### **CS 6220 Summer ’18 Project Report - Home Credit Default Risk**

**Author:** *Chia Yi Liaw , Mounica Subramani , Sharyu Deshmukh, Somya Bhargava*

**1. ABSTRACT**

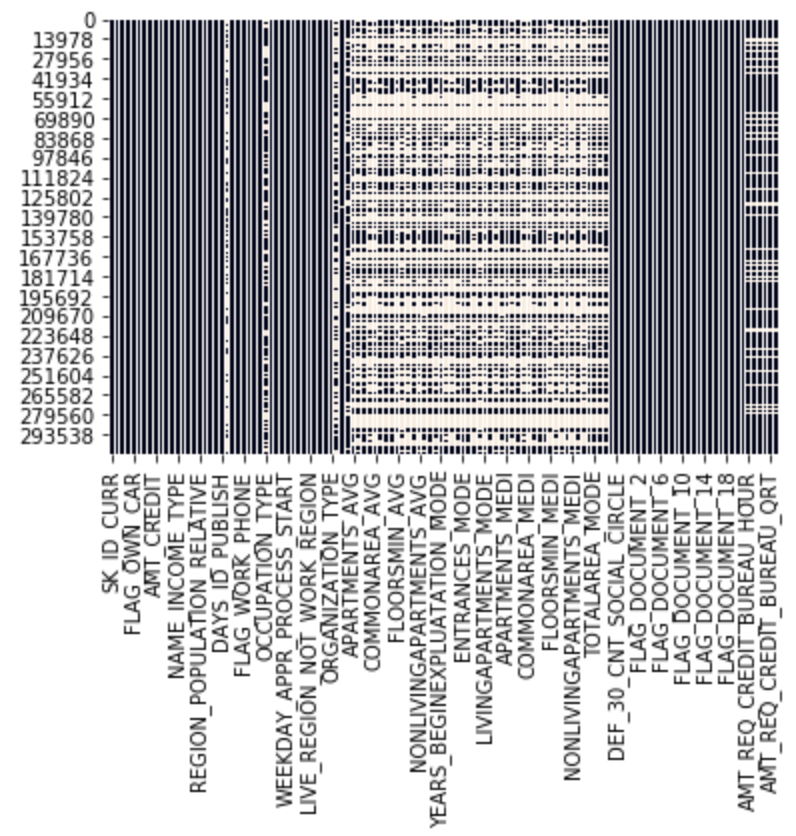
Loan repayment ability is a supervised classification problem. The objective of this project is to use historical loan application data from the Kaggle Home Credit Default Risk dataset to explore effective methods and create classifier to predict either loaner would be able to repay based on statistic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

**2. INTRODUCTION**

**2.1 Project Description**

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.

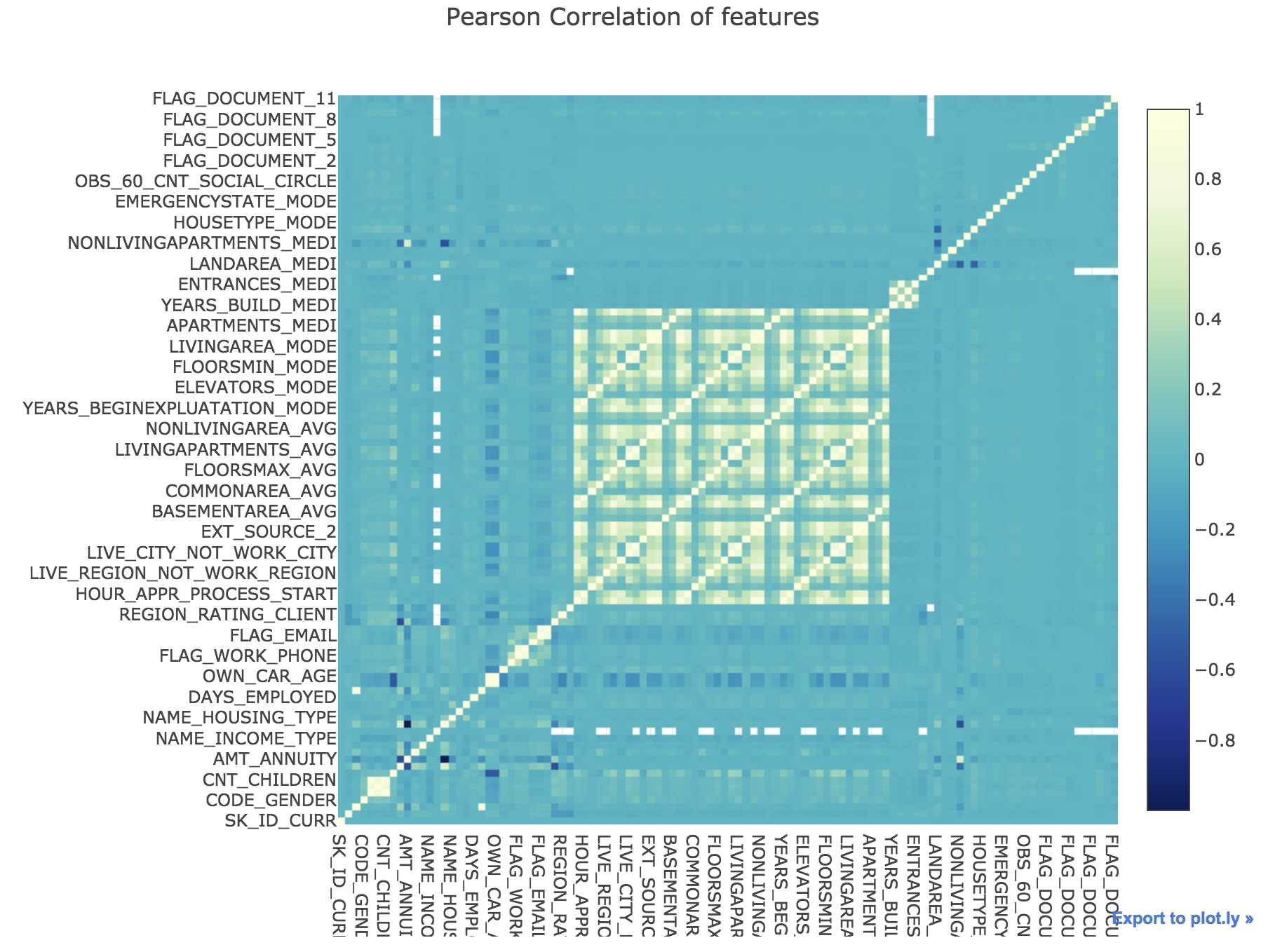
**Supervised**: The labels are included in the training data and the goal is to train a model to learn to predict the labels from the features

**Classification**: The label is a binary variable, 0 (will repay loan on time), 1 (will have difficulty repaying loan)

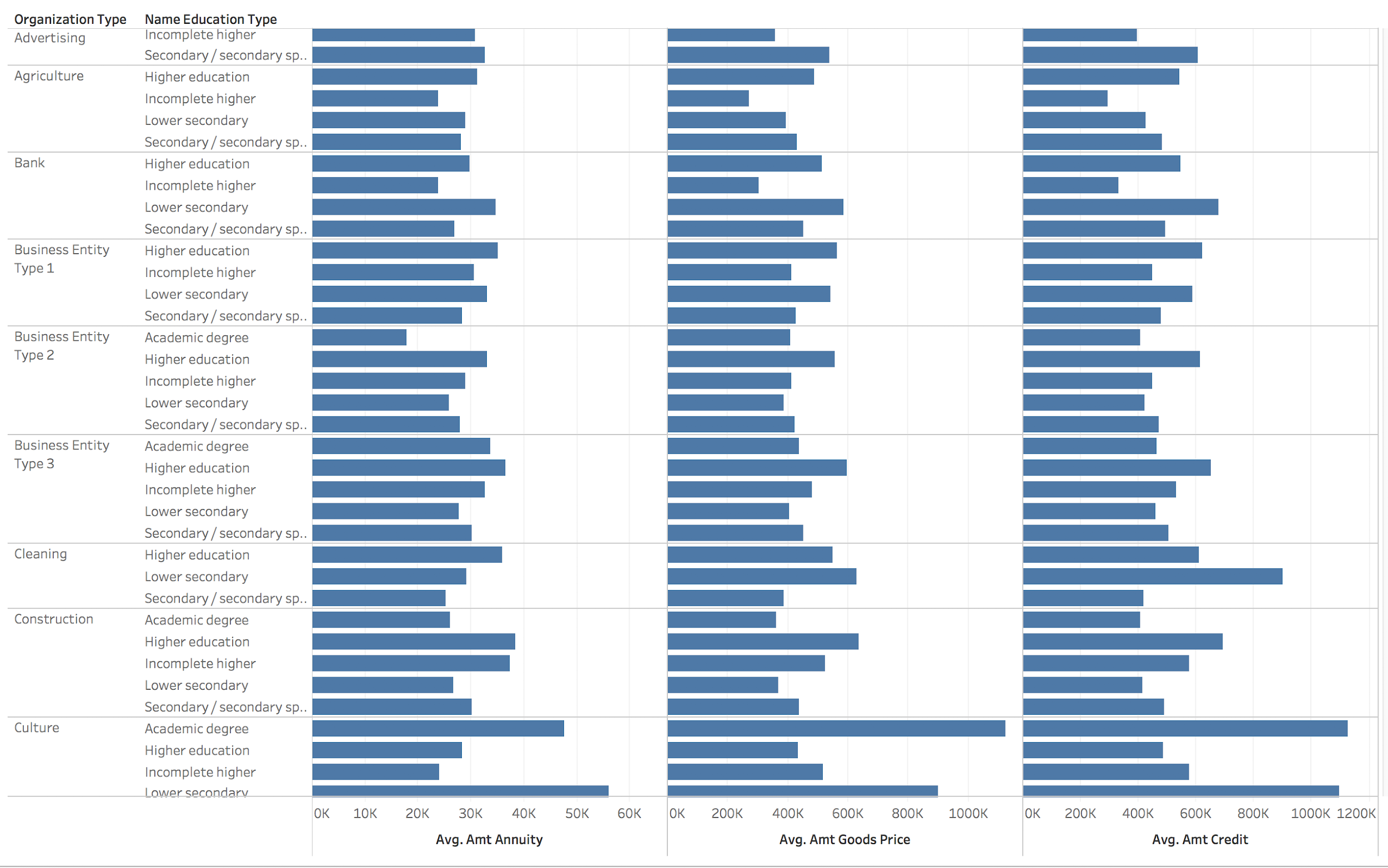
**2.2 . Exploratory Data Analysis**

Missing Value

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% of missing value. Imputation and deletion will be performed in the data pre processing stage,

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 LightGBM 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.

**2.3 Data Preprocessing**

Feature Selection

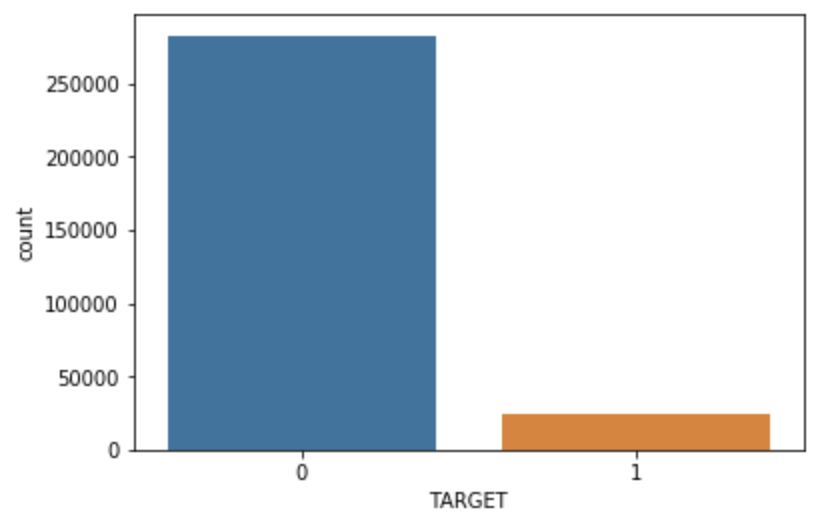
With the high dimensional and to reduce the variance of the model or chances of overfitting. There are 23 features contain more than 60% missing value, hence we decide to remove those features as it’s not suitable for training models. In addition, apply the LightGBM, removes the features that are collinear, low\_importance and zero\_importance, beside LightGBM, also run the random forest to rank the feature and then remove the features with the threshold = 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.

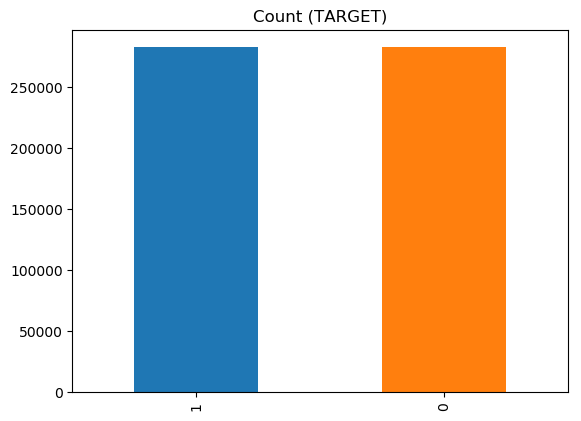
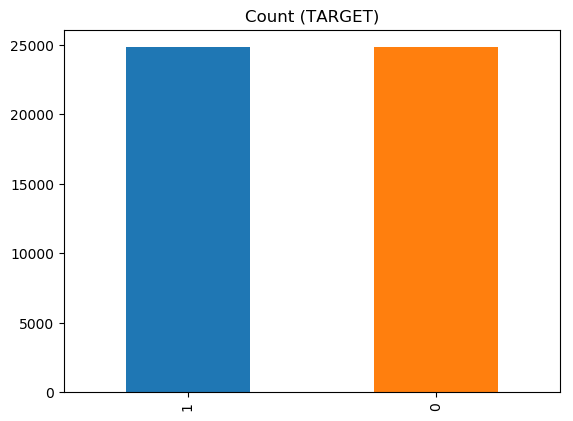
Encoding the Categorical Variable

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 

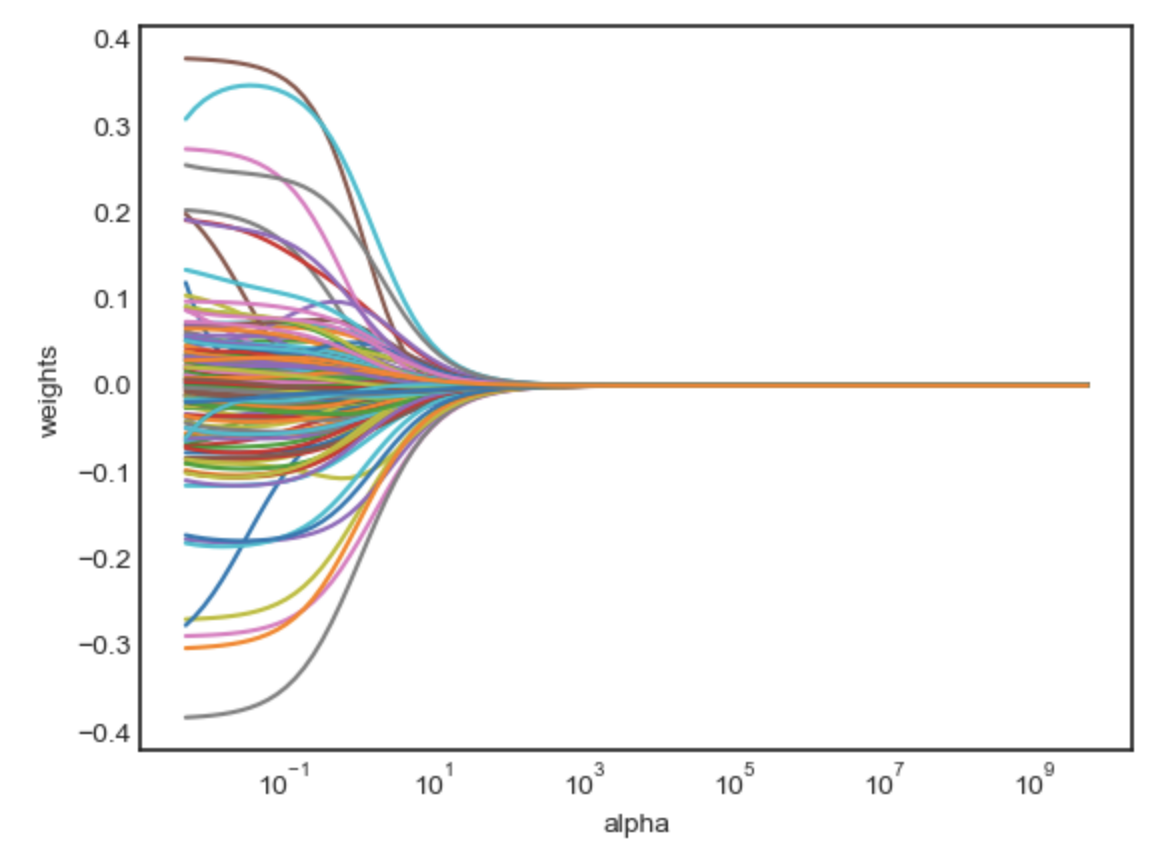
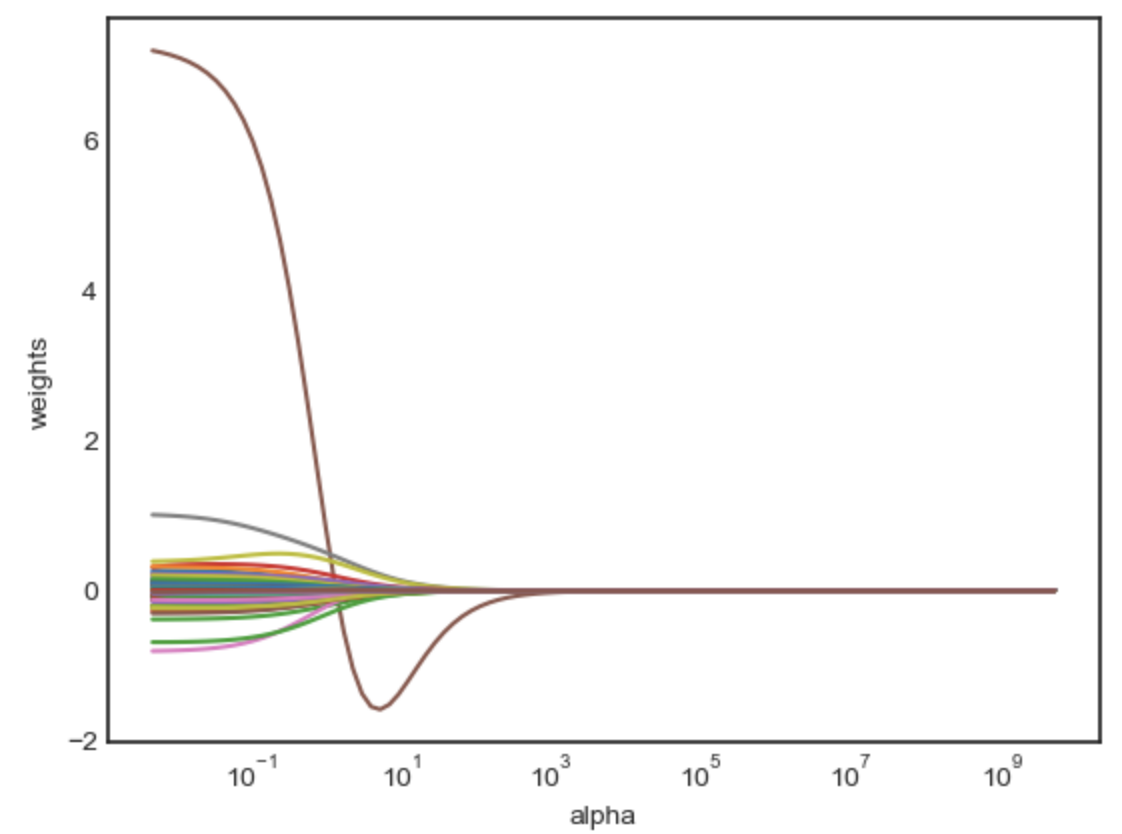
The below histogram show the imbalanced distribution of loan repayment in the dataset. The target variable defines if the loan was repaid by the borrower or not. From the graph, we find out that the data is highly imbalanced. Since, it might lead to incorrect result during modelling process, we needed resampling 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.  *(Distribution of original data)* 

*(After oversampling method) (After undersampling method)*

**3. METHODOLOGY**

**3.1. Modeling**

1. Logistic Regression

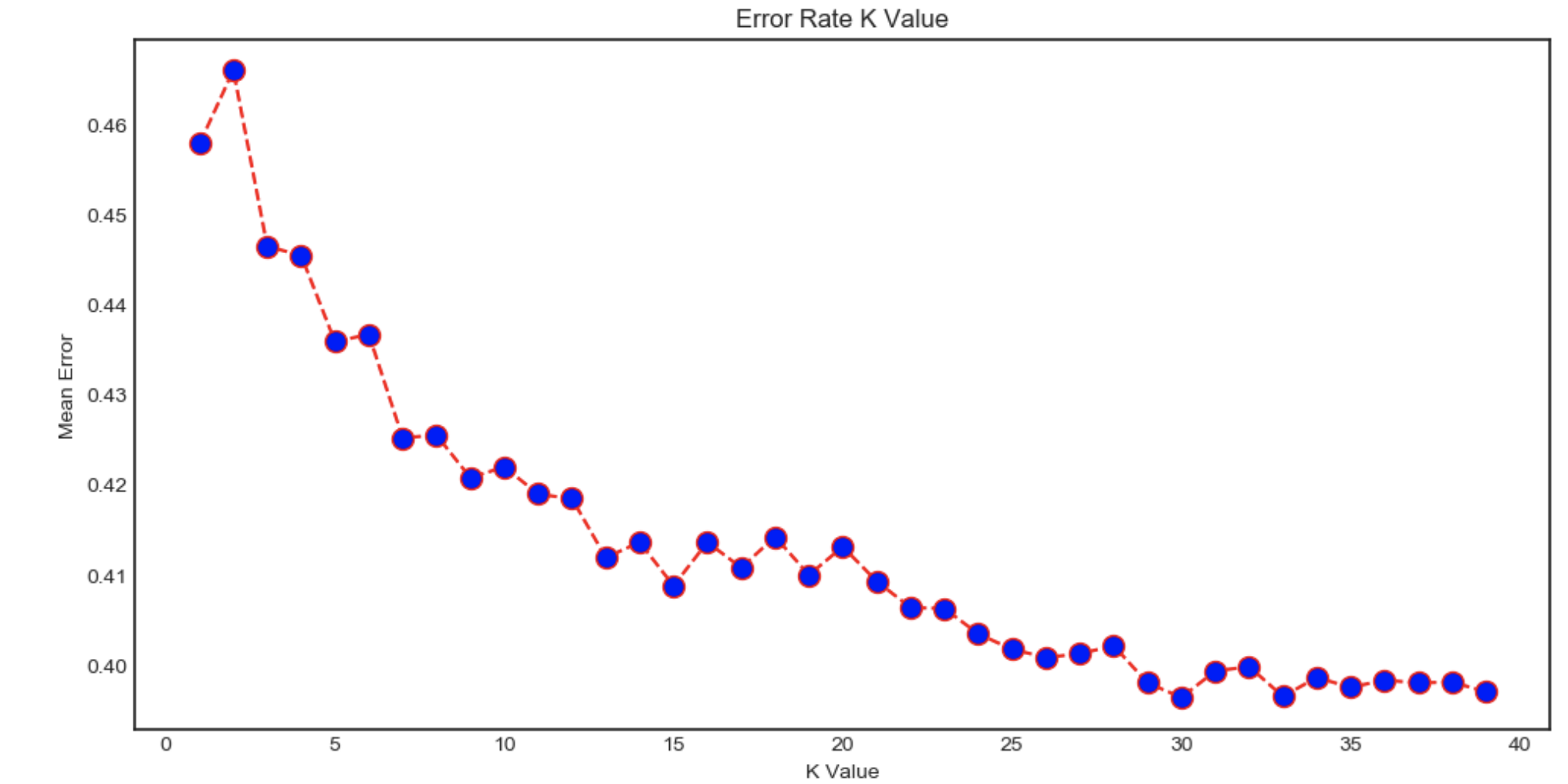
With the dependent variable is dichotomous in the home credit default risk, logistic regression modeling is a proper regression analysis to be applied in the project. The sigmoid function () to simulate events. In addition, when the estimation value is greater than 0.5 which it would classify the prediction as class 1, in contrast, suppose the estimation is less than 0.5, it will be classified to class 0. Ridge Regression perform the L2 regularization which minimize objective = LS Obj + α \* (sum of square of coefficients) and Lasso Regression perform the L1 regularization which minimize objective = LS Obj + α \* (sum of absolute value of coefficients). The below two graphs shows the logistic regression with L1 and L2 penalty. 

L1 Penalty Logistic Regression Patt L2 Penalty Logistic Regression Path

2. Support Vector Machine

The vector support machine (SVM) should perform well in high-order problems, On the other hand, 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 ( )

3. KNN

KNN model is to find the K value. In general, a larger k suppresses the effects of noise, making less distinct classification boundaries. If k is too high, all 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.

4. Gaussian Naive Bayes:

A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It’s 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. 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.

(add formula from the previous report)

**max features = p**

**n feature**

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).

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.

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 wea “learners” into a single strong learner in an iterative fashion.

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function

**3.1. Loss Function** The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported, and you can define your own. For example, regression may use a squared error and classification may use logarithmic loss. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

**3.2. Weak Learner** Decision trees are used as the weak learner in gradient boosting. Specifically, regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak but can still be constructed in a greedy manner.

**3.3. Additive Model** Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss. Generally, this approach is called functional gradient descent or gradient descent with functions.

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.

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.

11. XGBoost:

XGBoost or e**X**treme **G**radient **B**oosting 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

3.2 Hyperparameter 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:

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.

3.3 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.

3.4 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% or remained unchanged.

**4. RESULTS**

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

1. **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 = 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.

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

Precision = TP/TP+FP

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

Recall = TP/TP+FN

**4. 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.

F1 Score = 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.

**5. 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.

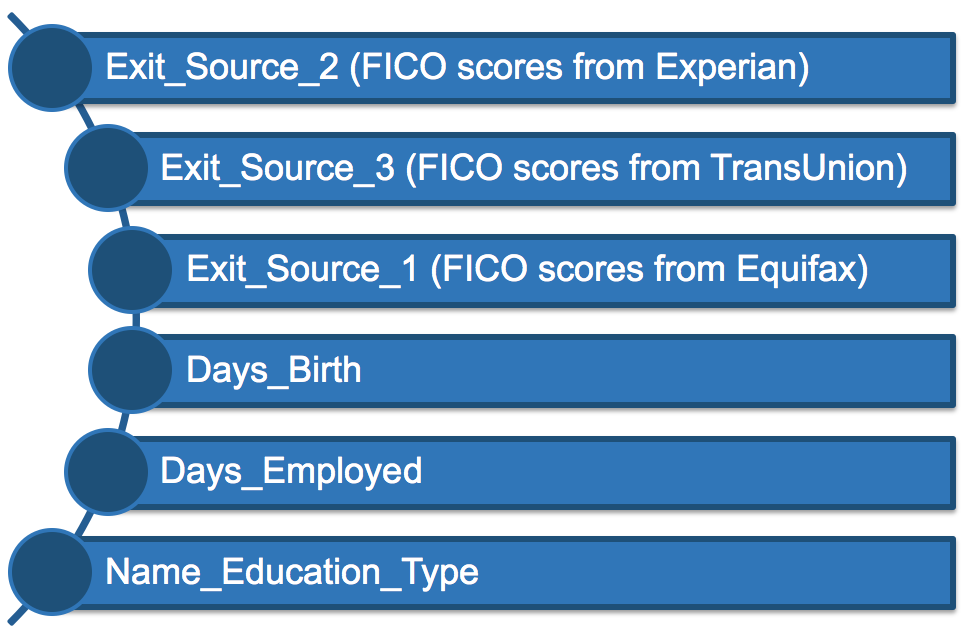
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **Accuracy Score** | **ROC Score** | **Precision** | **Recall** | **F1 Score** |
| Logistic Regression | 0.665 | 0.665 | 0.67 | 0.65 | 0.66 |
| K-Nearest Neighbors | 0.554 | 0.554 | 0.52 | 0.90 | 0.46 |
| Gaussian Naive Bayes | 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 | 0.682 | 0.682 | 0.682 |
| LightGBM | 0.748 | 0.749 | 0.69 | 0.68 | 0.68 |

**5. DISCUSSION**

We observed that 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 a finance provider wish to evaluate the risk of loan repayment ability. Company would like to have a model with higher recall value rather than the Precision. Because company wants to make sure the clients are able to repay the loan based on given operation data. 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

**6. FUTURE WORK**

For the pre-processing part, simple oversampling and undersampling methods were implemented by replicating datasets. On the other hand, there are more advanced data preprocessed method such resampling skewed data by the scale of client’s profile or other useful information available from the data. Perform manual feature engineering, make polynomial features and perform predictions based on them. Moverover, collect more data about our features in order to train the model better. In terms of improving the accuracy for the prediction , stacking/blending could be implied.

**7. 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.

**8. 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/](http://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/2017/09/understaing-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/](http://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](http://www.kaggle.com/c/home-credit-default-risk).
* Microsoft. “Microsoft/LightGBM.” *GitHub*, 12 Aug. 2018, github.com/Microsoft/LightGBM.