# Machine Learning Engineer Nanodegree
# Capstone Project Report

Vamsavardhan Thotakura

June 10, 2019

## 1 Definition

### 1.1 Project Overview

In the financial domain, banks are always looking for ways to help customers understand their financial health and identify which products and services might help them achieve their financial goals. One of the challenges in this business is to be able to get maximum customers registered for all the financial products. While each product has a specific monetary goal and intends to cater to a specific customer segment, it is not possible to go through all the customer data manually to figure out which products are suitable for them. The challenge here is, there are a lot of parameters specific to the customer, which could indicate his interest in a specific product. Given that there are numerous different financial products introduced in the industry continuously, it is essential to develop a model that could determine the customer interest in the specific product based on the various parameters.

In a typical day, I get at least 5 calls from marketing executives regarding various financial products, but most of them are not even relevant to me in any way. They make a call to every other customer in the database with the hope that a given product may be of interest to the customer. This process consumes much time and human resources if only a few of them become the actual customers of the specific product. The motivation of this work to build a model that can learn anonymous numerical data representing the existing customers of a bank and predict if a specific customer would go for that financial transaction.

### 1.2 Problem Statement

The problem statement is taken from the kaggle competition called "Santander-customer-transaction-prediction"[1]. Santander is a Spanish multinational commercial bank[2] and financial services company founded and based in Santander, Spain. The data science team is continually challenging machine learning algorithms, working with the global data science community to make sure they can more accurately identify new ways to solve the most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?. The competition provides An anonymized dataset containing numeric feature variables, the binary target column, and a string ID_code column. The numeric feature variables represent various characteristics of the customer and the financial product. The target variable shall indicate the possibility of the customer making a transaction corresponding to the financial product represented by the features in input, irrespective of the transaction value. There are 200 feature columns in the data, based on which target value has to be predicted by the model.

The Santander bank provided anonymized data set containing 200K records. The data provided for the competition has the same structure as the real data that the bank has to solve this problem of customer transaction prediction. The training data set with 200K records is used to train a machine learning model. This model shall identify the patterns in the data and build the relationship between the patterns and the target variable. Then, the model shall predict the target value for each record in the test data of another 200K records for which target value is not available. The predictions of the test data are submitted to kaggle as csv file to evaluate the model. In this problem, the target variable to be predicted by the model is clearly defined. Hence, one of the supervised machine learning algorithms is appropriate for the solution. The target variable in this problem is a boolean, meaning the model has to predict True or False. Classification class of machine algorithms are the best fit for these characteristics of the problem. The training data can be divided further into the training set and validation set. Validation set shall be used for model evaluation to identify the best model using various classification algorithms like support vector machines, random forest, gradient boosting techniques etc.

## 1.3 Metrics

Following metrics defined[7] for a typical binary classification problem are considered for evaluating the machine learning model

- **Accuracy score**: Accuracy score is the ration of correct predictions to the total number of input samples.

  $$Accuracy\ score = \frac{Number\ of\ correct\ predictions}{Totalnumber\ of\ predictions\ made}$$

  As mentioned in data analysis section, this data has class imbalance where 90% of the samples are classified as False. A naive model that always predict the target as 'False' irrespective of feature values may get an accuracy score of 0.9. But this do not indicate that model performed better. Model actually failed to identify positives as positives.

- **Loglos:** Logarithmic Loss or Log Loss, works by penalising the false classifications.When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

  $$Logarithmic\ loss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{i=1}^{M} y_{ij} * log(P_{ij})$$

  Similar to accuracy score, As logloss penalizes only false classifications, it do not correctly evaluate the model for class imbalanced data .

To overcome the limitations of above metrics and as per the kaggle competition evaluation guidelines[3], **area under roc curve ( AUC - ROC )** is the evaluation metric used to determine how good the model performed on the testing data. AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as

1s. By analogy, Higher the AUC, better the model is at distinguishing between customers making transaction vs not making transaction.

In the implementation, below Sklearn packages are imported and used to caluclate area under RUC curve for model evaluation - -

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
```

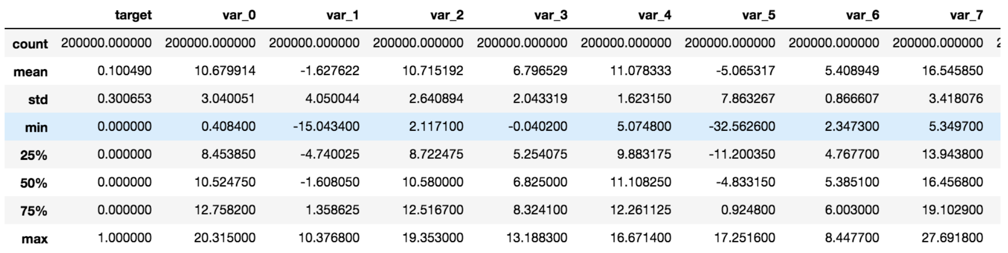
## 2 Analysis

### 2.1 Data Exploration

The data is available in two csv files -

- **train.csv:** The train data set with 202 columns and 200K rows.
  - **'ID_code:'** The index of each record in the data - train_0, train_1,... to train_199999
  - **'Target:'** The boolean target value to be predicted by the model
  - **'var_0, var_1,... to var_199'** - The 200 anonymous numerical input features representing the customer and financial product.
- **text.csv:** The test data set with 201 columns and 200K rows in this file
  - **'ID_code:'** The index of each record in the data - test_0, test_1,... to test_199999
  - **'var_0, var_1,... to var_199'** - The 200 anonymous numerical input features representing the customer and financial product.

Statistics of some of the input features are shown in Figure1



| | target | var_0 | var_1 | var_2 | var_3 | var_4 | var_5 | var_6 | var_7 |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 | 200000.000000 |
| **mean** | 0.100490 | 10.679914 | -1.627622 | 10.715192 | 6.796529 | 11.078333 | -5.065317 | 5.408949 | 16.545850 |
| **std** | 0.300653 | 3.040051 | 4.050044 | 2.640894 | 2.043319 | 1.623150 | 7.863267 | 0.866607 | 3.418076 |
| **min** | 0.000000 | 0.408400 | -15.043400 | 2.117100 | -0.040200 | 5.074800 | -32.562600 | 2.347300 | 5.349700 |
| **25%** | 0.000000 | 8.453850 | -4.740025 | 8.722475 | 5.254075 | 9.883175 | -11.200350 | 4.767700 | 13.943800 |
| **50%** | 0.000000 | 10.524750 | -1.608050 | 10.580000 | 6.825000 | 11.108250 | -4.833150 | 5.385100 | 16.456800 |
| **75%** | 0.000000 | 12.758200 | 1.358625 | 12.516700 | 8.324100 | 12.261125 | 0.924800 | 6.003000 | 19.102900 |
| **max** | 1.000000 | 20.315000 | 10.376800 | 19.353000 | 13.188300 | 16.671400 | 17.251600 | 8.447700 | 27.691800 |

Fig1: Statistics of input features

Below characteristics of the input data are observed -

1. All the 200 input features are of numeric type. There are not categorical or text feature columns.
2. None of feature columns have invalid (NaN) data.
3. The distribution of feature values are comparable to each other.
4. Target feature to be predicted is of boolean type.

5. Only 10% of the training samples have target 'True'. 90% of the have the target 'False'. The indicates that the training samples are not be balanced well w.r.t target. Data augmentation shall is done to overcome this as described in the later sections.

## 2.2 Exploratory Visualization

The anonymity of the input feature names in the data do not leave any possibility of identifying any correlation to target by feature names. There are 200 numerical features. Scatter plots of random few features to the targets do not indicate any specific pattern. Scatter plots of all feature values vs target look similar and are distributed well with the target. Figure2 below represent the scatter plots of randomly selected 9 features. It can be noted that none of them depict any specific pattern.

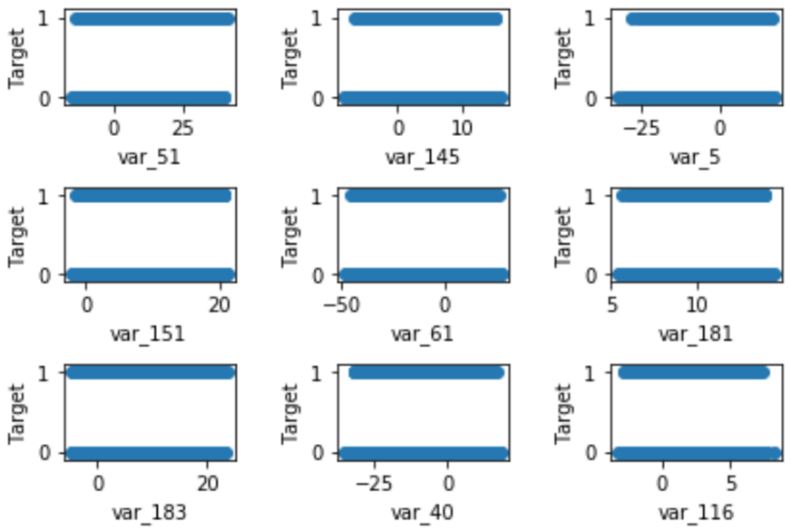


Fig2: Correlation between randomly selected input features and target

As mentioned in previous section, bar plot in Figure3 indicate that training data set samples are not balanced w.r.t target. This is a challenge as any model shall train more to predict "False" than "True".

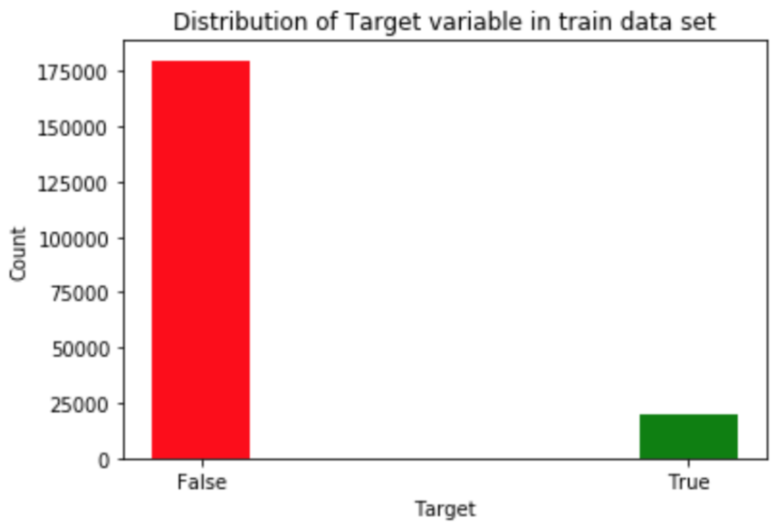


Fig3:Distribution to target variable in train data set

## 2.3 Algorithms and Techniques

The classifier used is LightGBM[4], a gradient boosting framework that uses tree based learning algorithm. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is used is because it focuses on accuracy of results. Since the data has 200K records, Light GBM shall not overfit the data.

Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

Light GBM sklearn API[5] has more than 100 parameters that can tuned. Listed below are the important ones with thier descriptions.

- **max_depth:** It describes the maximum depth of tree. This parameter is used to handle model overfitting. Reduce max depth to avoid overfitting.
- **min_data_in_leaf:** It is the minimum number of the records a leaf may have. The default value is 20, optimum value. It is also used to deal over fitting
- **feature_fraction:** Used when your boosting(discussed later) is random forest. 0.8 feature fraction means LightGBM will select 80% of parameters randomly in each iteration for building trees.
- **bagging_fraction:** specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.
- **early_stopping_round:** This parameter can help you speed up your analysis. Model will stop training if one metric of one validation data doesn't improve in last early_stopping_round rounds. This will reduce excessive iterations.

- **min_gain_to_split:** This parameter will describe the minimum gain to make a split. It can used to control number of useful splits in tree.
- **application:** This is the most important parameter and specifies the application of your model, whether it is a regression problem or classification problem. LightGBM will by default consider model as a regression model.

  regression: for regression

  binary: for binary classification

  multiclass: for multiclass classification problem
- **boosting_type:** defines the type of algorithm you want to run, default=gdbt

  gbdt: traditional Gradient Boosting Decision Tree

  rf: random forest

  dart: Dropouts meet Multiple Additive Regression Trees

  goss: Gradient-based One-Side Sampling
- **num_boost_round:** Number of boosting iterations, typically 100+
- **learning_rate:** This determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Typical values: 0.1, 0.001, 0.003…
- **num_leaves:** number of leaves in full tree, default: 31
- **device:** default: cpu, can also pass gpu. cpu is used for this project
- **metric:** specifies loss for model building. Set to 'auc' for this project.

Gradient boosting decision tree i.e. gbdt is used as boosting type for this project. GBDT is an ensemble model of decision trees, which are trained in sequence. In each iteration, GBDT learns the decision trees by fitting the negative gradients (also known as residual errors).So, the intuition behind gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better. Once we reach a stage that residuals do not have any pattern that could be modeled, we can stop modeling residuals (otherwise it might lead to overfitting). Algorithmically, we are minimizing our loss function, such that test loss reach its minima.

**Algorithm for gradient boosting decision tree:**

- **Step1:** Fit a simple decision tree on data
- **Step2:** Calculate error residuals. Actual target value, minus predicted target value

$$e1 = y - y_{predicted1}$$

- **Step3:** Fit a new model(another decision tree) on error residuals as target variable with same input variables

$$call\ it\ as\ e1_{predicted}$$

- **Step4:** Add the predicted residuals to the previous predictions

$$y_{predicted2} = y_{predicted1} + e1_{predicted}$$

- **Step5:** Fit another model on residuals that is still left. i.e.

$$e2 = y - y_{predicted2}$$

- **Step6:** Predict the target for Validation data and compute the metric passed i.e. ROC-AUC score.
- Step2 to Step6 is called as one **boosting round**
- **repeat steps 2 to 6** until
  - model starts overfitting or the sum of residuals become constant **or**
  - the set number of boosting rounds(*num_boosting_rounds*) is reached **or**
  - until validation score doesn't change for set number of times (*early_stopping_rounds*)

## 2.4 Benchmark

The benchmark model is a random forest classifier with all the default hyper-parameters provided by sci-kit learn library. Any solution to this problem should perform better than the simple random forest classifier with default parameters. Model built using random forest classifier shall predict whether the customer shall transact in future or not on validation data. Given below is the ROC curve for the benchmark model -

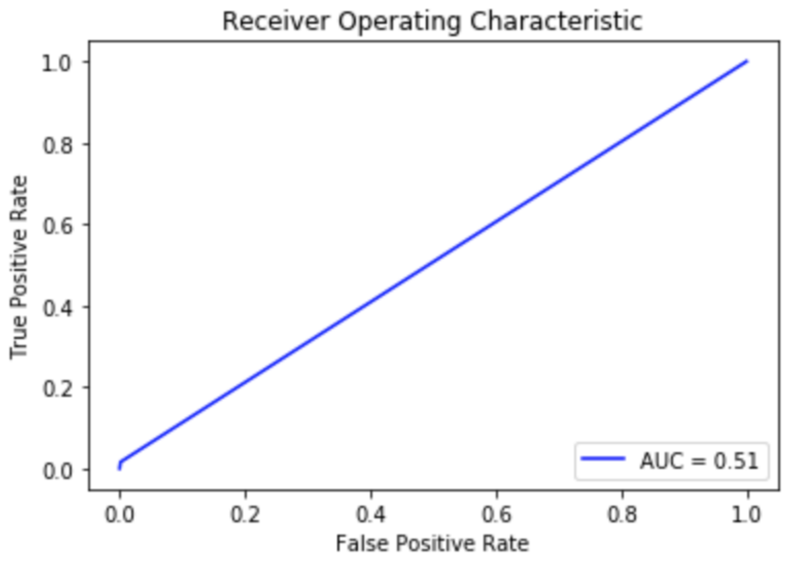


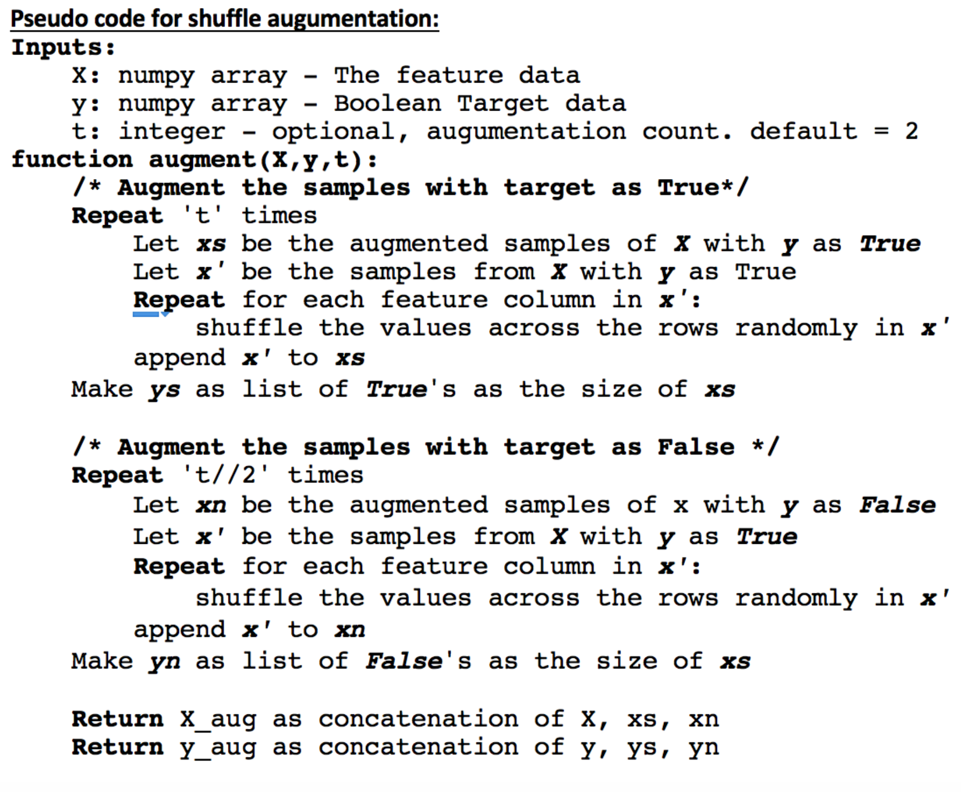
Fig4: ROC curve for benchmark model

To evaluate bench mark model, train data samples are split into train and validation sets. Confusion matrix computed on the predictions by model on validation samples indicated that model failed to classify positives. More False negatives are contributing to poor AUC score.

## 3 Methodology

### 3.1 Data Preprocessing

As mentioned in the exploratory data analysis section, the training data is augmented to make it balanced w.r.t target. This shall overcome the possibility of model training more for 'False' value of target. Infact, this is exactly the case with benchmark model, where model predicted a lot of positives as negatives. The augumentation logic is referenced from the kaggle public kernel[6] by kaggle user 'jiweiliu'. The pseudo code of the algorithm used for data augmentation is given below -



Pseudo code for shuffle augumentation:

```
Inputs:
    X: numpy array - The feature data
    y: numpy array - Boolean Target data
    t: integer - optional, augumentation count. default = 2
function augment(X,y,t):
    /* Augment the samples with target as True*/
    Repeat 't' times
        Let xs be the augmented samples of X with y as True
        Let x' be the samples from X with y as True
        Repeat for each feature column in x':
            shuffle the values across the rows randomly in x'
        append x' to xs
    Make ys as list of True's as the size of xs

    /* Augment the samples with target as False */
    Repeat 't//2' times
        Let xn be the augmented samples of x with y as False
        Let x' be the samples from X with y as True
        Repeat for each feature column in x':
            shuffle the values across the rows randomly in x'
        append x' to xn
    Make yn as list of False's as the size of xs

    Return X_aug as concatenation of X, xs, xn
    Return y_aug as concatenation of y, ys, yn
```

Fig5: Pseudo code for shuffle augmentation

### 3.2 Implementation

- Training and testing data files are loaded into pandas dataframes.
- Extract features dataframe 'X' and target dataframe 'y' from the training data.
- Training data is further divided into train and validation sets with 20% as validation data. The split is done with stratify on target 'y'
- Training sample excluding validation samples are augumented with more samples using the algorithm in above section. The samples with target True are tripled where as the samples with targe False are doubled.
- LGBMModel with default parameters is trained on training data. 0.57 ROC AUC validation score is acheived on validation data.

- Below hyper parameters are set based on the characteristics of the problem and their typical values used from past experience:
  - **Objective** set to *binary*
  - **metric** set to *auc*
  - **boosting_type** set to *gbdt*
  - **learning_rate** set to *0.01*
- Increased number of boosting rounds to 100000 and saw improvement in score to 0.61.
- Below Hyperv parameters of LGBM Model are tuned using grid search with small set of params at a time for multiple times. Listed below are various values of hyper parameters tried using grid search:
  - **num_leaves**: 31(default), 50, 25, 10,15,13
  - **bagging_freq**: 0(default), 10, 5, 3, 20
  - **bagging_fraction**: 1(default), 0.25, 0.5, 0,75
  - **feature_fraction**: 1(default), 0.5, 0.1, 0.01, 0.05
  - **min_data_in_leaf**: 20(default), 200, 150, 100, 50, 80
  - **num_boost_round**: 1000, 10000, 100000

  Achieved an AUC score of 0.77 on validation data set.
- The hyper parameters of the best model are saved in each trial and trained on full training set. This model got a similar score as validation set. **Hyper parameter values of final model:**
  - objective = "binary"
  - scale_pos_weight = 1
  - metric = "auc"
  - boosting = 'gbdt'
  - max_depth = -1
  - num_leaves = 13
  - learning_rate = 0.01
  - bagging_freq = 5
  - bagging_fraction = 0.4
  - feature_fraction = 0.05
  - min_data_in_leaf = 80
  - num_boost_round = 100000

### 3.3 Refinement

In the above mentioned appraoch, there are two issues to be solved

- Model is not trained at all with the same validation samples used for cross validation.
- Training performance is poor as early stopping param of LightGBM model is not used. It took approximately **5-6 hours** to train the model.

To improve on the above issues, Stratified K-fold split is used to create 5 folds. Below steps are performed for each fold.

1. Extract train sample and validation sample
2. Augment only train samples using data augmentation algorithm explained in previous section.
3. Create LGBM model with the tuned hyper parameters.
4. Train the model using augumented data with **early stopping** using auc score on validation data obtained from the split.
5. Predict the target for actual test data
6. Repeat the above three steps for five times on the same augumented data.
7. Make the final prediction on test data as the average of predictions done during all folds.

List of AUC scores on validation data in each fold is below -

- Fold 1 - 0.900684821079561
- Fold 2 - 0.9014076597363921
- Fold 3 - 0.9074235893904575
- Fold 4 - 0.9003955018553478
- Fold 5 - 0.8978753228999384

The time taken for above algorithm including both data augumentation and training for each fold is **2h 18min 43s**

## 4 Results

### 4.1 Model Evaluation and Validation

The predictions of the model and test data provided in the competition are submitted to kaggle and got private AUC score of .89 and public score of 0.90. This score exceeds the goal of 0.75 in the capsone proposal document by a large margin. The use of stratified K fold instead of normal KFold and data augmentation helped the model learn with balanced data w.r.t target variable. The early stopping improved the training performance of the model significantly from 4-5 hours to less than 2 hours on CPU.

### 4.2 Justification

The benchmark solution using random forest for this classification problem scored only 0.5 with ROC-AUC as the metric. The random forest suffered a lot due to imbalanced training data where training sample with target True is hardly 10% of the total training samples. The predictions tends to be False for most cases indicating a large number of false negatives. Gradient boosting models based on tree classifiers couple with data augmentation and stratified 5-fold cross validation scored 0.9 which is 80% improvement over benchmark model performance.

Being able to correctly classify 90% of the customers who would make transactions shall reduce a lot of man power to manually go through the data and contacting each and every potential customer to market financial products. Hence, the model has significantly solved the problem of identifying potential customers who would make a transaction on a particular financial product given the characteristics of the customer and the financial product

## 5 Conclusion

### 5.1 Free-Form Visualization

Below figure demonstrates how this model can be useful to solve the problem statement in financial domain -

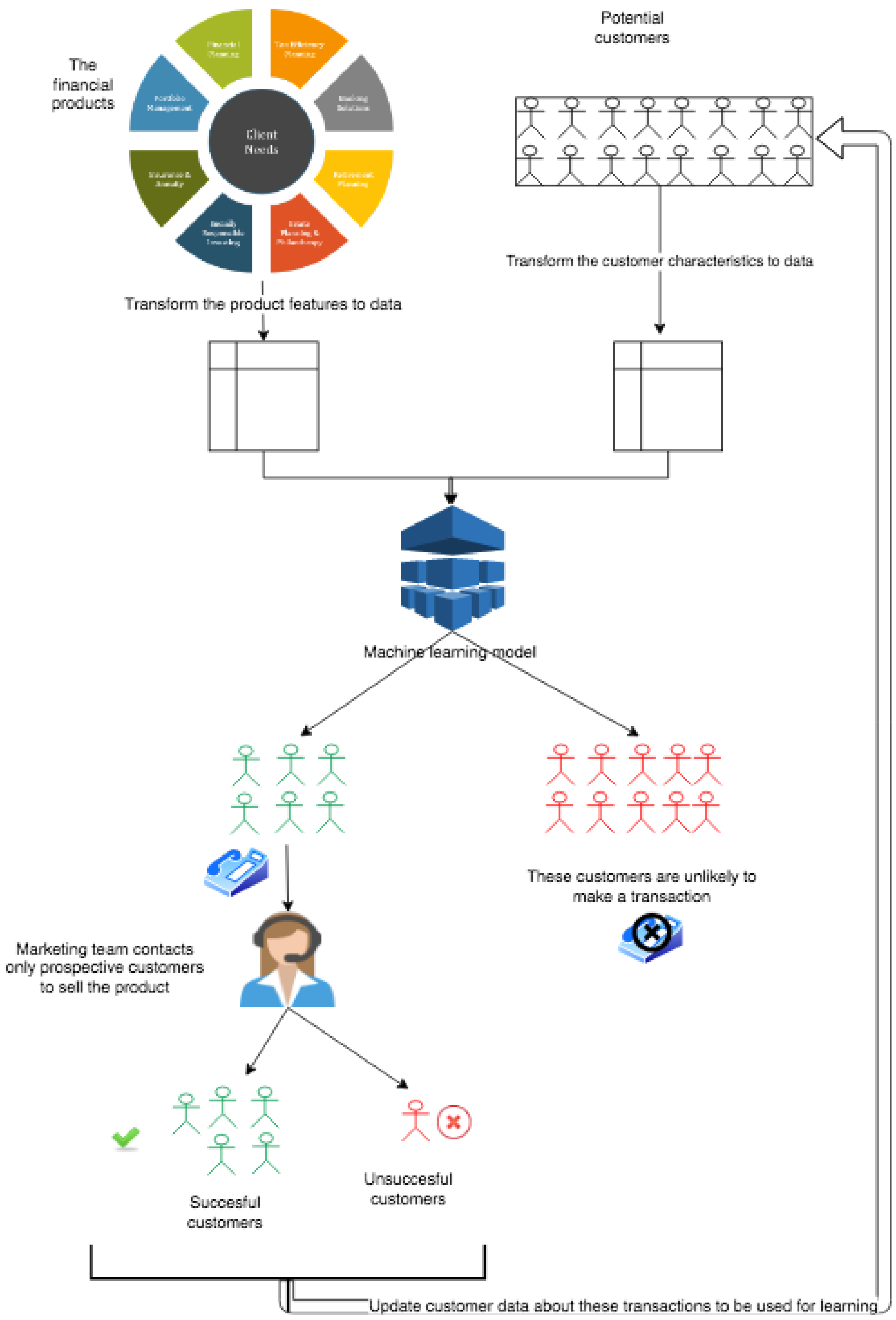

The financial products
Client Needs
Potential customers
Transform the product features to data
Transform the customer characteristics to data
Machine learning model
These customers are unlikely to make a transaction
Marketing team contacts only prospective customers to sell the product
Succesful customers
Unsuccesful customers
Update customer data about these transactions to be used for learning

Fig6: Machine learning model in financial domain

## 5.2 Reflection

One of the key aspects of financial data is security. Bank has lot of data about customers, their transactions, spending patterns, but cannot share the data to preserve confidentiality. That is why the data considered for the machine learning model implemented is anonymized and do not give any clue about what the values actually indicate about the customer or financial product. A manual scrutiny of such data to identify the potential customers is not possible without sharing details of what each value in the data actually represents about customer or the product. This competition is derived from real customer data and the model has considerable accuracy in identifying the potential customers. This would be a significant revenue savings for the banking organisations in-terms man power required for data scrutiny, marketing/sales in finding the prospective customers of the new products.

In this project, there are 200K samples in training set. Each sample consists of 200 numerical values, the anonymous data representing various features of customer data and financial product data. As the data is anonymous, no correlation to target can be found based on feature name and domain specific knowledge. The statistical metrics are calculated for these 200 features values. Scatter plots for random set of features are used to determine that none of these features correlate to target with a specific pattern. It is observed that there is class imbalance with target values. To overcome class imbalance, shuffle data augmentation is done to add more samples with target 1. Stratified 5 fold split is done to ensure training and validation sets are split with same proportion of target classes. For each fold, the training set is augmented with more samples. Augmented data is trained with LightGBM model with early stopping for better training performance. ROC-AUC score is computed on validation set in each fold. The steps of data augmentation, training and predictions is repeated for 5 times in each fold. Predictions for the current fold is computed as the average of predictions for 5 iterations. After 5 folds, average of predictions is computed as final prediction for the test data.

The most interesting part of this project was to deal with anonymized data. All the algorithms that I have done during the nanodegree provide good understanding about what each feature column represent. Given that there are 200 features, it is not possible to find the correlation for each feature in detail. I have spent lot of time training different classification models and tuning hyper parameters with out any significant improvement in score. After realizing the imbalance in the data w.r.t target, I started to explore the data augmentation approaches. That changed the course of my progress towards the solution.

### 5.3 Improvement

I think below algorithms or approaches can be explored for machine learning model

- Improve the performance of data augmentation logic as it may be a bottle neck when data size is 10x.
- Train the regular classification models like Extra Trees, Support vector machines, logistic regression with augmented data.
- XGboost is considered as another powerful classification algorithm along with LightGBM
- ensemble of multiple classification models trained independently
- Given the randomness of the data and assuming that all features are important for prediction, can train an individual LightGBM model per feature i.e. train 200 independent LightGBM classifiers. A voting or similar ensemble model can be used for the final prediction out of predictions by individual classifiers per feature.

## References

[1] https://www.kaggle.com/c/santander-customer-transaction-prediction

[2] https://en.wikipedia.org/wiki/Santander_Bank

[3] https://www.kaggle.com/c/santander-customer-transaction-prediction/overview/evaluation

[4] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-imp

[5] https://lightgbm.readthedocs.io/en/latest/Python-API.html#scikit-learn-api

[6] *LGB 2 leaves augment* https://www.kaggle.com/jiweiliu/lgb-2-leaves-augment

[7] https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38