Retargeting Optimization for Refinancing Car Loans: A Case Study of a US-based Lender

Lia Harutyunyan

Bachelor of Science In Data Science

COLLEGE OF SCIENCE AND ENGINEERING

AMERICAN UNIVERSITY OF ARMENIA

Supervisor: Anna Sargsyan

Yerevan, Armenia

***Abstract*— Gradient boosting methods have been proven to be a very important strategy. Many successful machine learning solutions were developed using the XGBoost and its derivatives. The aim of this study is to investigate and compare the efficiency of XGBoost, LightGBM and Logistic Regression methods on car loan retargeting problem. Car loan dataset is used in this work which contains 232 features and 9,500,000 records. For the purpose of the study, the features are analyzed and several techniques are used to rank and select the best features. For filling in the missing values the MissForest, KNN and other simpler strategies, like filling with 0s and mean/median methods have been compared. The implementation indicates that the LightGBM in combination with MissForest is faster and more accurate than Logistic Regression and XGBoost using a variant number of features and records.**

***Keywords*— Classifier, XGBoost(Extreme Gradient Boosting), LightGBM(Light Gradient Boosting Machine), Logistic Regression, MissForest, Auto, Loan refinancing**

INTRODUCTION

PROJECT BACKGROUND

The presented capstone is a part of the projects performed by Plat.ai company. The company is located in Armenia and works with customers from the US. Plat.ai works on the same name engine which is an artificial intelligence machine, designed to create real-time decisions. This tool adapts to customers' platforms or systems. The company gives personalized approach to data analytics and provides accurate insights into customers' data.

This project is a case study of a US-based lender and its outcome is a tool for retargeting optimization for refinancing car loans.

Description of the Task

The capstone project's main task is to build a classification model to optimize the retargeting process of one of US lenders. That is people, who have car loans in different agencies, to be analyzed based on different features (e.g. credit history, terms of previous loans, demographics). The target of the analysis is a binary variable. When the target is 1, then the customers who received the email with the offer accept that, when it is 0, then the customers reject the received offer. The main task is, based on 232 features available in the customer database, to identify the clients who are more likely to accept the offer of refinancing their loans. Overall, the data has both categorical and quantitative features. Some of the features presented in the list are: FICO autoscore, FICO delta autoscore, Mortgage loans, personal loans, tradeline information concerning different types of loans. Also, loan rates that the customer has now and rates that are offered for retargeting. To determine FICO Auto Scores, FICO first calculates the "base" scores, which are the traditional credit scores (which ranges between 350-800) then FICO adjusts the calculation based on industry-specific risk behavior to create tailored auto scores. A tradeline is a record of activity for any type of credit extended to a borrower and reported to a credit reporting agency. A trade line is established on a borrower's credit report when a borrower is approved for credit. The tradeline records all of the activity associated with an account.

With the rise of auto loans, many people tend to take their loans through different internet platforms. The main idea of this project is to construct a tool that will generate the preferable customers that have a high chance for accepting loan retargeting options. Having the list of 9.5 million users with previous decisions if they have accepted or denied the loan retargeting opportunity and using different models, the aim is to find the main model that will predict most accurately if the user is going to accept or reject the offer. The data has many missing values. Since for the modeling purpose it is very important to correctly fill the missing values, three different approaches have been experimented to get the best result.

Three models were constructed, these are LightGBM, XGBoost and Logistic Regression. As the problem is a binary classification, it is most common to evaluate the models using AUC scores. The goal of this study is to find the most accurate model that will give best predictions of the users who have high chances of agreeing to offer.

Data Source

The data is provided by Plat.ai's customer which is a lender company. The original data contains 232 features and 9.5 million loan owners. The target column is the “Lead Flag” column, which is a binary column of values 1 and 0, indicating if the customers responded to the offer sended by mail or not.

Literature review

Gradient Boosting

Boosting is used very widely for getting a strong model from a large number of relatively weak and simple models. [2] By weak models different machine learning algorithms whose accuracy can be only slightly better than random guessing are considered.

In the gradient boosting algorithm multiple models are trained sequentially. For each new model the Gradient Descent method [3] is used to gradually minimize the loss function. The general approach is to build a linear combination of simple models (basic algorithms) by reweighting the inputs. Each subsequent model (usually a decision tree) is built to give more weight and preference to previously incorrectly predicted observations. [10] .

Since the nodes in a decision tree consider a different branch of features for selecting the best split, all the trees are different. So, they can capture different outputs from the same data all the time. The Gradient Tree Boosting algorithm considers decision trees as the weak learners.
The gradient tree boosting algorithm is built sequentially: for each next tree, the model considers the errors of the last tree. The decision of every successive tree is built on the mistakes made by the previous tree. Gradient Boosting algorithms are mainly used for classification and regression problems.

XGBoost

XGBoost is a machine learning algorithm based on a decision tree and using the gradient boosting framework. In contrast with prediction tasks that use unstructured data (such as images or text), where an artificial neural network outperforms all other algorithms or frameworks, for small structured or tabular data, decision tree-based algorithms take precedence.

The XGBoost algorithm, short for Extreme Gradient Boosting, is a modified version of the gradient boosting algorithm. The working procedure of both is almost the same. The only crucial difference is that the XGBoost implements parallel processing at the node level. This makes it more powerful and fast than the gradient boosting algorithm. By including various regularization techniques and by setting the hyperparameters XGBoost reduces overfitting and improves overall performance. So, the speed is the biggest strength of XGBoost. The efficient use of Random Forest requires generating each tree in parallel on a cluster, and deep neural nets are usually run on GPUs, XGBoost can be run on a single CPU in less than a third of demonstrated by either of the other methods. [11] Using XGBoost lets one not worry about the missing values in the dataset. During the training process, the model itself learns where to fit the missing values. XGBoost is mainly used for classification problems but can be used for regression problems. Also this algorithm is highly effective in reducing the computing time and providing optimal use of memory resources.

So, XGBoost functions around tree algorithms. The tree algorithms consider the attributes of the dataset, the features or columns as the conditional node or the internal node. According to the condition at the root node, the tree splits up into branches or edges. The last branch that does not produce other branches is called a leaf node. Leaf nodes are results of completed splitting and they are taken as reaching the decision.

The concept of XGBoost revolves around gradient-boosted trees using supervised learning as the principal technique. Supervised learning basically refers to a technique in which the input data, generally the training data p_i having multiple features (as in this case), is used to predict target values s_i. The predicted value s_i helps to classify the problem, whether it is regression or classification.

The primary goal is to extract the right parameters from the training dataset. Instead of training all the trees in parallel, XGBoost optimizes the learned tree (training) and adds a tree at every step.

The example of using the XGBoost algorithm to construct a P2P loan default prediction model has shown the prediction accuracy rate of the XGBoost algorithm is 97.705% . [7]

LightGBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. Faster training speed and higher efficiency, lower memory usage, better accuracy are the advantages of LightGBM. It also supports parallel, distributed, and GPU learning, thus it is capable of handling large-scale data.

With increasing the number of features in the data the efficiency and scalability of the Gradient Boosting Model are lowering.

The Light Gradient Boosting Model or LGBM is used to solve this problem. It uses two types of techniques: Gradient Based on Side Sampling or GOSS and Exclusive Feature bundling or EFB.

GOSS excludes the significant portion of the data part which have small gradients. After that it uses the remaining data to estimate the overall information gain. The data instances which have large gradients actually play a greater role for computation on information gain. GOSS is used to get accurate results with a significant information gain while using a smaller dataset than other models.
EFB focuses on bundling the mutually exclusive features.

With the help of GOSS and EFB, LightGBM can significantly outperform XGBoost. [4]

So, LightGBM decreases the number of features by bundling features together. It speeds up tree learning. High dimensionality data has many mutually exclusive features. The mutually exclusive features never take zero values simultaneously. LightGBM safely identifies such features and bundles them into a single feature to reduce the complexity because the number of features is always less than the number of bundles.

LightGBM Decision Tree-type method has enabled an accuracy of 98% studied click patterns over a dataset that handles 200 million clicks over four days. [9] Protection methods to detect click frauds are highly demanded by e-commerce companies paying for the clicks.

Logistic regression

Logistic Regression is used to predict the probability of occurrence of some event based on the values of a set of features. To do this, the so-called dependent variable is introduced, which takes only one of two values - as a rule, these are the numbers 0 (the event did not occur) and 1 (the event occurred). So, it is a supervised ML algorithm used for binary classification problems (when target is categorical). Logistic Regression's range is bounded between 0 and 1. Logistic Regression does not require a linear relationship between inputs and output variables.

Logistic Regression uses “maximum likelihood estimation (MLE)” as a loss function, which is a conditional probability. If the probability is greater than 0.5 (as the default threshold), the predictions will be classified as class 0. Otherwise, class 1 will be assigned.

Random Forest and Boruta

Random forest algorithm consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest produces a class prediction and the class with the most votes becomes the model's prediction. The wisdom of crowds is the basic principle behind random forest, and it's a simple yet effective one. The algorithm is used for classification, regression and clustering problems. The main idea is to use a large ensemble of decision trees, each of which by itself gives a very low quality of classification, but due to their large number, the result is good.

For example, Logistic Regression models are used to study effects of predictor variables on categorical outcomes and normally the outcome is binary, such as presence or absence of disease. [13]

The Boruta algorithm is a wrapper built around the Random Forest classification algorithm. It is the implementation of the idea of assessing the importance of predictors using random permutations of their values. [6] The Boruta algorithm tries to capture all the important, interesting features in the data set with respect to an outcome variable. The time investment is significant for large datasets, but it is necessary to ensure selection of relevant features based on statistical significance.

Random Forest Extreme Gradient Boosting (RF-XGBoost) is used byNon Banking Financial Institutions, for example, for prediction whether to approve the loan or reject it. Here, a Random Forest classifier is used to get the importance of each feature. [5]

KNN

K-Nearest Neighbor in Missing Data Imputation is used to group the data set into different groups. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. So, KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then selects the K number of points which are closest to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data. [1] The selected class is the one that holds the highest probability. To improve the accuracy of missing value estimation for data, a combination of KNN-based feature selection and KNN-based Imputation is used.[8]

MissForest

MissForest is a machine learning-based data imputation algorithm that operates on the Random Forest algorithm. A study, conducted by the authors of this algorithm [12] in 2011 compared its imputation methods on datasets with randomly introduced missing values. As a result, MissForest outperformed all other algorithms in all metrics, including KNN-Impute, sometimes by over 50%.

As a first step, MissForest algorithm filles in the missing values using median/mode imputation. The missing values are marked as 'Predict'. The other rows are training rows. They are fed into a Random Forest model trained to predict. The generated prediction for that row is then filled in to produce a transformed dataset.

Methodology

I. Data cleaning and Feature engineering (*X1*)

As the dataset comes with 232 features, the first step of the work is evaluating and understanding each feature's importance for the idea of the project. This evaluation is done by getting the meanings for each feature and how it is represented in the dataset, what is the range of the features. The next step of the evaluation is measuring the number of missing values, are they easily replaceable in terms of their meaning. This brings to identifying the unimportant features, which means that these features can easily be dropped, so by this 61 of the features are eliminated from the main dataset. Having a cleaner dataset and continuing the investigations, features with missing values that can be easily replaced are separated, some features' NA values needed to be filled with specific techniques. So, 49 features' NA values were replaced by 0 and 28 features' NAs were replaced by columns' means. This step is a starting point, and appears to be in the data preparation part. As the data came until this step had already undergone several modifications, there was no need to go much deep into cleaning the data. Anyway having this amount of columns is not a very favorable option for continuing the work and identifying the models that can perform well. To overcome this problem feature engineering techniques can be used. With the help of models it is easy to identify the features with their importance evaluations. For the models to work properly, it is necessary to have data cleaned from the unimportant features.

When the dataset is free of the redundant columns and the missing values are partially handled, the models for getting the important features can be used. The feature engineering needs to be calculated by a model. For this purpose the right model should be identified, so several models were experimented and the one with the best result was chosen as the main model. There are several models that could be appropriate for detecting the feature importance of the dataset's features, these are LightGBM, XGBoost and Boruta. The last model was chosen as an experimental stage. The Boruta algorithm is a wrapper built on the Random Forest classification algorithm. The main problem with the Boruta is that it does not have any internal logic for replacing the NA values and also the time that it takes to run the model is too long. In contrast, the boosting algorithms do the task of filling missing values internally, so that there is no need to solve this problem manually at first. Even when the missing values were filled before running the Boruta algorithm, as it uses Random Forest, the task took too long to run. The boosting algorithms obviously outperformed Random Forest, as they do fill missing values and run the model in a very short amount of time. So, the most appropriate models for the data turned out to be the Gradient boosting decision tree techniques, like XGBoost and LightGBM. As these too give very similar results, the LightGBM was chosen as the main algorithm for feature selection because its performance speed and results compared to the XGBoost are better, this is a very base idea of the LightGBM model to work faster and give more accurate results. Firstly, the model was run on the whole dataset, further, when the list of features and their corresponding importance gains were evaluated, only ones with importance greater than 0 were selected. Having this list, the model was run again and from the results of the second run of the model, the main features needed to be selected by looking at the meanings of the features and how they can be related to the problem. The purpose was to find the first 50 most important features and to suppress others. Of course, the importance gain of each feature was a good measure, but also the main idea was to find features that would logically suit the problem that needed to be solved. That is having features important for qualifying the customers as ones that would be appropriate to choose and once that won't.

So, to accomplish this purpose, the features needed to be analyzed one by one and to leave only ones that would suit. Finally, the 50 features were selected and the other features were dropped from the datasets [Figure 1]. So, the result of this step is a smaller dataset with a limited number of features whose importance gains are appropriate for the further analysis. When the dataset with the 50 features was ready, the dataset was divided into train and test datasets. These two datasets are saved separately and they will be called in order to use them as train/test datasets. Further analysis should be done on these datasets, there is no other need for further splitting of the dataset.

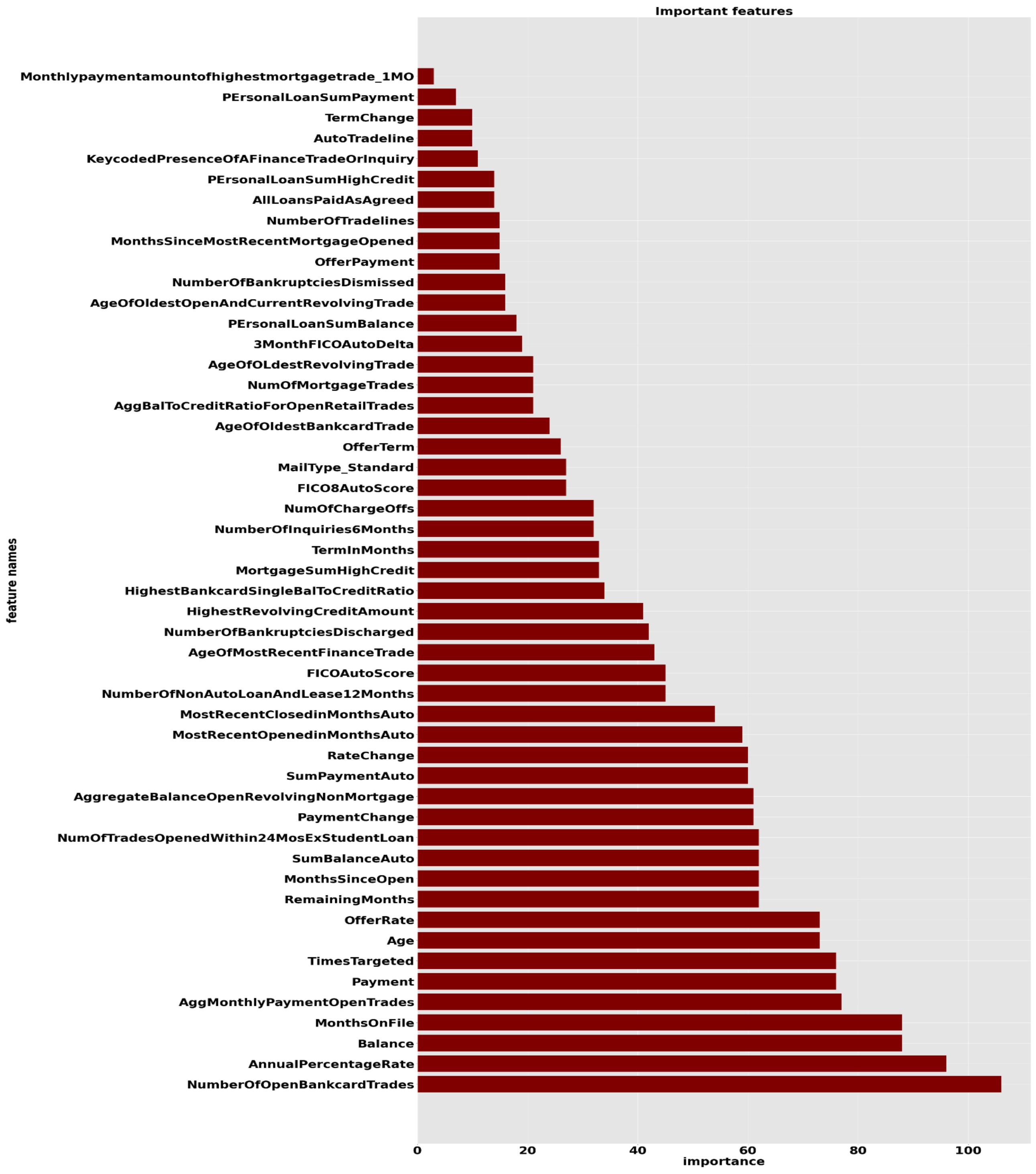


Figure 1: First 50 important features selected by logically fitting to criterias.

II. Data Sampling (X2)

As the dataset received from the customer included around 9.5 million rows, the approach for working with it was to use sampling techniques, to get the same dataset construction with only 10% of the original number of rows. This idea came from experiments with the dataset, which obviously took too long in time. For the purpose of getting an equivalent proportion of the target column's values, the approach of sampling the data with appropriate percentages of 1s and 0s was chosen. The main idea behind the approach is to divide the dataset into two datasets, based on splitting the target column's values of 1s and 0s to two datasests and to sample the 10% of each dataset. In order to get the right rows from each sub-dataset the random state was provided, which guarantees to always give the same results both for the divided dataset and for the whole dataset. After having the appropriate percentages of rows from each dataset, the two datasets are being concatenated. This step is a crucial point before starting any further research. For this given dataset, the sampling will not provide a misguidance for the models as it is quite homogeneous. Using the sample of the data helps in reducing the time and memory spent on filling the missing values and running the model.

III. Replacing nulls(X3)

The main dataset that should be used for fitting the models will result in having 50 columns and sampled to 10% of the original dataset. This should be the main shape for the dataset. Besides the shape of the dataset, it has a crucial issue with the number of missing values. Models should be run on datasets with no missing values. Here the dataset allows three methods for replacing the null values. Every method for replacing NA values is run on the train and test datasets separately as already mentioned above, the data with most important features is divided into train test datasets to escape splitting it every time of running each model. So, the first method that is used for this purpose is filling missing values with 0s. This is a very base and simplest method for imputing missing values. When the missing values are filled the two datasets are saved separately. The second method is to fill values with mean and median. When the mean is too close to the median, the first one is used to replace the nulls, when the mean is bigger than the median the last one is used to fill the values. As for the first case, when the mean and the median are close, the data has a symmetrical distribution, that means that imputing mean value will not change the distribution much. In contrast, when the mean differs from median, the distribution is skewed, the mean in this case is biased by the values at the far end of the distribution, so imputing missing values by median is more appropriate. Lastly, for filling the missing values, some model needs to be chosen to do the job. As the main idea for doing this is to get a more or less appropriate replacement for missing values, models like KNN would work. After some investigation and some experiments, it was obvious that running KNN would take too long, which is obviously a very inconvenient solution. So, after some research a model named MissForest turned out to deal with the job in a most accurate and fastest way. The model was run on the sample of the data and it took 14 hours to complete the job. The sample of the dataset and the main dataset do not give much difference as the dataset is quite homogeneous but as a matter of fact it would take around 7 days for the model to run on the whole dataset, having the exact same parameters of the computer. Finally, there are 6 datasets, 3 sets of train/test datasets each set with a different method of handling the missing values. Further analysis will be done on each of the datasets separately.

IV. Parameter tuning(X4)

When the shape of the datasets are modified and the missing values are filled, the datasets are ready for running the models on them. Before running the models, their parameters should be tuned. Based on the previous experiments done, the main models that can be considered well suited for the given data, are the models that work with gradient boosting algorithms, such as XGBoost and LightGBM. These models are ones that need to be analyzed and to choose one with the best score. In cooperation with these models, as the data is binary, Logistic Regression is also chosen as the main model. So, these three models are chosen as the base ones and the best one needs to be identified based on the roc-auc score. The best parameters for the gradient boosting models should be chosen. Their parameters need to be tuned. The basic parameters that need to be tuned for both LightGBM and XGBoost are learning rate which controls the contribution of all the vulnerable learners in the final output and max depth which is for the maximum depth of the individual regression estimators to limit the number of nodes in the tree. To be able to identify the best parameters, roc-auc scores are computed for each set of parameters, that is parameters are chosen in a range between 0.1 to 0.9 for learning rate and 1 to 5 for max depth. The checking is done through running a double for loop and one by one incorporating numbers from these ranges. The models are fitted on the train datasets and the prediction scores of the test datasets are saved. From experiments it was obvious that values more than or less than the ranges given, lower the roc-auc scores for the test dataset, even though they increase the roc-auc for the train dataset. So, the obvious overfitting was encountered for values other than these ranges.

Algorithm 1 is an example of the for loops that construct a dictionary of the best results for the specific model. The MODEL refers to XGBoost or LightGBM classifiers. The values of the parameters and the scores are saved in the results dictionary, which then will give the parameters with maximum scores.

Algorithm 1: parameter tuning

```
for i in range [0.1, 0.9]
   for j in range [0,5, 1]
            model = MODEL(learning_rate=i,max_depth=j)
            model.fit(x_train, y_train, eval_metric='AUC')
            Y0_MODEL = model.predict_proba(x_train)[:,1]
            Y1_MODEL = model.predict_proba(x_test)[:,1]
            results['learning_rate'].append(i)
            results['max_depth'].append(j)
            results['score_train'].append(roc_auc_score(y_train,
Y0_MODEL)
            results['score_test'].append(roc_auc_score(y_test,
Y1_MODEL)
```

This type of parameter tuning for LightGBM and XGBoost are done on three different datasets. These are ones filled with missing values with different methods. First dataset's missing values are filled with 0s, the second dataset's missing values are filled with mean/median and the last dataset's missing values are filled with MissForest. Finally, the learning rate and max depth parameters are chosen for each dataset for two boosting models. These parameters are going to be used in the further models. It is crucial to escape overfitting problems during parameter tuning. When the training roc-auc score is too much higher than the auc score for the testing dataset the model seemingly has overfitted the data. In such cases, the parameters need to be adjusted so that testing and training scores do not differ much. The final parameters that are chosen give a difference of the testing and training scores for less than 1 which is a good measure to choose the parameters as ones that will work fine for the models. From the training datasets' auc scores it is obvious that the best models have around 70% of accuracy based on the model type. So, puting into account that the difference of train and test scores should not differ much, the testing scores are expected to be not less than 70%. Lastly, the Logistic Regression is run and compared to Logistic Regression with regularization, it turned out that running Logistic Regression with

‘L1’ regularization gives the best result for roc-auc score, which is also important to note that it was approximately 70% for the both train and test datasets. The three models are ready to be fitted and predicted for the resulting datasets.

V. Final models(X5)

When the datasets and the values for the models’ parameters are ready, the final models can be fitted for doing the comparison. Each model should run on three datasets separately. The resulting test scores for each model run on each datasets should be saved in a data frame. Further, the data frame will provide the best model based on the test scores of each model. So the three different sets of the dataset are loaded and need to be used to fit three different models. These models are XGBoost, LightGBM and Logistic Regression. Each set’s train data is used to fit the model and test data is used to predict the target value and to get the preferred roc-auc score. So, we have overall 9 predictions and correspondingly 9 roc-auc scores to choose the best model. The first model to run is the LightGBM model, after getting the resulting roc-auc score for the test dataset it is saved for further comparison, the same is done with XGBoost. The last step for each dataset is to fit the Logistic Regression and to get the roc-auc score for it. For running Logistic Regression, as it was already mentioned it turned out that using regularization will give better results for the given data. So, Logistic Regression gives quite good results which are very similar to ones that were received with the boosting models. After firstly fitting the Logistic Regression model, the non significant features were detected and dropped from the dataset. The model was fitted again to get the best and most accurate predictions on the datasets for this type of model. Also, when running Logistic Regression on the data imputed by MissForest, the importance plot for the impact of the significant features on the target variable is generated [Figure 2]. This steps are done on each of the three datasets separately. And when all the models are finally fitted, there are overall 9 roc-auc scores, and even though there is not much difference between these scores, the highest score comes from LightGBM fitted on the dataset with MissForest as the model for filling missing values [Figure 3]. So, this model is chosen as the best model for this particular problem and this right model is going to be used for validation datasets and for getting the most accurate predictions for the customers that are going to be chosen as targeted customers. Below are the ROC curves for the LightGBM model that is run on the dataset whose missing values are filled by MissForest algorithm [Figure 4].

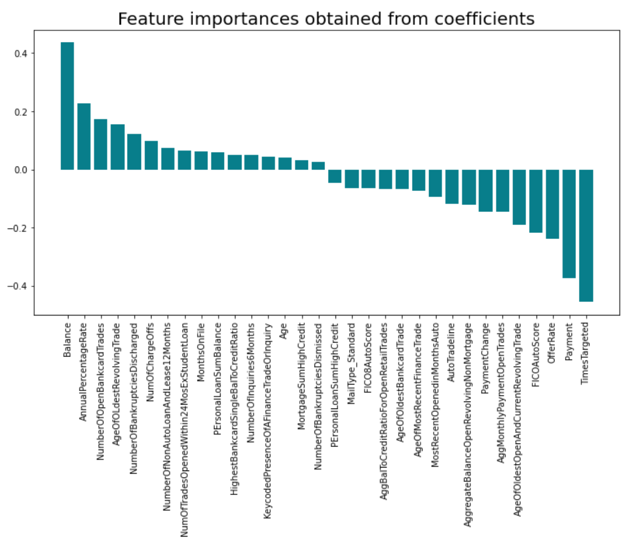
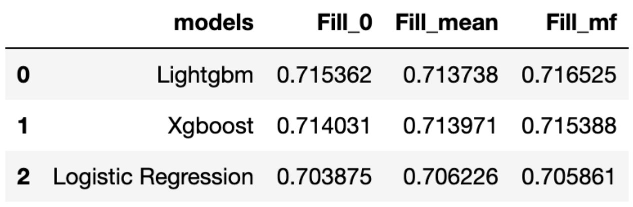


Figure 2: Feature impact on target



	models	Fill_0	Fill_mean	Fill_mf
0	Lightgbm	0.715362	0.713738	0.716525
1	Xgboost	0.714031	0.713971	0.715388
2	Logistic Regression	0.703875	0.706226	0.705861

Figure 3: Comparison table for Models

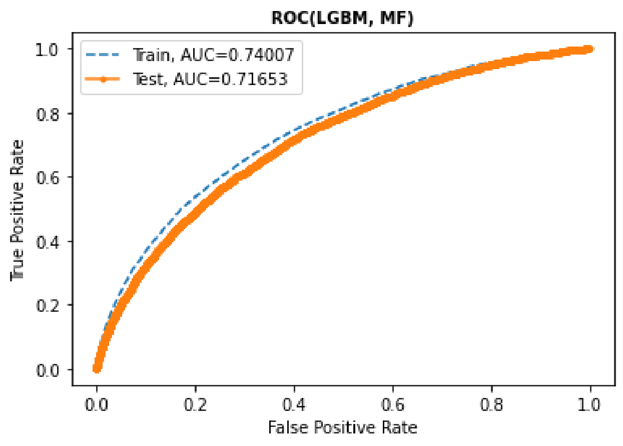


Figure 4: ROC curves and AUC scores for LightGBM and MissFores

VI. Validation(X6)

When the final model is chosen with the right parameters tuned and right method for missing values, the prediction on the validation dataset is done. For this purpose, a validation dataset is used, which comes with the same columns as the dataset that was used for training and predicting, except that it does not contain the target column. The dataset has to pass several phases before the prediction of the LightGBM model is done. As it is known that the data comes in a standard format, which means having standard column names and types as the train dataset, it is quite easy to get only columns important for the use. First of all, the data should be prepared by filling missing values of the features that are known how to be filled. So, some columns' NA values get filled by 0s other columns' NA values by means, this information about these features are known from the investigations done in the beginning. The step for dropping unnecessary features is missed for this part as, now after filling the missing values, it is needed to choose only columns that are in the important features list. So, 50 most important features are chosen. Finally, the dataset is ready for analysis, but prior to that the missing values still need to be filled. So, the MissForest model is used. After running MissForest and replacing all the nulls, LightGBM is trained on the previous training dataset and the trained model is used for predicting the target value for the clean validation dataset. The prediction is done on the validation dataset and the results of it will be the information about each customer's prediction probability for being a target customer or not. For identifying such customers, a probability threshold of 0.8 is set. So, the customers having predicted probability greater than 0.8 are considered as ones that are target for this problem and they are the ones needed to be separated and shown as the final results of the research. Having the customers probabilities and setting the threshold, a target column of binary values of 0s and 1s is made. By this column it is easy to separate all the customers who most probably will respond to the needs of the company. Finally, using this target column and the IDs of customers the table of the targeted customers is made and ready for use. For visualization purposes, to see customers from which part of the US are more likely to respond, their ZIPCodes are plotted on a map plot.[Figure 5]

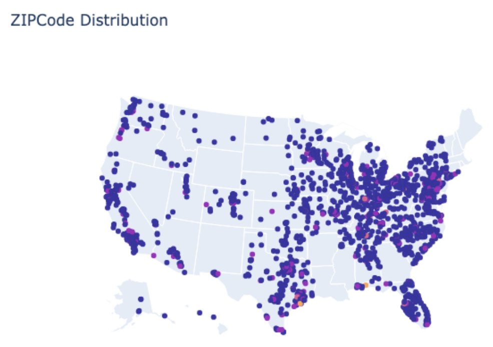


Figure 5: Visualization of distribution of customers through US

Conclusion

Boosting methods iteratively train a set of weak learners. The weight of the records is updated based on the regression results of the previous learners' loss function. In this study three methods were compared (XGBoost, LightGBM and Logistic Regression) in terms of performance and ROC-AUC scores. While in this experiment there was not much difference in time consumed by different models on the same system with the same dataset, LightGBM proved itself to be significantly more efficient than the other two models. Among missing value replacement approaches, MissForest algorithm outperformed simple methods of filling them with 0s and means. So, the combination of LightGBM and MissForest is proven to be the best prediction tool for car loan retargeting systems.

References

1. Christopher, A. 2021 “K-Nearest Neighbor.” *Medium* medium.com.
2. Daoud, E. A. et. al. 2019. “Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset.” *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering* 13 (1). https://publications.waset.org/10009954/comparison-between-xgboost-lightgbm-and-catboost-using-a-home-credit-dataset.
3. Friedman, J. H. et. al. 2001. “Greedy function approximation: A gradient boosting machine.” *Ann. Statist* 29 (5). https://doi.org/10.1214/aos/1013203451.
4. Ke, G. et. al. 2017. “Lightgbm: A highly efficient gradient boosting decision tree.” *Advances in neural information processing systems* 30. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.
5. Koduru, M.et. al. 2020. “RF-XGBoost Model for Loan Application Scoring in Non Banking Financial Institutions.” *International Journal of Engineering Research & Technology (IJERT)* 9 (7).
6. Kursa, M. B. et. al. 2010. “Feature Selection with the Boruta Package.” *Journal of Statistical Software* 36, no. 11 (https://doi.org/10.18637/jss.v036.i11).
7. Li, Zh. et. al. 2021. “Application of XGBoost in P2P Default Prediction.” *Journal of Physics: Conference Series* 1871 (6th International Symposium on Advances in Electrical, Electronics and Computer Engineering (ISAEECE 2021)).
8. Meesad, Ph. et. al. 2008. “Combination of KNN-Based Feature Selection and KNN-Based Missing Value Imputation of Microarray Data.” *3rd International Conference on Innovative Computing Information and Control*.
9. Minastireanu, E-A. et. al. 2019. “Light GBM Machine Learning Algorithm to Online Click Fraud Detection.” *Journal of Information Assurance & Cybersecurity* 2019. 10.5171/2019.263928.
10. Natekin, A., and A. Knoll. n.d. et. al. “Gradient boosting machines, a tutorial.” *Front Neurorobot.* doi: 10.3389/fnbot.2013.00021.
11. Sheridan, R. et. al. 2020. “Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships.” *J Chem Inf Model.* 56 (12). DOI: 10.1021/acs.jcim.6b00591.
12. Stekhoven, D., and P. Bühlmann. et. al. 2011. “MissForest--non-parametric missing value imputation for mixed-type data.” *Bioinformatics* . 28 (1). 10.1093/bioinformatics/btr597.
13. Todd, G. et. al. 2007. “Logistic Regression.” *Methods in Molecular Biology: Topics in Biostatistics* 404.