MMF2030H

Machine Learning for Finance

Final Project

Home Credit Default Risk

by:

Manyi Luo

Yitian Zhang

Ziyi Wang

Spencer Sillaste

January 2022

Contents

1 Introduction

Many people struggle to get loans due to insufficient or non-existent credit histories. This population is often taken advantage of by untrustworthy lenders. We strive to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this under-served population has a positive loan experience, we make use of a variety of alternative data to predict their clients' repayment abilities. This will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful. In this project, we implemented random forest, gradient boosted trees as well as logistic regression to characterize the client's payment ability based on various data sets containing client's telco and transactional information. We first did some data mining, merging work, followed by model construction. Afterwards, we compared the pros and cons as well as performance of these models, and identify the most suitable model for our data set.

2 Preliminary Data Processing

Before implementing the models, we have to first investigate the given 7 raw data sets. After examining each data set, we merged `application_train.csv`, `bureau.csv`, `bureau_balance.csv`, `POS_CASH_balance.csv`, `credit_card_balance.csv`, `installments_payments.csv` together with some data transformation. Note that `previous_application.csv` is not included in our final data set. The main reason is due to the computing power limitation of our computers, we want to limit the total number of features we put into our model to speed up processing. Also, we noticed that many features in `previous_application.csv` overlaps with features in `application_train.csv` but are historical information. Due to timeliness considerations, application information in the past is less timely to provide valuable information to our model compared with other data sets. Detailed code can be found on GitHub or in attached files.

2.1 Data Transformation and Merging

To merge the six data sets we discussed above, we went through each variable in each data set and did the following:

• bureau_balance.csv

For each SK_ID_BUREAU, we calculated the proportion of 0 of STATUS as it's an indication of 'good', which we believe may play a role in determine a client's payment ability. We dropped MONTHS_BALANCE as it's irrelevant to the client's behaviour. In this step, one new numerical were variables created.

Then, we merged bureau_balance.csv to bureau.csv w.r.t. the key SK_ID_BUREAU.

• bureau.csv

1) For each SK_ID_CURR, we calculated the proportion of Closed and Active of CREDIT_ACTIVE. In this step, two new numerical variables were created.
2) We removed CREDIT_CURRENCY as it's constant
3) We removed DAYS_CREDIT, CREDIT_DAY_OVERDUE, DAYS_CREDIT_ENDDATE, DAYS_ENDDATE_FACT and DAYS_CREDIT_UPDATE as these are irrelevant variables (time variable only relative to the application)
4) For each SK_ID_CURR, we calculated the proportion of Consumer Credit and Credit Card of CREDIT_TYPE. In this step, two new numerical variables were created.
5) We replaced NA with Active for variable CREDIT_ACTIVE, and calculated the proportion of Active for each SK_ID_CURR.
6) We replace NA with 0 for AMT_CREDIT_MAX_OVERDUE, and calculate average of AMT_CREDIT_MAX_OVERDUE for each SK_ID_CURR.
7) We NA with 0 for CNT_CREDIT_PROLONG, and calculated maximum of CNT_CREDIT_PROLONG

for each SK_ID_CURR.

8) We replaced NA with 0 for AMT_CREDIT_SUM, and calculated the sum of AMT_CREDIT_SUM for each SK_ID_CURR.

9) We replaced NA with 0 for AMT_CREDIT_SUM_DEBT, and calculated sum of for each SK_ID_CURR.

10) We replaced NA with 0 for AMT_CREDIT_SUM_LIMIT, and calculated sum of AMT_CREDIT_SUM_LIMIT for each SK_IF_CURR.

11) We calculated the proportion of Credit card and Consumer credit of CREDIT_TYPE for each SK_ID_CURR. In this step, two new numerical variables were created.

12) We removed AMT_ANNUITY as irrelevant.

13) We calculated average over all non-NA values for STATUS_0 for each SK_ID_CURR.

Then, we merged all new variables transformed from bureau.csv which also include the information from bureau_balance.csv to application_train.csv w.r.t. the key SK_ID_CURR.

Also, we picked some relevant features from installment_payment.csv, creit_card_balance.csv and POS_CASH_balance.csv to be merged to our final data set. Details of processing are as follows:

- installment_payment.csv

We picked AMT_INSTALMENT, AMT_PAYMENT and combine them into one variable indicating whether the client has paid out the installment on time. One new dummy variable is created following the criteria: if AMT_PAYMENT $\geq$ AMT_INSTALLMENT, then the dummy equals 1, other wise equals 0. Then we calculate the maximum of that dummy variable for each SK_ID_CURR.

- creit_card_balance.csv

We picked AMT_DRAWINGS_ATM_CURRENT, AMT_DRAWINGS_CURRENT, AMT_DRAWINGS_OTHER_CURRENT, AMT_DRAWINGS_POS_CURRENT, AMT_RECIVABLE, and AMT_TOTAL_RECEIVABLE to be included

in our final data set. We replaced all NA with 0, and calculated the average for each variable for each SK_ID_CURR.

• POS_CASH_balance.csv

1) We picked variable SK_DPD, replaced NA with 0, and made it into a dummy variable following the criteria: if the original value is 0 then kept it as 0, else replaced original value with 1. Then we took the maximum of the newly created dummy variable for each Sk_ID_CURR.
2) We replaced NA with 0 for CNT_INSTALMENT_FUTURE, and took the average of it for each SK_ID_CURR.

Then, we merged the new variables transformed from POS_CASH_balance.csv, creit_card_balance.csv and installment_payment.csv to application_train.csv w.r.t. the key SK_ID_CURR. Note that there are duplicated SK_ID_CURR in some data sets, under this circumstances, we calculated the average for each numerical variable and use the calculated average to present the corresponding feature value for each unique SK_ID_CURR. Note that after data transformation, there are no non-numerical variable (such as categorical variable or dummy variables) in POS_CASH_balance.csv, creit_card_balance.csv, installment_payment.csv, and Bureau_new.csv, thus, taking average to numerical variables in these data sets would be enough.

2.2 Quality assurance

First, we remove the irrelevant features and the features that we cannot explain from the dataset. We remove FLAG_MOBIL, FLAG_PHONE, FLAG_DOCUMENT_2, FLAG_DOCUMENT_3, FLAG_DOCUMENT_4, FLAG_DOCUMENT_5, FLAG_DOCUMENT_6, FLAG_DOCUMENT_7, FLAG_DOCUMENT_8, FLAG_DOCUMENT_9, FLAG_DOCUMENT_10, FLAG_DOCUMENT_11, FLAG_DOCUMENT_12, FLAG_DOCUMENT_13, FLAG_DOCUMENT_14, FLAG_DOCUMENT_15, FLAG_DOCUMENT_16, FLAG_DOCUMENT_17, FLAG_DOCUMENT_18, FLAG_DOCUMENT_19, FLAG_DOCUMENT_20,

FLAG_DOCUMENT_21.

Then, we aim to remove the variables that only have one value because the constant variable cannot improve the performance of the models. Since there are not any constant variables, we do not remove any features in this step.

In the end, we deleted variables related to ID, and we got a new data set with the target variable and all features we want. The data set has a size of 307511×118. The data set has 117 features, and the feature dictionary can be found in GitHub and attached files.

Then, we randomly select 70% data as the training set and use the rest 30% data as testing set. The training set and the testing set are disjoint, so we can avoid the bias which happens when people use the training set to test the performance of the model. In the training set, we note that there are much more data with `TARGET` $= 0$ than data with `TARGET` $= 1$, due to the imbalance of the data set, we would like to make the proportion of the two types of `TARGET` variable approximately equal. In this way, we may increase the prediction accuracy of our model and reduce the misclassification rate of identifying client with payment difficulties. There are 19,375 samples with `TARGET` $= 1$, so we random select 19,375 samples with `TARGET` $= 0$ from the training set and combine them with the 19,375 samples with `TARGET` $= 1$ into a new data set as our training set. In addition, validation set varies from different models, details about model validation will be discussed in later sections.

Since the missing values and the categorical variables may cause error when we fit the models, we do some additional data transformation before we fit the models. For the missing values, we replace `NA`'s with average for numerical variables, and replace `NA`'s with mode for categorical variables. Additionally, we make all categorical variables into several dummy variables using `get_dummies()` from `pandas`. For instance, variable `NAME_FAMILY_STATUS` has six different types, and we created five dummy variables to indicate which family status the does a client fall into. Note that we used five dummies to represent five of the six types

so as to avoid dummy variable trap. After these steps, the number of features increases to 239.

There is another technique to deal with missing values and categorical variables, which is weight of evidence (WOE). WOE may improve the data quality and the performance of the model. We will discuss WOE in later sections.

3 Model Construction

In this section, we constructed three models: Random Forest, Boosted Tree and Logistic Regression as candidate models for further comparison. Detailed code can be found on GitHub or in attached files.

3.1 Random Forest

Random forest (RF) can be seen as an extension of the Bagging algorithm. Based on Bagging ensemble with decision tree, RF further adds the selection of random attributes in the training process of decision tree. In RF, for each node of the base decision tree, a subset containing K attributes is randomly selected from the candidate attribute set of the node first, and then an optimal attribute is selected from the sub-set for partitioning. The "diversity" of random forest comes not only from the disturbance of samples, but also from the disturbance of attributes, which further enhances the generalization ability. RF can run efficiently on large data sets. Due to the sampling and attributes, the final trained model has strong generalization ability. Also, it has high tolerance for missing value of some features. In this case, we choose RF as one of our candidate models based on the characteristics of our data set.

3.1.1 Hyperparameter Tuning

When the model is of low complexity, machine learning is insufficient, there will be under-fitting issues, and generalization error will become larger. When the complexity gradually increases to the optimal model complexity, the generalization error reaches the lowest point (i.e., the highest accuracy). If the complexity still increases, the generalization error increases gradually from the minimum value, and there will be concerns of over-fitting. The goal, therefore, is to adjust the model complexity as close to the lowest generalization error as possible by constantly tuning hyperparameters.

The main function we implemented for random forest is `RandomForestClassifier()` from `sklearn.ensemble`. There are mainly five parameters for `RandomForestClassifier()` we could consider for hyperparamerter tuning:

- `n_estimators`: number of trees in the forest
- `max_features`: max number of features considered for splitting a node
- `max_depth`: max number of levels in each decision tree
- `min_samples_split`: min number of data points placed in a node before the node is split
- `min_samples_leaf`: min number of data points allowed in a leaf node

The following table explains the importance of each model and general direction of parameter tuning:

Parameter	Impact	Degree
`n_estimators`	Increases until the model is stable	$\star\star\star\star$
`max_depth`	Set to maximum depth by default Model gets more complex as it increases	$\star\star\star$
`min_samples_split`	Set to 1 by default Model gets simpler as it increases	$\star\star$
`min_samples_leaf`	Set to 2 by default Model gets simpler as it increases	$\star\star$
`max_features`	Equals $\sqrt{\text{Total Number of Features}}$ by default Model gets more complex as it increases	$\star$

More $\star$ means the parameter has a more significant impact on the model performance

There are some other parameters that may affect the model performance and efficiency, such as `min_weight_fraction_leaf`, which limits the minimum sum of all sample weights of leaf nodes; `max_leaf_nodes`, which can be set to prevent over-fitting by limiting the maximum number of leaf nodes; and `min_impurity_split`, which limits the growth of the decision tree. Due to our computers computing capability limitation and limited time for this project, we only consider the most important five hyperparameter that closely related to our model performance and efficiency as we discussed above.

We used `GridSearchCV()` from `sklearn.grid_search` to find the optimal parameters for random forest. Grid search is to search parameters, that is, within the specified parameter range, parameters are adjusted in turn according to the step length, and the adjusted parameter training learner is used to find the parameter with the highest accuracy on the verification set from all parameters. This is actually a process of training and comparison. `GridSearchCV()` can guarantee to find the parameter with the highest accuracy within the

specified parameter range, but this is also the drawback of grid search. It requires traversing all possible combinations of parameters, which is very time-consuming in the face of large data sets and multiple parameters. In this case, we could only identify the optimal parameters within a limited range based on our computer computing capability.

The following table summarizes the optimal parameters for our random forest model:

Parameter	Optimal Value
`n_estimators`	350
`max_depth`	12
`min_samples_split`	2
`min_samples_leaf`	50
`max_features`	17

3.1.2 Model Validation

Using the optimal parameters, we built our random forest model using `RandomForestClassifier()` on the training set we constructed. We sued k-fold cross-validation to validate our random forest model. In K-fold cross-validation, initial samples are divided into K pieces, part of which is reserved as test set of validation model, and the rest $K-1$ pieces are used for train set. Cross-validation is repeated K times, once per test. The advantage of this method lies in that randomly generated sub-samples are repeatedly used for training and verification at the same time, and the results are verified once each time. The 10-fold cross-validation is the most commonly used method.

We plotted receiver operating characteristic (ROC) curve for each fold. ROC curve is a performance measurement for classification problems at various threshold settings. ROC is

a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. A good model has AUC close to 1, which means it has a good measure of separability.

The following graph represents the ROC curve for the 10-fold cross-validation of our random forest model with optimized parameters.

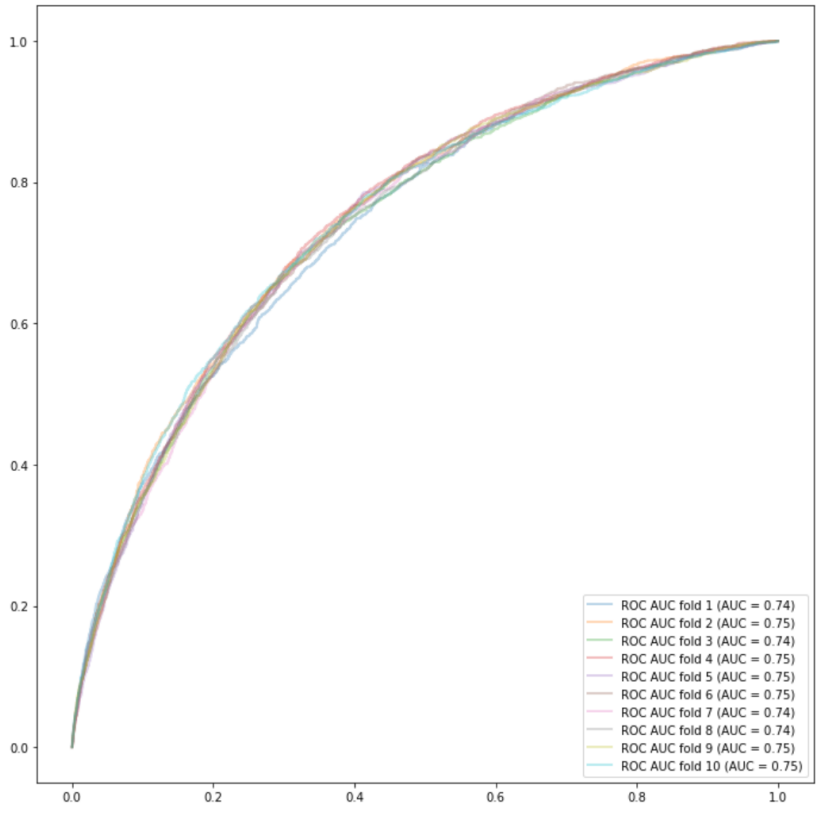

Figure 1: 10-fold Validation ROC Curve

As we can clearly see from the graph, the AUC calculated for each fold is areound 0.74 and 0.75, which averages to 0.746. This indicates a relatively good overall performance of our random forest with optimized parameters.

Additionally, we could use k-mean clustering to speed up the cross-validation process. The

following graph illustrates the 10-fold cross-validation ROC with k-mean clustering.

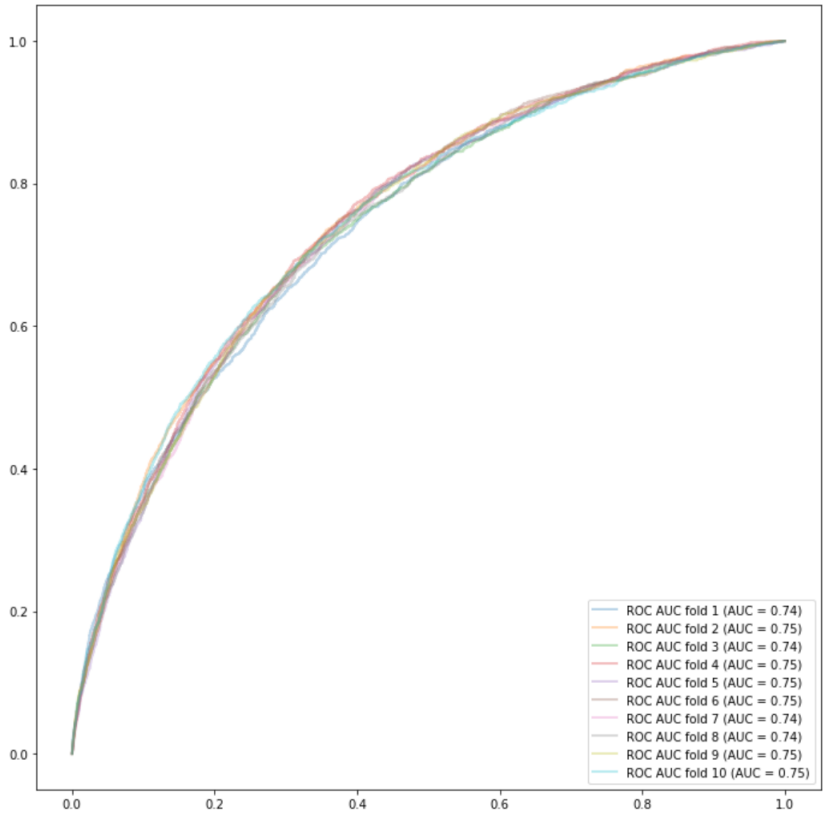

Figure 2: 10-fold Validation with K-mean clustering ROC Curve

Same as the previous 10-fold cross-validation ROC curve, the AUC calculated averages to 0.746.

3.1.3 Model Performance Evaluation

Having our model validated, we would like to use confusion matrix and some related indices to assess the performance of our random forest model. The reason to choose the confusion matrix and related indices calculated from it is that the confusion matrix provides more valuable information about the model performance and measure of separability. Calculating the confusion matrix can give us a better idea of the correctness of the classification model and the types of errors it makes. The confusion matrix defines and calculates the following:

- **True Positive**: we predicted that a client has payment difficulties and (s)he actually does.
- **True Negative**: we predicted that a client is of other cases and (s)he actually is.
- **False Positive**: we predicted that a client has payment difficulties but (s)he actually does not.
- **False Negative**: we predicted that a client is of other cases but (s)he actually has payment difficulties.

The confusion matrix is plotted as follows:

		Actual		
		Positive	Negative	Total
Predicted	Positive	57687	2350	60037
	Negative	27115	5100	32215
	Total	84802	7450	92252

Note that Negative refers to client with payment difficulties, and Positive refers to the client is of all other cases.

Based on the confusion matrix, we could calculate some statistics that help us evaluating the model performance:

- **Recall (Sensitivity)** $= \frac{TruePositive}{TruePositive+FalseNegative}$: from all the clients with payment difficulties, how many our model predicted correctly.
- **Precision** $= \frac{TruePositive}{TruePositive+FalsePositive}$: from all the classes predicted as clients with payment difficulties, how many are actually clients with payment difficulties.
- **F-Score** $= \frac{2\times Recall\times Precision}{Recall+Precision}$: it measures Recall and Precision at the same time by calculating the Harmonic Mean of the two.

- **Accuracy** $= \frac{TruePositive+TrueNegative}{TruePositive+TrueNegative+FalseNegative+FalseNegative}$: from all the classes (clients with payment difficulties and all other cases), how many of them have been predicted correctly by our model.
- **Total Misclassification Rate = 1 - Accuracy**: the percentage of total incorrect classifications made by the model.

The following table summarizes the above indices calculated based on the confusion matrix:

Statistics	Value
Recall	0.68
Precision	0.96
F-Score	0.80
Accuracy	0.68
Total Misclassification Rate	0.32

Note that all these indices measure our model predictability from different perspectives of correctness of classification and the type of errors it make. In our case, the calculated index values illustrating classification correctness (Recall, Precision, F-Score, Accuracy) are all close to 1, which represents a relatively good performance of our random forest model.

With our random forest model constructed and evaluated, we implemented several other models for comparison. We also compared our random forest model with the baseline random forest model with given parameters constructed in tutorials in later sections.

3.2 Gradient Boosted Trees

One of the machine learning models implemented to predict the risk of a client defaulting and therefore being unable to pay back a loan is the gradient boosted trees model. The gradient boosted trees model was chosen as it is a robust machine learning method that has been proven to work well and efficiently on a variety of binary classification problems with a large number of data points and features. As well, when compared with our other model, random forest, it allows for the comparison of the performance for different ensemble tree models as random forest uses bagging whereas gradient boosted trees uses boosting and so the pros and cons to each of these models can be examined. Gradient boosted trees construct decision trees sequentially, where the current iteration of the decision tree uses the information from the previous decision tree on how well it was able to make predictions to the data and uses that information to aid in constructing a decision tree with better predictive power. This is generally viewed more favourably than random forest models decision tree construction, as the use of previous decision tree information generally increases the predictive power of gradient boosted trees. Some of the potential downsides to the gradient boosted trees model is that it is very dependent on the hyperparameter values, and so not finding the optimal parameters may lead to a model with less predictive power than possible. There is also a larger variety of parameters that can be tuned. As well, the ensemble boosting algorithm leads to the model easily overfitting. In order to construct the most accurate gradient boosted trees model for the default risk dataset, hyperparameter tuning is performed and the features with low correlations to the target data are removed for increased performance and interpretability. To verify that the model is making accurate predictions, a repeated k-fold cross validation is performed on the training set to ensure generalizability of the model and the predictions on the testing set are examined using a variety of metrics, including a receiver operator characteristic curve, a confusion matrix and a variety of single number classification metrics.

3.2.1 Correlation Matrix

Before the gradient boosted trees model was built and tuned, a correlation matrix for the training data was constructed. The correlations of the features with the target data were examined and the features with a very small absolute correlation to the target were removed, specifically, less than 0.01. This value was chosen as it is less than the median correlation of ≈ 0.02 between all the features and the target, and so only the features that contribute the least to making predictions in the model are removed.

3.2.2 Hyperparameter Tuning

To find the optimal hyperparameters for the gradient boosted trees model, a sequential grid search cross-validation method is performed. Individual or a small collection of parameters are hyper-tuned using grid search cross validation and once they have been tuned, a new parameter or collection of parameters are tuned using the model with the partial set of tuned parameters. The `XGBClassifier` from the `XGBoost` package in Python is used for the gradient boosted trees model. The hyperparameters are tuned using training set data where the `RepeatedStratifiedKFold` method from the `sklearn` package has been applied so that the hyperparameters do not overfit to the training set. There were 10 K-Folds used, and they were reshuffled and repeated 3 times. The hyperparameters that will be tuned and their descriptions are given below,

- `n_estimators`: number of decision trees that are gradient boosted
- `max_depth`: max number of levels in each decision tree
- `learning_rate`: How much information should the next decision tree learn from the previous decision tree.
- `subsample`: The percentage of data that is sampled before the decision trees are constructed. This is done for every boosting iteration.

- `colsample_bytree`: The percentage of column data that is sampled before each decision tree is constructed.
- `colsample_bytree`: The percentage of column data that is sampled before each decision tree is constructed.
- `reg_alpha`: Strength of L1 regularization on the model weights.
- `reg_lambda`: Strength of L2 regularization on the model weights.

All the parameters are tuned on the training set with cross validation for a collection of different parameter values, and a box and whiskers plot of the accuracy is used to determine the best value from the collection. The number of trees is first tuned, and the optimal value is found to be, 180 as can be seen from the plot below.

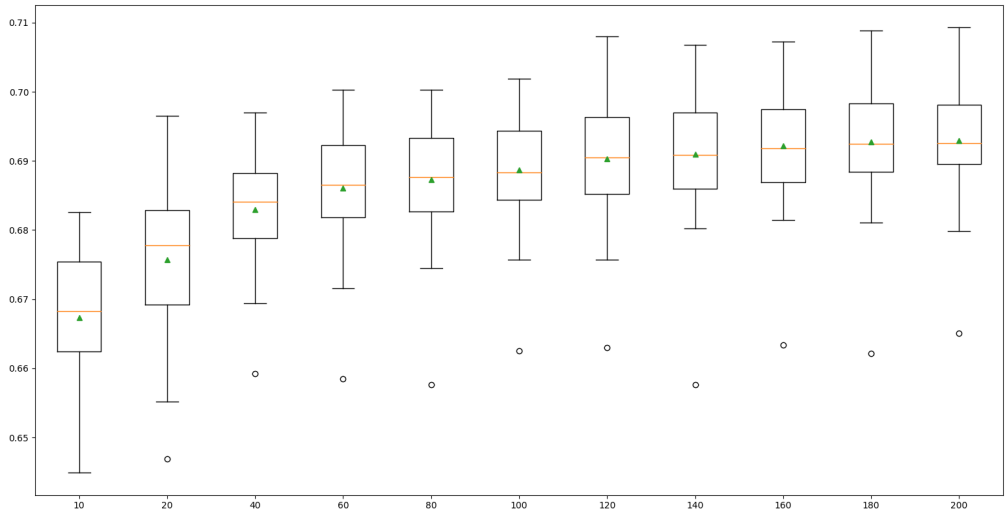

Figure 3: Box and whisker plot of the accuracy as a function of the number of trees in the gradient boosted trees model.

The box and whisker plot shows the median value as a green triangle and the mean value as an orange line. The bottom and top of the box represent the first and third quartile of the cross-validation distribution respectively, and the whiskers or lines coming out the box

represent the maximum and minimum values in the distribution excluding outliers. It can be seen in the box and whisker plot for the number of trees that the value with the highest accuracy is about 180 trees. This is chosen as it that has a very similar median accuracy to 160 and 200 trees, however, the maximum and minimum as well as the quartiles fall in a more narrow range than 200 trees, this indicates that the model is fitting well to most of the different cross-validation sets, and it has slightly higher accuracy than 160 trees.

With the best number of trees found, that number is implemented into the model and a similar process is performed for the next hyperparameter being tuned. This process of finding the optimal hyperparameter value using the box and whisker plot, implementing the optimized hyperparameter value into the model, and then using that model to find the optimal value for the next hyperparameter is repeated until all the hyperparameters have been tuned. The collection of box and whisker plot for the other hyperparameters can be found in the appendix. The summary of the optimal values for the hyperparmeters is given in the table below.

Parameter	Optimal Value
`n_estimators`	180
`max_depth`	3
`learning_rate`	0.1
`subsample`	0.9
`colsample_bytree`	0.3
`reg_alpha`	0.3
`reg_lambda`	1

The hyperparameters `n_estimators`, `max_depth` and `learning_rate` have the most significant impact on the performance of the model, whereas the other hyperparameters have a

much less significant impact on performance.

3.2.3 Model Validation

To verify that the gradient boosted trees model with the tuned hyperparameters is able to accurately make predictions about the default rate and that the model is an improvement over the model with the default hyperparameters, the ROC curve and corresponding AUC on the 10-fold partitioned total data set is compared for the default model and the tuned model. The ROC curve plots the true positive rate against the false positive rate and is useful for identifying how the model makes classification predictions. The AUC is the area under the ROC curve and is a single number metric for identifying the performance of the model, a perfect model has an AUC of 1. The ROC curve for each fold in the hyperparameter tuned gradient boosted trees model is given below, and the AUC for each fold is given in the legend.

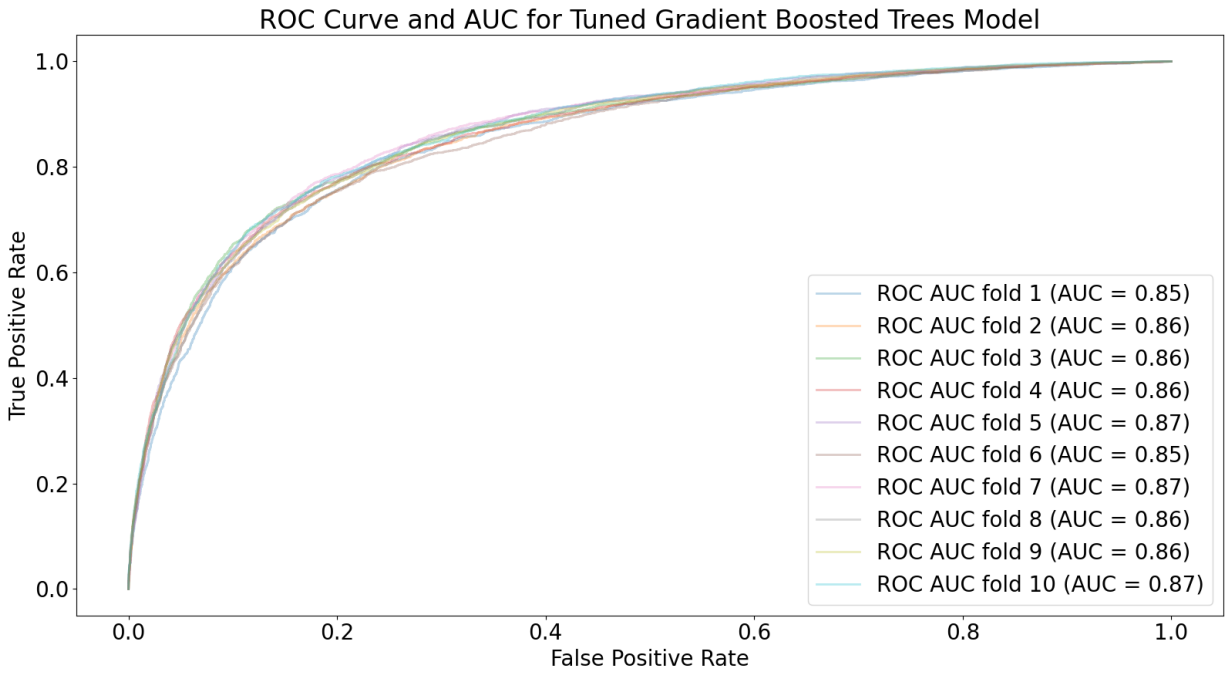

Figure 4: ROC curve and AUC for the hyperparameter tuned gradient boosted trees model on the 10-fold dataset.

It can be seen from the plot that for all the 10 folds, the ROC curve and the AUC are quite

similar, this indicates that the model is able to generalize well. The average AUC for all folds is 0.862 a very accurate value for the predictive power of the model. The shape of the ROC curve is consistent throughout the different false positive rates, this indicates that the model makes similar predictions on all data points, that is, there is not a specific subset of data points where the model performs particularly well or unwell. Moving on, the default ROC curve and AUC for the default hyperparameter gradient boosted trees model is given below.

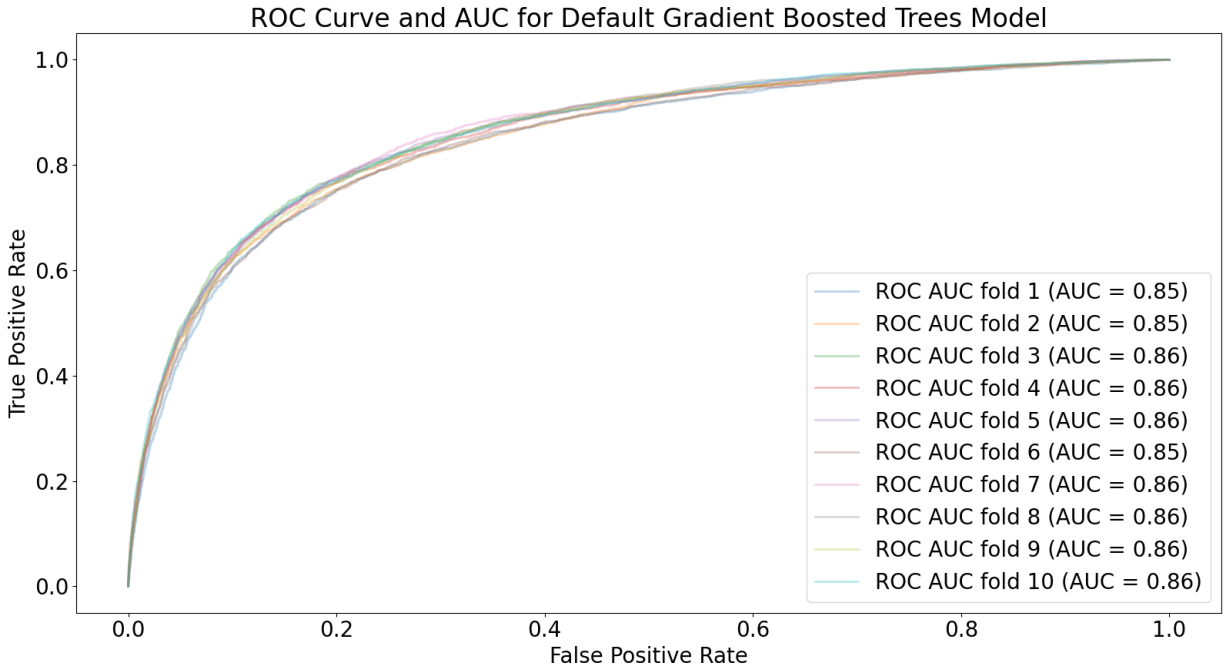

Figure 5: ROC curve and AUC for the default hyperparameter gradient boosted trees model on the 10-fold dataset.

Similar to the tuned model, the default model has an ROC curve shape and AUC values that are similar for all 10 folds. However, the AUC values are lower, with an average value of 0.858. This indicates that the tuned hyperparameters did improve the overall predictive power of the model. With the tuned model validated, the performance of the model can be investigated using a variety of performance metrics.

3.2.4 Model Performance Evaluation

To evaluate the predictive power of the gradient boosted trees model in predicting the chance a client defaults on a loan, the following metrics are used, confusion matrix, recall, precision, F-score, accuracy and total misclassification rate. Beginning with the confusion matrix, which is given below.

		Actual		
		Positive	Negative	Total
Predicted	Positive	58652	5139	63791
	Negative	26150	2311	28461
	Total	84802	7450	92252

From the confusion matrix, it can be seen that the majority of the actual target data points are positive, whereas the predicted target data points are more split between the positive and negative. From the confusion matrix it is clear where the model is faltering, it is losing most of its predictive power in misclassifying actual positive target variables as negative. Moving on to the single number metrics given in the table below.

Statistics	Value
Recall	0.69
Precision	0.92
F-Score	0.79
Accuracy	0.66
Total Misclassification Rate	0.34

From the table, it can be seen that the precision of the metric of the model is much better than all of the metrics. The F-score is the next best, unsurprisingly, as it is a function of the precision. The recall and accuracy are both fairly low compared to the precision, but the actual values indicate that the model can still make accurate predictions about default rates.

The total misclassification rate is similar to the recall and accuracy, that is $0.34 = 1 - 0.66$. The high value for the precision indicates that for clients that actually do not default, the model prediction is misclassifying only a small amount as likely to default. So, the model will work well on a default risk problem where minimizing the number of false positives is very important. The model should be able to generalize well, but if the training set has outdated data then the model may not be able to make accurate predictions on the present test set. As well, gradient boosted trees model are likely to overfit, so even though the hyperparameter selection has been performed to avoid this issue, it still may be slightly present, and thus the model would not generalize perfectly.

The feature importance plot of the top 20 most important features in the gradient boosted trees model is also investigated and is given below.

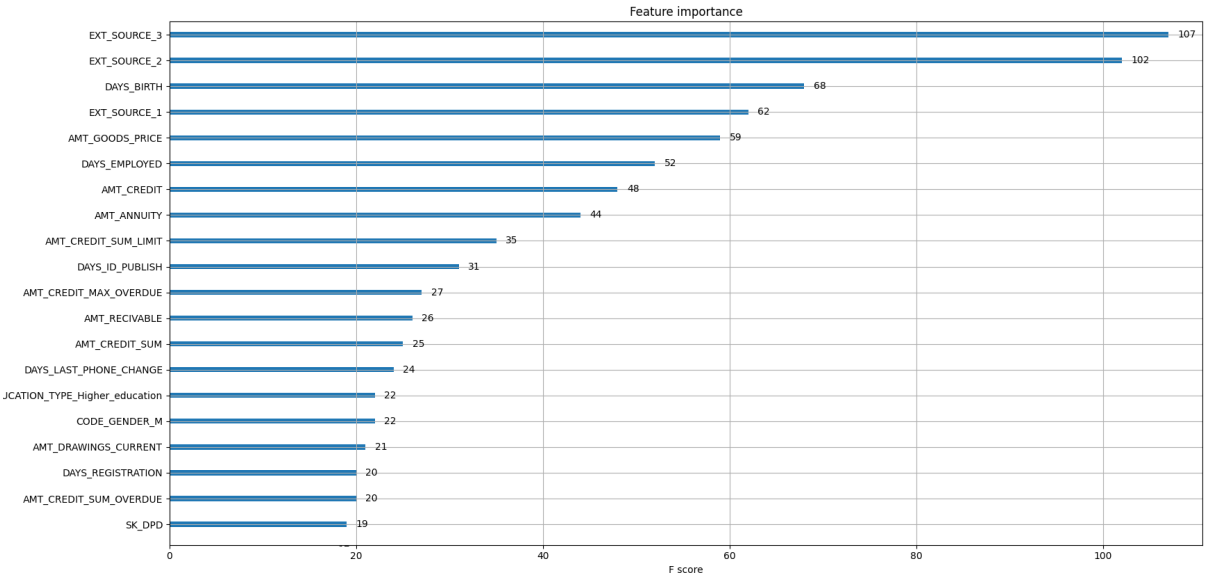

Figure 6: Feature importance plot for 20 most influential features in the tuned gradient boosted trees model.

Observing some top features in the feature importance plot, it is interesting to note that the three external source score are significant to making accurate predictions with the model. This makes sense as they may come from another credit scoring source and so if this external

credit source has a reliable model it will be a good indicator of a client's likelihood of defaulting. An interesting feature that appears in the top 20 most important is `CODE_GENDER_M`, which is a binary indicator if the client is male. This is unexpected as other financial features would naively seem to be more important in deciding loan default likelihood rather than gender, but this is not the case for the gradient boosted trees model. Additionally, the `DAYS_BIRTH` feature that counts how many days since the client's birth is very important in model prediction. The use of these features may become troublesome in countries where credit risk models can include only very limited personal information, such as Canada, as the model could become sexist or ageist, violating the countries policies. It is also interesting that the three features `DAYS_ID_PUBLISH`, `DAYS_LAST_PHONE_CHANGE` and `DAYS_REGISTRATION`, regarding changes to the loan application client information are so important for the model to make predictions, naively, it would be expected that financial features would be more important. Other features in the top 20 also seem reasonable given the context of predicting the clients' ability to pay back a loan. There are not any suspicious features other than the one previously mentioned that were more surprising than suspicious.

3.3 Logistic Regression

3.3.1 Logistic Regression without Feature Engineering

In statistics, logistic regression is a regression model where the dependent variable is categorical. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Therefore, in our case, we would use the logistic model to predict how capable each applicant is of repaying a loan.

Mathematically, suppose we have n independent variables (explanatory variables): $x_1, x_2, x_3, \cdots, x_n$, and one binary response variable Y, which is whether the client has payment difficulties in our case ($Y = 1$ means that the client has payment difficulties, and $Y = 0$ refers that the client is of all other cases). Assume there exists a linear relationship between the logit (or

log-odds) of event that $Y = 1$ (i.e. $log(odds) = log\frac{p}{1-p}$, where p is the probability of $Y = 1$) and the explanatory variables $x_1, x_2, x_3, \cdots, x_n$. We could express the mathematical form as follows:

$$log(odds) = log\frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

where $log(odds)$ is the log-odds, $log(\cdot)$ is the natural logarithm, β_i for $i \in \{1, 2, \cdots, n\}$ are the parameters of the logistic model.

Variable clustering is a method to reduce multicollinearity by grouping the correlated variables. Since logistic regression requires the explanatory variables to be independent of each other, we use variable clustering to reduce multicollinearity among variables before we fit the logistic regression model.

We use the data set whose characteristic variables are transferred to dummy variables to train this model. There are 239 variables in the data set. These 239 are separated in 98 clusters. The variables in the same cluster show the similar information. We select the variable with the lowest $1 - R^2$ ratio to best represent the cluster and the variable with the highest information value (information value measures the predictive power of a variable) to keep the strongest predictor from each cluster. After that, we select 149 variables which are in Appendix 2.

Then, we fit the step-wise logistic regression model. Specifically, step-wise logistic regression based on AIC selection criteria is conducted using R. Then, there are 58 significant variables (plus an intercept) are selected as our explanatory variable for the logistic regression model. The final model is:

$$log(odds) = log\frac{p}{1-p} = 1.208 + \sum_{i=1}^{58} \beta_i x_i$$

where 58 variables and their estimated coefficients are in Appendix 3.

To evaluate the performance of the logistic regression model, we plot the ROC curve and calculate the corresponding AUC.

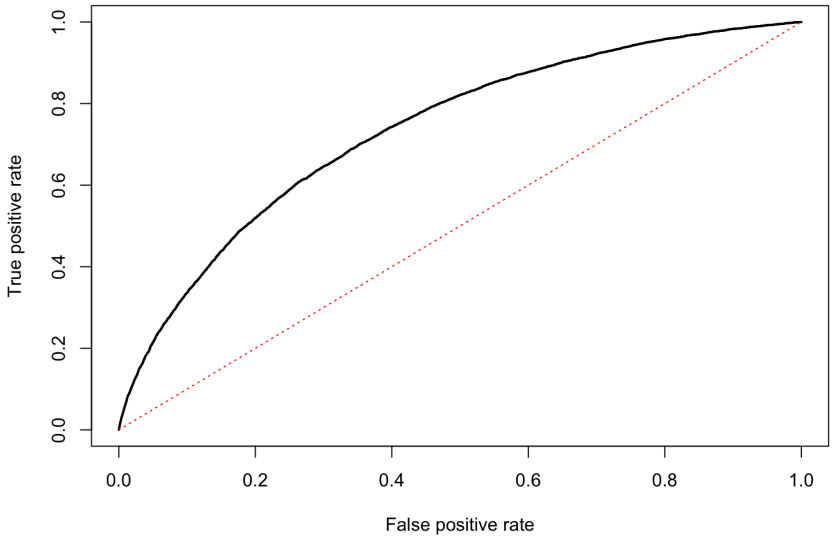

Figure 7: ROC for Logistic Regression

The AUC of the logistic regression model is 0.735 which is lower than the AUC's of the previous machine learning models with optimized parameters, so we can conclude that random forest and boosted trees with optimized parameter outperform the logistic regression model using the same feature set.

Then, we calculate the confusion matrix and some related indices to assess the performance of the logistic regression model.

		Actual		
		Positive	Negative	Total
Predicted	Positive	57035	2445	59480
	Negative	27767	5005	32772
	Total	84802	7450	92252

Note that Negative refers to client with payment difficulties, and Positive refers to the client is of all other cases.

Based on the confusion matrix, we calculate the following indices to further evaluate the performance of the logistic regression model.

Statistics	Value
Recall	0.67
Precision	0.96
F-Score	0.79
Accuracy	0.67
Total Misclassification Rate	0.33

The previous indices evaluate our model predictability from different perspectives of correctness of classification and the type of errors it make. In our case, the calculated index values illustrating classification correctness (Recall, Precision, F-Score, Accuracy) are all close to 1, which represents a relatively good performance of our logistic regression model.

3.3.2 Feature Engineering

Since the AUC of the logistic regression model is lower than the AUC's of the Random Forest and Boosted Trees, we perform additional feature engineering to improve the performance of the logistic regression.

Weight of Evidence (WOE) illustrates the predictive power of an independent variable in relation to the dependent variable. WOE is based on the probability density functions for two binary events. In our case, WOE_i for group i of a variable is defined as:

$$WOE_i = \ln(\frac{\text{Distribution of Accounts without Payment Difficulties}}{\text{Distribution of Accounts with Payment Difficulties}})$$

For a continuous variable, we split the data into 10 bins. For a categorical variable, each

category is a group. For all kinds of variables, missing values are grouped separately. Then, we use the previous formula to calculate WOE of each bin.

By doing this, we do not need to create dummy variables for the categorical variables, which reduce the number of variables and the execution time. Also, we do not need to fill NA with zero or mode of data, and NA can also provide useful information. In addition, WOE can can reduce the influence of outliers.

The following graph shows the WOE's for EXT_SOURCE_3.

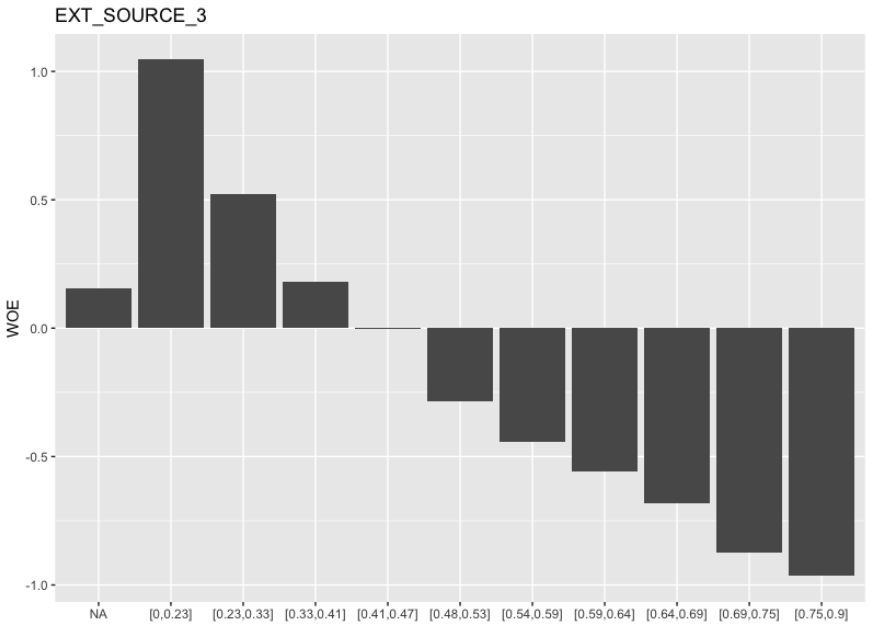

Figure 8: WOE

From the previous graph, we know that the values of EXT_SOURCE_3 are split into 10 bins, and missing values are grouped separately. Furthermore, WOE decreases as EXT_SOURCE_3 decreases.

Since WOE is based on the probability density functions for two binary events, and the response variable of the logistic regression model is also binary, so WOE suite logistic regres-

sion well. Thus, we transform all variables to WOE. In the next section, we will use WOE to fit the logistic regression and evaluate the performance of the model.

3.3.3 Logistic Regression with Feature Engineering

After transferring data to WOE, we fit the logistic regression model again.

First, we use variable clustering to reduce the multicollinearity among variables. There are 117 variables in total, and we separate them in 25 clusters. We select the variable with the lowest $1 - R^2$ ratio and the variable with the highest information value from each cluster. After that, we select 41 features which are in Appendix 4.

Then, we fit the step-wise logistic regression model. There are 32 significant variables (plus an intercept) are selected as our explanatory variable for the logistic regression model. The final model is:

$$log(odds) = log\frac{p}{1-p} = -0.0132 + \sum_{i=1}^{32} \beta_i x_i$$

where 32 variables and their estimated coefficients are in Appendix 5.

Then, we plot the ROC curve and calculate the corresponding AUC.

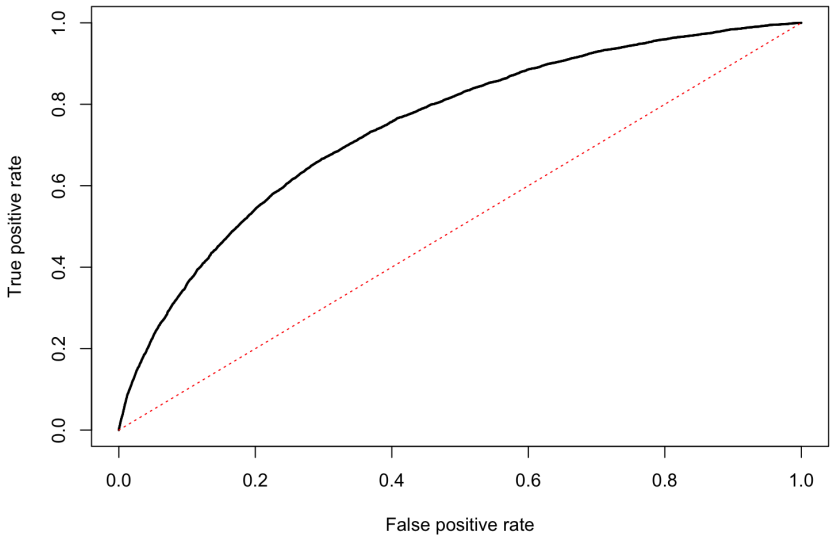

Figure 9: ROC for Logistic Regression after Performing Feature Engineering

The AUC of the logistic regression model is 0.745 which is same as the AUC of Random Forest and get closer to AUC of the Boosted Tress, which shows that feature engineering improves the performance of logistic regression.

Then, we calculate the confusion matrix and some related indices to assess the performance of the logistic regression model after performing feature engineering.

		Actual		
		Positive	Negative	Total
Predicted	Positive	57677	2351	60028
	Negative	27125	5099	32224
	Total	84802	7450	92252

Note that Negative refers to client with payment difficulties, and Positive refers to the client is of all other cases.

Based on the confusion matrix, we calculate the following indices to further evaluate the performance of the logistic regression model after performing feature engineering.

Statistics	Value
Recall	0.68
Precision	0.96
F-Score	0.80
Accuracy	0.68
Total Misclassification Rate	0.32

By the previous table, we know that Recall, F-Score and Accuracy increase by 0.01 and Precision does not change after performing feature engineering. Thus, we can conclude that feature engineering improves the performance of logistic regression model.

4 Model Comparison

The baseline model is a random forest model with default parameters. Using the testing set, we know that the AUC of the baseline model is 0.72, and the confusion matrix of the baseline model is:

		Actual		
		Positive	Negative	Total
Predicted	Positive	54151	2426	56577
	Negative	30651	5024	35675
	Total	84802	7450	92252

Then, we calculate the indices based on the previous confusion matrix and include the result in the following table which contains the indices of all the models based on the same testing set.

	Baseline Model	Random Forest	Boosted Trees	Logistic Regression 1	Logistic Regression 2
AUC	0.72	0.745	0.750	0.735	0.745
Recall	0.64	0.68	0.69	0.67	0.68
Precision	0.96	0.96	0.92	0.96	0.96
F-Score	0.77	0.80	0.79	0.79	0.80
Accuracy	0.64	0.68	0.66	0.67	0.68
Total Misclassification Rate	0.36	0.32	0.34	0.33	0.32

Random Forest and Boosted Trees refer to the corresponding machine learning model with optimized parameters, Logistic Regression 1 refers to the logistic regression model without feature engineering, and Logistic Regression 2 refers to the logistic regression model with feature engineering.

Beginning by comparing the AUC between the models, it can be seen that all other models outperform the baseline model in this metric, with the gradient boosted trees model having the best value. The logistic regression 2 and the random forest model have very similar values. For the recall of the models, once again, all other models outperform the baseline model and once again the gradient boosted trees model has the best value. For the precision, all the model except gradient boosted trees perform exceptionally well, the gradient boost trees precision is lower than the rest. For the F-score a similar theme is seen as in the precision as F-score is a direction function of precision. For the accuracy, the random forest model and logistic regression model perform the best, with the baseline model performing the most poorly. The total misclassification rate follow the inverse trends as described for accuracy. It is particularly interesting that the random forest model and logistic regression 2 model have the same scores for all metrics to the second decimal place. This indicates that these two models are equally valid and make very similar predictions to the likelihood

of client default. It is also unsurprising that the logistic regression model 2 out scores the logistic regression 1 model in all metrics, indicating that the feature engineering did improve prediction power. When comparing, the two decision tree based models, the random forest and gradient boosted trees, both perform similarly with the random forest performing better on some metrics and the gradient boosted trees performing better on other metrics. These differences indicate that depending on the context of the problem and types of classifications you need, each model will perform well in different areas. Another interesting difference between the two decision tree models is that the tuned 10-Fold AUC for the gradient boosted trees is significantly larger than that of the random forest model. This may indicate that the gradient boosted trees model is overfitting to the training set, as it performs significantly better on the 10-fold data set, where overfitting is much more difficult.

5 Conclusion

A variety of models to predict the likelihood that a client will default on a loan have been constructed. The various datasets of client information were merged and consolidated into a single table so that training and testing sets could be constructed from the full table. Various features with irrelevant information were removed, and similar features were merged together. Missing values for the features were replaced with an appropriate value. The categorical variables were replaced with dummy variable counterparts to ensure that the models can make accurate use of those features. Features were engineered from the available information to allow the models a greater set of features to make predictions. The cleaned and processed data was then split into a 70% training set and 30% test set. A random forest model was constructed, and its hyperparameters tuned to improve model performance. The model was evaluated using a variety of different classification metrics to demonstrate its predictive power. A gradient boosted trees model was constructed with tuned hyperparameters as well. The model was also evaluated using classification metrics, and a feature importance plot was generated to help understand how the model is using different features to make predictions. A logistic regression model was then constructed on the same set of features

as the two machine learning models and evaluated in the same manner. A collection of features were then engineered specifically for the logistic regression model, such as weight of evidences. The logistic regression model was then fit to the data set with the newly engineered features and evaluated in the same way as the other models. The performance of all the models was then compared, including a baseline model that is a random forest model with default parameters. It was found that the performance of the tuned random forest model and the logistic regression model with the engineered features was nearly identical. It was found that the gradient boosted trees model performed better for some metrics but worse in others. In general, the baseline model and logistic regression model with the default data set performed the worst. The random forest model and the logistic regression model with the feature engineered data set performed the best. The gradient boosted trees model was in the middle performance-wise, with the model being potentially better than the others if the problem requires very specific metrics to be accurate. Overall, the models were able to make accurate predictions on the likelihood of a client defaulting on a loan, and the models appear to generalize well. The most accurate models could be deployed to a company with some oversight needed, as the models do not have an extremely high predictive power and sometimes misclassify data points.

6 Appendix

Appendix 1

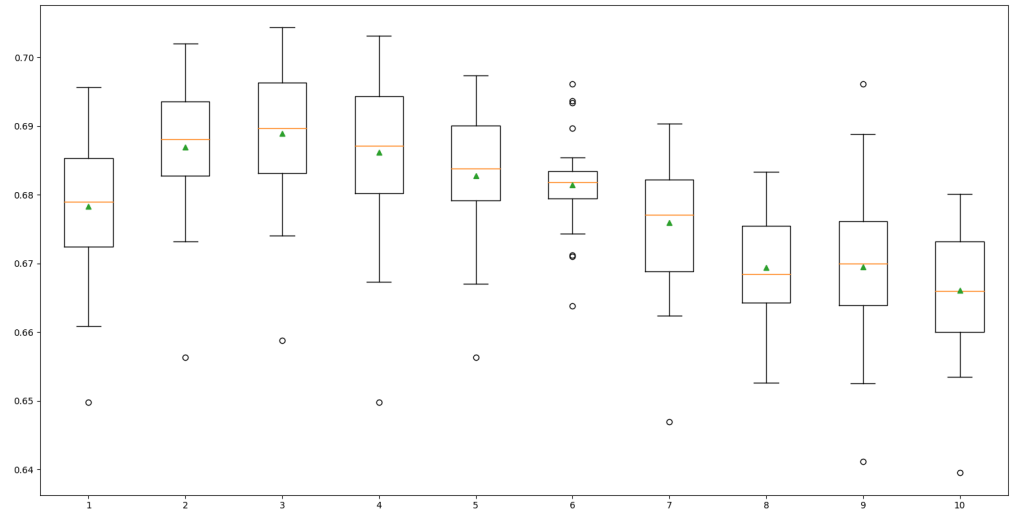

Figure 10: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

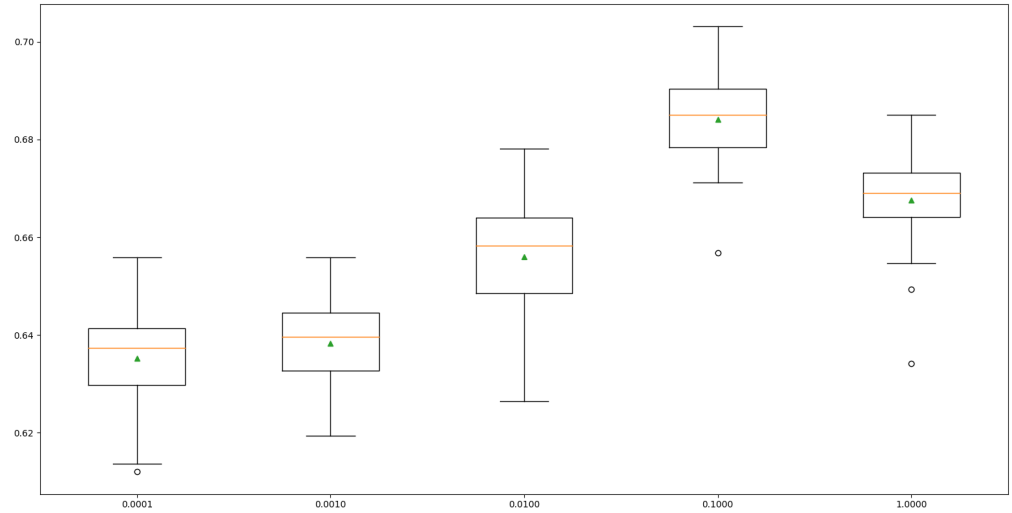

Figure 11: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

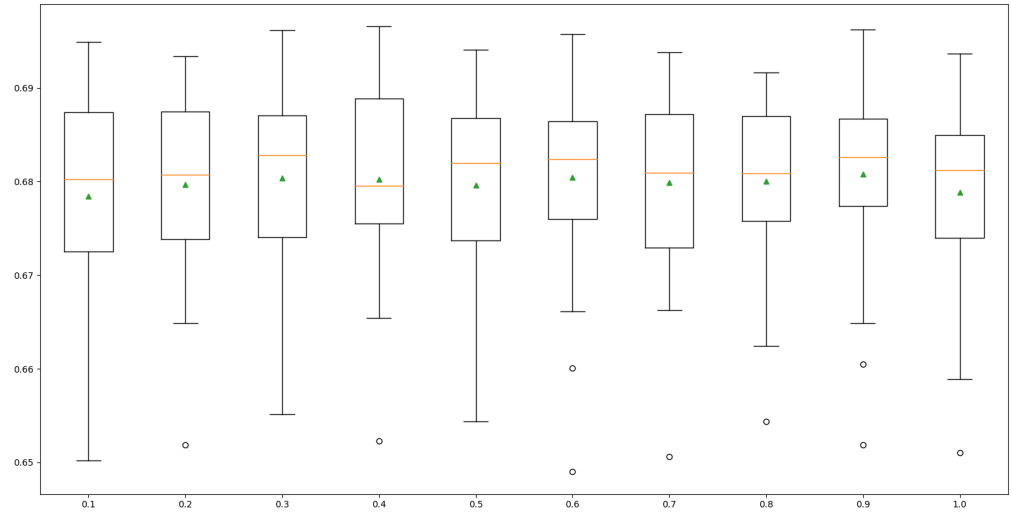

Figure 12: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

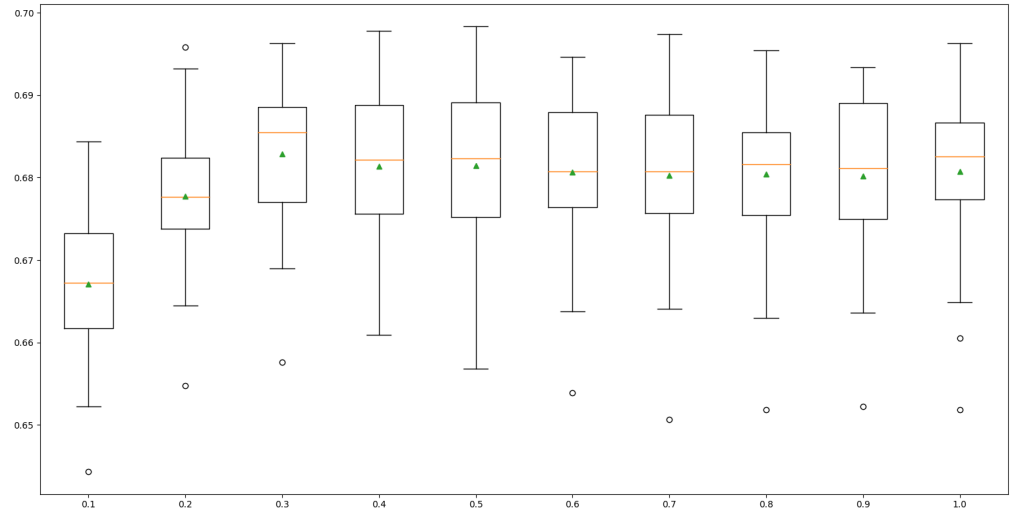

Figure 13: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

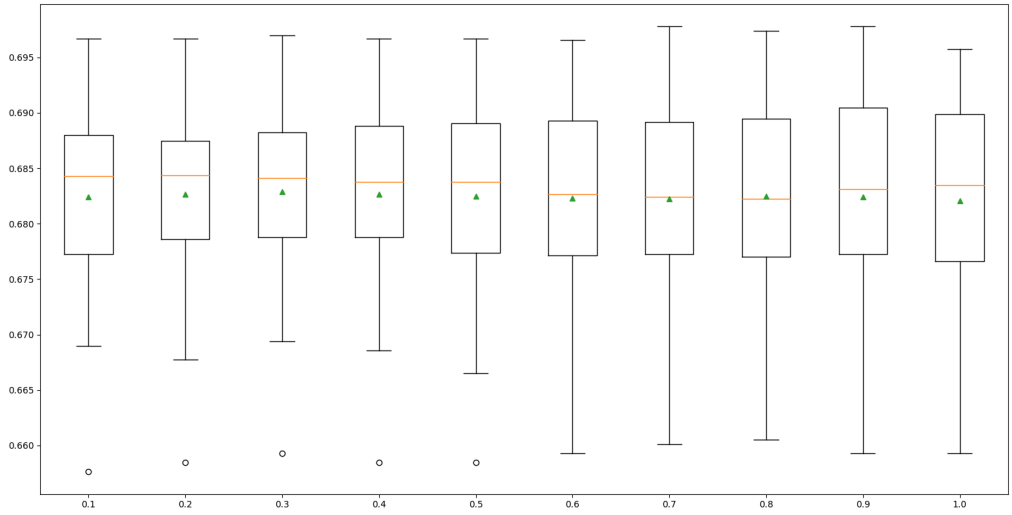

Figure 14: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

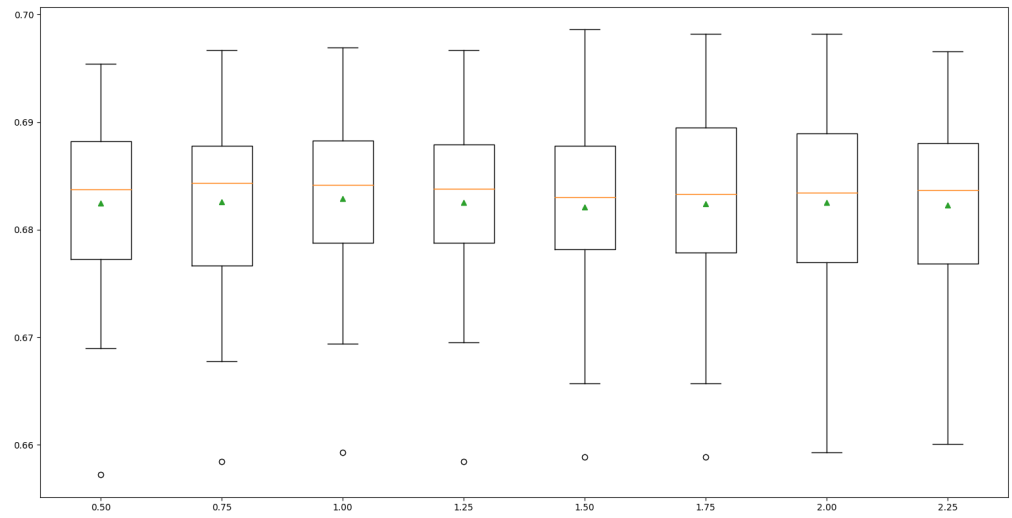

Figure 15: Box and whisker plot of the accuracy as a function of the tree depth in the gradient boosted trees model.

Appendix 2

Explanatory Variable		
TOTALAREA_MODE	CREDIT_ACTIVE	NAME_FAMILY_STATUS_Civil_marriage
APARTMENTS_AVG	CREDIT_TYPE_CREDIT_CARD	ORGANIZATION_TYPE_Industry_type_11
DAYS_EMPLOYED	FLOORSMIN_MEDI	ORGANIZATION_TYPE_Business_Entity_Type_2
ORGANIZATION_TYPE_XNA	FLOORSMIN_MODE	DAYS_ID_PUBLISH
FLOORSMIN_AVG	ORGANIZATION_TYPE_Services	DAYS_REGISTRATION
YEARS_BUILD_AVG	OCCUPATION_TYPE_Private_service_staff	WALLSMATERIAL_MODE_Others
AMT_GOODS_PRICE	ENTRANCES_AVG	ORGANIZATION_TYPE_Construction
FLOORSMAX_AVG	ENTRANCES_MEDI	ORGANIZATION_TYPE_Transport_type_4
FLOORSMAX_MEDI	NAME_FAMILY_STATUS_Married	NAME_INCOME_TYPE_State_servant
AMT_DRAWINGS_POS_CURRENT	NAME_FAMILY_STATUS_Separated	ORGANIZATION_TYPE_School
AMT_DRAWINGS_CURRENT	CNT_INSTALMENT_FUTURE	ORGANIZATION_TYPE_Industry_type_5
NONLIVINGAPARTMENTS_MEDI	AMT_REQ_CREDIT_BUREAU_WEEK	ORGANIZATION_TYPE_Industry_type_13
CODE_GENDER_M	AMT_REQ_CREDIT_BUREAU_DAY	FONDKAPREMONT_MODE_reg_oper_account
NONLIVINGAREA_AVG	ORGANIZATION_TYPE_Self_employed	FONDKAPREMONT_MODE_reg_oper_spec_account
NONLIVINGAREA_MEDI	HOUR_APPR_PROCESS_START	FLAG_CONT_MOBILE
FLAG_OWN_REALTY_N	NAME_EDUCATION_TYPE_Lower_secondary	WALLSMATERIAL_MODE_Monolithic
EXT_SOURCE_2	WEEKDAY_APPR_PROCESS_START_MONDAY	FONDKAPREMONT_MODE_not_specified
REGION_RATING_CLIENT	WEEKDAY_APPR_PROCESS_START_WEDNESDAY	SK_DPD
OCCUPATION_TYPE_Core_staff	ORGANIZATION_TYPE_Military	OCCUPATION_TYPE_High_skill_tech_staff
ORGANIZATION_TYPE_Kindergarten	WEEKDAY_APPR_PROCESS_START_SUNDAY	ORGANIZATION_TYPE_Telecom
NAME_CONTRACT_TYPE_Cash_loans	WEEKDAY_APPR_PROCESS_START_SATURDAY	ORGANIZATION_TYPE_Agriculture
NAME_EDUCATION_TYPE_Higher_education	OCCUPATION_TYPE_Accountants	WALLSMATERIAL_MODE_Mixed
EMERGENCYSTATE_MODE_No	ORGANIZATION_TYPE_Bank	ORGANIZATION_TYPE_Insurance
HOUSETYPE_MODE_block_of_flats	BASEMENTAREA_AVG	ORGANIZATION_TYPE_Industry_type_3
REG_REGION_NOT_WORK_REGION	BASEMENTAREA_MEDI	ORGANIZATION_TYPE_Transport_type_2
OBS_60_CNT_SOCIAL_CIRCLE	REG_CITY_NOT_LIVE_CITY	NAME_HOUSING_TYPE_Co_op_apartment
OBS_30_CNT_SOCIAL_CIRCLE	NAME_HOUSING_TYPE_Rented_apartment	OCCUPATION_TYPE_Cleaning_staff
COMMONAREA_AVG	WALLSMATERIAL_MODE_Panel	ORGANIZATION_TYPE_Trade_type_6
COMMONAREA_MEDI	WALLSMATERIAL_MODE_Stone_brick	ORGANIZATION_TYPE_Realtor

Figure 16: Explanatory Variables

YEARS_BEGINEXPLUATATION_MEDI	ORGANIZATION_TYPE_Industry_type_10	WEEKDAY_APPR_PROCESS_START_FRIDAY
YEARS_BEGINEXPLUATATION_AVG	ORGANIZATION_TYPE_Industry_type_9	ORGANIZATION_TYPE_Postal
DEF_30_CNT_SOCIAL_CIRCLE	ORGANIZATION_TYPE_Religion	HOUSETYPE_MODE_terraced_house
DEF_60_CNT_SOCIAL_CIRCLE	OCCUPATION_TYPE_IT_staff	ORGANIZATION_TYPE_Industry_type_12
NAME_FAMILY_STATUS_Widow	ORGANIZATION_TYPE_Transport_type_3	ORGANIZATION_TYPE_Restaurant
CNT_FAM_MEMBERS	EXT_SOURCE_1	ORGANIZATION_TYPE_Industry_type_1
LANDAREA_MEDI	FLAG_EMAIL	OCCUPATION_TYPE_Realty_agents
NAME_TYPE_SUITE_Unaccompanied	DAYS_LAST_PHONE_CHANGE	ORGANIZATION_TYPE_Trade_type_1
NAME_TYPE_SUITE_Family	NAME_INCOME_TYPE_Commercial_associate	ORGANIZATION_TYPE_Industry_type_7
REG_CITY_NOT_WORK_CITY	ORGANIZATION_TYPE_Trade_type_3	OCCUPATION_TYPE_Secretaries
LIVE_CITY_NOT_WORK_CITY	ORGANIZATION_TYPE_Culture	ORGANIZATION_TYPE_University
AMT_CREDIT_SUM	ORGANIZATION_TYPE_Emergency	ORGANIZATION_TYPE_Cleaning
AMT_CREDIT_SUM_DEBT	ORGANIZATION_TYPE_Trade_type_7	ORGANIZATION_TYPE_Police
ORGANIZATION_TYPE_Medicine	ORGANIZATION_TYPE_Housing	NAME_TYPE_SUITE_Other_A
EXT_SOURCE_3	ORGANIZATION_TYPE_Electricity	ORGANIZATION_TYPE_Industry_type_2
OCCUPATION_TYPE_Security_staff	ORGANIZATION_TYPE_Government	ORGANIZATION_TYPE_Advertising
FONDKAPREMONT_MODE_org_spec_account	OCCUPATION_TYPE_Low_skill_Laborers	AMT_PAYMENT_GREATER_EQUAL_INSTALMENT
NAME_HOUSING_TYPE_House_apartment	ORGANIZATION_TYPE_Business_Entity_Type_1	AMT_CREDIT_MAX_OVERDUE
NAME_HOUSING_TYPE_With_parents	ORGANIZATION_TYPE_Business_Entity_Type_3	ORGANIZATION_TYPE_Mobile
OWN_CAR_AGE	ORGANIZATION_TYPE_Other	ORGANIZATION_TYPE_Trade_type_4
FLAG_OWN_CAR_N	OCCUPATION_TYPE_HR_staff	

Figure 17: Explanatory Variables

Appendix 3

Explanatory Variable	Coefficient
EXT_SOURCE_2	-2.021
EXT_SOURCE_3	-1.451
CREDIT_ACTIVE	0.5798
EXT_SOURCE_1	-0.5999
CODE_GENDER_M	0.3487
NAME_EDUCATION_TYPE_Higher_education	-0.2956
ORGANIZATION_TYPE_XNA	-29.27
DAYS_EMPLOYED	0.00007862
CNT_INSTALMENT_FUTURE	0.02075
NAME_CONTRACT_TYPE_Cash_loans	0.4709
SK_DPD	0.3274
AMT_DRAWINGS_CURRENT	0.00001037
DEF_30_CNT_SOCIAL_CIRCLE	0.2075
FLAG_OWN_CAR_N	0.2658
FLOORSMAX_AVG	-0.4714
DAYS_ID_PUBLISH	0.00005257
NAME_FAMILY_STATUS_Married	-0.2707
OWN_CAR_AGE	0.009632
REGION_RATING_CLIENT	0.1199
AMT_CREDIT_SUM_DEBT	5.254E-08
NAME_FAMILY_STATUS_Widow	-0.2731
DAYS_LAST_PHONE_CHANGE	0.00006725
NAME_INCOME_TYPE_State_servant	-0.1482
ORGANIZATION_TYPE_Bank	-0.5153
REG_CITY_NOT_LIVE_CITY	0.1675
AMT_DRAWINGS_POS_CURRENT	-7.98E-06
CNT_FAM_MEMBERS	0.06826
ORGANIZATION_TYPE_Self_employed	0.09294
NAME_INCOME_TYPE_Commercial_associate	-0.09546

Explanatory Variable	Coefficient
ORGANIZATION_TYPE_Postal	0.3796
WALLSMATERIAL_MODE_Panel	-0.06376
ORGANIZATION_TYPE_Industry_type_9	-0.3155
OCCUPATION_TYPE_Core_staff	-0.1164
DAYS_REGISTRATION	8.835E-06
WEEKDAY_APPR_PROCESS_START_SUNDAY	-0.1512
ORGANIZATION_TYPE_Industry_type_10	-1.437
OCCUPATION_TYPE_High_skill_tech_staff	-0.149
WEEKDAY_APPR_PROCESS_START_MONDAY	-0.0813
FONDKAPREMONT_MODE_reg_oper_account	0.105
WALLSMATERIAL_MODE_Others	0.3335
NAME_FAMILY_STATUS_Civil_marriage	-0.09284
FLAG_CONT_MOBILE	-0.5873
ORGANIZATION_TYPE_Restaurant	0.2706
OCCUPATION_TYPE_Accountants	-0.1546
OCCUPATION_TYPE_Low_skill_Laborers	0.2509
ORGANIZATION_TYPE_Military	-0.3244
ORGANIZATION_TYPE_Medicine	-0.1483
ORGANIZATION_TYPE_Electricity	-0.4378
WEEKDAY_APPR_PROCESS_START_SATURDAY	-0.07204
ORGANIZATION_TYPE_School	-0.1535
ORGANIZATION_TYPE_Housing	-0.2208
ENTRANCES_MEDI	-0.323
ORGANIZATION_TYPE_Industry_type_12	-0.695
NAME_TYPE_SUITE_Unaccompanied	0.05255
ORGANIZATION_TYPE_Transport_type_3	0.2674
NAME_HOUSING_TYPE_House_apartment	-0.05936
ORGANIZATION_TYPE_Police	-0.2498
ORGANIZATION_TYPE_Construction	0.1136

Figure 18: Explanatory Variables and Coefficients

Appendix 4

Explanatory Variable		
WOE_ENTRANCES_MEDI	WOE_AMT_GOODS_PRICE	WOE_CODE_GENDER
WOE_NONLIVINGAREA_MODE	WOE_AMT_CREDIT	WOE_DAYS_REGISTRATION
WOE_AMT_CREDIT_MAX_OVERDUE	WOE_REG_REGION_NOT_WORK_REGION	WOE_NAME_FAMILY_STATUS
WOE_AMT_REQ_CREDIT_BUREAU_WEEK	WOE_REG_CITY_NOT_WORK_CITY	WOE_NAME_HOUSING_TYPE
WOE_AMT_DRAWINGS_CURRENT	WOE_FLOORSMAX_AVG	WOE_OWN_CAR_AGE
WOE_AMT_TOTAL_RECEIVABLE	WOE_ELEVATORS_AVG	WOE_AMT_INCOME_TOTAL
WOE_LIVINGAPARTMENTS_MEDI	WOE_CNT_INSTALMENT_FUTURE	WOE_FLAG_WORK_PHONE
WOE_COMMONAREA_MODE	WOE_AMT_PAYMENT_GREATER_EQUAL_INSTALMENT	WOE_FLAG_OWN_REALTY
WOE_DAYS_EMPLOYED	WOE_EXT_SOURCE_3	WOE_NAME_CONTRACT_TYPE
WOE_FLAG_EMP_PHONE	WOE_CREDIT_TYPE_CREDIT_CARD	WOE_CNT_FAM_MEMBERS
WOE_DEF_30_CNT_SOCIAL_CIRCLE	WOE_FLOORSMIN_AVG	WOE_NAME_TYPE_SUITE
WOE_OBS_30_CNT_SOCIAL_CIRCLE	WOE_FLOORSMIN_MEDI	WOE_WEEKDAY_APPR_PROCESS_START
WOE_EXT_SOURCE_2	WOE_EXT_SOURCE_1	WOE_DAYS_LAST_PHONE_CHANGE
WOE_REGION_RATING_CLIENT	WOE_OCCUPATION_TYPE	

Figure 19: Explanatory Variables

Appendix 5

Explanatory Variable	Coefficient	Explanatory Variable	Coefficient
WOE_EXT_SOURCE_3	0.82828	WOE_FLAG_WORK_PHONE	0.56509
WOE_EXT_SOURCE_2	0.72548	WOE_NAME_FAMILY_STATUS	0.19102
WOE_EXT_SOURCE_1	0.52119	WOE_WEEKDAY_APPR_PROCESS_START	1.41596
WOE_OCCUPATION_TYPE	0.44782	WOE_AMT_DRAWINGS_CURRENT	0.19974
WOE_AMT_TOTAL_RECEIVABLE	0.52129	WOE_DAYS_REGISTRATION	0.13752
WOE_DAYS_EMPLOYED	0.46479	WOE_LIVINGAPARTMENTS_MEDI	0.10339
WOE_CNT_INSTALMENT_FUTURE	1.13406	WOE_AMT_GOODS_PRICE	0.3055
WOE_AMT_CREDIT	0.23431	WOE_AMT_REQ_CREDIT_BUREAU_WEEK	-0.77751
WOE_NAME_CONTRACT_TYPE	1.04525	WOE_CREDIT_TYPE_CREDIT_CARD	0.29755
WOE_OWN_CAR_AGE	0.73706	WOE_FLAG_OWN_REALTY	-0.81984
WOE_CODE_GENDER	0.58491	WOE_NAME_TYPE_SUITE	0.3257
WOE_DEF_30_CNT_SOCIAL_CIRCLE	0.62921	WOE_FLAG_EMP_PHONE	-0.17554
WOE_ELEVATORS_AVG	0.34246	WOE_CNT_FAM_MEMBERS	0.26871
WOE_AMT_CREDIT_MAX_OVERDUE	0.49256	WOE_NAME_HOUSING_TYPE	0.0385
WOE_DAYS_LAST_PHONE_CHANGE	0.24117	WOE_AMT_PAYMENT_GREATER_EQUAL_INSTALMENT	-0.01
WOE_REGION_RATING_CLIENT	0.26594	WOE_REG_REGION_NOT_WORK_REGION	-0.08107

Figure 20: Explanatory Variables and Coefficients