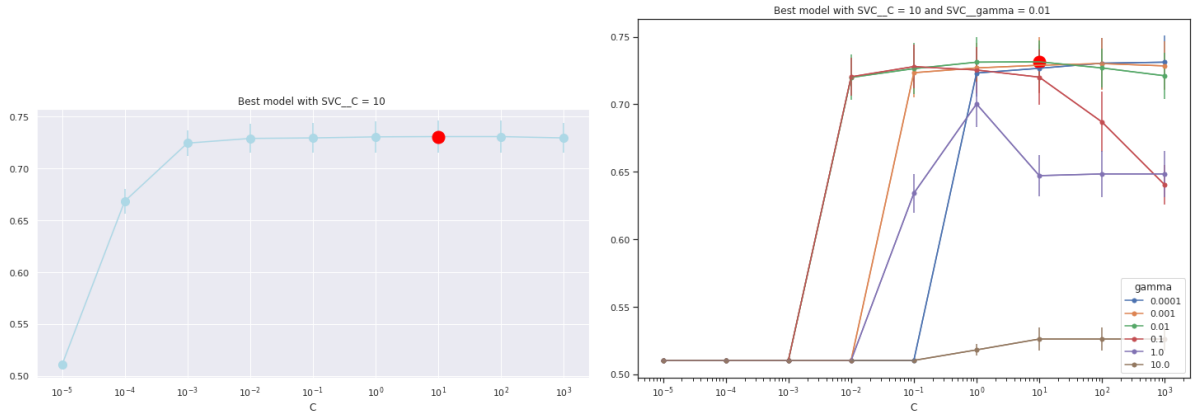
5 SVM

The fifth model we have trained is the Support Vector Classifier, we have applied both the linear and the radial. In order to obtain the best hyperparamters for both models we have made a grid search chossing the ones with best accuracy.

The linear SVC has one hyperparameter, `C`, that allows more or less margin in the hyperplane that classifies the observations and it affects to the flexibility od the model. On the other hand, the radial SVC, apart of `C`, has another hyperparameter `gamma` that varies the shape of the border that classfies the observations.

The following plots (Figure 5.1 show in both models the accuracy of the full range of hyperparameters tryed. We can see how the maximum is obtained in both models in non-bordering points.

Figure 5.1: SVM: Hyperparameters plots



(a) Tree

(b) Radial

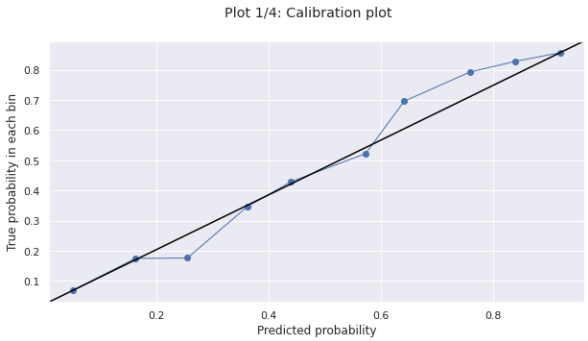
The optimum hyperparameters are: `C = 10` in both cases and `gamma = 0.01` in the radial one. Regarding the accuracy we have the following results:

	Avg CV Accuracy	Accuracy (train)	Accuracy (test)
Linear SVC	0.72693	0.73	0.74
Radial SVC	0.72844	0.74	0.74

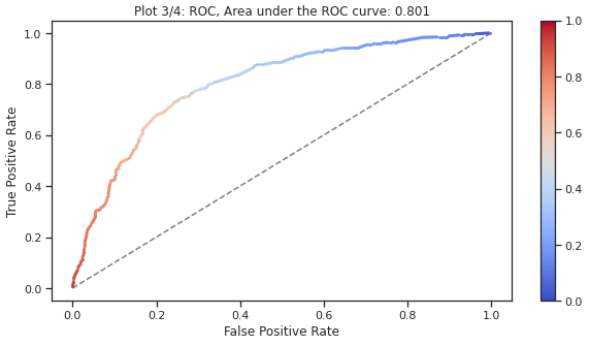
We can see how the radial SVC performs slightly better than the linear one.

Lets analyze the performance plots of both. Starting with the linear, the calibration plot (Figure 5.2 (a)) almost fits the straight line, but tends to subestimate the probabilities. The ROC curve (Figure 5.2 (b)) is similar to the ones in other models and pretty equilibrated. And the histogram (Figure 5.2 (c)) is neither precise nor smooth.

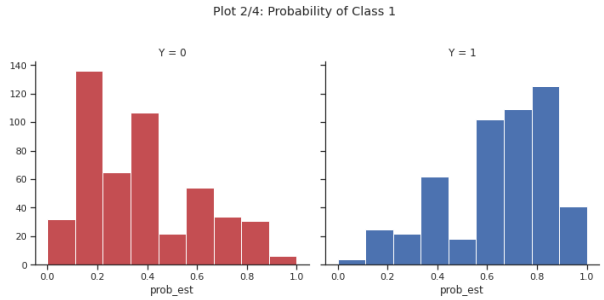
Figure 5.2: Linear SVC: Performance plots



(a) Calibration plot

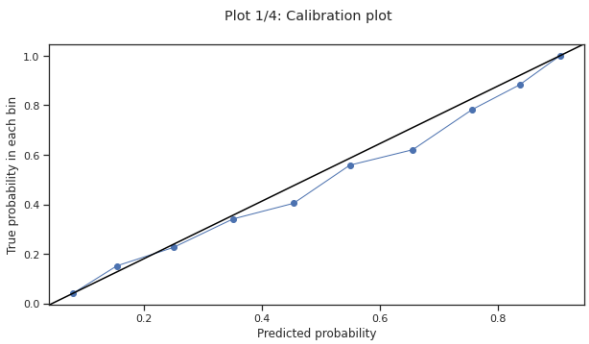


(b) ROC curve

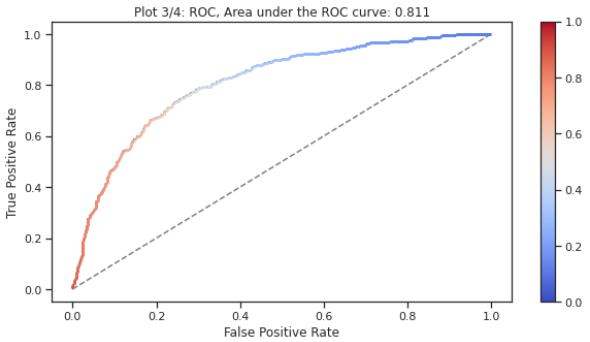


(c) Class 1 histogram

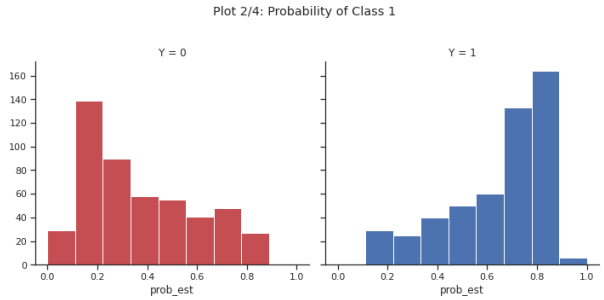
Figure 5.3: Radial SVC: Performance plots



(a) Calibration plot



(b) ROC curve



(c) Class 1 histogram

Following with the radial, the calibration plot (Figure 5.3 (a)) almost fits the straight line, but tends to subestimate the probabilities. The ROC curve (Figure 5.3 (b)) is very

similar to the linear model one. And the histogram (Figure 5.3 (c)), comparing to the linear SVC is well-distributed, however it has some problems in the extremes.

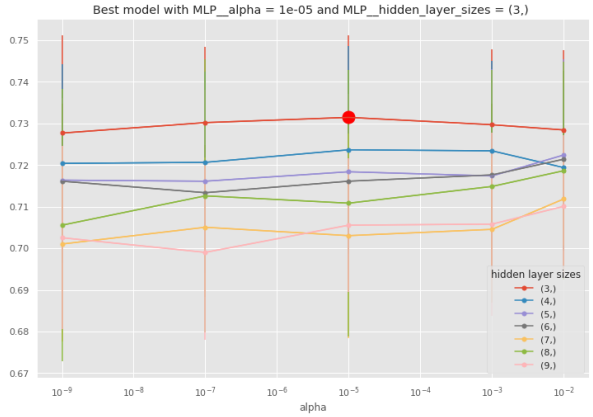
In general, the radial SVC performs better, not only according to the accuracy but also to the sensitivity and specificity.

6 Multi-layer Perceptron

The sixth model we have trained is the Multi-layer perceptron. As usual, in order to obtain the best hyperparameters we have performed a grid search. In this particular case we have a lot of possibilities in the layers choice. We have tried with all possibilities between one and three hidden layers from three to nine perceptrons each. Apart from this, there is another hyperparameter `alpha` the regularization term. There are more parameters like the maximum iterations or activation function that are common to all the hyperparameters variants.

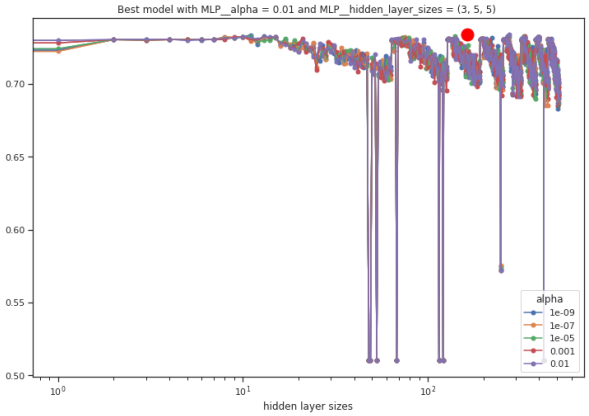
The following plots (Figure 6.1) show the different optimum points of hyperparameters choice depending on the number of hidden layers. Once you have more than one hidden layer the hyperparameters grid plot becomes a bit messy, however, it keeps being interesting.

Figure 6.1: MLP: Hyperparameters plots



(a) 1 layer

(b) 2 layers



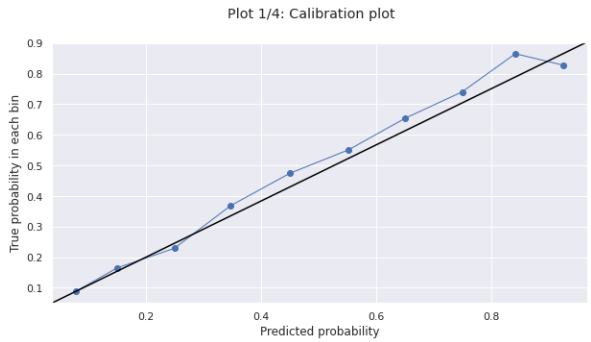
(c) 3 layers

Among all the hyperparameter choices the best one is `gamma = 0.01` with three hidden layers (3, 5, 5). The accuracy obtained with this configuration is:

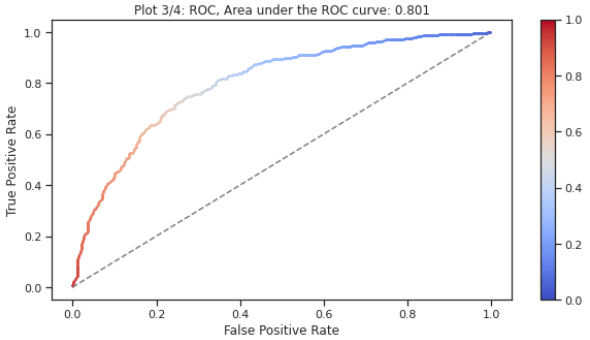
Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.729747	0.73	0.73

Both train and test accuracy are equal, therefore, there is no overfitting. The calibration plot (Figure 6.2 (a)) adjust almost a straight line, though it tends to give more probability than the real and in the higher values we have more failure. The ROC curve (Figure 6.2 (a)) is pretty much in the average of the assignment and has nothing special. And, finally, in the histogram we can se that the model works well in the intermediate probability values but not in the extremes.

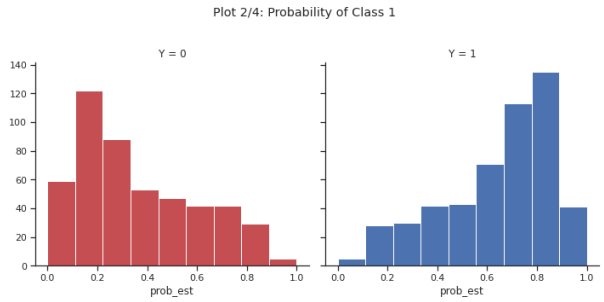
Figure 6.2: MLP: Performance plots



(a) Calibration plot



(b) ROC curve



(c) Class 1 histogram

7 Random Forest

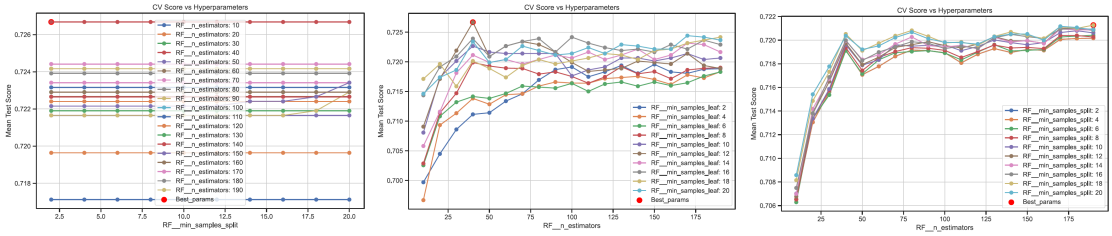
Second to last and as an additional predictive model, we have decided to train a random forest model. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [4].

As with the previous models ,we start training the model with all the variables that we have left in the data set as inputs after the EDA. First of all, it is necessary to take into account that, as the decision tree, the random forest has several hyperparameters. Specifically, the optimal value is to be determined in the same way as before, that is, by making a sweep of the possible values of each hyperparameter and 10 Cross-Validation for each of them, selecting as optimum the one with the lowest error value.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

The hyperparameters thus determined were again `min_samples_split`, specifying the possible values from 2 to 21 with jumps of 2 by 2, `min_samples_leaf`, specifying the possible values from 2 to 21 with jumps of 2 by 2 and `n_estimators` (number of trees to be used in the model), specifying the possible values from 10 to 200 with jumps of 10 by 10. It is important to mention that in the case of the random forest, the hyperparameter `max_features` (number of variables randomly sampled as candidates at each split), was not introduced in the hyperparameter grid but was initially defined as the number of columns in the training inputs.

Figure 7.1: RF: Hyperparameters GridSearchCV plots



After that, the grid plot of the hyperparameters is obtained (Figure 7.1). In each of them, the optimal hyperparameter obtained can be observed. In this case, the optimal hyperparameters are as follows `n_estimators = 40`, `min_samples_split = 2` and `min_samples_leaf = 12`.

The measurements obtained are:

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.717878	0.81	Data 0.73

Since the two values of accuracy in training and test differ, we can conclude that we have fallen into the overfitting of training. However, we have obtained that the accuracy if we would consider all obs. 0 or 1 would be 0.5 ('No Information Rate'), as our accuracy is much higher, we also conclude that it is better to launch this model than to assign all observations a 0 or 1.

Figure 7.2: RF train and test Confussion Matrices

```
Confusion Matrix and Statistics
          Prediction
 Reference    0    1
         0 1548  399
         1  344 1686

Accuracy: 0.81
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.63
Mcnemar's Test P-Value: 0.05
Sensitivity: 0.83
Specificity: 0.8
Precision: 0.82
Recall: 0.8
Prevalence: 0.51
Detection Rate: 0.42
Detection prevalence: 0.52
Balanced accuracy: 0.81
F1 Score: 0.81
Positive label: 0
```

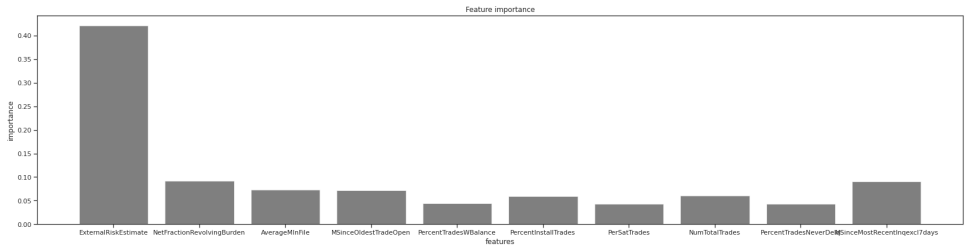
(a) Train

```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 348 139
         1 126 382

Accuracy: 0.73
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.47
Mcnemar's Test P-Value: 0.46
Sensitivity: 0.75
Specificity: 0.71
Precision: 0.73
Recall: 0.71
Prevalence: 0.51
Detection Rate: 0.38
Detection prevalence: 0.52
Balanced accuracy: 0.73
F1 Score: 0.72
Positive label: 0
```

(b) Test

As a consequence, we thought of decreasing the possible value specified in the hyperparameters in order to slow down or decrease the growth of the tree and avoid this over-fitting. However, the reduction of the hyperparameters would cause the same effect as the one produced in the decision tree, i.e. to obtain very few significant variables at the time of branching, but the clear advantage that the Random Forest model provides when considering how all variables are significant at a greater or lesser level (Figure 7.3) will be lost.Therefore, it was decided to leave this additional model in this way, even though it is known to cause some over-fitting.

Figure 7.3: Random Forest features importance plot



8 XGBoost

Finally, as an additional prediction method, we have decided to implement XGBoost. XGBoost is a supervised machine learning method for classification and regression, although in this case it will only be used for classification. XGBoost is short for "extreme gradient boosting". This method is based on decision trees and is an improvement over other methods, such as random forest and gradient boosting. We will use it because among its advantages are that it works well with large and complex datasets when using various optimization methods. As we have done in previous methods, we will use all the variables selected in preprocessing as inputs. Regarding the hyperparameters, we find:

- `max_depth`: The maximum depth of a tree, it is used to control over-fitting as higher depth will allow model to learn very specific relations to a particular sample.
- `learning_rate`: It is the step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features, and this shrinks the feature weights to make the boosting process more conservative. It makes the model more robust by shrinking the weights on each step. (It ranges between 0 and 1).
- `n_estimators`: represents the number of trees that will be adjusted sequentially during the boosting process. Each tree is fitted in a way that corrects the errors of the previous model. The value of n_estimators determines the number of iterations or tuning stages that will be performed. A larger value of n_estimators will generally lead to a more complex and powerful model, but can also increase the risk of overfitting if set too high.

In order to tune the hyperparameters we will use Cross Validation trying [100, 200, 500, 600] for `n_estimators`, [0.01,0.05,0.1] for `learning_rate` and [2,4,6,8] for `max_depth`. Once we have trained the model, it is obtained that the optimal values for the hyperparameters in this model are `max_depth = 4`, `learning_rate = 0.01` and `n_estimators = 500`.

With these values for the hyperparameters, we achieve an average value for the accuracy in the cross-validation equal to 0.726.

Avg CV Accuracy	Accuracy (train)	Accuracy (test)
0.726	0.78	Data 0.73

While in terms of the goodness of fit and the predictive capacity of the model, accuracy of the train and accuracy of the test set respectively, an accuracy = 0.78 is obtained in the train and accuracy = 0.73 in the test. That is, from the data that make up the train, we are able to predict them more correctly than predicting new values. With this case, we manage to correctly classify 73% of the new observations.

Figure 8.1: XGBoost train and test Confussion Matrices

```
Confusion Matrix and Statistics
          Prediction
 Reference    0    1
         0 1473  474
         1  397 1633

Accuracy: 0.78
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.56
Mcnemar's Test P-Value: 0.01
Sensitivity: 0.8
Specificity: 0.76
Precision: 0.79
Recall: 0.76
Prevalence: 0.51
Detection Rate: 0.41
Detection prevalence: 0.53
Balanced accuracy: 0.78
F1 Score: 0.77
Positive label: 0
```

(a) Train

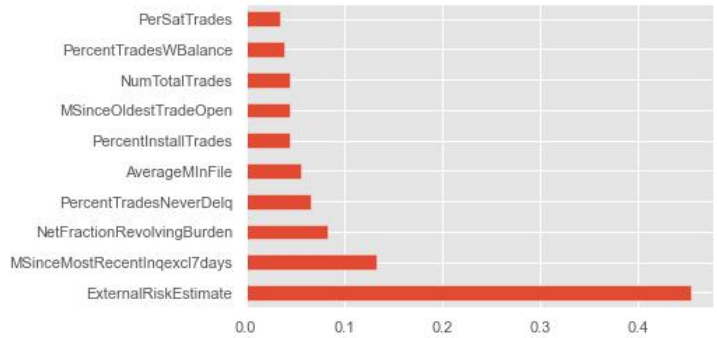
```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 346 141
         1 123 385

Accuracy: 0.73
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.47
Mcnemar's Test P-Value: 0.3
Sensitivity: 0.76
Specificity: 0.71
Precision: 0.74
Recall: 0.71
Prevalence: 0.51
Detection Rate: 0.39
Detection prevalence: 0.53
Balanced accuracy: 0.73
F1 Score: 0.72
Positive label: 0
```

(b) Test

Consequently, this model has been a little bit overtrained, it faces overfitting.

Regarding the most significant variables in the model, as the following graph shows, it is mainly `ExternalRiskEstimate` and `MSinceMostRecentInqexcl7days` that have the greatest relevance in the model.

Figure 8.2: XGBoost features importance plot



As was the case with past models, we obtained intermediate accuracy values; we have not been able to improve the models with this new contribution.

9 Creativity and innovation

Given that the models in which we obtained the best accuracy turn out to be those in which there is over-fitting, an investigation has been carried out for possible alternative techniques that avoid or reduce the possibility of over-fitting, that is, preventing the model from the patterns of the data in the training set which are learned. A possible technique to avoid over-fitting is known as data augmentation. Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. It includes making minor changes to the dataset or using deep learning to generate new data points. In this way, and automatically changing the training set data, the possibility of the model being able to learn the patterns in the data is avoided.

Another new idea that we wanted to contribute was the creation of the variable 'PerSatTrades'. This idea arises when we obtained that the correlation between the variables NumTotalTrades and NumSatisfactoryTrades was very high and, consequently, one of them had to be eliminated.

However, in order not to lose information, we wanted to calculate the variable 'PerSatTrades' which informs us about the percentage of TotalTrades that are Satisfactory Trades', a new variable that no longer has a correlation with the original one that we do maintain.

In parallel, we have also decided to train two additional models to those already proposed. These chosen models have been the Random Forest and the XGBoost.

Random Forest and XGBoost are two widely used machine learning algorithms for classification tasks. Both models are based on the assembly of trees, the first being an assembly of independent trees, while XGBoost is based on the assembly of sequential trees using gradient boosting.

We chose Random Forest for its tendency to produce robust and accurate models due to combining multiple trees and being robust to outliers.

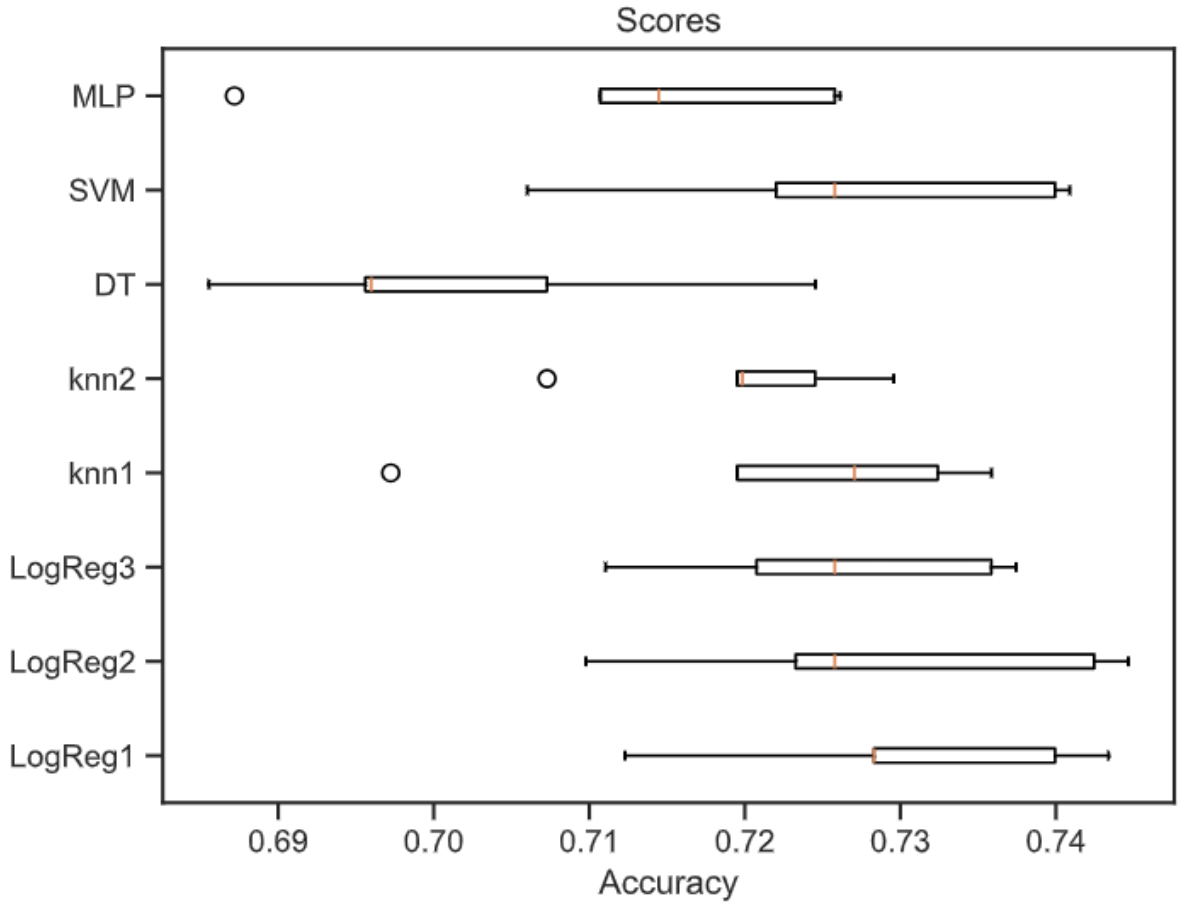
And we also tested XGBoost because of its high performance and accuracy and its regularization options that help prevent overfitting.

10 Model comparation

Once we have all the models trained with their respective optimal structures, we are able to compare all of them. To make these comparisons, we will base ourselves on the 'accuracy' values, both those obtained in the CrossValidation with 5 folds, the 'accuracy' obtained in the train (relative to the goodness of the fit), and the one obtained from the test (the which informs us about the predictive capacity of the model). In addition, let's also compare the area under the ROC curve of each model.

Model	avg_cv_acc	acc_train	acc_test	sensi	speci	ROC
Logistic Regression (2.1)	0.7305	0.73	0.74	0.75	0.72	0.792
Logistic Regression (2.2)	0.7292	0.73	0.74	0.76	0.72	0.792
Logistic Regression (2.3)	0.7262	0.73	0.74	0.74	0.74	0.794
KNN (3.1)	0.7224	0.73	0.74	0.73	0.75	0.792
KNN (3.2)	0.72014	0.73	0.74	0.74	0.74	0.796
Decision Tree (4)	0.71797	0.7	0.7	0.76	0.64	0.746
Linear SVC (5)	0.72693	0.73	0.74	0.77	0.71	0.801
Radial SVC (5)	0.72844	0.74	0.74	0.78	0.71	0.811
MLP (6)	0.729747	0.73	0.73	0.76	0.71	0.807
Random Forest (7)	0.717878	0.81	0.73	0.75	0.71	0.803
XGBoost (8)	0.726	0.78	0.73	0.76	0.71	0.804

Figure 10.1: Cross-Validation Accuracy Boxplots for each model



The conclusions based on this table and the plot of those accuracies, will be analyzed in the next chapter related with the conclusions of the Assignment.

11 Conclusions

Before training any type of modeling, an EDA (Exploratory Data Analysis) and preprocessing of this data must be carried out to guarantee the quality of the data, select relevant

features, prepare the data for the models and avoid problems in the modeling . In the data set used in this work, it was necessary to eliminate a large number of outliers that were useless due to missing values in the output variable. An input variable also had to be eliminated due to its high correlation with another of the variables, and we have created a new one in its place to capture more information. Moreover, we have also eliminated inconsistent negative values in the data set. Based on the results obtained and given the small difference in improvement in terms of accuracy between the models, the choice of the optimal model for the specific application that concerns us can be approached in several ways.

Firstly, if we are looking for a better result in terms of the predictive capacity of the model, the optimal model would be the SVM. This is because it is the one that gives the best accuracy results without falling into over-fitting compared to Random Forest or MLP, which, although they provide good results in terms of accuracy, fall into over-fitting, providing better results in training than in the test.

On the other hand, if we focus on the interpretability of the model and given that all models give very similar results in terms of predictive capacity, the optimal option for this application would be the logistic regression model. Additionally, this model requires less computational consumption of resources and its execution and obtaining results is faster.

Thus, depending on the needs and priorities of the project, the choice of the optimal model will vary.