Home Credit Default Risk Classification

TEAM: Titans

Parithimalan A
IMT2018055, Titans
IIIT Bangalore
Bangalore, India
Parithimalan.A@iiitb.org

Nachiappan S K
IMT2018047, Titans
IIIT Bangalore
Bangalore, India
Nachiappan.SK@iiitb.org

Mudit Goyal
IMT2018045, Titans
IIIT Bangalore
Bangalore, India
Mudit.Goyal@iiitb.org

***Abstract*—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. To overcome this problem, we have created a machine learning model using the existing customers data to predict whether a client will repay the loan or not.**

***Index Terms*—Machine Learning, Classification, Home Credit, credit history, prediction, Cross Validation, Feature Engineering, LightGBM, XGBoost, Catboost, Random Forest, Ensemble, Adaboost, SVM, SMOTE**

I. Problem Statement / Business Problem Description

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data–including telco and transactional information–to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so 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.

II. Dataset Description

- **application Train/Test.csv**
  - o This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).
  - o Static data for all applications. One row represents one loan in our data sample.
- **bureau.csv**
  - o All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample).
  - o For every loan in our sample, there are as many rows as the number of credits the client had in the Credit Bureau before the application date.
- **bureau_balance.csv**
  - o Monthly balances of previous credits in the Credit Bureau.
  - o This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.
- **POS_CASH_balance.csv**
  - o Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.
  - o This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.
- **credit_card_balance.csv**
  - o Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.
  - o This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.
- **previous_application.csv**
  - o All previous applications for the home credit loans of clients who have loans in our sample.
  - o There is one row for each previous application related to loans in our data sample.
- **installments_payments.csv**
  - o Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample.
  - o There is a)one two for every payment that was made plus b)one row each for missed payment
  - o 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**
  - o A csv file which contains the descriptions of various data files.

III. Data processing Tasks

A. Visualizations and Inference

The columns in apptrain.csv table can be divided into 3 kinds:

i) Column With Categorical features:
These features are visualised using histogram with values corresponding to TARGE value 0 and 1.
Example 1:

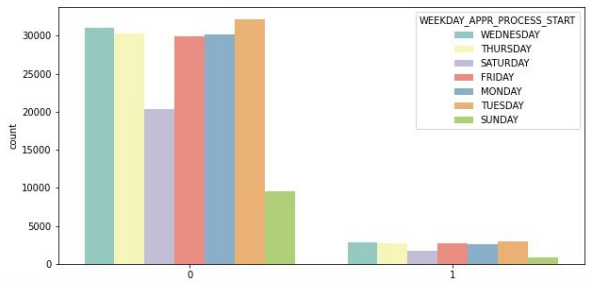

Fig. 1. Day in which application was received vs TARGET

Through this plot we see Most activity happens on weekdays and fewer on weekends. The effect they have on Target value is proportionally the same.
Example 2:

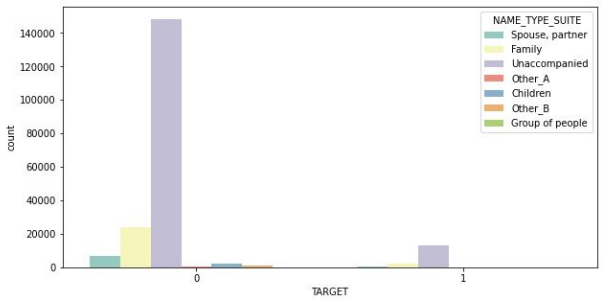

Fig. 2. NAME_TYPE_SUITE vs TARGET

Most features are of this type where the data is skewed towards one category; this is just one example and most features are like this.
So with this in mind we have decided to use Label encoding for these features as with one hot encoding this will simply increase the columns with redundant 0s. Another reason is that we can simply convert this as a 2 category feature with 1 feature as the most occurring one and the other with the rest. This is automatically done when fixing outliers.
Example 3:
This is another kind of example where there are only 2 categories. All categorical features come under these 3

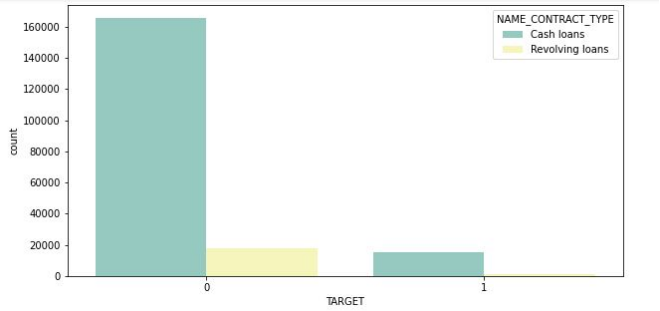

Fig. 3. NAME_CONTRACT_TYPE vs TARGET

kinds of examples and their visualization is available in the notebook.

ii) Columns with 3 or less distinctive numeric value:
These features are visualised using histogram with values corresponding to TARGET value 0 and 1.
Example 1:

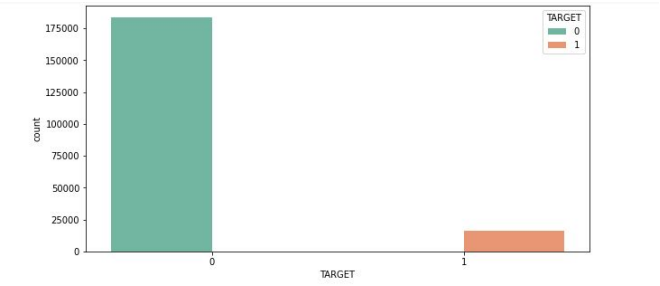

Fig. 4. Histogram of TARGET

This column is the TARGET column which we are going to predict. We can see that this is an unbalanced(skew) set.
Example 2: This a feature with only 1 value which will

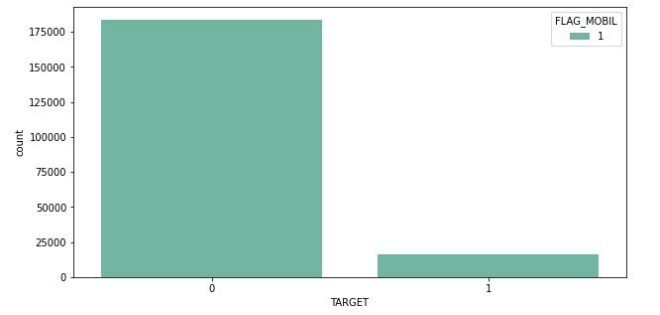

Fig. 5. Histogram of FLAG_MOBIL vs TARGET

not affect the TARGET value, so we have removed this column. All features of this type are removed.
Example 3:
This type of feature has 2 values but the count of one of the values is negligible so we have removed this column. All features of this kind are removed.

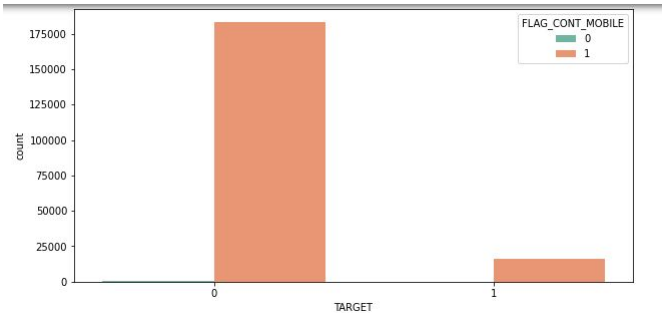

Fig. 6. Histogram of FLAG_CONT_MOBIL vs TARGET

Example 4:

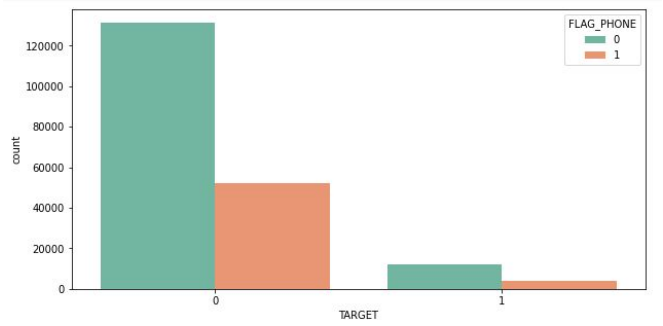

Fig. 7. Histogram of FLAG_PHONE vs TARGET

All other features fall into this example. These features have been used further for the model.

iii) Features with more than 3 distinct values:
These features are visualised using KDE plot. A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram.KDE represents the data using a continuous probability density curve in one or more dimensions.
Example 1:

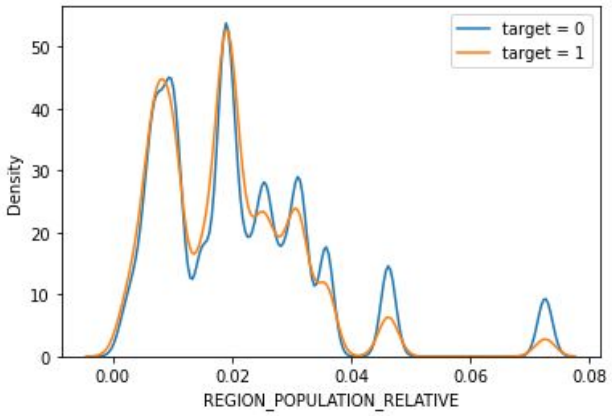

Fig. 8. Density of REGION_POPULATION_RELATIVE

Most features fall into this category where the continuous probability density curve has the same shape for TARGET value 0 and 1.
Example 2:

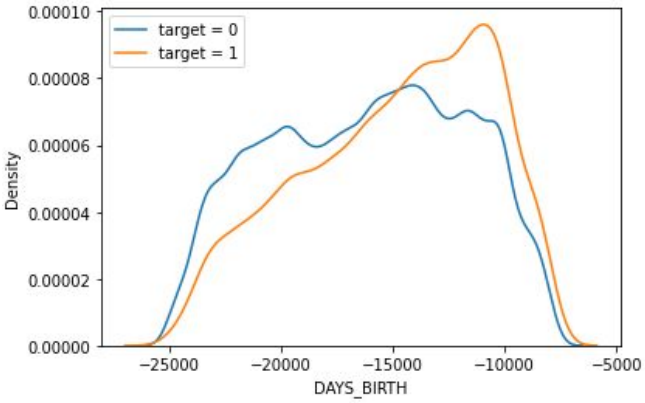

Fig. 9. Density of DAYS_BIRTH

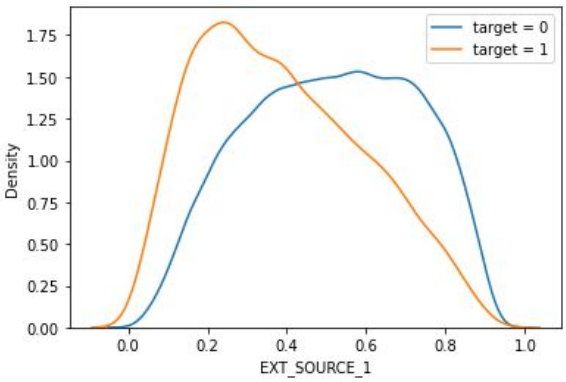

Fig. 10. Density of EXT_SOURCE_1

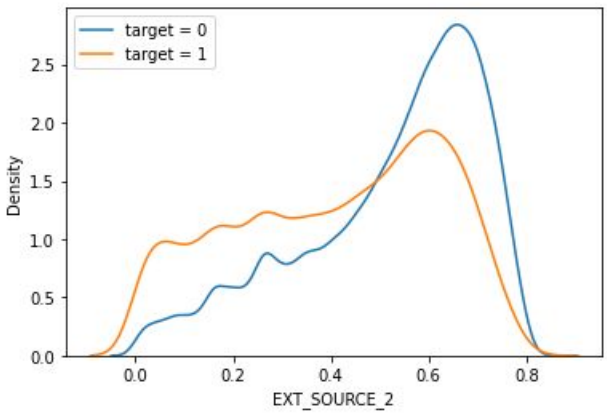

Fig. 11. Density of EXT_SOURCE_2

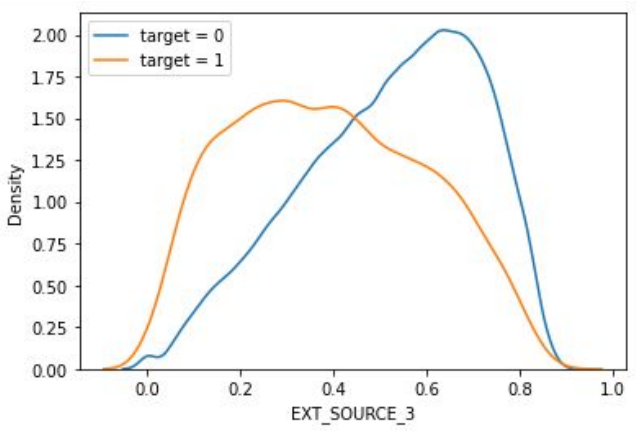

Fig. 12. Density of EXT_SOURCE_3

These 4 features have different continuous probability density curves for TARGET 0 and 1 and we can see the good separation between them. We can also see they act the opposite way from around the midpoint of the graph. By seeing this we can create polynomial features for these 4 to increase their difference and that could contribute to better performance of the model.

B. Data Cleaning & Preprocessing

1) Missing Values: A Generic Function RemoveNaN has been made to find the columns in which there are missing values(i.e NaN) and fill those according to the dictionary(This key has column names as key in which NaN values are present) provided. The Function is made in such a way that it displays the unique values of a column, and updates the dictionary.
Pseudo Code:
RemoveNaN(Dataframe,update_dict):

- o Find the columns/features in which NaN values are present.
- o For each such column find the "Imputation Method" from the dictionary.
- o Now, Replace all the missing values with the Imputation method retrieved

2) Imputations: The same function which identifies the missing values is used to Impute those values also. The Imputation Methods the RemoveNaN function accepts are

a Mean value
b Minimum value
c Maximum value
d Standard Deviation Value
e Most occurring value(Mode)
f Median
g Some constant Value/ String

Any one of the above methods has been chosen for each Column with missing value and the imputation method has been stored in a dictionary.
Now the dictionary along with the dataframe is fed into the RemoveNaN function to get all the missing values imputed.

3) Encoding: The Non-numeric columns in each of the dataframes/tables if the categorical values are in some implicit order and the number of categories is greater than 10 then they are encoded using the label encoder or the one-hot encoding depending on the number of unique categories in the column.
If the number of the unique values in the column are less than 10. It's optionally chosen to one hot encode the column.

4) Computation/Merging Data Sources: The data is spread across multiple tables and has to be merged to a single data table for that to be fed to the Model. Again one more generic function has been made which takes a child dataframe, parent dataframe, key based on which the two frames are to be merged, Method_dictionary for merging.
There can be multiple entries in the child table corresponding to one key in the parent table. All such values have to be aggregated/computed to one value so that it can be merged to the corresponding key in the parent table.
The method dictionary has the column name as keys and aggregation function as values. Our function retrieves the aggregation method from the dictionary and computes a single value which is put in the new column in the parent table.
The non-numeric columns are processed before merging the tables. The various computation methods available are

a It can ignore column
b Mode set of values
c Mean of the values
d Any predefined function

Pseudo Code:
Remove/ Transform non-numeric fields in the child table.
Merge(child df, parent df, key, method):
For each column in child df:

- Retrieve the agg_method from the dictionary
- group the columns value based on the key.
- Apply the function retrieved on the set of values with the same key.
- Merge the child table to the parent table based on it the keys in the parent table.

Return the new merged table/dataframe.

C. Feature Engineering

1) Feature Reduction: We have removed features that

- Are single valued
- Have more than 60 % NaN values
- Highly correlated features
- Have very less correlation with the TARGET (< 0.005)

2) New Features: We have created many new features that might help the model get a better score. We have created a couple of features called `'Penalty_days'` and `'Penalty_amount'` that is the difference between the actual pay date and expected pay date and actual amount paid

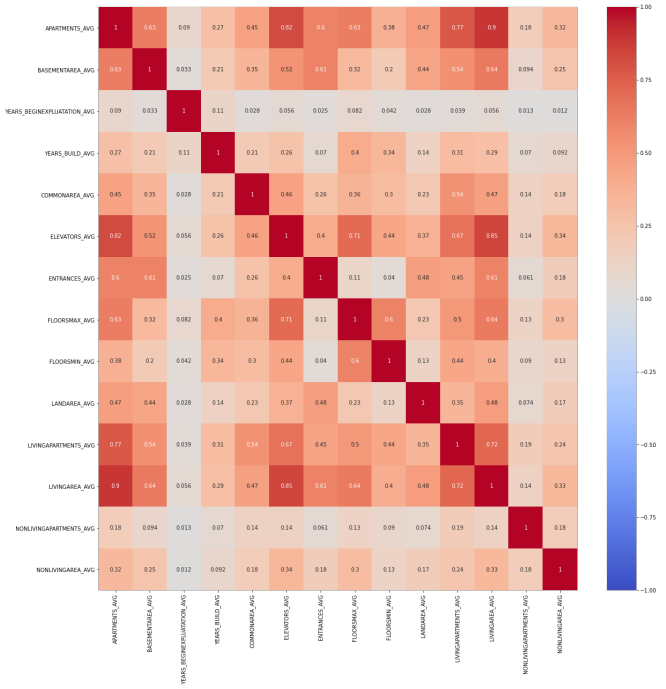

Fig. 13. Correlation Heat Map

and expected amount to be paid. These features were created as someone who has penalties would be less likely to get a good credit score.

We have created more features by taking combinations of features and taking their ratios. The features used were 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY_1', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'AMT_ANNUITY_2', 'AMT_APPLICATION', 'AMT_DOWN_PAYMENT', 'RATE_DOWN_PAYMENT', 'DAYS_CREDIT', 'AMT_CREDIT_SUM', 'AMT_CREDIT_SUM_DEBT'.

This resulted in 132 new features but many of them had inf and NaN values. So after this step we have removed such features and finally ended up with 41 new features.

After Visualisation of some features and seeing their feature importance we have created polynomial features of degrees 2 and 3 for 4 features namely 'EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3', 'DAYS_BIRTH'.

3) Oversampling using SMOTE: SMOTE stands for Synthetic Minority Oversampling Technique and is an oversampling technique used to increase the samples in a minority class. We have used this to increase the data samples of the minority class. This method did not work well.

D. Post-Merge Preprocessing

-> Drop Key columns as the keys are required only to identify the records across the tables.

-> Create New Features.
-> Removing Outliers.
-> Remove highly correlated columns
-> Remove columns that have very less correlation with the target.
-> Normalise values(For quicker training of model)

1) Outliers: We have removed the outliers for features that are not normalised in the data set.

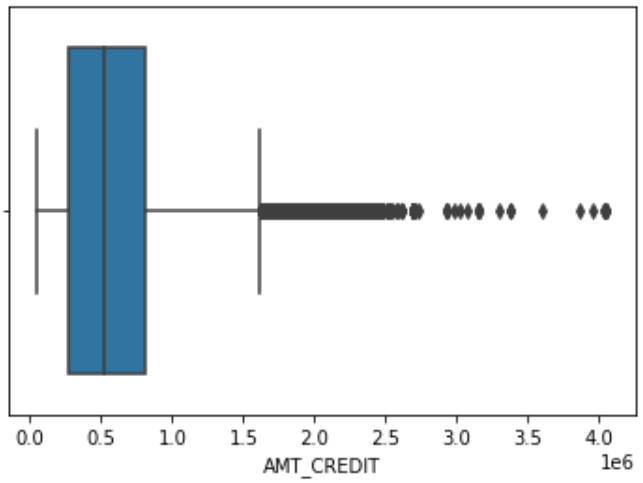

Fig. 14. Example of outliers (outliers of feature AMT_CREDIT)

Also some features have a single value that take up over 50% of the data which would mean any other data would be outlier to it and we will be left with a single unique value only, so we haven't removed outliers here as they may hold key data that helps the model.

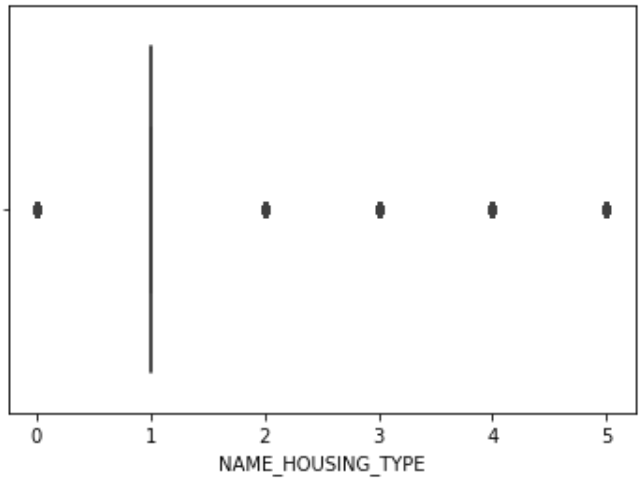

Fig. 15. Figure showing the case of categorical features in which outliers should not be removed

2) Scaling/Normalization: The dataset already consists of some data that are Normalised, so we have scale all the other features.

IV. Model Training

We started with basic models like Logistic Regression, Naive Bayes, Decision Trees which didn't give good scores. We also tried LinearSVM which gave better scores. But, we couldn't use SVM with a polynomial / rbf kernel as we have a very large dataset, it is impractical to use sklearn. We tried SVM with GPU support from another library called thundersvm, but it didn't give good results. Later, we tried ensemble classifiers like Random Forest, Gradient Boost, Adaboost, XGBoost, LightGBM, Catboost which gave a significant increase in our scores. Some models like SVM, Decision Trees give the approximate probabilities, therefore we used CalibratedClassifierCV module using isotonic scaling with cross validation to give calibrated probabilities which gave good results. But after trying it on other models like Catboost, XGboost, Lightgbm also increased the score. We tried combining several models by simple weighted average which increased the score significantly. At last, we tried stacking ensemble (blending) using base estimators as Random Forest, XGBoost, LightGBM, Logistic Regression, Catboost and meta estimator as Logistic Regression but it didn't give a better score.

Here are the models we used for ensembling:

o Catboost

TABLE I
CATBOOST HYPER PARAMETERS

Hyperparameters	Values
border_count	5000
depth	4

o Light Gradient Boosting Machine
Hyper parameters found for Lightgbm was done using Optuna library LightGBMTunerCV.

TABLE II
LIGHT GBM HYPER PARAMETERS

Hyperparameters	Values
feature_pre_filter	False
lambda_l1	0.0007099817067951657
lambda_l2	2.2721620492879536e-07
num_leaves	64
feature_fraction	0.4
bagging_fraction	0.9938087600649544
bagging_freq	1
min_child_samples	20
learning_rate	0.01
n_estimators	1000
class_weight	balanced

o XG Boost
Refer Table III

o Random Forest
We tuned a random forest model and it improved the score individually, but after we ensembled all the models, it gave the best result with default parameters.

TABLE III
XG BOOST HYPER PARAMETERS

Hyperparameters	Values
N_estimators	1000
learning_rate	0.01
scale_Pos_weight	3

V. Results

A. Evaluation Metric

AUC - ROC curve is a performance measurement for classification problems at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much a model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

B. Comparision of different machine learning techniques/Models

TABLE IV
COMPARISION OF DIFFERENT MACHINE LEARNING TECHNIQUES/MODELS

Model	CV Score	Public Score	Private Score
Catboost	0.77697	0.76416	0.77345
LightGBM	0.77585	0.76325	0.77146
XG Boost	0.77570	0.76339	0.76801
Logistic Regression	0.76094	0.75620	0.76476
Random Forest	0.75105	0.74705	0.75811
Ensembled	0.77910	0.78013	0.78711

VI. Conclusion

We would like to conclude that we were able to come up with an efficient model to predict whether the client will repay the loan or not. This project can be helpful for home credit lenders in identifying the trustworthy clients Whose past credit history is not available.

Acknowledgment

We would like to thank Professor G Srinivasa Raghavan, Professor Neelam Sinha as well as all the Teaching Assistants – Tushar Anil Masane, Shreyas Gupta, Amitesh Anand, Arjun Verma, Bukka Nikhil Sai, Divyanshu Khandelwal, Kotha Tejas, Mohd Zahid Faiz, Saurabh Jain, Tanmay Jain, and Vibhav Agarwal. We would happily say that we had a great learning experience while working on the project.

References

- https://github.com/frenzytejask98/ML_TA_IIITB_2020
- https://scikit-learn.org/stable/
- https://seaborn.pydata.org/examples/index.html
- https://www.kaggle.com/jsaguiar/lightgbm-7th-place-solution
- https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

- https://medium.com/analytics-vidhya/how-to-install-and-run-thundersvm-in-google-colab-de1fe49eef85
- https://github.com/ankane/thundersvm
- https://catboost.ai/docs/concepts/about.html
- https://lightgbm.readthedocs.io/en/latest/
- https://xgboost.readthedocs.io/en/latest/
- https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
- https://www.kaggle.com/c/home-credit-default-risk/discussion/63499
- https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/
- https://medium.com/@nupur94/calibration-of-models-45721a221da6
- https://www.overleaf.com/latex/templates/ieee-conference-latex-template/hkfsmxcvymyk