Machine Learning Assignment-2

Pavan Nagesh, D18128936, TU059

# Introduction and Review of Existing Work

This project is intended to review the work carried out by different kagglers on a dataset and identify the underlying gaps to be filled. The dataset chosen for this project is called Santander Customer Prediction (<https://www.kaggle.com/lakshmi25npathi/santander-customer-transaction-prediction-dataset>)**.** This dataset set was released for a competition in Kaggle where the aim is to predict if a customer will make a transaction or not. It has two csv files one is training set and other is testing set. There is a total of 202 columns and 2,00,000 rows in the training set including target and index column, whereas in test set there are 201 columns and 2,00,000 rows and it doesn’t have the target column. This is a binary classification problem where the class labels to be predicted is either ‘0’(customer does not make transaction) or ‘1’(customer makes transaction). All the columns are numeric and there are no categorical variables.

In total there are 19 kernels for this dataset. The most common models that are built are Logistic regression, Gaussian Naive Bayes, Random Forest, XGBoost Classifier and Light GBM classifier. From the exploratory analysis included, checking the frequency distribution of each of the variables and the target variable. It was seen that the target variable is imbalanced and one of the three different techniques were employed as to balance the dataset namely ROSE, SMOTE and Random under sampling. Tough some of the kernels do balance the target before modeling but most of the kernels do not. Data Exploration also included observation of skewness, kurtosis, and variance in the data. Most of the kernels concluded that there is no need of data preprocessing as the data distribution showed that they were almost normally distributed with slight skewness and variance and hence the models were built with unscaled data. Further feature engineering was performed using Principle Component Analysis(PCA) to understand structure of high dimensional data but none of them truly reduced the dimensionality and retained the original dimension of the data. Further some of them also used Random Forest Classifier to identify the most important features and the results showed that about 25 features could be ignored, but these features were not ignored as considerable amount of information will be lost.

Various models were built some with default parameter values and other models were tuned with hyper parameters. The technique used to search hyperparameters for the model was Grid Search. The metrics selected to measure the performance of the model in all the kernels is Accuracy and AUC. From the analysis of all the models from all the kernels it was found that LightGBM performed the best in with AUC of 0.89 with the default model parameters, 0.952 for balanced data using SMOTE and 1 for imbalanced dataset. XGboost Classifier (AUC of 0.81 for default model parameters and 0.82 for model tuned with hyperparameters for balanced data using random under sampling) and Gaussian Naïve Bayes classifier(AUC of 0.88 with default model parameters, 0.88 for model tuned with hyperparameters for balanced data using random under sampling and 0.86 for balanced data using SMOTE ) model also showed good performance. From the analysis all the kernels it can be concluded that models showed better performance for unbalanced and unstandardized data when compared to models with balanced and standardized data.

# Gaps and Plan

From the review of existing work, it was identified that the boosting algorithms like LightGBM and XGBoost show better performance than other models for binary classification with imbalanced data. There do exist other boosting algorithm like CatBoost which is much faster than its counterparts for bigger datasets and none of the kernels have showed implementation using this model. Also, for such boosting algorithms it is essential to select a proper set of hyperparameters for getting better performance. Though one of the kernels did use grid search on hyperparameters to tune the model but not all the important parameters like max\_depth, learning rate were identified and grid search for finding optimal parameters is very time consuming and requires more computational power. To overcome the complexity of the Grid search random search of hyperparameters can be used which randomly subsets a set of parameter values and it is faster compared to grid search. The results showed that models performed best for imbalanced dataset, but for any machine learning model it is essential to balance the dataset. Hence to get an optimal prediction of the target following is the plan which is implemented:

* The data is checked for skewness and kurtosis, based on the result the data is standardized.
* Target attribute is balanced using ROSE.
* In total 8 different Xgboost and CatBoost models are built
* Two variants of Xgboost and Two Variants of CatBoost models are built one each for balanced and unbalanced dataset using grid search for hyper parameter tuning.
* Two variants of Xgboost and Two Variants of CatBoost models are built one each for balanced

and unbalanced dataset using random search for hyper parameter tuning.

* The models are build using both balanced and unbalanced dataset to monitor how the model performs in both the cases. The metric used for evaluation is AUC.
* Further to improve the model performance 3-fold cross validation is done

# Implementation

## Exploratory Data Analysis

EDA is one of the major initial steps for any machine learning projects. It helps in understanding the structure of the data and the underlaying key points which further helps in making informed decisions on building of models. The EDA for this project is done based on Descriptive Statistics and Visualizations.

**Descriptive statistics**

The descriptive statistics of both train and test set is obtained using the summary() function in R. From the output, it can be observed that there is a total of 202 features in the test set which includes target attribute and index column and the test set consist of 201 features similar to train set but without the target column and the total number of rows are 2,00,000 for both the sets. The summary statistics shows that all the features except the target and index are of floating type and target is of integer and index is character type. The mean values for both the train and test set for each feature is distributed over a large range and the standard deviation is also considerably large for both the train and test sets. All the summary statistics mean, min, max, standard deviation values are close for both the train set and test set. The distribution of target variable from the fig.1. shows that the data is imbalanced, about 90% of the values have the class label 0 meaning that 90% of the customers will not do transaction and 10% of the values have the class 1 meaning that 10% of the customers do transaction. For the model to give unbiased results it is essential to balance the values for class labels. Next both the train and set are checked for missing values and it showed that there are no missing entries in any of the rows or columns.

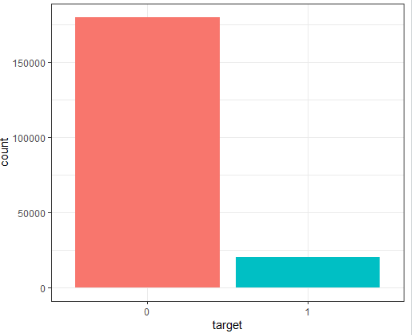
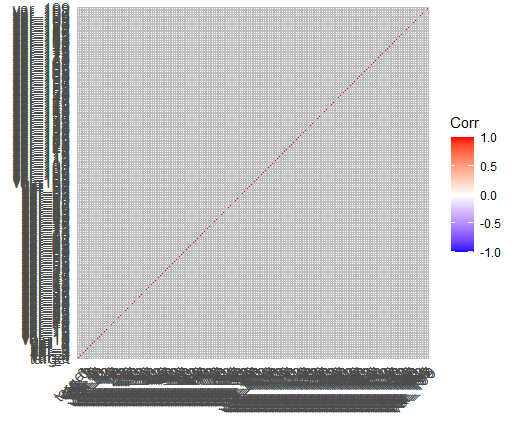
 

Fig.1. Target distribution Fig.2. Correlation Heat Map

The skewness and kurtosis are calculated for each columns, since the values were to be computed for each of the columns in the data frame *apply()* was used from R as skewness and kurtosis were passed as parameters for the function. These values were computed for both the train and test set separately. For the distribution to be considered as normal the expected range of value for skewness is -1.96 to +1.96 and for kurtosis it is -2 to +2. When computed skewness for features of train set, the values were in the range -0.34017 to 0.26741 which are close to zero and kurtosis values are in the range -0.81183 to -0.01402. Hence it can be said that the features of train set are normally distributed. Similarly, the skewness values for features of test set range between -0.32886 to 0.26868 and the kurtosis values range between -0.82601 to -0.02379. Hence it can be said that the features of test set are normally distributed.

**Visualization**

Further Frequency distribution of each features in train set with respect to target class labels 0 and 1 were examined. The plots showed that most of the features have almost similar distribution for both the class labels expect for few features such as var\_0, var\_2, var\_6, var\_12, etc. From the plot it can also be said that the distribution for each of the variables is normal with accepted degree of skewness and kurtosis. Similarly, the frequency distribution of features in the test set also showed normal distribution with accepted degree of skewness and kurtosis.

The correlation between the features and the target was examined which showed that there was negligible correlation with values nearing to 0 indicating no correlation. The covariance between features was also examined and fount that there is no correlation within the features. The correlation and covariance were visualized using correlation heatmap as shown in fig.2.

\*Note: EDA is inspired from two kernels (<https://www.kaggle.com/vi20027804/customer-transaction-prediction#2.1-Exploratory-Data-Analysis(EDA)> and <https://www.kaggle.com/lakshmi25npathi/santander-customer-transaction-prediction-using-r>)

## Feature Engineering

From EDA it was seen that the data is not balanced, there are no missing values, data is normally distributed, there is no correlation between features and the target also within features and the mean values of each columns are distributed over a large rand and SD is also very high for both train and test set.

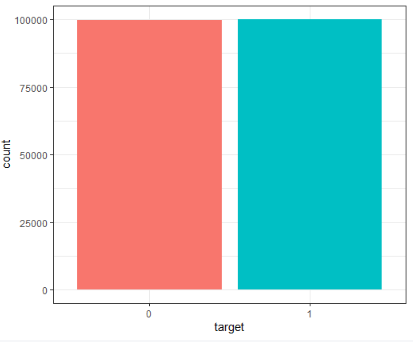


Fig.3. Target distribution after oversampling

The feature engineering is done firstly to balance the train dataset and secondly to standardize both train and test sets to have a mean of 0 and SD of 1. To balance data the technique used is Random Over Sampling Examples (ROSE), ROSE function from ROSE package in R is sued which creates a sample of synthetic data by enlarging the features space of minority and majority class examples. Operationally, the new examples are drawn from a conditional kernel density estimate of the two classes, as described in Menardi and Torelli (2013). The features are standardized by scaling the values using *scale()* function and now the mean and SD are 0 and 1 respectively.

## Model Building

From previous works it was identified that LightGBM and Xboost classifiers both performed well on imbalanced dataset and both showed good results with the default model parameters. In this project one more prominent boosting algorithm called CatBoost is evaluated against Xboost and is expected to show better results than its counterpart. It is also essential to fine tune the model with appropriate values of hyper parameters and hence Grid search technique and Random Search is used to find the optimal values for hyper parameters of the models. These models are further compared using balanced and imbalanced datasets where the model settings are same for both the cases which helps in better comparing the model performance in different scenarios.

**Data Partition**

Since the unknown target labels are to be predicted for the test set, train set is used to split the data further into 70% for training and 30% validation. From catools package *createDataPartition()* is used to split the data frame into training and validation sets. Two data partitions are done, one using the balanced training set and the other using unbalanced training set and models are built on both these data.

**XGBoost Classifier using Balanced Data tuned using Grid Search**

The XGBoost algorithm only accepts data in matrix format and hence first the training and validation sets created from balanced data is converted to its corresponding matrices. To do this the target column and all the other features are separated and stored into two different variables one containing the values of the target and the other containing the values of rest of the features. Using *xgb.Dmatrix()* function the dataframe containing features apart from the target is converted to XGBoost matrix. This is done for both the training set and the validation set. Since the model is a binary classification model, the levels of the target are changed from “0” and “1” to “no” and “yes”. A list of parameters is selected and also the values for the parameters are set. Since the optimal model parameters are not known at the start, a range of values are given, and grid search is performed to find the best set of parameter values with which the model performance is high. The maximum depth of the tree to be created is set to 5,10,15 and 20 using a vector format to *max\_depth* parameter and *colsample\_bytree* is set to take to values from 0.5 to 0.9. The learning rate is set to 0.3 to reduce overfitting. Finally, the number of trees to be created by the model is set to 100 in *nrounds* parameter. All these parameters are set using *expand.grid()* from caret package. Using *trainControl()* the model is set to train using 3-fold cross validation. Finally the model is trained using the *train()* and the method parameter is set to “xgbTree” to indicate that the model to be trained is a XGBoost classifier and the grid search parameters list and the cross validation parameters are passed to the function. To compute the time taken by the model to train *Sys.time()* is called at the start and at the end of the model training process and the difference of these two times are computed to et overall time taken by the model to train which was **3.169779 hours**. The model is trained with each of the combination of the parameter values and the result obtained by the best parameter is selected to finally complete the model training.

Next the class labels of the validation set are predicted using *predict().* The predicted labels are “yes” and “no” and hence the levels are changed to “0” and “1”. Using the *roc()* the roc is plotted, and AUC is computed for predicted class labels which was found to be **0.729.** Further the confusion matrix is computed with the predicted class labels and the actual class labels of the validation set. Using the *gglpot()* the confusion matrix is visualized where the data passed is the table given by the output of *confusionMatrix()* which is then converted to a dataframe. The x and y axis are set to “Reference” and “Prediction” indicating the actual and predicted values. Using *geom\_tile()* the plot created to represent the format of confusion matrix where “Reference” and “Prediction” values are passed to the *geom\_text()* to print the corresponding values in respective tiles. The overall accuracy of XGBoost classifier using balanced data with grid search hyperparameter tuning was found to be **73%.** The features selected by the model was obtained by *varImp()* which gives out the most important features and its output showed that all 200 features were used by XGBoost. A bar chart was also plotted to see the top 20 features.

**XGBoost Classifier using Imbalanced Data tuned using Grid Search**

The model is created using imbalanced training dataset which is split into train and valid sets with ratio of 70% and 30% and XGBoost model is trained and tuned using same hyperparameters grid search where parameter values are same as set for previous XGBoost model created above. The model is trained on different parameter values which are selected by grid search technique. The time taken by this model to train is **2.922677 hours** and the ROC plot for predicted class labels for validation set show that the AUC value is **0.6296.** Further confusion matrix is plotted similar to done for previous model and the accuracy was found to be **78%.** The variable importance plot showed that the model used 198 features.

**CatBoost Classifier using Balanced Data tuned using Grid Search**

For CatBoost classifier the balanced training dataset is used which is split into training and validation set. Further the class labels are separated from each of the sets and stored separately and the class labels are changed to “yes” and “no”. The hyperparameter values for the grid search are set where the maximum depth of the tree is set to 6,7,8,9 and 10 which is passed as a vector, lower values are selected s more deep the tree structure becomes the complexity of the model increases and also requires more computational power. The iterations to take place is set to 500 and 600 also to reduce overfitting different values are set for the learning rate 0.3,0.4 and 0.5. The grid search creates number of models from all the subset of the parameter values to get the most optimum parameter values for the model. The model is trained with 3-fold cross validation similar to the one created in Xboost classifier. Using *train()* from caret package CatBoost model is trained with different values of the parameters selected from Grid search. The model computation time is checked using *Sys.time()* and it was seen that CatBoost model took **2.272147 hours** to train.

The class labels for the validation set is predicted and the ROC is plotted and the AUC for the predicted class labels is computed using *roc()* which gave a value of **0.744.** The confusion matrix is created and visualized in the similar fashion as done in the previous models which showed the overall model accuracy to be **74%.** Variable importance plot is alco created similar to previous models and it was found that all the 200 features were selected.

**CatBoost Classifier using Imbalanced Data tuned using Grid Search**

For this model it is build and trained in similar fashion as with the CatBoost model tuned using hyperparameter grid search where the parameters selected, and the different values are same. The only difference is the data with which the model is trained, the imbalanced training dataset is used which is split into training and validation set with same split ration 70:30. The model is trained using different parameter values which are selected using grid search technique. The time taken by the model is **1.93766 hours.** The ROC plot showed the AUC value for the predicted class labels to be **0.658.** Confusion matrix is created, and the model accuracy is found to be **91.5%.** Variable importance plot shows that the model used all the 200 features to train the model.

**XGBoost Classifier using Balanced Data tuned using Random Search**

Here the model is built using same balanced dataset but the method of selecting hyperparameters to tune the model is different. Here Random search is use instead of grid search which randomly selects the parameters rather that selecting all the subsets of the parameter values and building the model using all the subset values. The random search of parameters is done using the *trainControl().* The matrices of the training and validation sets are created similar to other XGBoost models. The search parameter of *trainControl()* is set to “random” to indicate that the model while training should select the parameter values randomly. The values for hyper parameters are not set as done for grid search method rather the function randomly selects the parameter values and builds model for different sets of randomly selected values and finally checks for the parameter values of the model which performed the best and these parameters are selected to finally train the model using 3-fold cross validation. The time taken by the model to train is **1.375804 hours.** The validation class labels are predicted, and ROC plot is do not to find AUC for predicted class labels and the value was found to be **0.641.** The confusion matrix is created and visualized similar to previous models and the accuracy was found to be **91.7%.** The variable importance plot also shows that the model used all 200 features.

**XGBoost Classifier using Imbalanced Data tuned using Random Search**

The model is built and trained using the partition of imbalanced dataset and random search method was employed to search for most optimal hyperparameters. The model settings are similar to the XGBoost model created using balanced dataset and tuned using random search method. The computation time for model training is **1.74246 hours.** The ROC plot for the predicted class labels showed that the AUC value was **0.645.** Confusion matrix is created and visualized which showed the overall model accuracy to be **91.8%**

**CatBoost Classifier using Balanced Data tuned using Random Search**

Another variant of CatBoost model is created using the balanced dataset and tuned with the random search of hyperparameters. The model is built and trained using similar fashion as previous CatBoost models with the grid search techniques. The difference is the use of random search and not the grid search for hyperparameters. The rand search setting is done in the *trainControl()* where the search parameter is set to “random” which randomly selects values for the model parameters and trains the model using different set of parameters randomly selected by the model. Finally, the model is trained using the most optimal parameter settings. The computation time taken by model to train with 3-fold cross validation is **11.23678 mins.** The ROC plot showed the AUC value for predicted class labels to be **0.671** and the confusion matrix is plotted which showed the overall model accuracy to be **91.82%.** All the 200 features were selected by the model for training purpose.

**CatBoost Classifier using Imbalanced Data tuned using Random Search**

The final variant of Catboost is built with similar model settings as of the previous CatBoost model with random search of hyperparameters except that the data used in the model is imbalanced data. The computational time was found to be **15.98863 mins.** The ROC plot showed the AUC value to be **0.658.** The confusion matrix plotted showed the overall model accuracy as **91.74%.** This model also used all the 200 features.

## Model Evaluation

In total there are 8 models created 4 variants each of XGBoost and CATBoost classifiers. From the initial EDA it was observed that the data is highly imbalance with 90% data belonging to class “0” and 10% belonging to class “1”. If the models are evaluated based on the accuracy for such imbalance dataset the classifier will have higher accuracy as it will predict more of “0” because the data for this class is more and classification done by such models cannot be considered reliable. One of the most used metrics for binary classification and also for imbalanced data is AUC(Area Under the Curve). The AUC is observed from the Receiver Operator Characteristics(ROC) plots where the y-axis represents the sensitivity i.e. true positive rate and x-axis represent 1-specificity i.e. false positive rate. The AUC unlike accuracy which considers a single threshold to give the classification probabilities, AUC uses the average of values from different thresholds. Hence the AUC is used as the measure of metric to evaluate the models.

Below table summarizes the results of all the models where four models are created using the unbalanced dataset and four using the balanced dataset. Under each category two variants of XGBoost fine-tuned using grid search and random search is built and two variants CATBoost fine tuned using grid search and random search is built. The time taken by each model, the AUC and the accuracy are also recorded.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Balanced/Unbalanced Dataset | Type of search for Hyperparameters | Model Name | Computational Time | Area Under the Curve(AUC) | Accuracy |
| Unbalanced Dataset | Random Search of Hyperparameters | XGBoost Classifier | 1.74246 hours | 0.645 | 91.8% |
| CATBoost Classifier | 15.98863 mins | 0.658 | 91.74% |
| Grid Search of Hyperparameters | XGBoost Classifier | 2.922677 hours | 0.631 | 78% |
| CATBoost Classifier | 1.93766 hours | 0.658 | 91.5% |
|  | | | | | |
| Balanced Dataset  (Random Over Sampling of Examples) | Random Search of Hyperparameters | XGBoost Classifier | 1.375804 hours | 0.641 | 91.7% |
| CATBoost Classifier | 11.23678 mins | 0.671 | 91.82% |
| Grid Search of Hyperparameters | XGBoost Classifier | 3.169779 hours | 0.729 | 73% |
| CATBoost Classifier | 2.272147 hours | 0.744 | 74% |

Table.1. Model Evaluation Summary

From the table it can be seen that CATBoost has outperformed the XGBoost for both the balanced and unbalanced datasets as the AUC of CATBoost is more than XGBoost . The time take is also less compared to XGBoost classifier. Thus, it can be said that the CATBost better performs for imbalanced dataset as well as balanced dataset. The computational time for XGBoost model which is fine tuned using grid search takes on an average 3 Hrs for 3-fold crass validation which is way to long and also it consumes all the 8 cores. The XGBoost model fine-tuned using random search method considerably takes lesser time compared to grid search but the AUC of this model is less compared to the AUC of the XGBoost model tuned using grid search where the AUC was found to be 0.729 for the balanced dataset. Also, for the unbalanced dataset the XGBoost model tuned using grid search and random search methods dint show much of difference in the AUC values. Hence it can be concluded that the XGBoost model fine-tuned using grid search performed better than the same model fine-tuned using random search. This is because of the values for the parameter that is selected randomly which results in only 18 variants of XGBoost model using different values of hyperparameters which are selected randomly whereas in grid search the parameter values are passed while the model training which results in creating 120 different variants of XGBoost with different subsets of the hyperparameter values . But when considering the computation time random search method is much faster than the grid search approach.

The CATBoost model also showed similar results where the model tuned using grid search showed better performance with AUC of 0.744 compared to random search where the AUC was 0.64 for balanced data. When the same model is built for unbalanced data the AUC of both the variants of CATBoost tuned with grid search and random search respectively had same AUC of 0.658 which means that the parameters that were selected randomly through random search also yielded the same performance as that of the model for which the parameter values were manually set and grid search was employed to search for optimal set of parameters. Also, the time taken to train CATBoost model tuned with random search was much faster, with average computation time 15 Min compared to model tuned with grid search which took on an average 2 Hrs to train. Thus, it can be concluded that the CATBoost model performed best when tuned using grid search by considering the AUC but when the computation time is considered random search method outshines the grid search.

By looking at the accuracies it can be observed for all the variants of XGBoost and CATBoost the accuracies are very high for the imbalanced dataset as the “0” class label is more than “1” the model is tend to classify most of the data as “0” and hence the accuracy is not considered as an accurate measure of metric.

Since there is no rule of thumb as on what the right set of parameters is to be set for model, different values were tested with and finally selected the range of values. From the variable importance of the model it was seen that models used all the features except for the XGBoost model for unbalanced dataset which was tuned using grid search which used 198 features. The most important feature recorded by all the models was var\_81 with maximum gini gain

Thus, it can be concluded that CATBoost model performed best compared to XGBoost in both the hyperparameter search methods where the AUC was 0.744 using grid search and 0.671 using random search. The performance of CATBoost was also better that its counterpart for imbalanced dataset with AUC of 0.658 for both grid search and random search.

## Reflection

As presented in section1 where the works done by many kagglers was reviewed from the analysis it was found that most of them retained the results of the models which were built using imbalanced dataset and concluded that to be the best model including the XGBoost. But the results from an imbalanced dataset is always unreliable. To analysis the performance of XGBoost classifier one of the kernels was selected which followed the similar data processing steps as done in this project. The data is checked for skewness and kurtosis and found to within the acceptable range and also the data is normalized. The data was balanced using Random Under Sampling where the majority class was reduced to have equal data for both the class labels. Grid search technique was used to search for optimal parameters for XGBoost and the model was trained using stratified fold with n value equal to 4. Only one hyperparameter was set which was learning rate to reduce overfitting and the value was 1e-06 and rest of the model parameters were set to default. The grid search showed that the model best performed with the values of learning rate set at 1e-06, gamma = 10, lamda set to 1e-06 and min\_child\_weight = 0.1. The metrics to evaluate the model was set to AUC and the AUC of XGBoost model was found to be 0.829.

The results obtained by the kernel are good but since the method employed is under sampling this results in considerable amount of data loss. Further the grid search performed in only one parameter and the rest are set to default. Also, the default maximum tree depth for XGBoost is 6 and with this small tree not much information can be gained as the learning rate is also close to 0 very small amount of information is extracted from each tree created. Though the AUC is high for this model it is not completely reliable as the data is under sampled and the model is overfitted as the min\_child\_weight is very low which indicates that smaller groups in the tree are created and lesser value results in model to be highly overfitted and it is recommended to have bigger values.

The XGBoost model created in this project is trained using ROSE where the minority class labels data is increase indicating that there is no data loss. Further Grid and Random search techniques shows that the model fine-tuned using grid search gave better performed with AUC of 0.729. The model created has reduced overfitting by using bigger values for min\_child\_weight with value equal to 2 which results in model that is more reliable. The max\_dept parameter also have range of values indicating that more information is extracted from the tree and also the tree created is not too deep which may result in inducing overfitting and increasing complexity.

Thus, it can be said that though the AUC of the XBoost model created in the kernel is more compared to the model created in this model, the overfitting is reduced without any loss of data. Further CATBoost has shown best performance in terms of computational time and AUC. The model can be better fine-tuned using different parameter values and the results are expected to improve.

## Learning Outcomes

There exists 5 types of boosting algorithms GBM, LightGBM, XGBoost, CatBoost and H2o. During the course of this project two of the type of Boosting algorithms were studied namely XGBoost and CatBoost also how any models can be tuned, basically there are two types of methods to search the most optimal parameter values for the model, grid search and random search. Both of these methods were implemented. During the implementation it was observed that there is no set rules on what the values of the parameters should be and the models were built using different combinations of parameter values. There still are other combinations which can be tested upon. Since the model training took almost 3hrs it was experienced that if these models are run on GPU the training will be much faster but due to lack of resource the models were trained using CPU and not GPU. While tuning the hyperparameters it was experienced in may cases where the model was over tuned which resulted in lot of incorrect predictions and also the model was highly overfitted. Hence it is always recommended to start off with smaller values and smaller subsets.

Not all of the boosting algorithms can be easily installed in R by installing the model packages, LightGBM model required to have visual studio installed and the latest Rtools. This particular package had to be installed externally and not in R. Initially the required files were downloaded from gitHub and the r script was built in visual studio but resulted in may errors and hence only implementation of XGBoost and CatBoost was don’t in this project. Different feature selection algorithm was tried and studied namely “Boruta” and was implemented with CPU which was taking more time and hence in future projects this type of feature selection method will be tested using GPU.

References

N, L. (2020). Santander Customer Transaction Prediction Dataset. Retrieved 13 May 2020, from <https://www.kaggle.com/lakshmi25npathi/santander-customer-transaction-prediction-dataset>

Prabhakaran, S. (2020). Caret Package – A Practical Guide to Machine Learning in R – Machine Learning Plus. Retrieved 13 May 2020, from <https://www.machinelearningplus.com/machine-learning/caret-package/>

How to produce a confusion matrix and find the misclassification rate of the Naïve Bayes Classifier?. (2020). Retrieved 13 May 2020, from <https://stackoverflow.com/questions/46063234/how-to-produce-a-confusion-matrix-and-find-the-misclassification-rate-of-the-na%C3%AF>

Usage examples - CatBoost. Documentation. (2020). Retrieved 13 May 2020, from <https://catboost.ai/docs/concepts/r-usages-examples.html#selecting-hyperparameters>

Berhane, F. (2020). Extreme Gradient Boosting with R. Retrieved 13 May 2020, from <https://datascienceplus.com/extreme-gradient-boosting-with-r/>

Beginners Tutorial on XGBoost and Parameter Tuning in R Tutorials & Notes | Machine Learning | HackerEarth. (2020). Retrieved 13 May 2020, from <https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/>

Menardi, G., & Torelli, N. (2012). Training and assessing classification rules with imbalanced data. *Data Mining And Knowledge Discovery*, *28*(1), 92-122. doi: 10.1007/s10618-012-0295-5