CS 6220 Summer '18 Project Report : Home Credit Default Risk

Chia Yi Liaw
liaw.c@husky.neu.edu

Mounica Subramani
subramani.m@husky.neu.edu

Sharyu Deshmukh
deshmukh.s@husky.neu.edu

Somya Bhargava
bhargava.so@husky.neu.edu

1 Abstract

Loan repayment ability is a supervised classification problem. The objective of this project is to use historical loan application data from the Kaggle Home Credit Default Risk dataset to explore effective methods and create classifier to predict either loaner would be able to repay based on statistic method and supervised machine learning.

2 Business Understanding

Home Credit is a finance provider that focuses on serving the unbanked population. Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. So, Home Credit tries to broaden financial inclusions for unbanked population. The Home Credit Default Risk challenge is a standard supervised machine learning task where the goal is to use historical loan application data to predict their clients' repayment abilities based on datasets provided.
Supervised: The labels are included in the training data and the goal is to train a model to learn to predict the labels from the features Classification: The label is a binary variable, 0 (will repay loan on time), 1 (will have difficulty repaying loan)

3 Data Understanding

The data is provided by Home Credit, a service dedicated to provided lines of credit (loans) to the unbanked population.
There are 7 different sources of data:

- application_train or application_test: the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating 0: the loan was repaid or 1: the loan was not repaid.
- bureau: data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- bureau_balance: monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- previous_application: previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- POS_CASH_BALANCE: monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.

- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.

- installments_payment: payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment. Moreover, we are provided with the definitions of all the columns (in HomeCredit_columns_description.csv) and an example of the expected submission file.

  In this project, we will stick to using only the main application training and testing data. This will let us establish a baseline that we can then improve upon.

4 Data Preparation

4.1 Exploratory Data Analysis

- Missing values: The Dataset contains 122 features with 307511 entries. As the respect of missing value, The distribution map gives better evaluation and understanding of missing data. From the visualization , we found out that missing value is concentrated in several features. In addition, there are 23 feature contains more than 60 percent of missing value. Imputation and deletion will be performed in the data pre processing stage.

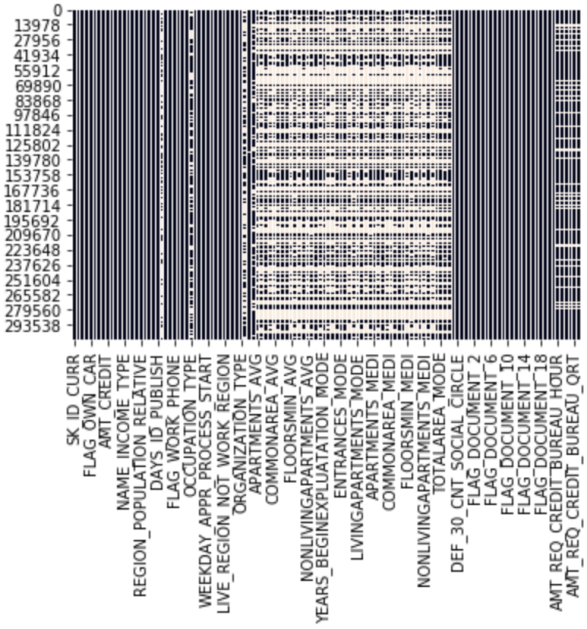

Figure 1: Missing Value Map

- Feature correlation : By calculating the r-square to determine the linear correlation between features. Pearson feature correlation quantifies the degree to which a relationship between two variables. By analyzing correlation feature, we implement the random forest method and Light-GBM to improve calculation efficiency. Pearson correlation heat map shows the correlation between predictors, the lighter green indicates the higher correlation.

  Taking the top 5 features chosen from random forest, Organization Type and Education Type of client's versus the average amount annuity ,goods prices and credit. Using Tableau to show the visualization shown on the right, with client's organization type in culture and academic degree education, has relatively high average amount annuity, good prices and credit.

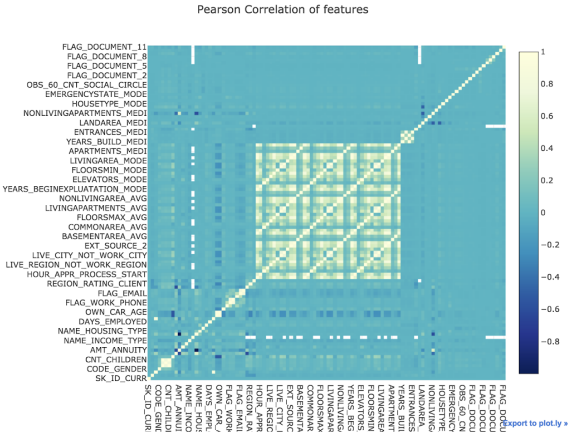

Figure 2: Feature Correlation Map

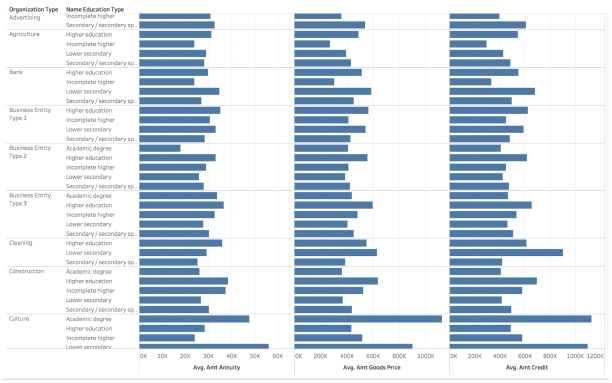

Figure 3: Top Five Features vs Average of Annuity, Goods Price, Amount Credit

4.2 Data Preprocessing

- Feature selection : With the high dimensional and to reduce the variance of the model or chances of overfitting. There are 23 features contain more than 60 percent missing value, hence we decide to remove those features as it's not suitable for training models. In addition, apply the LightGBM, removes the features that are collinear, low importance and zero importance, beside LightGBM, also run the random forest to rank the feature and then remove the features with the threshold equal to 0.0003. We have also used an ensemble technique where we used LightGBM gradient boosting model on the result of random forest. The outputs so generated are stored in csv format so that we can train them all and compare the results.

- Encoding the Categorical Variable : Encoding transforms categorical features to a format that works better with classification and regression algorithm. By using the one hot encoding for multiple categories and label encoding for binary categorical features, datasets would be better processed.

- Imputation of Missing Value: After feature selection , There are still some columns with missing data. Hence, will input the missing value with mode which is the most frequent value in the column, moreover, it would apply well on the categorical variable.

- Dealing with Imbalance Data: The below histogram show the imbalanced distribution of loan repayment in the dataset. The target variable defines if the loan was repaid by the borrower or not. From the graph, we find out that the data is highly imbalanced. Since, it might lead to incorrect result during modeling process, we needed re-sampling of the dataset. As a result, we have used both over and under-sampling using various to train the model. By looking at the evaluation, we will choose the better sampling method.

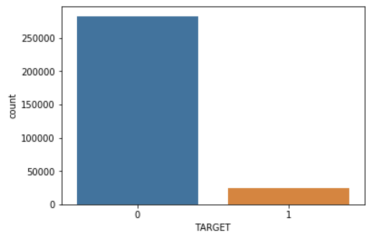

Figure 4: Original Distribution

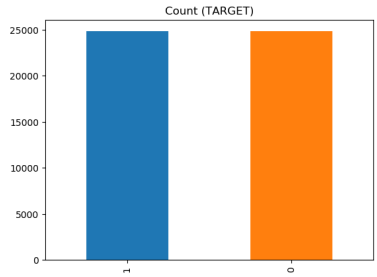

Figure 5: After Under-sampling

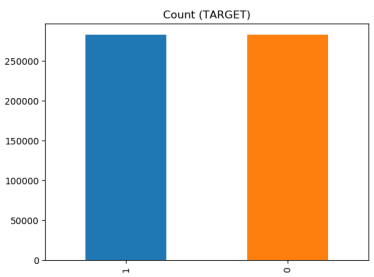

Figure 6: After Over-sampling

5 Modeling

5.1 Methods

5.1.1 Logistic Regression

With the dependent variable is dichotomous in the home credit default risk, logistic regression modeling is a proper regression analysis to be applied in the project. The sigmoid function () to simulate events. In addition, when the estimation value is greater than 0.5 which would classify the prediction as class 1. In contrast, suppose the estimation is less than 0.5, it will be classified to class 0. Ridge Regression performs the L2 regularization where

$$\text{Minimize Objective} = \text{LS Obj} + \alpha * (\text{sum of square of coefficients}) \tag{1}$$

and Lasso Regression performs the L1 regularization where

$$\text{Minimize Objective} = \text{LS Obj} + \alpha * (\text{sum of absolute value of coefficients}) \tag{2}$$

The logistic function is defined as follows:

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

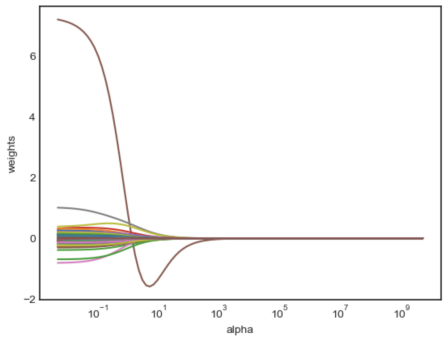

Figure 7: L1 Penalty Logistic Regression Patt

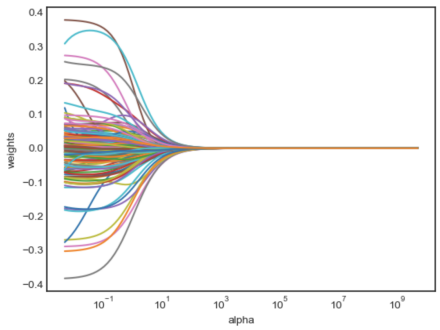

Figure 8: L2 Penalty Logistic Regression Path

5.1.2 Support Vector Machine

Support Vector Machines (SVM) are based on the idea of finding a hyperplane that best divides a dataset into two classes. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVM should perform well in high-order problems, On the other hand, it's hard to predict the actual boundary shape of a data with more than 60 features. The cost was set at 1.0 in regarding to the cost() function

$$\min_{\theta} \sum_{i=1}^{m} [y^i cost_1(\theta^T x^i) + (i - y^i) cost_0(\theta^T x^i)] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

5.1.3 K-Nearest Neighbors

prediction would be 0. The higher k is, the smoother the model would be. However, when k is small, it would be easily affected by noise and perform under fitting. In addition to the mean error rate = 0 when K value between 30 to 40 , adjusting k value would influence the accuracy. Comparing from Euclidean distance () and Manhattan Distance () to get a better model.

$$Euclidean\ Distanced(x,y) = \sqrt{\sum_{i=1}^{n}(x_2 - y_2)^2}$$

$$Manhattan\ Distanced(x,y) = \sqrt{\sum_{i=1}^{n}|x_2 - y_2|}$$

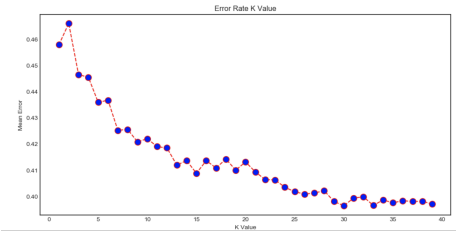

Figure 9: K-value vs Mean Error Rate

5.1.4 Gaussian Naive Bayes

A Gaussian Naive Bayes algorithm is specifically used when the features have continuous values. When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contains a continuous attribute x. We first segment the data by the class, and then compute the mean and variance of x in each class. (copy formula and text from wikipedia)

5.1.5 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of over fitting to their training set. Regression Trees are known to be very unstable, in other words, a small change in your data may drastically change the model. The Random Forest uses this instability as an advantage through bagging resulting in a very stable model. We simply estimate the desired Regression Tree on many bootstrap samples (re-sample the data many times with replacement and re-estimate the model) and make the final prediction as the average of the predictions across the trees. As one of the best use cases for random forest is feature selection, we examined which variables work best/worst in each tree.

$$maxfeatures = \sqrt{nfeature}$$

5.1.6 Extra Trees Classification

Adding one further step of randomization yields extremely randomized trees, or ExtraTrees. These are trained using bagging and the random subspace method, like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Instead of computing the locally optimal feature/split combination (based on, e.g., information gain or the Gini impurity), for each feature under consideration, a random value is selected for the split. This value is selected from the feature's empirical range (in the tree's training set, i.e., the bootstrap sample).

5.1.7 Bagging Classifier

Decision trees are sensitive to the specific data on which they are trained. If the training data is changed (e.g. a tree is trained on a subset of the training data) the resulting decision tree can be quite different and in turn the predictions can be quite different. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees. It creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples - where N is the size of the original training set; many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set.

5.1.8 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion.
The objective of any supervised learning algorithm is to define a loss function and minimize it. The mean squared error (MSE) as loss is defined as:

$$Loss = MSE = \sum (y_i - y_i^p)^2$$

where, y_i = ith target value, y_i^p = ith prediction, $L(y_i, y_i^p)$ is Loss function

We want our predictions, such that our loss function (MSE) is minimum. By using gradient descent and updating our predictions based on a learning rate, we can find the values where MSE is minimum.

$$y_i^p = y_i^p + \alpha * \delta \sum (y_i - y_i^p)^2 / \delta y_i^p$$

which becomes, $y_i^p = y_i^p - \alpha * 2 * \sum (y_i - y_i^p)$

where, α is learning rate and $\sum (y_i - y_i^p)$ is sum of residuals

So, we are basically updating the predictions such that the sum of our residuals is close to 0 (or minimum) and predicted values are sufficiently close to actual values.

5.1.9 AdaBoost

In general Boosting refers to the problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb. AdaBoost or Adaptive Boosting was the first realization of boosting that saw remarkable success in application. The weak learners in AdaBoost are decision trees with a single split, called decision stumps for their shortness. AdaBoost works by weighting the observations, putting more weight on difficult to classify instances and less on those already handled well. New weak learners are added sequentially that focus their training on the more difficult patterns. This means that samples that are difficult to classify receive increasing larger weights until the algorithm identifies a model that correctly classifies these samples. Predictions are made by majority vote of the weak learners' predictions, weighted by their individual accuracy.

5.1.10 LightGBM

Light GBM is a gradient boosting framework that uses tree-based learning algorithm. It splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'. Leaf wise splits lead to increase in complexity and may lead to overfitting and it can be overcome by specifying another parameter max-depth which specifies the depth to which splitting will occur.

5.1.11 XGBoost

XGBoost or eXtreme Gradient Boosting is an implementation of gradient boosting machines created by Tianqi Chen and now is part of a wider collection of open-source libraries developed by the Distributed Machine Learning Community (DMLC). XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Specifically, it was engineered to exploit every bit of memory and hardware resources for tree boosting algorithms. The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters. The algorithm was developed to efficiently reduce computing time and allocate an optimal usage of memory resources. Important features of implementation include handling of missing values (Sparse Aware), Block Structure to support parallelization in tree construction and the ability to fit and boost on new data added to a trained model

5.2 Hyper-parameter Tuning using Grid Search

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc. The grid search provided by GridSearchCV exhaustively generates candidates from a grid of parameter values specified with the param grid parameter. For instance, the following param grid:

$$['C' : [1, 10, 100, 1000], 'kernel' : ['linear'],$$

$$'C' : [1, 10, 100, 1000], 'gamma' : [0.001, 0.0001], 'kernel' : ['rbf'],]$$

specifies that two grids should be explored: one with a linear kernel and C values in [1, 10, 100, 1000], and the second one with an RBF kernel, and the cross-product of C values ranging in [1, 10, 100, 1000] and gamma values in [0.001, 0.0001]. The GridSearchCV instance implements the usual estimator API: when "fitting" it on a dataset all the possible combinations of parameter values are evaluated, and the best combination is retained.

5.3 Polynomial Features

In an attempt to improve the performance of the modes, we used an approach to describe the relationship between dependent and independent variables more sophisticatedly. For this, we transformed the data by using polynomial features. The derived feature matrix has columns representing X's (or the independent variables), x squares, and x cubes. This idea of improving a model not by changing the model, but by transforming the inputs, is fundamental to many of the more powerful machine learning methods.

5.4 PCA for Dimensionality Reduction

Principal component analysis is a fast and flexible unsupervised method for dimensionality reduction in data. Its behavior is easiest to visualize by looking at a two-dimensional dataset. Rather than attempting to predict the y values from the x values, the unsupervised learning problem attempts to learn about the relationship between the x and y values. In principal component analysis, this relationship is quantified by finding a list of the principal axes in the data and using those axes to describe the dataset. The vectors represent the principal axes of the data, and the length of the vector is an indication of how "important" that axis is in describing the distribution of the data—more precisely, it is a measure of the variance of the data when projected onto that axis. The projection of each data point onto the principal axes are the "principal components" of the data. Using PCA for dimensionality reduction involves zeroing out one or more of the smallest principal components, resulting in a lower-dimensional projection of the data that preserves the maximal data variance. For our dataset, we observed that the accuracy and roc score of models like SVM, knn and Gaussian Naive

Bayes improved considerably. While for every other algorithm, it either got reduced by 1-2 percent or remained unchanged.

6 Results and Evaluation

To evaluate the performance of the models, following techniques were used.

- Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Accuracy is a great measure only for a symmetric dataset where values of false positive and false negatives are almost same. Therefore, other parameters have to be looked at to evaluate the performance of the model.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

where

TP = True Positive: A true positive is an outcome where the model correctly predicts the positive class.
TN = True Negative: A true negative is an outcome where the model correctly predicts the negative class.
FP = False Positive: A false positive is an outcome where the model incorrectly predicts the positive class.
FN = False Negative: A false negative is an outcome where the model incorrectly predicts the negative class.

- Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP}$$

- Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$Recall = \frac{TP}{TP + FN}$$

- F1 Score: The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F1Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

This score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, especially if there is an uneven class distribution.

- ROC curve: The ROC curve is a fundamental tool for diagnostic test evaluation.
In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two target groups. An area of 1 represents a perfect test; an area of 0.5 represents a failed test. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test

The table of scores obtained from each evaluation technique is as listed below

Evaluation Scores					
Models	Accuracy Score	ROC Score	Precision	Recall	F1 Score
Logistic Regression	0.665	0.665	0.67	0.65	0.66
K-Nearest Neighbors	0.554	0.554	0.52	0.90	0.46
Gaussian Naive Bayes	0.536	0.534	0.52	0.90	0.46
Adaboost	0.676	0.676	0.68	0.67	0.67
Random Forest	0.671	0.671	0.68	0.65	0.67
Support Vector Machine	0.666	0.666	0.67	0.65	0.66
XGBoost	0.685	0.686	0.68	0.68	0.68
LightGBM	0.748	0.753	0.69	0.68	0.68

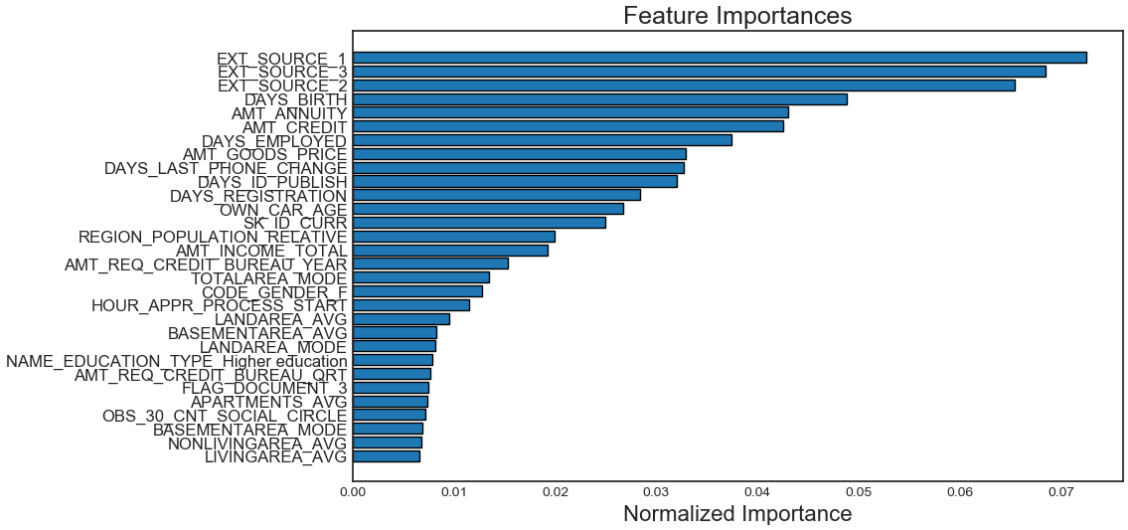

Figure 10: Feature Importance

7 Discussion

We observed that the LightGBM performs outstandingly as compared to all the other models.
In general, it was observed that the tree based models are performing better on this dataset.

Consider if a finance provider wishes to evaluate the risk of loan repayment ability from customer's side, the company would want to ensure that the clients are able to repay the loan. Here, Precision and Recall values on the operation data come into picture. A high precision value implies that the model returned substantially more relevant results than the irrelevant ones whereas a high recall value implies that the model returned most of the relevant results. Hence, the company would like to have a model with higher recall value rather than the precision.
Meanwhile, the precision cannot be below a threshold to make sure the Home credit will not take every loaner as a person who can't have a consistent repay capability since a precision can indicate a large number of false positives.

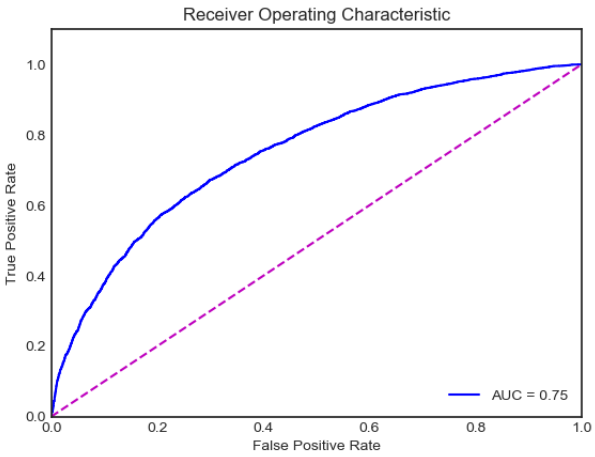

Figure 11: ROC Curve for LightGBM

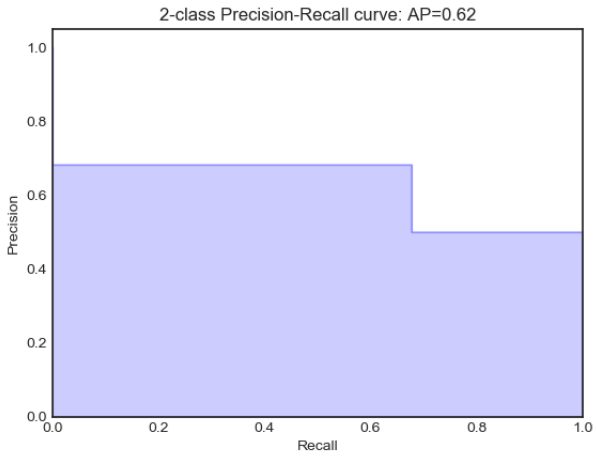

Figure 12: Precision-Recall Curve for LightGBM

8 Future Work

For the pre-processing part, simple oversampling and under-sampling methods were implemented by replicating datasets. On the other hand, there are more advanced data preprocessing methods which would be interesting to explore such as re-sampling skewed data by the scale of client's profile or other useful information available from the data.
Perform manual feature engineering, make polynomial features i.e. while two variables by themselves may not have a strong influence on the target, combining them into a single interaction variable might show a relationship with the target and perform predictions based on them.
Moreover, collect additional data about our features in order to train the model better.
In terms of improving the accuracy for the prediction , stacking/blending could be implemented.

9 Conclusion

The most influential features to indicate client's repayment ability were discovered from random forest feature selection method. Among the most relevant features includes the FICO score which is client's credit scores most lenders use to determine your credit risk from Experian, TransUnion and Equifax. The next essential feature is the day of birth , how many days has client's been employed and the education type such as secondary or higher education. Base on the typical loan repayment assessment, these are essential criteria to examine client's repayment eligibility. The higher FICO score history, the less credit risk a client would have. Especially focus on the credit report from Experian and TransUnion which gives the highest importance among the features. Moreover, with the higher education background , older age and stable employment, clients tend to have solid loan repayment on schedule and obtain financing in the future. In conclusion, we recommend Home Credit to evaluate the loan application considerably by reviewing the above features from client.

10 References

- Srivastava, Tavish. "Introduction to KNN, K-Nearest Neighbors : Simplified." Analytics Vidhya,27 Mar. 2018, www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.
- Patel, Savan. "Chapter 2 : SVM (Support Vector Machine) - Theory – Machine Learning 101 – Medium." Medium, Augmenting Humanity, 3 May 2017, medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.
- "Missing Values in Data." Statistics Solutions, www.statisticssolutions.com/missing-values-in-data/.
- Chatzidimitriou, Kyriakos. k-Nearest Neighbors, ml-tutorials.kyrcha.info/knn.html.
- Ray, Sunil, and Business Analytics and Intelligence. "Understanding Support Vector Machine Algorithm from Examples (along with Code)." Analytics Vidhya, 14 Sept. 2017, www.analyticsvidhya.com/blog/ support-vector-machine-example-code/.
- Pedregosa, F. "Scikit-Learn." 1.4. Support Vector Machines - Scikit-Learn 0.19.1 Documentation, 2011, scikit-learn.org/stable/index.html.
- Jain, Aarshay. "A Complete Tutorial on Ridge and Lasso Regression in Python." Analytics Vidhya, 17 May 2018, www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/.
- VanderPlas, Jake. "Feature Engineering." Introducing Scikit-Learn | Python Data Science Handbook, jakevdp.github.io/PythonDataScienceHandbook/05.04-feature-engineering.html.
- "Home Credit Default Risk | Kaggle." Countries of the World | Kaggle, www.kaggle.com/c/home-credit-default-risk.
- Microsoft. "Microsoft/LightGBM." GitHub, 12 Aug. 2018, github.com/Microsoft/LightGBM.