Home Credit Default Risk Prediction

## Tanishk Parihar, Sarthak Tandon, Pranav Kottoli Radhakrishna

### School of Information Studies, Syracuse University

#### Syracuse, NY, USA, {tparihar, stando01, pkottoli}@syr.edu

<https://ist707-project.herokuapp.com/>

## Abstract

Evaluating and predicting the repayment ability of the borrowers is important for the banks to minimize the risk of loan payment default. The major goal of the project is to find a more robust way to assess whether a candidate will default on a loan using the data provides by the Home Credit Organization. We performed analysis using 3 different models and multiple layers of data. We conclude by determining how complex the data should be and which is the best model to use for prediction, along highlighting the challenges and future work.

# Introduction

Machine learning is a subset of artificial intelligence which involves the study of computer algorithms which learn from examples. It has a wide variety of applications ranging from recommender systems and natural language processing to fraud detection. Machine Learning algorithms can be broadly classified into supervised and unsupervised learning algorithms.

Typically, supervised learning algorithms learn a function that maps an input to an output based on examples. These algorithms build a mathematical model based on a sample of data called training data. These models are subsequently used to make predictions on new datasets which the model has never seen before. The training data used to build these models has to be labelled. This data can be thought of as a set of examples where each example is a pair consisting of an input object and a desired output value.

The label of the training data can either be continuous or categorical in nature. If the label is continuous, it is considered a regression problem. If the label is categorical, the problem is one of classification and it requires classification algorithms to solve.

In this report, we will apply classification-based machine learning algorithms to a loan default risk dataset.

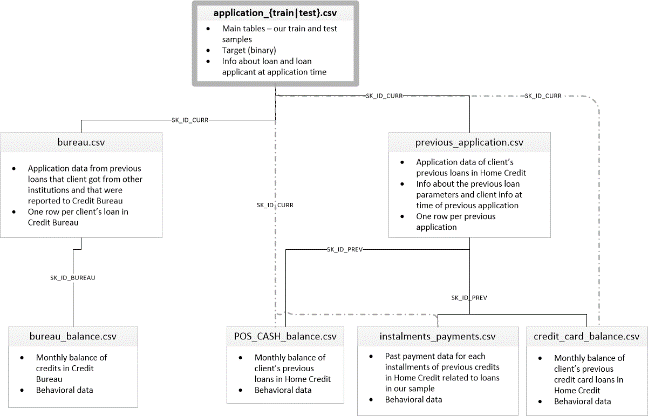
# The Finance Domain

For our experiments, we chose to work on predicting if an applicant can repay the loan or not. A loan is the lending of money by one or more individuals, organizations, or other entities to other individuals, organizations etc and a recipient (i.e., the borrower) incurs a debt who is usually liable to pay interest on that debt until it is repaid as well as to repay the principal amount borrowed. The major goal of the project is to find a more robust way to assess whether a recipient will default on a loan or not.

The dataset we used is from Kaggle by an organization called Home Credit. The data set consist of multiple files “application\_train” – main training, “bureau” & “bureau balance” – credit related data of the client, “previous\_application” – previous loan applications data, “credit\_card\_balance” & “installment\_payments” – loan and credit card payment data. We also did our analysis in three steps, starting with just the main data and later adding other tables. The target variable was binary – 0 if the applicant will not default in loan repayment and 1 otherwise. Predictive models like Logistic Regression, Ensemble method and Gradient Boosted Models were used to make our prediction. The dataset can be found online at https://www.kaggle.com/c/home-credit-default-risk/overview

# Data Processing

We executed each of the machine learning models on three versions of data and compared the results. The first version consisted of only the application train file, the second version consisted of application train, bureau and bureau balance and the final version consisted of all the files mentioned above.



Version 1

The application\_train dataset that we downloaded from Kaggle had issues like large number of missing values, anomalies and unsuitable form of data that required transformation. We performed various data preprocessing steps to prepare our dataset for the modeling process. First, we removed all the columns in which more than 50% of the values was missing. We did that because it would be impractical to impute such large number of missing values. In some columns, after using exploratory data analysis, we found that the mean and median values can be a good replacement for the missing values. Also, we plotted correlation matrix to check for multicollinearity problem and dropped those columns which had this issue. The number of days recorded were negative as they were relative to the current loan application. For such columns, we converted the days into number of years. Finally, we made use of mice imputation for filling the rest of missing values.

Version 2

First, we performed exploratory data analysis on each of the 3 files. We found several missing values which were imputed using Multiple Imputation by Chained Equations (MICE). We used one hot encoding to convert categorical data to numeric data.

Once the data cleaning had been dealt with, the datasets had to be merged. The ‘application\_train’ dataset contains a primary key namely ‘SK\_ID\_CURR’ which is present as a foreign key in ‘bureau’. Similarly, ‘bureau’ contains a primary key ‘SK\_ID\_BUREAU’ which is present as a foreign key in ‘bureau\_balance’. The first step was to aggregate the ‘bureau\_balance’ dataset on the ‘SK\_ID\_BUREAU’ attribute. We calculate the minimum, maximum, mean and sum of our attributes. This process was repeated on the bureau dataset, where the ‘SK\_ID\_CURR’ attribute was used to aggregate the data. The final step was to merge all 3 datasets based on the primary keys.

The merged dataset contained some missing values due to the absence of some primary key values in the child dataset. Attributes with greater than fifty percent missing values were dropped. The remaining attributes were imputed with their medians.

Version 3

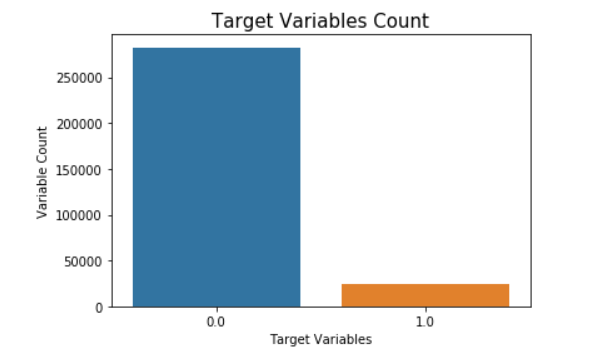
Version 3 build upon the work done by version 1 and 2. After bureau and bureau balance have been added, we will further add instalment related details to the training file. However, as will the case of balance related data, the merging process is not straight forward. We first started aggregating the cash balance data and aggregated the numerical and categorical variable using the functions created for grouping the values using 'SK\_ID\_CURR'. We however wanted to maintain the sequence of the information since it also had the data for the previous instalment and current instalments via the 'SK\_ID\_PREV' & 'SK\_ID\_CURR' keys. So we decided to aggregate twice, starting with 'SK\_ID\_PREV' and the again with 'SK\_ID\_CURR'. We then joined the data using 'SK\_ID\_CURR'. We did the same with the data pertaining to credit card info and previous application.

Now that the data is joined, we must fix the issue of missing values. As in the previous versions, we first removed all columns with more than 50% missing values. After that was done, since were dealing with numerical variables, we imputed the remaining features using median.

Once all the data was joined and values imputed, we now had 1300+ columns. Before running any other model and optimizing the same, we need to know that all the columns are important for training. Running a model with 1300 columns is time consuming a waste of computation as well if most of them are useless. Hence, we fed the model to a Gradient Boosting Model with base hyperparameters. The aim is to get the feature importance of the of all the columns and remove those which have no importance. Through this exercise we were able to get the number of useful columns down to 83.

Under Sampling

During data analysis, we found that the target variables are present in imbalanced proportions.



This is detrimental to model training as the models will be trained better for one target variable than the other, giving good accuracy numbers, but bad figures for Recall and AUC score.

One way to counteract the situation is to under sample the data where the overrepresented class is randomly brough down to same count as the underrepresented class. As a result, the learning models will be trained on equal amount of target variables. One advantage of this is that the model is uniformly trained on both classes. We realized that the model performed the best on the under sampled validation data. However, we realized that once we tested the model on the testing data, the model showcased sever underfitting issues resulting in abysmal metric numbers.

As a result, we decided to add class weights corresponding to the value count of each target variable, where the underrepresented class will have a high penalty compared to the other one.

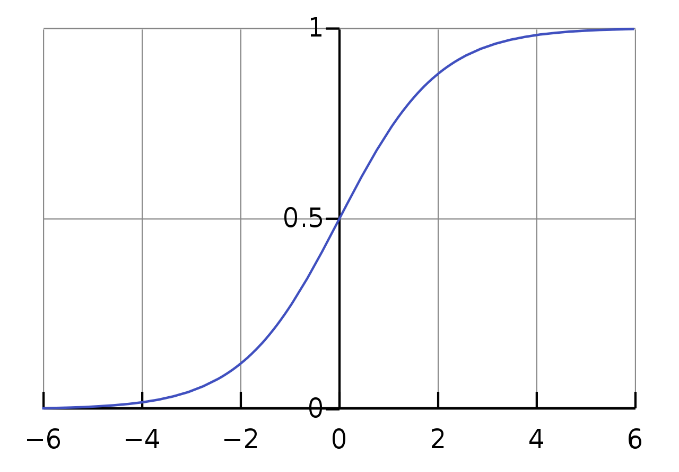
# Machine Learning Models

We applied three different classification algorithms to the dataset: Logistic Regression, Random Forest and Light Gradient Boosting. These approaches have proven to be quite effective in dealing with classification-based machine learning problems in the past.

Logistic Regression

The Logistic Regression is a very popular model that is used for binary class problems. The objective of logistic regression, like linear regression, is to model the mean of the response variable, given a set of predictor variables. However, the response variable of a logistic regression is binary rather than continuous in nature. Logistic function is named after the logit function that is used by the model to perform the binary classification. The logistic function, also known as the sigmoid function, is an s-shaped curve which is used to map the weighted inputs to 0 or 1.





For our base model, we used the logistic regression model from the scikit-learn package with default parameters. We then tuned different hyperparameters to improve the performance of our model.

Random Forest

Random Forest is an ensemble learning method. Ensemble learning is a technique which aims to improve classification accuracy by aggregating the predictions of multiple classifiers. Ensemble methods generate a set of base classifiers and performs a final classification by taking a vote based on the predictions made by each of the base classifiers.

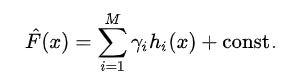
Random Forest is specifically designed for Decision Tree classifiers. It votes on the values generated by a decision tree where each tree is built on an independent set of random vectors. This randomization is done to reduce the correlation between decision trees to improve the generalization error of the random forest.

A random subsection of attributes is selected for each split of each decision tree built for the random forest. Each tree is then allowed to grow to its maximum depth to minimize bias. It should be noted that this can sometimes lead to overfitting due to increased variance. Once all the trees have been built, a prediction can be made using majority voting.

To deal with the class imbalance problem, we tune the weights assigned to each class to help us maximize our recall.

Gradient Boosting

Gradient Boosting is another example of Ensemble Learning. It is a sequential algorithm that keeps adding predictors to the ensemble model with each one correcting its predecessor. The difference lies in the fact that the model tried to fit the new predictors to the residual errors made by the previous predictor.



The variation we used for our project was Light Gradient Boosting. Why Light Gradient Boosting is different is that while other model grows trees horizontally(level-wise), it grows it trees vertically(leaf-wise). In order to grow, the model chooses the leaf with max change in loss. The primary benefit of the Light Gradient Boosting is the changes to the training algorithm that make the process dramatically faster, and in many cases, result in a more effective model.

# Model Evaluation

After execution all the models were evaluated and compared.

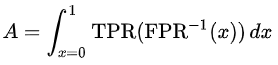
Evaluation Metric

If case of determining if an individual will default in a loan or not, we need to keep the following assumption in mind:

We need to reduce the false negatives as we need to reduce the chances of incorrectly predicting if a recipient will default on a loan.

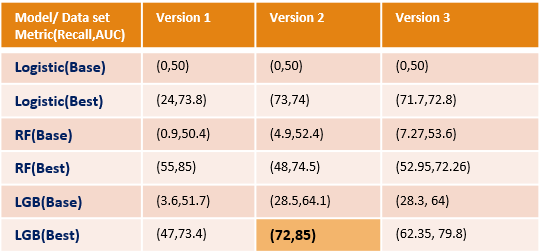
Given that the target variable is imbalanced, we need to make sure that our model can differentiate between the two classes properly.

Based on these factors, we will be using recall and AUC score as the metrics to determine the best model.



Test Results

The Test results for our models are as follows:



Out of the 18 models we ran, the best results we received was using Light Gradient Boosting, on the 2nd Version of the data, with Recall of 72% and AUC score of 85%. Best Result aside, we can make observe that:

* All base models perform abysmal recall scores. This is because target variable is imbalanced. Another reason is that base model does not adjust class weights. Those models are unable to predict ‘1’ properly Once class weights are adjusted and models are further optimized, we get more stable results.
* The performance of the best models is all over the place as well. We cannot say that Light Gradient Boosting is the go-to model for all cases. In our project we had the flexibility of working with the main dataset as well as the sister data sets. But if business requirement states that we only use the main data or use all the of it, then Random Forest and Logistic Regression perform best respectively. Hence it is always advisable to start with the simple model and move to mode complex ones, eliminating none of them.
* We mentioned under sampling before and we testing our models using the same. However, we got deceptively good results on the under sampled training and testing set. Further testing revealed a clearer picture, which was that the model was underfitting. Hence, we decided not got down that path any further.
* While under sampling was underfitting the data, in case on Version 3, excess data was causing overfitting, which was reflected in the model performances.

Coming back to the best model, these are the hyperparameters which were used to the achieve the needed results:

* n\_estimators=10000
* objective = 'binary',
* learning\_rate = 0.02,
* reg\_alpha = 1.0,
* reg\_lambda = 0.8
* subsample = 0.8

Lastly, we want to see how good our best model is really is. As mentioned in the beginning, were working on the data set for a Kaggle competition. Looking at the leader board, the Best AUC is 81.7%. Does that mean, our model is better than the best model on Kaggle? The comparison is not that direct. For starters, our best score is on the test set of the training data. We did not run our model on the testing data since there no way to measure our score as the competition is over. But if our model’s performance does not sway to extreme ends, our model can still stand among the best submissions.

# Conclusion

After gathering all the results from the models that we tested, we found that Light Gradient Boosting provided the best results. The combination of main files and credit related data yielded the best results. We generated feature importance from the tree-based models to understand the features that helped in predicting if a applicant would default or not. Some features that were common in the top 10 variables of our models were Ext\_Source2, Ext\_Source3, Amount Credit and Days Employed. Although our model performed well in predicting the defaulters from the actual number of defaulters, but still more work needs to be done to further improve its performance. We believe that further improvement will help the organisation to avoid giving loans to defaulters and in this way, they could extend their profit margin.

References:

* https://www.amazon.com/Introduction-Data-Mining-Pang-Ning-Tan/dp/0321321367
* https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c
* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3912194/
* <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>
* <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks>
* Olson, David L.; and Delen, Dursun (2008); Advanced Data Mining Techniques: ISBN 3-540-76916