# Home Credit Default Risk

Can you predict whether an applicant is capable of repaying a loan?

-Achyut Bhandiwad, Suraj Vadvadgi

## Introduction:

Paying off unexpected expenses, building a home, purchasing vehicle etc. would cost. So people tend to take a money from the bank but they struggle to get loans due to insufficient or non-existent credit history. Companies such as Home credit helps people by providing a positive and safe loan experience with little or no credit history. And these lenders are exposed to default risk if the individuals are unable to return their loans on companies' liability. So the company Home Credit makes use of variety of data like telco and transactional information to predict their client's repayment abilities in order to approve the applications of deserved candidates. Therefore we will be using various statistical and machine learning techniques to make these predictions.

## Problem description:

The objective of this project is to predict whether or not an applicant will be able to repay a loan using historical data of user requesting a loan. This is a standard supervised classification task because the training data (applicant's previous history) contains labels which helps the model to predict desired output from its attributes. And the response variable is binary value 0(repay loan on time), 1(difficulty in repaying the loan). The aim is to preprocess the applicant's data and to use supervised learning methods to predict the target output.

## Related work:

While Home credit is currently using various statistics and machine learning algorithms to make these predictions, we decided to try with various other methods to improve prediction accuracy and unlock the full potential of datasets provided. So we will be combining multiple learning algorithms (Ensemble techniques such as XGB or LightGBM) to obtain better performance than a single/weak classifier alone.

## Dataset description:

The datasets are provided by Home Credit Group. There are 8 different data files:

- application_{train|test}.csv: This is the main table containing training and testing data samples of each applicants. Information about loan and loan applicants at the time of application.

  The training data has 307511 observations (each one a separate loan) and 122 features (variables) including the TARGET (the label we want to predict 0-credit repaid, 1-loans which weren't repaid on time).

- Bureau.csv: Credit Bureau keeps tracks of all client's previous application data in any financial institutions. There may be multiple rows depending upon the users previous credits.
- Bureau_balance.csv: Monthly balances of previous credits in Credit Bureau. It has one row for each month of history of every previous credit. Behavioral data.
- POS_CASH_balance.csv: Monthly balances of client's previous POS (point of sales) and cash loans at Home Credit.
- credit_card_balance.csv: Monthly balance of previous credit card loans that the applicant has with Home Credit.
- previous_application.csv: Application data of client's previous loans in Home credit. Information about the previous loan parameters and client info at the time of previous application.
- installments_payments.csv: Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample. One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.
- HomeCredit_columns_description.csv: This file contains descriptions for the columns in the various data files.
  In this project we will just stick to the main table application training and testing data to work with.

## Preprocessing techniques:

The Data preprocessing step was the most challenging and important part of any machine learning project, since the data collected from various sources will be in raw format which has to be formatted/cleaned for our analysis/modelling. We started of this process with Exploratory Data Analysis (an approach to analyze the datasets to find relations, anomalies, patterns, main characteristics using visual methods). The aim of EDA is to extract useful features within the data to maximize the accuracy of our model. The processing steps are as follow:

- The training dataset has 122 columns (containing 65 float values, 41 integers and 16 categorical values). So to deal with categorical values we either use Label Encoding(holds good for less than 2 categories in a column) provided by Scikit-Learn or one Hot Encoding(categories >2) using get.dummies() function.
- The Encoding techniques used in previous step is only for training data, so in order to make the extra columns reflect in testing data we apply align() to dataframe.
- The outliers/anomalies were removed from datasets.
- We used Feature Engineering technique i.e, domain knowledge of the data to create useful features in order to increase the predictive power of ML algorithms.
- The Principal Component Analysis was made to remove highly correlated data correlation matrix to reduce data columns.

## Proposed solution and methods:

Instead of using a single model to predict what if we use multiple techniques to have better accuracy. Here comes the Ensemble techniques, which combines several machine learning algorithms to decrease the variance (Bagging) and bias (Boosting). These methods are grouped either sequentially (ex Adaboost) or parallel (ex Random Forest) to improve the performance. XGBoost a famous algorithm implementation because of its execution speed and model performance had been recently dominating in all machine learning competitions. And final we ended up using Light GBM (fast, distributed, high-performance). It is tree based learning algorithm in which the tree grows leaf wise (vertically) where as in other algorithms it grow level wise (horizontally). And found Light GBM performed superiorly compared to other methods.

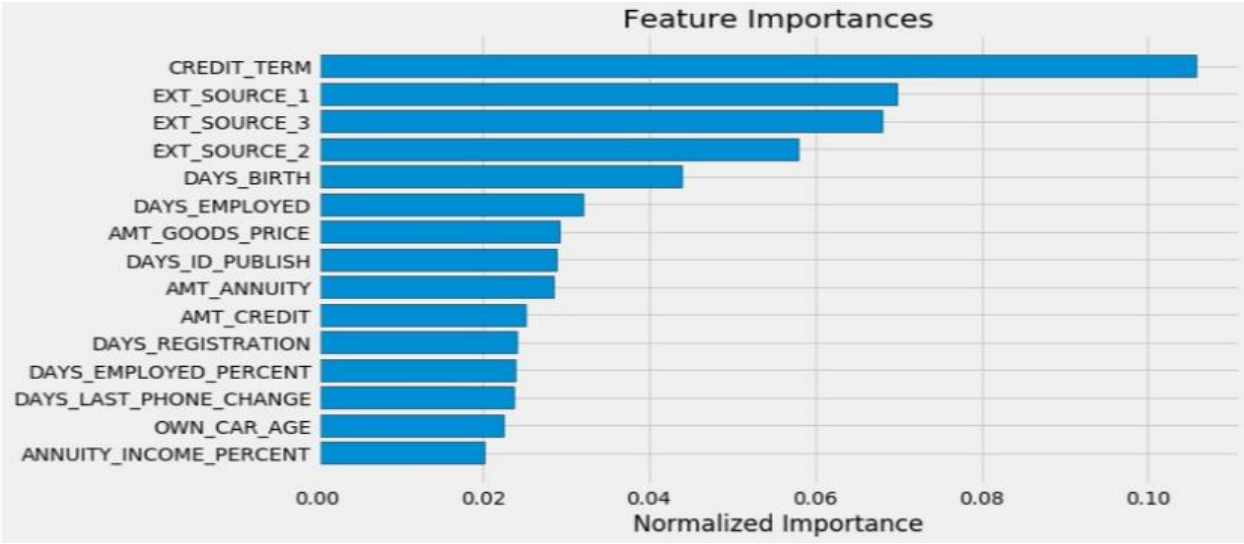
The features of Light GBM are:

- More efficient and trains the model very fast.
- Less usage of Memory.
- Increased accuracy
- Parallelism and GPU learning is supported.
- Stable over huge amount of data

## Experimental Results and Analysis:

We started implementing this problem statement in python language since many inbuilt library functions are available handy and online sources like googlecolab, kagglenotebook helps in speeding up the execution time. The average run time of this code was approximately 30mins.

After selecting programming language the next task was feature extraction where we spent 60% of our time to do this since as we know the bad/poor quality data(may lead to overfitting high variance, high bias) will hurt the performance of model with unexpected results. The important explanatory variables in our data were:



In Machine Learning, there's something called the "No Free Lunch" theorem. It states that there is no such thing as the best algorithm. Some algorithm works good and some not depending upon the data sets and it's especially relevant for supervised learning (predictive modeling). If your training set is small, high bais/low variance classifiers such as Naïve Bayes has best accuracy over the low bais/high variance classifiers like KNN. And accuracy of classifiers depends on many other factors. So you can't say which algorithm perform better always because it depends upon various factors depending upon your dataset. We started our modeling with various methods like Decision Tree, Neural Networks, Support Vector Machines, Gaussian Naïve Bayes, Logistic Regression, K Nearest Neighbors, Bagging, Random Forest, AdaBoost, Gradient Boosting, XGB classifiers of Scikit-Learn library by tuning their respective parameters to check which classifiers works best.

But our observations says, Ensemble techniques worked well. The ROC Area under the curve values were 0.70572(Random Forest Method) and 0.7622(LightGBM Method). So these to algorithm implementation outperformed the results. There were many model evaluation parameters like precision, recall, F-statistics but we used Accuracy parameter to predict the model adequacy. And we used k-fold cross validation technique to make the model learn training and testing dataset till the best accuracy is meet.

The K-fold cross validation values are:

| K-fold value | Train | Valid |
|---|---|---|
| 0 | 0.810061 | 0.762336 |
| 1 | 0.802217 | 0.765188 |
| 2 | 0.831552 | 0.769092 |
| 3 | 0.806661 | 0.764769 |
| 4 | 0.811583 | 0.763609 |
| Overall | 0.812415 | 0.764973 |

## Conclusion:

In summary, the goal of any machine learning problem is to find a single model which learns the target function correctly. Rather than making one model and hoping it outperforms with that of other methods we aggregate the algorithms to form one final model. So there comes the Ensemble Techniques, which also have gained popularity in recent days, so we have used Light GBM for our model training and testing purpose which gave the accuracy of 0.76.

## Contribution of Team Members:

We are a team of two. One of us worked on preprocessing of data and the other on modeling this data to find the predictions.

**References:**

https://www.kaggle.com/willkoehrsen/start-here-a-gentle-introduction/notebook
https://scikit-learn.org/stable/index.html
https://lightgbm.readthedocs.io/en/latest/