Phase 4: Predicting Offer Acceptance in BPI Challenge 2017

Nick van Daelen (2053691) Nikos Perdikogiannis (2030624) Ralph Kaijim (1299131)

March 11, 2024

1 Introduction

The dataset contains event logs from a loan application process. A customer submits an online application for a loan. The application is then checked for completeness and correctness. Afterwards, an employee assesses the application and decides whether to approve, decline or cancel the application. If it is approved, an offer is sent to the customer, which the customer can accept, reject or cancel the application. Multiple offers per application can be sent. In our understanding, the part of the process that is about sending offers is of primary importance since it is when the terms of the loan are finalized. In an ideal scenario, each accepted loan process should involve optimal terms for the institution and be dealt with in the minimal time possible.

In practice, this is often hard to achieve due to the negotiation that will inherently take place between the employee and the customer. The customer wants to maximize their personal benefit, while the employee aims to maximize the benefit for their institution. To assist the employees in this task, we propose a model that will use characteristics from the application and the offer to be sent, to determine the outcome of the offer without sending it. This way, the employees will have a way of evaluating an offer for a respective application, or even compare different offers in a quantitative manner, before sending them to the customer. This provides the employees with an extra tool in their arsenal, which should make the process of finalizing the terms of the loan faster and/or with more favorable terms for the institution.

In the following sections we will further explain the relevance of such a model for the institution, diving into the specifics of how it can be used to benefit the institution. Next, we will talk about the feasibility of such a task, exploring how we can use the available data in our favor, and look into how such a model can be used in practice. We will briefly explain which machine learning models we trained for this task, and how they compare according to quantitative metrics. Finally, we will explain how we propose such a model to be used optimally in accordance with the institutions priorities.

2 Analysis

2.1 Relevance

For companies offering loans to its customers, there are two major considerations that must be thought through before making an offer. The first consideration is obviously whether or not the company is willing to agree to the terms of there offer. Assuming the company only makes offers that they would agree to, the second component that must be taken into account is whether or not the other party will find the terms agreeable and accept the offer. This is a crucial consideration for two main reasons.

- First, to avoid wasting time and resources making offers that are likely to be rejected. Inefficient use of resources can be costly for a company, and by understanding the likelihood of offer acceptance, the company can focus its efforts on potential customers who are more likely to accept, optimizing its outreach strategies. By streamlining the targeting process, the company can allocate its resources more effectively, ensuring that time and effort are directed towards individuals with a higher probability of accepting loan offers.
- Second, to identify offers that are too generous to the customer, and therefore can be adjusted to make it more favorable to the company with little risk of rejection. Recognizing instances where the terms may be overly favorable allows the company to refine its loan offerings, balancing

competitiveness with profitability and mitigating the potential for financial losses. By fine-tuning the terms of loan offers, the company can strike a balance between attractive customer propositions and financial prudence, ensuring a sustainable and profitable lending model.

Our prediction model focuses on this consideration, and aims to provide the company with additional information regarding the likelihood of the customer accepting a given offer. With this additional information, the company is better equipped to make offers that are more likely to be accepted, but also just as importantly, avoid making offers that are much more favorable to the customer than necessary.

Generally, for prediction models with a binary posterior feature (in this case, offer accepted or offer rejected), the focus is on the classification of a case into one of these two groups. While this classification is useful in it's own right, what is perhaps more relevant is the predicted probability of acceptance, especially when it comes to to optimizing loan terms for competitiveness and profitability. For instance, if the model predicts than one offer has a sixty-five percent chance of being accepted, and another has a ninety-five percent chance, both will be classified as a 1 (offer accepted) even though only the second offer represents a potential opportunity to offer terms more favorable to the company. Additionally, observing the changes of these probabilities that result from changes to the terms helps the company better understand what, in particular, customers value more or value less when it comes to the terms of a loan. For these reasons, in addition to providing the predicted outcome, we will also ensure that the predicted probability of acceptance is also made available for the company to use.

Lastly, its important to understand the strengths and limitations of a model and it's prediction task. A prediction model such as this is only valuable if used appropriately. It may be tempting to see predictions made by a computer and want to automate the entire process of making offers. However, this would be unwise without evaluating the feasibility of the prediction task and analysing the empirical results of testing the model, to determine to what extent is it appropriate to make decisions based solely on the numbers, and when it is less appropriate and domain expertise should be highly valued. We begin by discussing the feasibility of the prediction task below.

2.2 Feasibility

On the feasibility side, predicting the outcome of an offer is by all means a hard task. It depends on a plethora of factors, not all of which are measurable by the dataset or even identifiable by a human employee. For instance, a customer might be overly optimistic they can get a better offer and reject an objectively good one. They may consider the interest rate as too high judging based on a loan they took a couple of years ago when the interest rates were much lower. Some customer who gambles a lot might reject an offer just because they think they can get a better one. In general, the customers will not use purely rational criteria when they judge an offer, and this makes the prediction task pretty hard for a computer alone. However, an employee, through the interactions with the customer, might be able to leverage his/her experience dealing with clients and use the outcome of the prediction model in a way that increases the rate of acceptance of his/her offers.

As we already mentioned, our goal is to predict the outcome of an offer. In particular, the model will predict the probability of an offer being accepted based on some predetermined features of the offer. Despite the difficulty of the task, we can identify some features that will most definitely affect the decision of the customers, if they were to judge by purely rational criteria. Those features are mainly the offer attributes combined with some information about the customer's needs, and are listed below.

- *Offered Amount* and *Requested Amount*
  We deem these two features to be of primary importance to the outcome of the offer. A low *Offered Amount* when compared to the *Requested Amount* in the customers application will most definitely result in a rejected offer, because it will not satisfy the customer's needs. Assuming the *Requested Amount* is the actual amount needed for the customer, the customer cannot accept such an offer.

- *Monthly Cost* and *Number of Terms*
  These two features combined determine the cost of the loan to the customer. The total cost for an individual might be deemed too high when comparing it with the benefits of getting the loan.

In another case, the *Montly Cost* alone might be too high to cover, depending on one's wages. In such case the employee, after discussions with the customer would decrease the *Monthly Cost* and increase the *Number of Terms* in a later offer.

- *Loan Goal*
  The *Loan Goal* by itself does not have anything to do with the outcome of the offer, but it might affect the aforementioned features by indicating the necessity of the loan. For example, a customer that wants to buy a new motorbike might be able to tolerate a lower *Offered Amount*, and go with a more inexpensive bike. On the other hand, a customer that needs the loan for an existing loan takeover might be willing to deal with a higher *Number of Terms* in order to get the exact amount he/she requested.

- *First Withdrawal Amount*
  In some cases the amount the customer will first be able to withdraw can be of primary importance. This will likely depend on the *Loan Goal*, in cases in which the customer needs money quickly.

- *Application Type*
  The *Application Type* determines whether the customer is a new customer or an existing one. This feature encodes information about the customer, and/or their potential experience with credit institutions.

To further enhance the accuracy of our model, we added information about the events that take place before the offer is sent. This should benefit the model since it encodes the history of the case up to the moment of prediction, which is all the customer knows about the application process. Therefore, the previous events might affect the probability of an offer being accepted.

2.3 Usability of the prediction model

The prediction models we implemented use the features introduced in section 2.2 as inputs and calculate the probability of the offer being accepted for output. Such a model will be used by the employees as a way to evaluate the offer before sending it.

The input features our model uses can be split in two categories: offer attributes and application attributes. The application attributes are the *Application Type*, the *Loan Goal*, the *Requested Amount* and the sequence of previous events. The offer attributes are the *Offered Amount*, the *Monthly Cost*, the *Number of Terms*, and the *First Withdrawal Amount.* These are features that change on every offer, in contrast with the application attributes. The employee would load the application attributes to the model, and then play around the offer attributes by testing values and taking note of the probability of acceptance predicted by our model to decide which offer to send.

For instance, consider an employee dealing with a new credit application for a loan to buy a car, requesting $\$20,000$. The employee would input these application attributes in the model, and then start testing values for the offer attributes. The probability of acceptance can be used as an evaluation of the offer to be sent. The employee will use such evaluation along with other information he/she has obtained through the customer to determine which offer to send. The better the terms are for the customer, the worse they are for the credit institution, therefore the employee needs to find a balance that will satisfy the customer, and still being profitable for the institution.

2.4 Quantitative Evaluation and Model Selection

To find a model that works well on the data, we train several models and use quantitative metrics to compare them. The baseline model is a random guess whether a loan will be approved the same way a coin toss does. So, this will be around a 50/50 outcome between the model predicting that the loan would be granted or not. Then, we train several machine learning models and compare them on a set of quantitative metrics.

To evaluate the models, we divide the data in five parts, train the model on four of the five parts, and then use the part not trained on to evaluate the model performance. We do this process five times, with every iteration a different part of the data, and evaluate the model. This is called 5-fold cross-validation and ensures that the model evaluation is robust, since the overall performance of the model is determined by the mean of the performance metric over the different iterations.

We use two metrics for the evaluation, *accuracy* and *F1-score*. The *accuracy* shows the percentage of the cases that were predicted correctly. The data is almost balanced between offers that were accepted and offers that were not accepted, so accuracy is a fitting evaluation metric for the loan application use-case. The *F1-score* is a more comprehensive evaluation metric that is widely used in prediction tasks. It combines information about how many cases that were predicted as accepted were actually accepted, with information about how many of the cases that were actually accepted were predicted as such. Table 1 shows the accuracy and F1-score of the different machine learning models we have tried over the course of this project.

Model	**Mean accuracy**	**Mean F1-score**
Baseline model	0.4987	0.5278
Logistic regression - SGD	0.5813	0.6593
Random Forest	0.5960	0.6555
SVM	0.5902	0.6925
Neural Network	0.6357	0.7081

Table 1: Model performance comparison

As you can see is the baseline accuracy around the expected 0.5 (50%) and are all the models a significant increase in performance. The best performing model, according to both these metrics, is the Neural Network model. The mean accuracy of 0.63 (63%) is 4 percentage points higher than the second best model in terms of accuracy, the Random Forest. Also, it is the best according to the mean F1-score, being slightly better than the SVM.

2.5 Qualitative Evaluation and Threshold adjustments

When dealing with classification models, an important decision to be made is where to set the threshold. The threshold is a value between 0 and 1, where cases with a predicted probability above the threshold are classified as 1s (Offer Accepted) and cases below the threshold are classified as 0s. Using 0.5 as a threshold is a common approach, but not the only possibility. For instance, if false positives are a bigger problem than false negatives, we'd want to err on the side of caution by setting a higher threshold, maybe 0.75, so that the predicted positives are more likely to be true positives as well. This depends on the specific context of a prediction model, and we will let domain experts at the bank weigh their priorities and make this decision. If we create a confusion matrix for various thresholds like we do in figure 4, you'll notice that when the threshold is set low, almost all of the offers that were accepted are correctly identified as such, resulting in a high value for true positives (bottom right). However, with the bar set so low to achieve this true positive rate, the trade off is that many offers that were rejected are also predicted to be accepted, leading to another large value for false positives (Top Right). We observe the opposite effect with an especially high threshold. With a threshold of 0.9 we correctly identify almost all of the rejected offers (True Negatives), but also incorrectly identify most of the accepted offers.

Figure 1 conveys similar information in a different way. Here we see that as the threshold increases, so does the proportion of cases classified as negatives (green line). As a result, precision (Red line) gets closer to 100 percent since only the cases that the model is very confident are positives are classified as such. The trade off, however, is that NPV (Orange line) decreases with threshold since almost all cases, including accepted offers, are classified as negatives leading to many false negatives. The same happens in reverse with especially small thresholds. It is at the discretion of the process owner how they want to set the thresholds and how they want to use the resulting predictions. What is important is an understanding of the effect of adjusting the threshold and correctly interpreting the classifications in that context. There are many possibilities, but we have a few recommendations of our own.

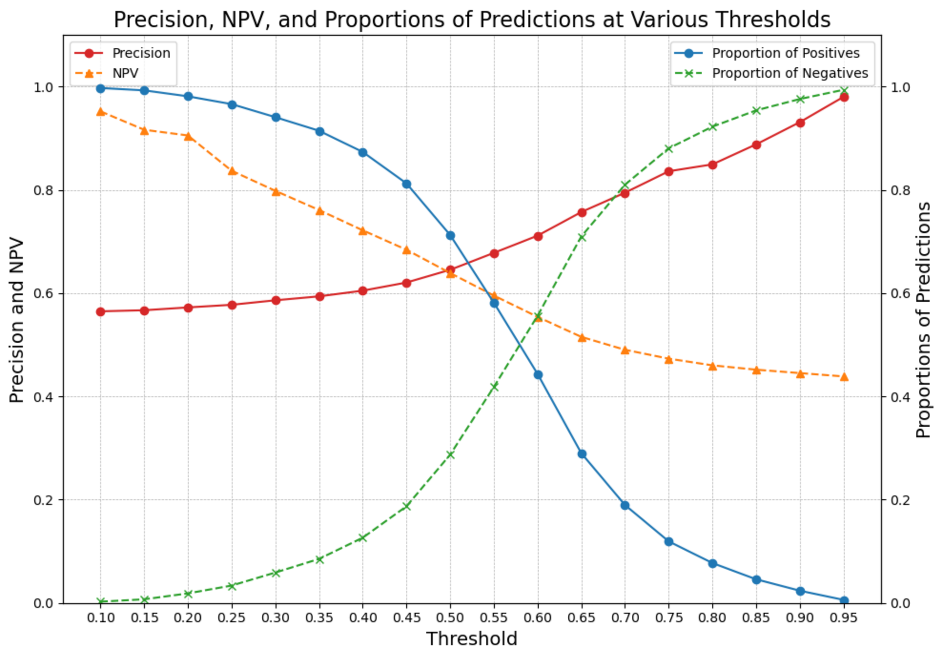


Figure 1: **Resulting Precision, NPV, proportion of Positives, Proportion of Negatives for various Classification Thresholds.**
Precision = True Positives/(True Positives + False Positives)
NPV = True Negatives/(True Negatives + False Negatives)
Proportion of Positives = Proportion of cases classified as 'Accepted'
Proportion of Positives = Proportion of Cases Classified as 'Rejected'

The highest classification accuracy comes when the threshold is set at around 0.5. This is no surprise as the data is relatively balanced. So, if we're purely interested in correctly predicting the highest proportion of cases, 0.5 is the way to go. However, a model that can predict the correct outcome 63% of the time is not useful for a financial institution. This accuracy is far to low to be considered a viable replacement for human decision making. However, that's not to say that the model can never be used to accurately predict outcomes. Therefore, we found a different way of extracting value out of the model by using the model prediction in certain specific cases. The model returns a predicted probability of the loan application being approved, so if we only predict the cases with a very low and high predicted probability we can create a use-case for our model for the process owner. We call the cases with these probabilities high confidence case. For the models in Table 1, we used all the cases where if a case had a probability of 0.5 or above we predicted the loan to be approved. We propose the following: if a case has a probability of lower then 0.15 we predict the loan is not going to be approved, when the probability is above 0.8 we predict the loan to be approved and all the values in between we say that the model is inconclusive. If these thresholds move away from zero and one, we decrease the number of times the model is inconclusive but also decrease the accuracy of the model. This trade-off will be mostly decided by the strategy of the process owner. The current thresholds result in a 0.855 accuracy on the high confidence cases and this includes 8.4% of all the prediction moments.

Reliably being able to classify almost ten percent of the cases in the test set is better than nothing, but for the rest of the cases there are other ways to extract value that goes beyond classifying the offer into one of two outcomes. As previously mentioned, the predicted probability of loan offer acceptance is a crucial metric. This probability serves as a valuable signal to lenders, indicating the need for adjustments in loan terms. When the predicted probability approaches 1, it suggests that the offer may be overly generous, prompting a reevaluation for potential adjustments that align with the company's interests. Conversely, probabilities close to 0 signal an opportunity to make the offer more attractive to the customer, enhancing the model's value by guiding lenders towards optimal and tailored loan terms.

Furthermore, by analyzing how the model's probabilities change following modifications to the offer terms, the process owner gains new and unique insights into how these adjustments impact customers' perceptions of the offer. This approach provides a more nuanced understanding of the customer base, discerning their preferences and priorities. To reiterate, while the classification accuracy of the model is not nearly high enough for us to recommend using it to replace the human element of creating offers, it can definitely still be used to inform the decisions of the people who are making the offers.

3 Conclusion and discussion

3.1 Summary

Overall, predicting the outcome of an offer is a hard task but not an infeasible one. As we explained in the previous sections, our final model can be used to give the employees valuable insight into the attributes of an offer. Using the model as an automated way to evaluate offers is not advised, due to the low 63% accuracy with 0.5 threshold. The task at hand is fairly complex, and the employees should leverage their knowledge and experience to fine tune the attributes of the offer. Our model will be used to predict the probability of an offer being accepted so that the employees find an agreement with the customer while sending the minimum number of offers needed.

3.2 Discussion

Out of a handful of different approaches we looked into, modeling this prediction problem with a neural network gave us the most accurate prediction. The good news is that this model is able to predict with more accuracy than the naive predictor. The bad news is that it is not making predictions with enough accuracy that the classification of outcomes should be blindly trusted. There are a number of potential explanations for the difficulty of making these predictions with high accuracy, many of which were detailed in earlier sections including the diversity of past experiences of applicants, the unpredictability of a persons risk preferences, and a persons eagerness to negotiate an offer to maximize their own value. Thus, this model should not be expected to make decisions by itself. Instead, the model, and the predictions it makes should only be used to inform the decisions of those whose job it is to create offers to send to applicants. This is a use where the model can be quite effective, especially when considering not just whether the model predicted an offer would be classified, but also looking at the probability the model calculates to make that classification.

Another benefit of the type of model we chose is that it is not a pure classification model, but instead a regression model that classifies based on predicted probabilities. This is useful because the lender can extract much more information from these predicted probabilities that it could from just a 1 or a 0 representing the predicted classification like a pure classification model would. Potential uses for this information are further summarized in the implications section of this conclusion.

Lastly, we want to touch on a fairly common situation that isn't covered very well by this model. The model does not differentiate between whether it was the initial offer presented to an applicant or if the individual had previously received an offer, subsequently declining it. In this situation, the structure of the first offer, and the applicant's response to it, tells the lender a great deal about the priorities and preferences of that specific individual. With this extra information, the model's output becomes less relevant and should be used sparingly in these situations.

3.3 Implications

As previously discussed, a lender would be interested in knowing the likelihood of an offer being accepted for two major reasons. First is to avoid wasting time and resources putting together an offer that will be rejected. A rejected offer results in no new business for the company and a company needs to write new loans regularly in order to grow. The second reason they'd like to predict the outcome of an offer is to avoid being to generous with the terms of the agreement. If the lender were to know that the applicant would still accept an offer, even if the terms were less favorable, the lender would regret making the offer more favorable than it needs to be, as then it becomes less valuable for the company.

If offer acceptance were able to be predicted with 100 percent accuracy, the company would see massive improvements in both of these areas. However, by the nature of this prediction task, a 100 percent accuracy, or even anywhere close to that, is not feasible for reasons discussed before. However, the low accuracy does not mean that the information we get from the model is useless. Because the model gives us the probability of acceptance in addition to the predicted classification, the company is able to use that information, in conjunction with standard techniques, to make better offers and see improvements in both of the two, aforementioned areas. This information can be used to achieve this in a couple different ways.

One way we discussed that this model could be used is to only trust its classification if the probability is exceptionally high or low. The specific ranges we looked into were predicted probabilities below 0.1 or above 0.85, where any offer where the model assigns a value below 0.1 would be classified as a rejection and anything above 0.85 an accepted offer. With this approach we are able to have confidence in the classifications made by the model (with an accuracy of almost 90 percent), but it has the obvious drawback that many of the cases (more than 90 percent) will not be classified as the model assigns them probabilities somewhere between 0.1 and 0.85.

Being able to reliably classify nearly ten percent of the cases in the test set is a step in the right direction. However, for the remaining cases, there are alternative avenues for extracting value beyond a binary classification. As previously noted, the predicted probability of loan offer acceptance emerges as a vital metric. This probability functions as a valuable signal to lenders, indicating the necessity for adjustments in loan terms. When the predicted probability approaches 1, it suggests potential over-generosity in the offer, prompting a reevaluation for adjustments that align with the company's interests. Conversely, probabilities nearing 0 present an opportunity to enhance the offer's attractiveness to the customer, serving as a cue for for the lender to consider improving the attractiveness of the offer, if the potential business presented by the applicant is worth it.

It's crucial to emphasize that these predicted probabilities should be utilized to inform the judgment of the lender, not as a sole basis for decisions. Human expertise, contextual insights, and business considerations must complement the model's output to ensure a well-rounded and informed decision-making process. Furthermore, the model could be used as a tool to improve a lender's judgement and help them learn more about the market they operate in. Examining the shifts in the model's probabilities when offer terms are adjusted yields the process owner fresh and distinctive insights into how these changes influence customers' perceptions of the offer. This method facilitates a more nuanced comprehension of the customer base, enabling the identification of their preferences and priorities. As a quick example, consider a hypothetical scenario where the lender assumes that customers prioritize lower monthly payments over a more extended repayment period (assuming the present value remains constant). Conversely, the lender might prefer shorter terms with higher monthly payments. Using the predictive model, the lender can systematically adjust these terms and observe how the predicted probabilities of acceptance change. If the model indicates a significant increase in acceptance rates with the lender's preferred terms, it validates their assumptions and highlights the efficacy of the chosen concessions. This data-driven approach allows the lender to align offerings more closely with customer preferences, enhancing the likelihood of acceptance and overall customer satisfaction.

3.4 Final Recommendation

To reiterate, our recommendation is not to use this model to replace human decision making in creating offers entirely. We recommend using the predictions our model makes to augment the information available to those who are deciding the terms of the offers they're sending to applicants. By utilizing a combination of the strategies outlined above, the company is giving it's employees an additional tool to help them make better decisions and gain a better understanding of the preferences and priorities of customers. This is done without having to make any radical alterations to their current processes, and avoids the risk that comes with relying too heavily on statistical models. Instead, human expertise, contextual insights, and business considerations remain the cornerstone, guiding the construction of offers with a balanced and more informed approach.

Appendices

A Prediction setup

We decided to start with three models besides the baseline model. These were a Logistic Regression classifier, a Support Vector Machine classifier and a Random Forest classifier. The main reason we started with logistic regression is that it is designed for binary classification problems, making it a logical first choice. As an added benefit, it can give us clear insight into how features affect the probability of the outcome of a loan application. Ideally, we would like the logistic regression to be the best fitting model due to its interpretability. Interpretability is also why we decided to dummy encode the categorical variables: to see if certain values of categorical features had a bigger impact.

The next models we decided to explore were a Random Forest classifier and a Support Vector Machine (SVM). Random Forest is a model that often performs rather well with the default setting, which makes it a good model for initial exploration. Both Random Forest and SVM work well with complex data, so we expected them to perform better than the Logistic Regression classifier. As you can see in table 1, they indeed performed better, but we didn't see a significant improvement. Therefore, we decided to nuild a Deep Neural Network to encapsulate even more complexity in our model, which ended up being our final model.

For the evaluation, we chose k-fold cross-validation with F1-score and accuracy to evaluate our models. We use a confusion matrix to get insight into the kind of mistakes the model makes. We use both F1-score and accuracy because there is a slight class imbalance of 54.7% accepted loan applications to 45.3% denied loan applications. This imbalance, however, is slight, so the accuracy is still very relevant. To evaluate our models, we used k-fold cross-validation to give us a more robust model evaluation.

B Relization of process models

B.1 Preprocessing

For the preprocessing phase, we start by cleaning the event log. For that we use the *pandas* library in Python. We start by filtering out all the events that have a "lifecycle:transition" value other than "COMPLETE". Next, we convert the "time" column to a timestamp so that it can be processed. We create a new column named "EventCount", which includes the cummulative number of events up to the point of prediction.

We define the moment of prediction as the moment in which the first offer is sent, ie. the first "O_Create_Offer" event. As prior features we obtain the following offer attributes along with some application attributes:

- Monthly Cost
- First Withdrawal Amount
- Number of Terms
- Offered Amount
- Requested Amount
- Loan Goal
- Application Type

We use the StandardScaler from the sklearn library to standardize the first 5 columns which contain numerical values. This ensures our data will be approximately normally distributed, which many machine learning algorithms require to work well. For the "Loan Goal" and the "Application Type" we use dummy encoding since they are categorical variables. As for the prefix encoding, we use Label Encoding on the event names to get a list of 23 possible events. Then, we obtain all the prefixes, defined as the sequence of events before the moment of prediction, that is before the first "O_Create_Offer" event. Dummy encoding is also used to encode the prefix for every case.

B.2 Models

For all the models discussed in this section, we use K-Fold cross validation with k=5. This ensures robustness in our evaluation methods. The evaluation metrics are accuracy and F1-score, which are presented in table 2.

Naive Predictor The Naive Predictor is our baseline model. It basically predicts either Accepted (1) or Rejected (0) in a random fashion. As the theoretical probability suggests, this approach has around 50% accuracy rate.

Logistic Regression For the Logistic Regression model, we used the SGDClassifier from the sklearn library. This method uses Stochastic Gradient Descent (SGD) learning, and we used the default parameters for training.

Support Vector Machine For the SVM we use the LinearSVC from sklean library, with parameters $c = 0.0001$ and max_iter = 10000.

Random Forest For the Random Forest, we use the RandomForestClassifier from sklearn, with parameters n_estimators = 45 and max_depth = 35.

Deep Neural Network For the Neural Network, we use the Keras library. The architecture of our Neural Network classifier is shown in figure 2. For the training, we use the Adam optimizer with a learning_rate = 0.001. We use validation based early stopping with patience = 15 to stop the training if there is no improvements. For the loss function, we use binary crossentropy with accuracy as the metric of choice. A batch_size = 32 is selected, and we train up to a maximum of 200 epochs.

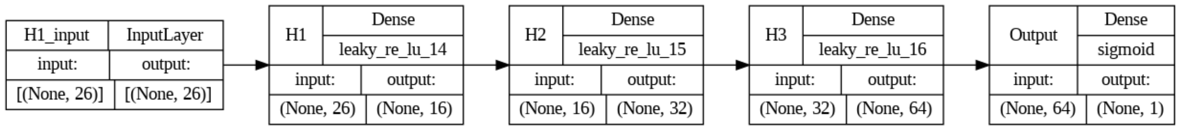


Figure 2: The architecture of our Neural Network model. It includes 3 hidden layers with 16, 32, and 64 neurons, with the Leaky ReLU activation function and the Sigmoid for the output layer.

C Evidence

For the following models, we filter out all the events with "lifecycle:transition" being anything other than complete, and trim the traces on the first instance of O_Create_Offer, keeping the prefixes and suffixes of the cases. The process models were constructed using the inductive visual miner in ProM Lite 1.4.

C.1 Process model before first Offer

The process model shown in figure 3 shows the part of the process before the first offer is sent. In most cases, the O_Create_Offer comes right after the application is accepted, but in some cases this part is more complex. We use prefix encoding to encode the events that come before O_Create_Offer, and we dummy encode the prefix of the cases to train our model.

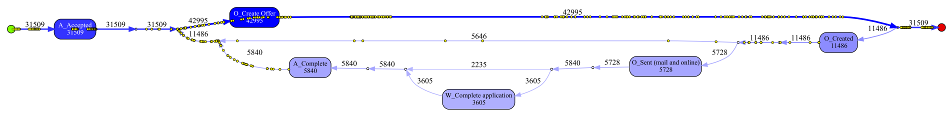
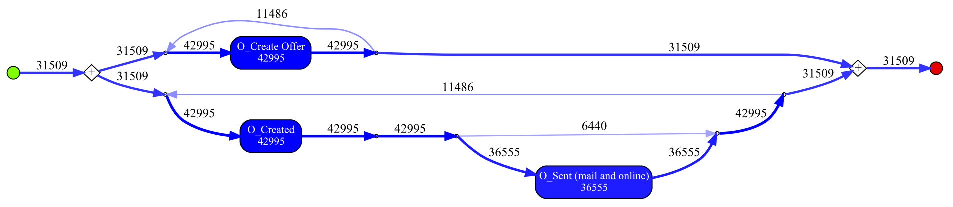


Figure 3: The process before the first O_Create_Offer event.

C.2 Process model after first Offer

The process model shown in figure C.2 shows the part of the process after the first offer is sent. This high level process model shows the main events happening. We can see that multiple offers can be sent afterwards, or the case can end with a single offer being sent.



D Analysis of prediction model quality

D.1 Quantitative analysis

Below you can find table 1 with the numeric values of all the metrics discussed in the sections above. We trained and tested these models and used the mean metric values for the comparision. The selected neural network model outperforms the rest in both mean accuracy and mean F1-score. K-Fold cross validation is used to ensure the robustness of our models. The metric values across all k are shown for reference.

	Metric	K=1	K=2	K=3	K=4	K=5	**Mean Metric**
Naive Predictor	**Accuracy**	0.4982	0.4941	0.5080	0.4994	0.4937	**0.4987**
	F1-score	0.5284	0.526	0.5352	0.5267	0.5228	**0.5278**
Logistic Regression	**Accuracy**	0.5910	0.5922	0.5415	0.5922	0.5896	**0.5813**
	F1-score	0.6920	0.7158	0.4576	0.7135	0.7176	**0.6593**
SVM	**Accuracy**	0.5931	0.586	0.582	0.5955	0.5945	**0.5902**
	F1-score	0.6942	0.6868	0.6867	0.6974	0.6972	**0.6925**
Random Forest	**Accuracy**	0.6008	0.5944	0.5924	0.5959	0.5967	**0.5960**
	F1-score	0.6594	0.6549	0.6518	0.6554	0.6560	**0.6555**
Neural Network	**Accuracy**	0.6324	0.6369	0.6327	0.6330	0.6436	**0.6357**
	F1-score	0.7071	0.7203	0.6940	0.6982	0.7208	**0.7081**

Table 2: The values of each metric across all 5 folds along with their mean for all classifiers.

D.2 Qualitative analysis

In figure 4 we see the confusion matrices with thresholds ranging from 0.1 to 0.95. As discussed in earlier sections, the threshold can be altered to affect the predictions of our model, depending on whether we want more False Positives or True Negatives. If we choose a threshold of < 0.15 for negatives and $0.80 <$ for positives, we get 0.855 accuracy as mentioned in section 2.5 above

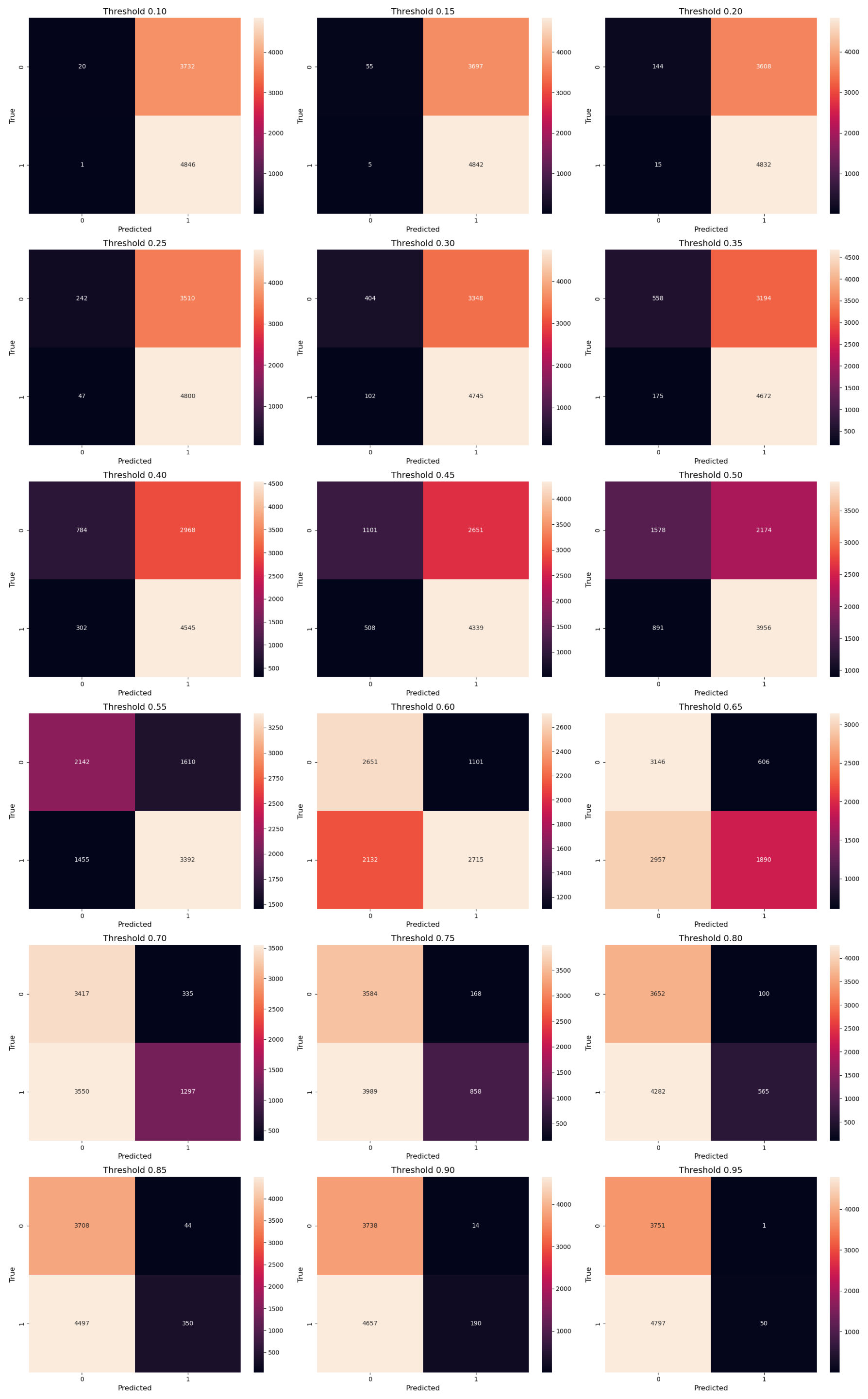


Figure 4: **Confusion Matrices with threshold set at different levels**