Predicting Loan Offer Selectivity Using Process Mining

Abd Alrhman Abu Sbeit (2043017),   
Gianluca Bortoluzzi (2064375),   
Huy-Phuong Lê (0916939)

# Introduction

In this project, we analyze a loan application process between a financial institution and its clients. The process owner is the bank, and the main external stakeholder are the loan applicants.

In an ideal flow, the process starts with a client application for a loan. The client specifies the requested loan amount, and delivers the required documents to the process owner. The business owner then creates an offer (or multiple) and sends them to the applicant, while checking the applicant documentation. Several scenarios are possible at this point:

1. The client refuses all offers from the bank
2. The client selects one of the received offers

In the first case, the process ends, which is a scenario we want to avoid. In the second case, the client accepts the offer. The process owner then decides whether to end the process by completing or deleting the application. In principle, this decision is based on an internal economic evaluation.

In this analysis, we primarily focus on predicting whether a client will *select* an offer made by the bank. The task of the model will therefore be *predicting the choice of the client*, not the outcome of the entire process. Firstly, this provides insight into client characteristics, allowing for further qualitative analysis. Secondly, the process owner will be able to decide whether to reduce efforts on applicants likely to reject each offer, or conversely increase efforts.

For this prediction task, we employ several process mining, data analysis and prediction techniques. In the next sections, we use process mining tools to comprehensively understand and visualize the complex loan application process. We then use data analysis for feature selection based on correlation analysis. After feature selection, we implement and compare two predictive models, namely Random Forest and Gradient Boosting. Our results show a better prediction accuracy for this task with a Random Forest model (88% accuracy) than Gradient Boosting (68% accuracy), and both significantly outperform a naive classifier for this dichotomic prediction task (½ = 50%). In the end, we conclude the study by discussing and summarizing the implication, relevance and shortcomings of the project. Finally, in the appendix we discuss the prediction setup, explaining how the data was handled, and present some technical evidence as well as further analysis.

# Analysis

This section expands upon the relevance of the chosen prediction task, as well as detailing the analysis performed. Next, we explain the implementation of the prediction models, their evaluation, and their results.

## Prediction task

In this analysis, we primarily focus on predicting whether a client will *select* an offer made by the bank. The reason we focus on *selection* and not *acceptance* is related to the provided data, which shows that an offer can be *Accepted: True, False* and *Selected: True, False*. The necessary condition for an application resulting in a loan is to have an offer which has the value *True* on both case attributes. If an offer is *Accepted* but not *Selected*, data shows that the bank continues to send new offers until an offer is *Selected*, or the process ends. Therefore, we focus on predicting the choice of the client with respect to the offer, instead of the entire process.

Studying the choice of the client is interesting both for inference and description reasons. From an inferential point of view, the process owner is interested in knowing the possible outcomes of an application to establish a data-based strategy. On the other hand, from a descriptive perspective, knowing and understanding feature importance can be useful for conducting further research.

The prediction task can be summarized as the following research question:

*How likely is a loan applicant to select an offer, based on the offer attributes?*

In the next paragraph, we will cover the steps taken to reach the prediction models. Starting with the data pre-processing, we then obtained significant insight on the data and investigated the best prior features for our model.

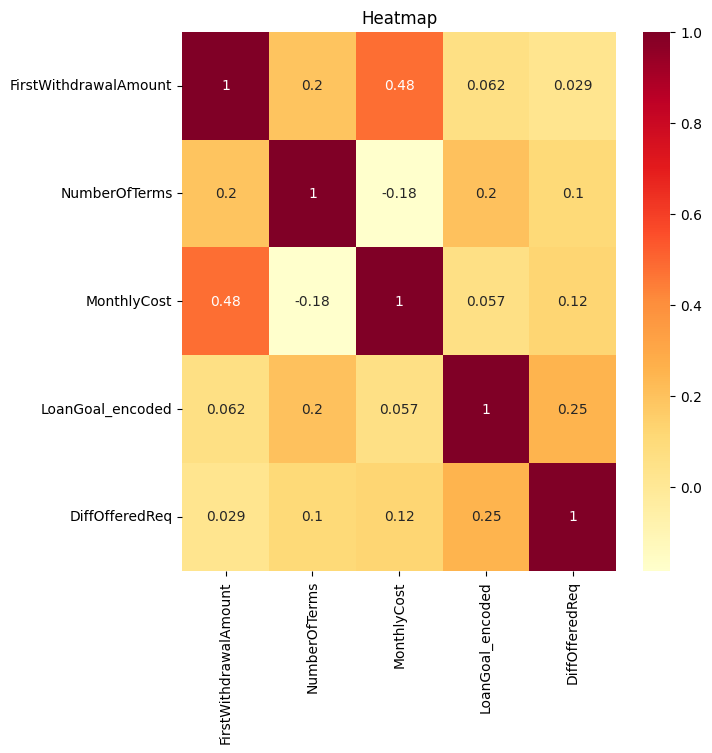
### Data pre-processing

To reduce the amount of noise from the dataset, we conducted a preliminary data exploration to identify traces that are incomplete and do not meet a frequency threshold. By analyzing the traces of the provided data in the process mining software Disco, we noticed that 87% of the trace variants only have one case. Hence we decided to only select traces that have 10 cases or more.

Next, we excluded events that do not end with the event *complete* by projection of the *lifecycle:transition* event attribute, to avoid including incomplete traces in our analysis and have a clearer view of the process. After filtering the data, we performed data exploration to identify the most informative features for our prediction tasks, discussed in the next section.

### Data Exploration and Feature Selection

To identify potential relationships between the features contained within the provided dataset, we explored the data using a correlation heatmap. Contextual and practical reasons have been considered too. For the chosen prediction task, only the offer attribute can have a meaningful impact. Therefore, given the prediction task described in Section 2.1and the business context, we selected the following prior features: *Monthly Cost, Number of Terms*, *Difference between Offered and Requested Amount*, *Loan Goal* and *First Withdrawal Amount*. Figure 1 shows the correlations between the analyzed features, and we can clearly see that they are not strongly pairwise correlated. This allows us to build a model knowing that all these features will contribute to the prediction.

*Figure 1: Correlation heatmap. This heatmap shows the correlation between each feature as values within each cell. The darker a cell, the stronger the correlation.*

#### Monthly Cost

This feature represents the amount to be paid monthly by a client to settle the loan if the offer is selected. Given that this amount is based on *Offered Amount*, it was expected to find great variability in the data. The average monthly cost when analyzing the full dataset is close to €270. However, if we consider the interquartile range (IQR), the mean value results in €232. Lastly, we observed that this feature is related to *Offered Amount* and *Requested Amount*. *Figure 2* shows that most of the values for *Monthly Cost* are centered around this mean value. There is a right skew, suggesting that there are some, but not many, larger monthly costs within the data.

#### Number of Terms

*Number of Terms* refers to the number of payback terms that a client agrees to upon accepting a loan offer. This feature is most related to *Offered Amount* and, transitively, also with *Requested Amount*. In *Figure 2*, we can see that the range of “Number of terms” is quite wide. As some terms have a much higher occurrence, there may be some standard or default number of terms associated with these loans.

#### First Withdrawal Amount

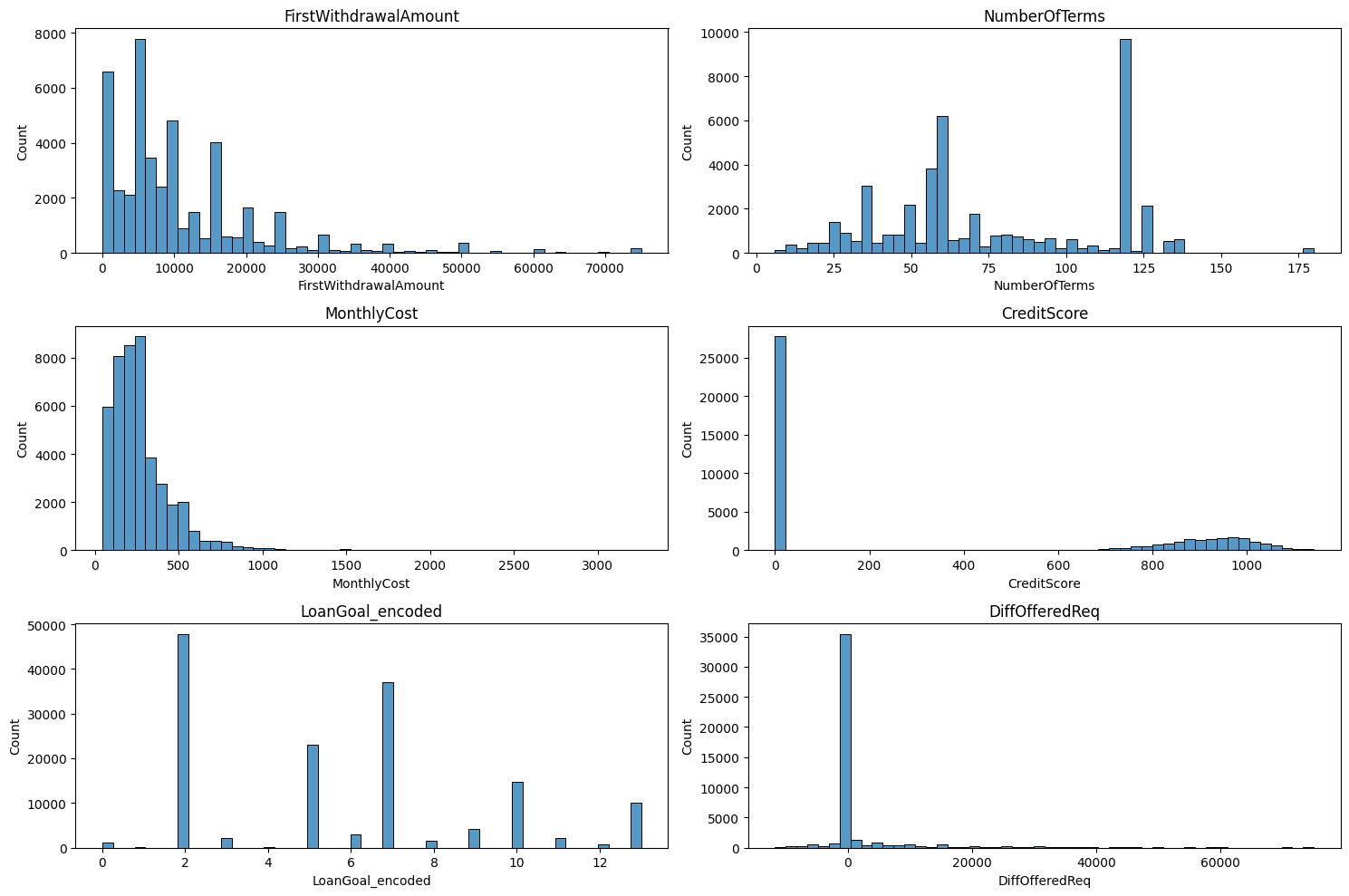
This feature represents the first amount a client can withdraw after the process ends successfully.

#### Loan Goal (Encoded)

*Loan Goal (Encoded)* represents the reason why a client takes the loan. This feature was constructed by applying label encoding to transform the original categorical feature into a numeric feature.

#### Difference Offer Request

*DiffOfferedReq* represents the difference between the requested loan amount by the client and the amount offered by the bank. This feature was constructed by simply taking the difference between these two original features.



*Figure 2: histogram of each feature. The x-axis denotes the value of the feature, and y-axis denotes the frequency of that value occurring in the dataset.*

The selected features offer a starting point for the model training process, by providing empirical evidence of relationships within the data which is required for effective model training. In the next section, we discuss the model training process.

To explore the data further, we performed other analyses described in the Appendix. In those analyses we explore other features that should be relevant to the task, but for several reasons can’t be used in the model.

## Implemented models

To implement models that satisfy the prediction tasks defined in Section 2.1, we relied on two different methods to comparatively assess their performance. These models are both based on decision trees: Random Forest Classifier and Gradient Boosting Classifier. In the next sections, we discuss their implementations and results.

### Random Forest Classifier

The first prediction model we implemented is a Random Forest Classifier. This model consists of a group of regression trees, which are independently trained on a randomly selected subset of features, and predict the outcome through a ‘majority vote’. A distinguishing feature of Random Forest models is the deliberate inclusion of randomness throughout the construction of individual trees, as they bootstrap samples during the construction of trees, and select random subsets of features for each split. This deliberate randomness decorrelates the trees within the ensemble, and enhances the model's resistance to noise, which is part of our dataset, and improves model generalizability.

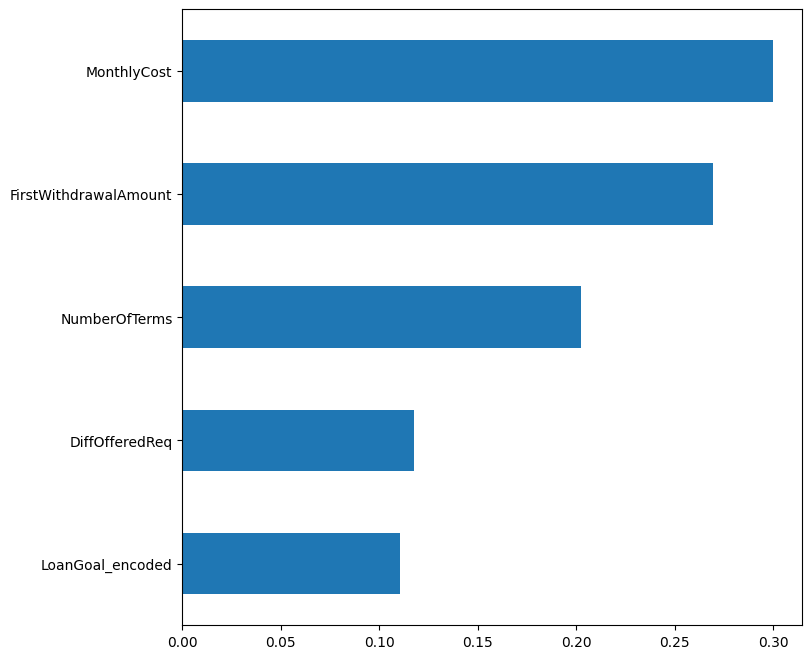
#### Model implementation and evaluation

With the features selected as described in Section 2.2, we proceeded by implementing a Random Forest Classifier in Python using the scikit-learn package. We use these features, namely *DiffOfferedReq,* *Loan Goal*, *First Withdrawal Amount*, *Monthly Cost*, and *Number of Terms*, as the independent variables of our model, and train the model on these features to predict the dependent variable *Offer* *Selected*.

For model training and evaluation, we divide the data into training, validation and test sets. We train the model on the training and validation data, and use the remaining test set to evaluate the performance of the model. This split ensures that the model is evaluated on new and unseen data. More details about the model training and evaluation process can be found in Section 4.2 in the Appendix.

Using this training and evaluation setup, the selected features, and the Random Forest Classifier model, we obtained test accuracy of 0.88. This means that the model is able to correctly predict a client’s decision to accept or reject an offer in 88% of the cases. This is significantly higher than a naive classifier, where random classification would result in an accuracy of 0.5 for this binary classification task. This means that the model is effective in extracting information from the selected features to predict whether a client will accept or reject an offer.

Besides the prediction accuracy, we also obtain insights into the importance of each feature in the model. These importances are shown in Figure 4. These values are computed by the model by analyzing the ability of each feature to enhance the pureness of the leaves of the tree; intuitively, this means that a higher value implies that a feature has a higher separation ability. In other words, distinguishing whether a loan will be accepted or rejected is more strongly dependent on that particular feature.



*Figure 3: importances of each feature as calculated by the Random Forest Classifier. The y-axis denotes the features, and the x-axis denotes the feature importance.*

As Figure 3 shows, notably, *Monthly Cost* appears to be the most important feature, followed by *First Withdrawal Amount*, *Number of Terms*, Difference between *Offered Amount and Requested Amount*, and finally *Loan Goal*. It is worth noting that none of these features are near zero, hence it is likely that they each play a non-trivial role in classifying whether an applicant will accept or reject a loan offer.

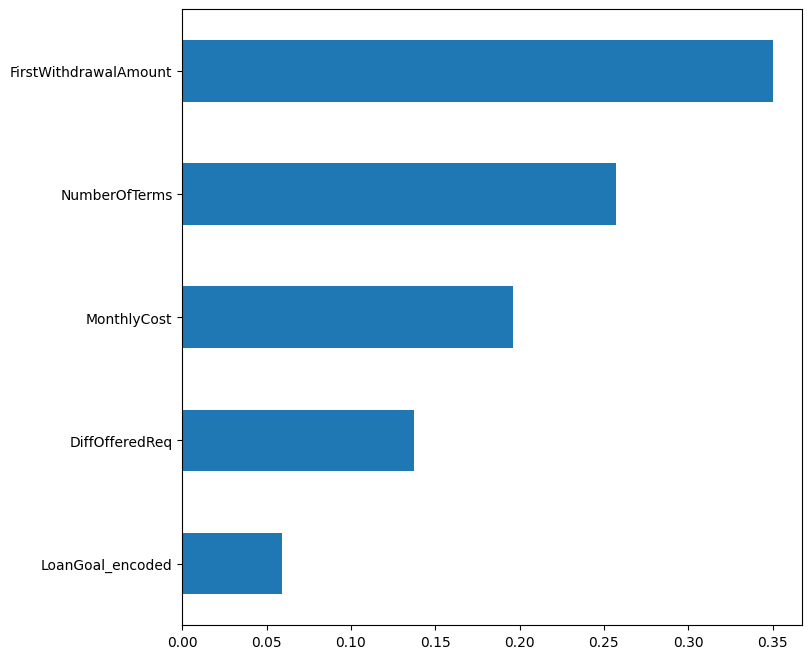
However, the monthly cost, first withdrawal amount and number of terms of a loan do seem to play a bigger role than the difference between offered amount and requested amount, and the loan goal. Using this information, the bank can for example customize loan offers in such a way that prioritizes the attractiveness of monthly cost, first withdrawal amount and number of terms for a client.

### Gradient Boosting Classifier

Gradient Boosting is an optimization technique that adjusts its predictions by sequentially training a tree to approximate the negative residuals of the previous one. The algorithm then keeps performing gradient updates in the Gradient Descent fashion.

#### Model implementation and evaluation

For the Gradient Boosting Classifier, we use the exact same training and evaluation setup as the Random Forest Classifier. Using this setup, we obtained a test accuracy of 0.68 for the Gradient Boosting Classifier model, hence the model is able to correctly predict a client’s decision to accept or reject an offer in 68% of the cases. While this is still higher than a naive classifier (50%) this accuracy is considerably lower than the Random Forest Classifier (88%)



*Figure 4: Importances of each feature as calculated by the Gradient Boosting Classifier. The y-axis denotes the features, and the x-axis denotes the feature importance.*

Figure 4 depicts the feature importances associated with each feature by the Gradient Boosting Classifier. As this figure shows, notably unlike the Random Forest model, *First Withdrawal Amount* is the most important feature, followed by *Number of Terms*, *Monthly Cost*, Difference between *Offered Amount and Requested Amount*, and finally *Loan Goal*. This difference could be explained by the different ways in which each model extracts patterns from the data. However, since the Random Forest Classifier has a higher accuracy, the insights from that model seem to be more reliable. However, by constructing additional models, it can be seen that different insights can be obtained from the data.

In the conclusion, we discuss the potential impacts of these model insights for the business.

# Conclusion and Discussion

In this work, we focused on predicting a client choice to *select* an offer made by the process owner. To build a predictive model, we then analyzed the available dataset to select the best prior features for our specific task. We found out that the offer attributes were indeed the best predictor for the offer selection.Two distinct prediction models have then been implemented and evaluated. Now, based on the specific requirements of our machine learning model, we can determine which algorithm would be more advantageous. Since our primary objective is to maximize the overall accuracy of predictions, without giving significant weight to incorrect predictions, the Random Forest model yields to be more accurate in predictions.

The model predictions will provide the business with the insights on what the clients value the most while choosing to select an offer. It will reduce the amount of offers sent that are more likely to be rejected, and the time and money spent by the business on the application, while enhancing offer creation. Given that the features are all related to the offer, our model is of particular interest during the offer creation process. Each offer is created by studying on a set of information available on the client. Thanks to our model, the evaluation of the best offer for the client can be integrated with the knowledge of feature importance. Some characteristics of the offer are more relevant than others to conclude successfully the process and are therefore considered more important by the client.

The model relevance also highlights some of its limitations. Firstly, we can note that our models can predict the offer selection outcome based on the offer characteristics. This is not of particular use if the process owner wants to know what client characteristics are most important, nor if the owner is interested in what steps of the process can influence the choice. This is due to the limitations of the available features, discussed further in the Appendix (e.g. waiting time, credit score); and the available information regarding the process context, as for the meaning of some steps of the process or event attributes.

To conclude, we would advise the process owner to use our model during the offer creation process. Before sending an offer to the customer based solely on a risk assessment, consult the model feature importance graph and statistic to understand what aspects are more important to the customer. Run the attribute of different offers to understand which one is more likely to be accepted.

Moreover, a recommendation to allow for future deeper analysis. The business should record extra information about the clients and fix the credit score data recording. Having more accurate information about the client and the process would increase the accuracy of our model and will make it possible to implement different models and prediction tasks, which will pay back positively to the business through an increase in successful loan applications.

# Appendix

The choice of this prediction task came after a heavy examination of the data set and the information available to us. We identified a distinction between offer acceptance and offer selection. The data reveals instances where an offer is accepted but not selected. This distinction is foundational for our prediction task, aligning with the complexity of the loan application process. Also, many cases have multiple offers sent to the client for one loan application, which can even increase the complexity of the process more, and use more resources from the business side. Throughout the appendix, we are showing more details about the data, and the models being used in this experiment.

## Prediction setup

Tools and packages used:

* Disco: for creating process models and data filtering
* Python: for model training and evaluation
  + Python packages:
    - PM4PY: for dealing with process mining data in Python
    - scikit-learn: for training and evaluating predictive models in Python
    - pandas: for data processing
    - seaborn: for visualizations
    - matplotlib: for visualizations
    - plotly: for visualizations
* R: for data exploration

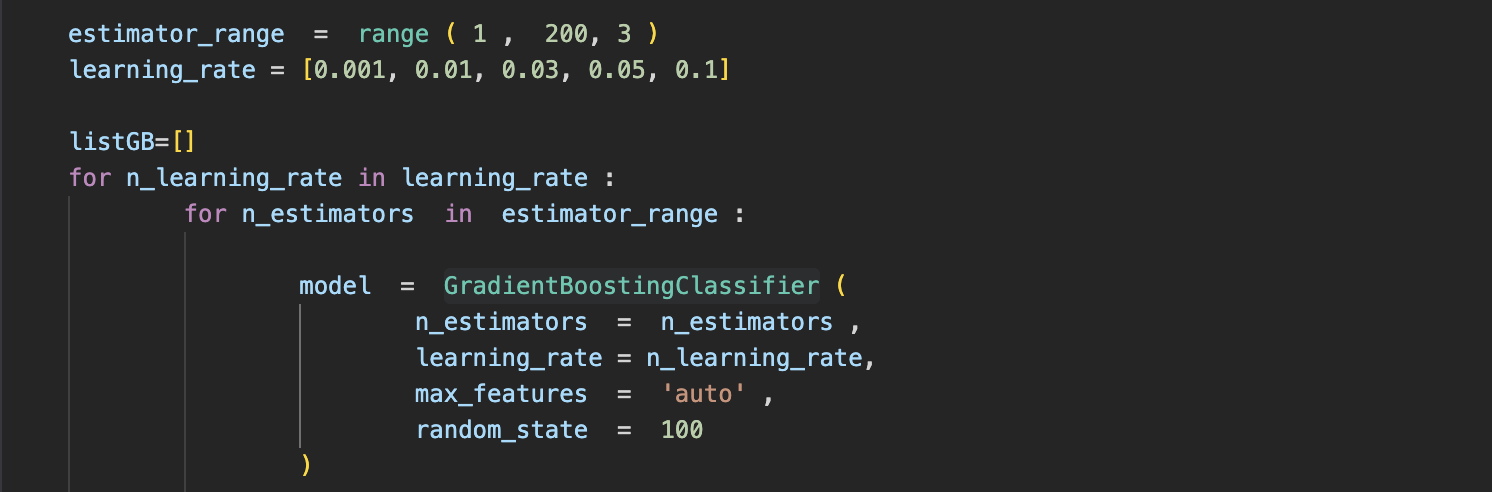
Data preparation

* Data cleaning
  + in Disco, we filtered the data by excluding trace variants with less than 10 cases (selection)
  + then we only keep events with *lifecycle:transition* set to *complete* (projection)
  + next, we exported the filtered dataset as a .CSV file
* Feature selection
  + We analyzed the data using correlation heatmaps and histograms
* Feature encoding
  + Encoding the data categorical data into a numerical one is crucial in machine learning modeling

## Realization of prediction model

By following the implementation on Github[[1]](#footnote-1), you will notice that the model is done over multiple Jupyter Notebooks, starting from **A\_data\_loading** where we are doing trace analysis, process discovery, data analysis, feature engineering, and some visualization, then, **B\_data\_analysis**, most of the remaining work is done in this notebook, starting from feature engineering, feature encoding, data visualization, split-train-test Random Forest model, 5-fold cross validation Random Forest model, and Gradient Boosting Classifier model.

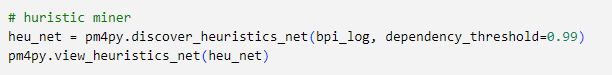
Although we implemented two models, the Random Forest model was trained and validated over two train and validation techniques, **split-train-test**, were we split the data into 70% training set and evaluated the model on 30% testing set, the other technique is **Cross Validation**, were we didn’t split the data, but the model was trained and evaluated five times using five different training and testing sets. In the Gradient Boosting model, the same 70% training and 30% testing sets were used, also a tuning technique was used on this model on order to find the optimal number of estimators, and the optimal learning rate as shown in figure 6, where we left the usage of number of features to the model to decide. The tuned parameters correspond to a constant learning rate of 0.1 and a number of estimators (how many trees are produced) of 196.

  
*Figure 5: Gradient Boosting Classifier tuning method*

During the model implementation, we tried to do different types of visualizations, techniques, and preprocessing. **Data\_loading\_and\_analysis** and **sandbox** files contain most of the preprocessing and visualizations that are not taken into consideration when we implement the final model. **Implementation**, **randomclassifier**, and **Decision\_Tree** files contain the first implementations of the Random Forest model, in **Decision\_Tree** file, we tried to use prefixes technique which in our case wasn’t useful and lead to data leakage which led to a **100%** accurate model.

### Piece of evidence 1: process model (compulsory!)

We created multiple visualizations of the process model using ProM, Disco and pm4py. Here, we show an example of the implementation in pm4py using the “pm4py.discover\_heuristics\_net” function on the pre-processed data:



*Figure 6: Heuristics net pm4py*

The resulting process model is shown in Figure 7. We can obtain various insights on this model. For example, the process can end in three ways:

1. A\_Pending, if the client selects an offer;
2. O\_Refused, if the client refuses all offers;
3. O\_Cancelled, if the process owner chooses to end the application.

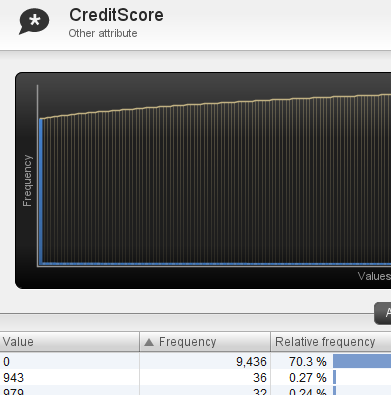
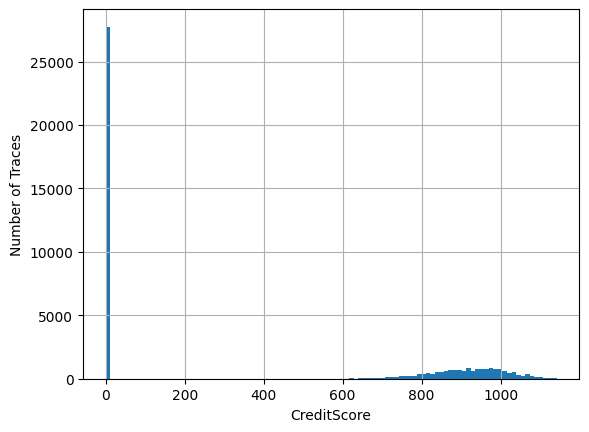


*Figure 7: Visualization of the loan process*

### Piece of evidence 2: process model / figure / table

Credit score feature

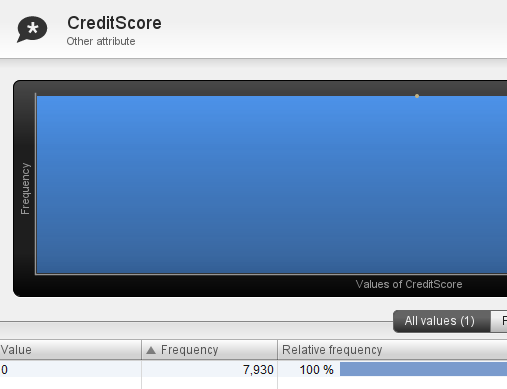
*CreditScore* initially appeared as a reliable feature for outcome prediction. However, upon closer inspection we identified some irregularities corresponding to this feature: a higher credit score for a particular client should result in a lower risk of loan for this client, however, we observed that 100% of the cases that have a non-zero credit score, also have selected an offer. Therefore, commonly employed data imputation strategies, such as replacing zero values with median or mean values would only shift the problem, especially because most cases have a zero value for this feature (as shown in *Figure 8*). The most probable explanation for this data characteristic is that the credit score was recorded in the dataset after the case had ended. We decided to exclude this feature from the model, as a 100% accuracy could be achieved by simply including this feature in the model.



*Figure 8 & 9: Visualizations of the class distributions for the feature CreditScore*

Moreover, we discovered that for cases that did not select any offer, the credit score is equal to zero. This can be verified by adding the following filters to the dataset in Disco (other than the one specified in the section pre-processing):

1. Attribute > filter by: *CreditScore* > Keep Selected > deselect empty row
2. Attribute > filter by: *Selected* > Forbidden > *true*

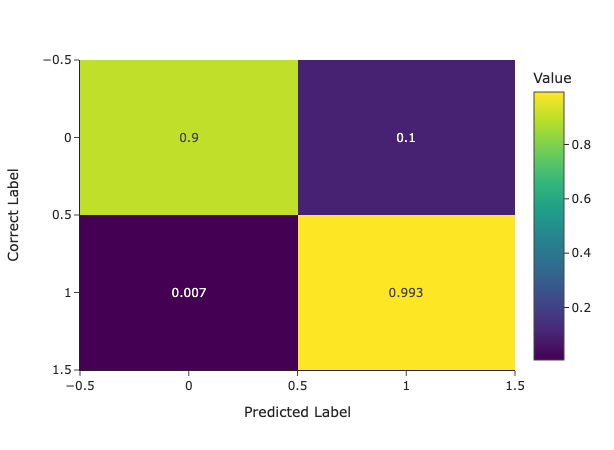
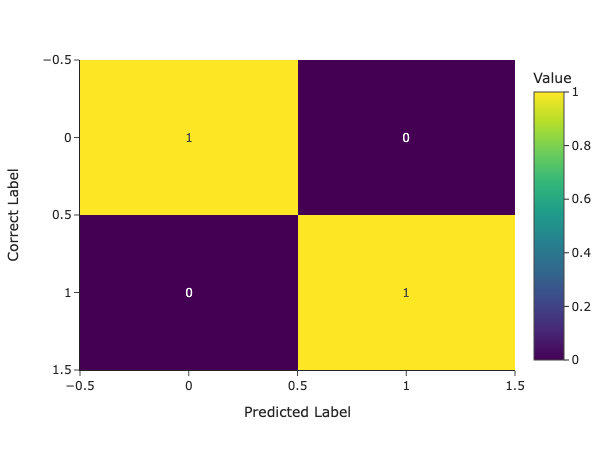


*Figure 10: Relative frequency credit score being equal to zero is 100%*

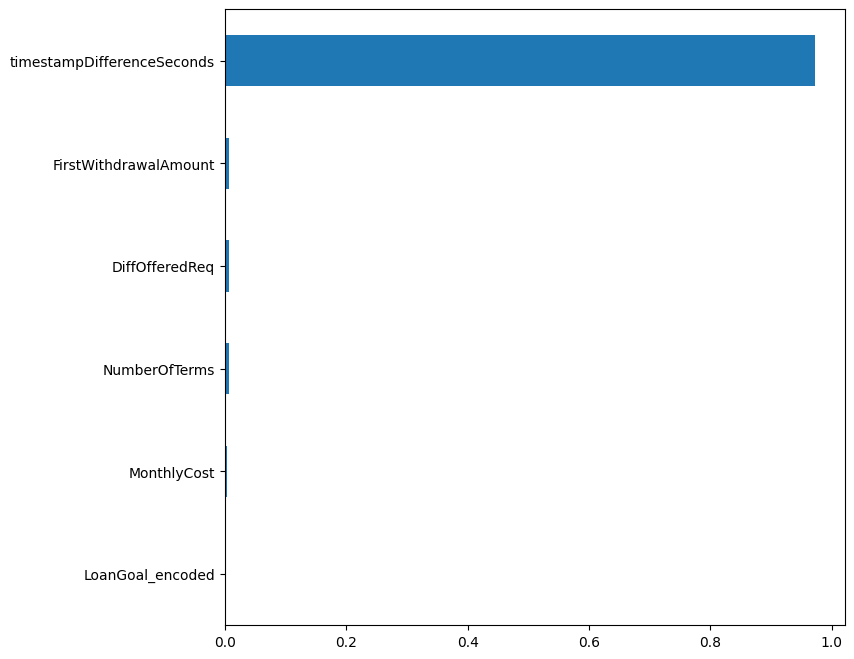
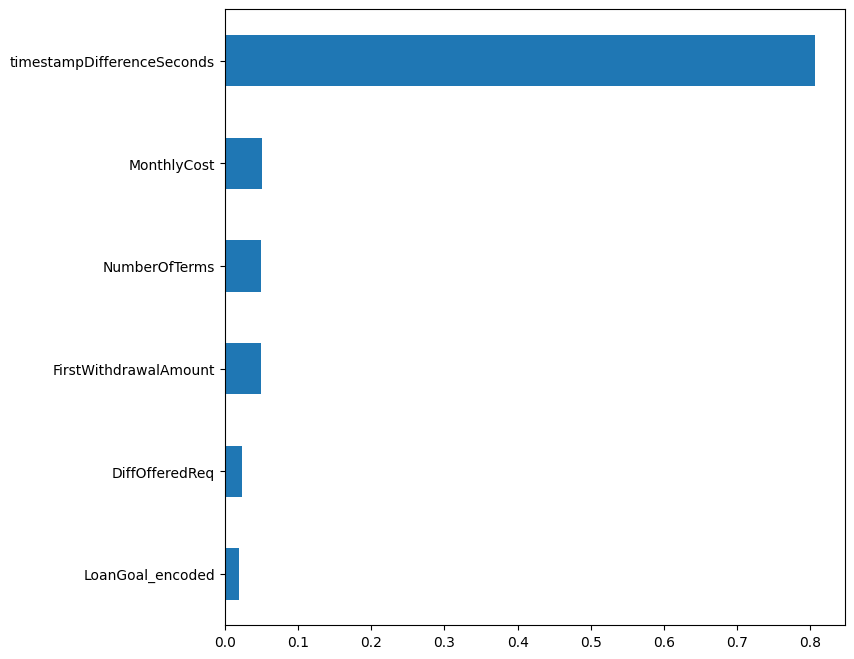
Waiting Time feature

An important feature to add to a choice prediction model would have been the client “waiting time”. At the same time, from a business perspective it is of utmost importance to understand the impact of waiting periods on the clients choices.We operationalized waiting time, in the context of our prediction task, as the time waited between the start of each case up until the reaching of one out of two events: “O\_Returned” and “O\_Cancelled”. The reason why this feature has not been selected is related to the available data. In fact, when performing Random Forest and GBC using this data we obtain 99.9% and 94% accuracy. At the same time the feature importance of waiting time in the two models reaches 80% and 97% with respect to all other features.This is due to he fact that the waiting time, starting at O\_Created” for “O\_Returned” is 8.6 days[[2]](#footnote-2), while “O\_Cancelled” waiting time is 27.3 days. This disproportion is probably due to data scarcity or business choices that we cannot see from the data. The feature has been removed from the model after these considerations.

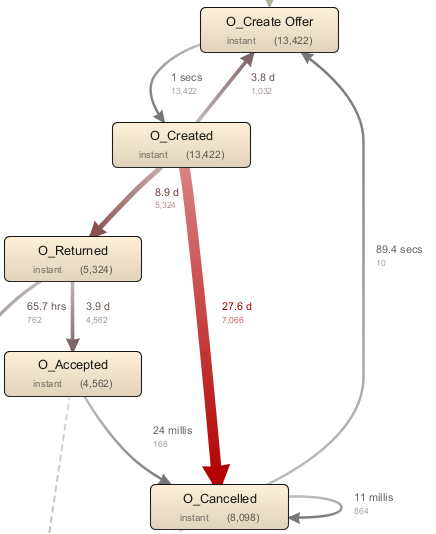
Using the waiting time between creating an offer and getting a response from the client as a feature, this leads to 99.9% accuracy using RF model with 80% of the prediction based on this single feature, and 94% accuracy using GBC model with 97% prediction based on this single feature. The following shows more information about using this information as a feature.



*Figure 11: The difference between the confusion matrix for RF and GB models respectively, when taking waiting time into consideration*



*Figure 12: The difference between the feature importance for RF and GB models respectively, when taking waiting time into consideration*



*Figure 13: The average time taken between offer related activities*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **“Selected”** | **Mean** | **Median** | **Min** | **Max** |
| **True** | 8.6 | 7 | 0 | 30 |
| **False** | 27.3 | 30.7 | 0 | 46 |

*Table 1: Comparing between the time taken between selecting an offer and rejecting it in* ***days***

## Analysis of prediction model quality

### Quantitative analysis

Random Forest Classifier

Table 2 provides additional evaluation metrics for the Random Classifier Model. The recall score indicates that for all accepted loans, 88% of the cases were correctly identified by the model as accepted. On the other hand, the precision score indicates that 90% of the ‘accepted’ predictions by the model were correct. The F1 score is the harmonic mean of precision and recall.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy Score** | **Recall Score** | **Jaccard Score** | **F1 Score** | **Precision Score** |
| 0.88 | 0.84 | 0.77 | 0.87 | 0.90 |

*Table 2: evaluation metrics and results of the Random Forest Classifier model*

Gradient Boosting Classifier

The results obtained with the Gradient Boosting Classifier are shown in Table 3. As can be seen, this model performs worse than the Random Forest Classifier on all metrics.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy Score** | **Recall Score** | **Jaccard Score** | **F1 Score** | **Precision Score** |
| 0.68 | 0.61 | 0.48 | 0.65 | 0.69 |

*Table 3: evaluation metrics and results of the Gradient Boosting Classifier model*

Total model prediction errors by both models

Table 4 shows the number of total predictions made by each model per class of the confusion matrix. This table shows that the Random Forest model is both less prone than the Gradient Boosting model in terms of false positive and false negative errors. We calculated these numbers by getting the total number of actual acceptance and rejection, then multiplied it with the percentages from figures 3 and 5A.

The Random Forest model shows more false negative errors than false positive errors. As a result, there’s a higher proportion of offers that would have been accepted by the applicant but are predicted to be rejections. Depending on business context, further model optimizations or model calibrations could mitigate this issue to an extent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **True-Positive** | **True-Negative** | **False-Positive** | **False-Negative** |
| **Random Forest** | 18148 | 21113 | 1936 | 3380 |
| **Gradient Boosting** | 13218 | 17287 | 5762 | 8310 |

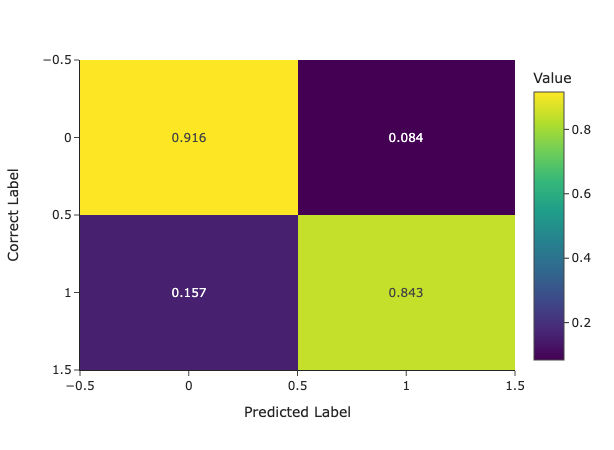
*Table 4: Comparing between the Number of Prediction Errors for the Random Forest and Gradient Boosting models*

The Random Forest model shows more false negative errors than false positive errors. As a result, there’s a higher proportion of offers that would have been accepted by the applicant but are predicted to be rejections. Depending on business context, further model optimizations or model calibrations could mitigate this issue to an extent.

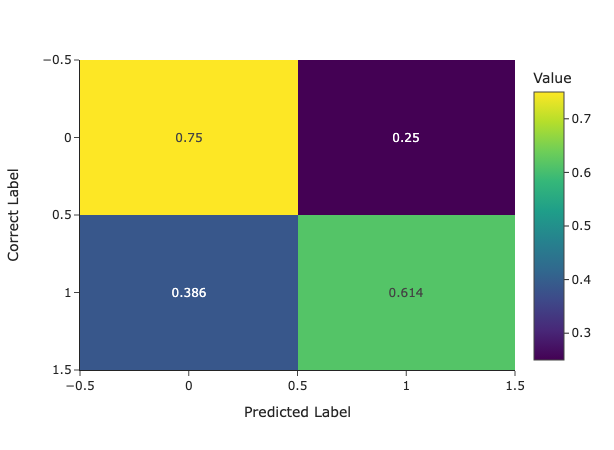
### Qualitative analysis

The identified hierarchy of feature importance in section 2.2 figures 3 and 4, presents strategic opportunities for the bank. Customizing loan offers with a focus on optimizing Monthly Cost, First Withdrawal Amount, and Number of Terms can enhance the attractiveness of offers to potential clients. This approach aligns with the observed priorities in client decision-making, providing actionable insights for the financial institution to refine its lending strategies and improve the acceptance rates of loan offers.

In Figures 14 and 15, the prediction errors of the both models are shown. As can be seen, the Random Forest Classifier is slightly better at identifying which applicants are likely to reject a loan than accept them, although the discrepancy is not very large (<0.07% difference). Compared to the Random Forest model, the Gradient Boosting model is both less accurate at identifying which applicants will reject or accept a loan as it is both more prone to making false positive and false negative errors.



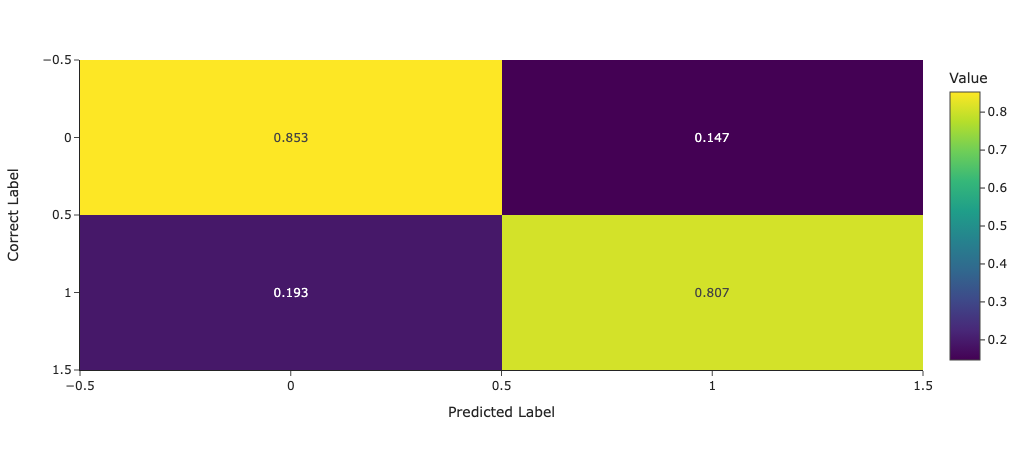
*Figure 14: Prediction errors of the Random Forest Classifier. Top left cell: true negatives, top right cell: false negatives, bottom left cell: false positives, bottom right cell: true positives*



*Figure 15: Prediction errors of the Gradient Boosting Classifier. Top left cell: true negatives, top right cell: false negatives, bottom left cell: false positives, bottom right cell: true positives*

## Full dataset prediction results

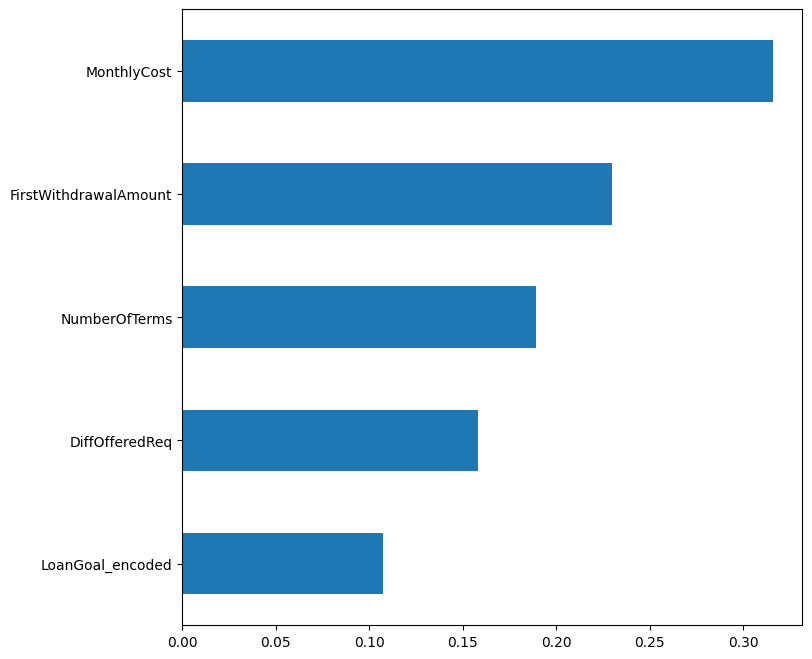
As expected, our results show a better prediction accuracy for this task on the full dataset[[3]](#footnote-3) with a Random Forest model (**82.9%** accuracy) compared to Gradient Boosting model (**60.8%** accuracy) but overall worse prediction accuracy compared to the run over the filtered dataset[[4]](#footnote-4) (**88%** Random Forest accuracy, and **68%** Gradient Boosting accuracy), and both significantly outperform a naive classifier for this dichotomic prediction task (½ = 50%). The feature importance for both models is still the same, for Random Forest ***Monthly Cost***appears to be the most important feature, followed by ***First Withdrawal Amount***, ***Number of Terms***, **Difference between *Offered Amount and Requested Amount***, and finally ***Loan Goal***, and ***First Withdrawal Amount***is the most important feature, followed by ***Number of Terms***, ***Monthly Cost***, **Difference between *Offered Amount and Requested Amount***, and finally ***Loan Goal*** for Gradient Boosting model.



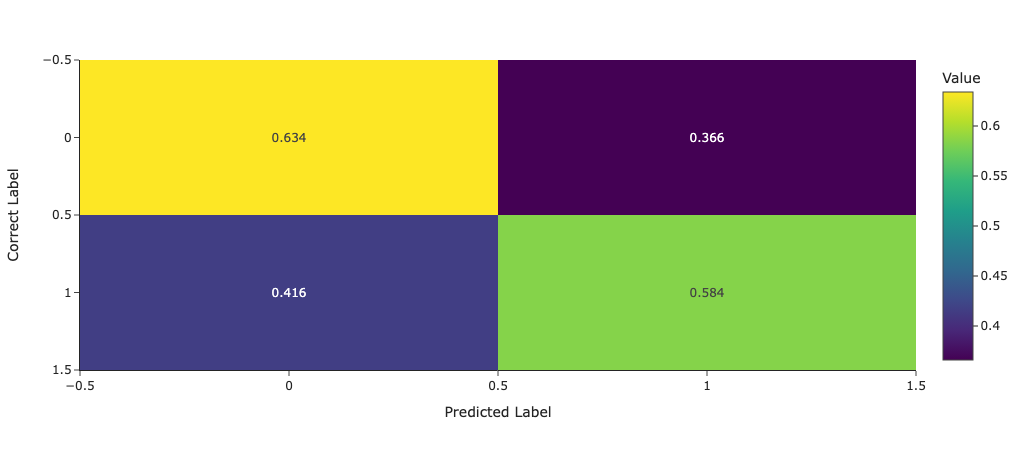
*Figure 16: Prediction errors of the Random Forest Classifier over the full dataset.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy Score** | **Recall Score** | **Jaccard Score** | **F1 Score** | **Precision Score** |
| 0.829 | 0.806 | 0.705 | 0.827 | 0.849 |

*Table 5: evaluation metrics and results of the Random Forest model*



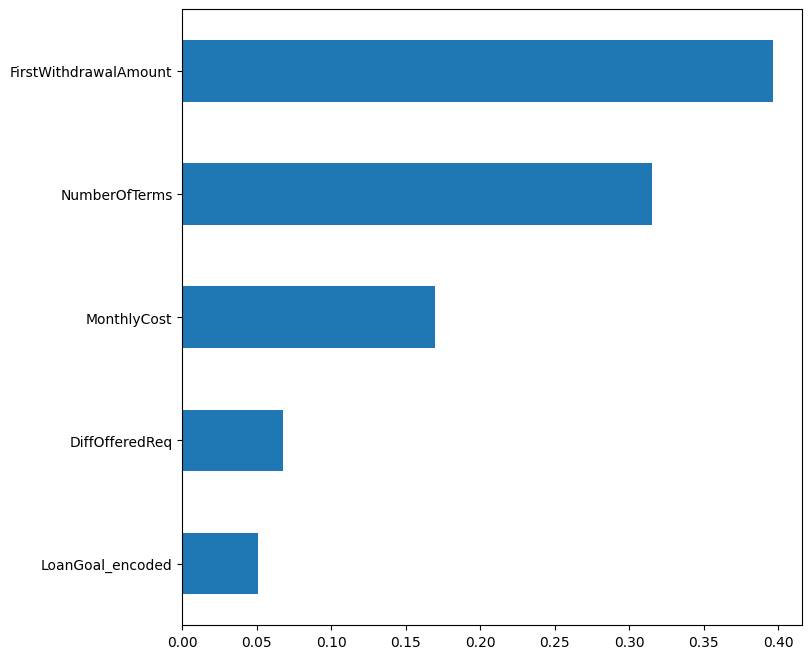
*Figure 17: Importances of each feature as calculated by the Random Forest over the full dataset.*



*Figure 18: Prediction errors of the Gradient Boosting Classifier over the full dataset.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy Score** | **Recall Score** | **Jaccard Score** | **F1 Score** | **Precision Score** |
| 0.608 | 0.583 | 0.43 | 0.601 | 0.62 |

*Table 6: evaluation metrics and results of the Gradient Boosting Classifier model*



*Figure 19: Importances of each feature as calculated by the Gradient Boosting Classifier over the full dataset.*

1. Github repository: <https://github.com/abu-sbeit/adv-pm> [↑](#footnote-ref-1)
2. The data for waiting time is different in the figure with respect to the table and text because, for visualization purposes, the graph in the figure is based on filtered data and has the sole purpose of graphically showing that the difference was clearly visible to the naked eye. [↑](#footnote-ref-2)
3. BPI Challenge 2017, <https://canvas.tue.nl/courses/25379/files/folder/data> [↑](#footnote-ref-3)
4. Filtered and preprocessed BPI Challenge 2017, <https://github.com/abu-sbeit/adv-pm/blob/main/data/bpi_challenge_offer.csv> [↑](#footnote-ref-4)