Machine Learning Engineer Nanodegree

Capstone Project

Rangesh Sripathi
Dec 25, 2020

I. Definition

Project Overview

Customer Churn is one of the most important and challenging problems for businesses such as Credit Card companies(Banks), Telecommunication service providers etc.. Customer Churning is relatively applicable to most customer facing sectors. Customer churn is directly proportional to Company outlook and it is a vital process to understand customers better..

In this project, I have trained and tested a binary classifier capable of predicting churning customers for a credit card company. The final model post evaluation uses the gradient boosting algorithm and was trained on the data provided by Leap-Analytics listed under Kaggle

Problem Statement

A manager at the bank is disturbed with more and more customers leaving their credit card services. They would really appreciate it if one could predict for those who are getting churned so they can proactively go to the customer to provide them better services and turn customers' decisions in the opposite direction.

This is a classic labeled supervised classification task hence algorithms that are related to supervised classification machine learning algorithms such as Logistic Regression, Random Forest,SVM ,KNN,Adaboost and LightGBM could be evaluated and the best model with recall score would be adopted.

Metrics

Models would be evaluated across Precision ,Accuracy , Recall. Model with a higher Recall rate is preferred since non-churning customers predicted as churned would not harm the business.

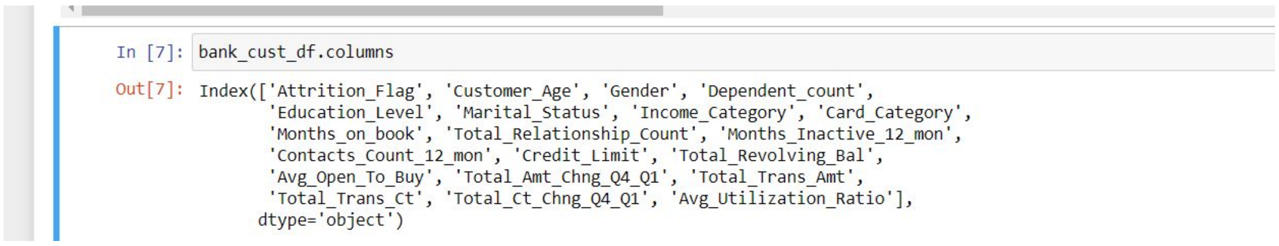
Recall = (TP/TP+FN)

Model thus with higher Recall is the evaluation criteria

II. Analysis

Data Exploration

The dataset provided by LeapAnalytics is on Kaggle of CSV formats which holds 1.44 MB of data. Dataset consists of 10,127 customers with 20 Labels listed below



```
In [7]: bank_cust_df.columns

Out[7]: Index(['Attrition_Flag', 'Customer_Age', 'Gender', 'Dependent_count',
               'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category',
               'Months_on_book', 'Total_Relationship_Count', 'Months_Inactive_12_mon',
               'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
               'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
               'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
              dtype='object')
```

Files :

BankChurnCustomers.csv

Observations:

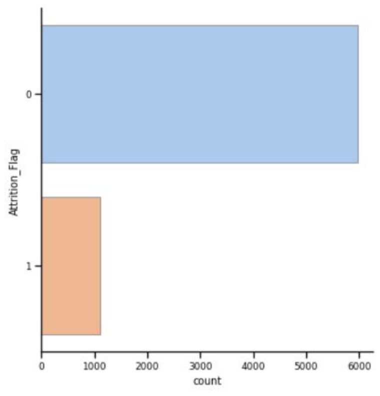
- ❏ We have only 16.07% of customers who have churned - Imbalance Dataset
- ❏ Dataset is very well pre-processed and it does not contain missing values or nulls
- ❏ Attrition_Flag is the Target Variable which we would be using for predicting whether a customer would be churning or non-churning customer
- ❏ Features include both Categorical and Non-Categorical
- ❏ Encoding techniques for Categorical features would be decided upon feature importance against the Target variable.

Exploratory Visualization

Attrition_flag is the Target variable . Listed below are some comparisons between Features and Target variables.

Imbalanced Dataset between Churned VS Un-churned Customers:

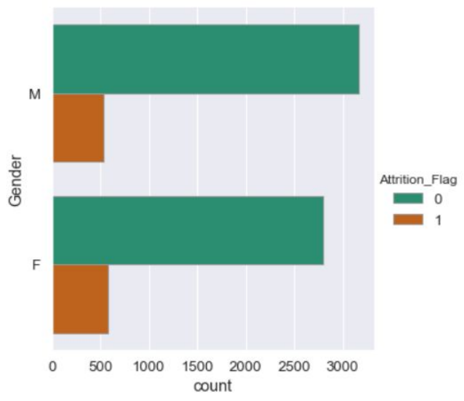
```
In [63]: sns.catplot(y="Attrition_Flag", kind="count",
                     palette="pastel", edgecolor=".6",
                     data=bank_cust_df)

Out[63]: <seaborn.axisgrid.FacetGrid at 0x2582d109dc0>
```



Visualization between Gender and Attrition_Flag

Inference : Churning is equally distributed between Male / Female Customers

```
In [92]: sns.catplot(y="Gender",kind="count",
                     palette="Dark2", edgecolor=".6",hue="Attrition_Flag",
                     data=bank_cust_df)

Out[92]: <seaborn.axisgrid.FacetGrid at 0x25839cc0fd0>
```

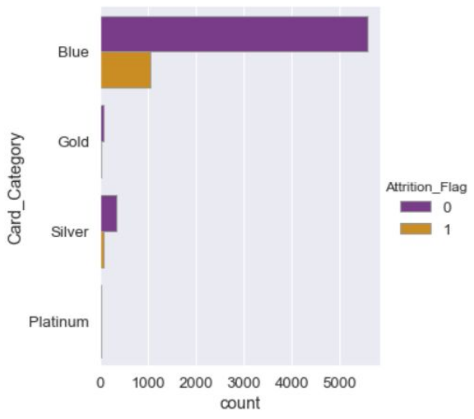


Visualization between Card Category and Attrition_Flag

Inference : Blue Card holder has higher churn

```
n [91]: sns.catplot(y="Card_Category",kind="count",
                    palette="CMRmap", edgecolor=".6",hue="Attrition_Flag",
                    data=bank_cust_df)
```

```
ut[91]: <seaborn.axisgrid.FacetGrid at 0x2583a5d7370>
```

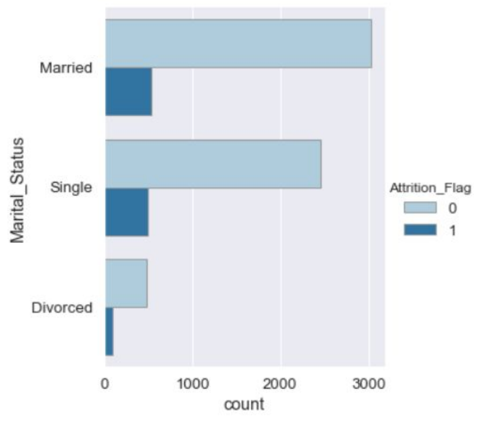


Visualization between Marital Status and Attrition_flag

Inference : Singe / Married are frequent churned customers

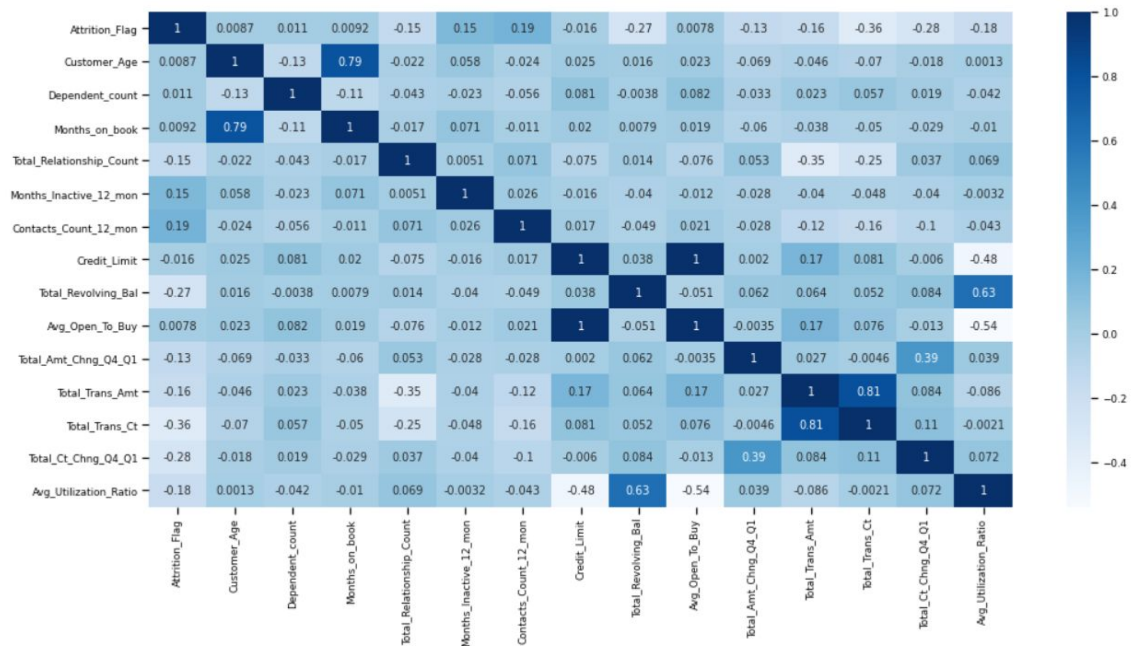
```
In [93]: sns.catplot(y="Marital_Status",kind="count",
                     palette="Paired", edgecolor=".6",hue="Attrition_Flag",
                     data=bank_cust_df)
```

```
Out[93]: <seaborn.axisgrid.FacetGrid at 0x2583a60c5b0>
```



Correlation - Pearson Matrix / HeatMap

```
In [39]: plt.figure(figsize=(15,8))
         sns.set_context(context='notebook',font_scale=.8)
         sns.heatmap(bank_cust_df.corr(method='pearson'),cmap='Blues',annot=True);
         plt.tight_layout()
```



Algorithms and Techniques

This is a Labeled Supervised Binary Classification problem , hence we would be using Logistic Regression, KNN,SVM Classifier ,Random Forest , Ada-Boost and LightBGM. Boosting techniques are essential since there is an imbalance between churned and non-churned customers.

Logistic Regression :

Logistic Regression is a classification algorithm. It estimates discrete values (Binary values like 0/1, yes/no, true/false) based on a given set of independent variable(s). Simply put, it basically, predicts the probability of occurrence of an event by fitting data to a *logit function*. The values obtained would always lie within 0 and 1 since it predicts the probability. since we are dealing with Classification Task Linear Regression is one potential algorithm.

Reference : https://www.edureka.co/blog/classification-algorithms/#logisticregression

Decision Tree Classifier :

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.Random Forest is one such variation of Decision Tree and is an ensemble model used for classification problems.

Reference : https://scikit-learn.org/stable/modules/tree.html

SVM-Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where the number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoiding over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

Reference : https://scikit-learn.org/stable/modules/svm.html

Boosting Techniques :

Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers.We have an Imbalanced Dataset and it is essential we use Boosting technique.

This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances. Boosting is used to reduce bias as well as the variance for supervised learning. It works on the principle where learners are grown sequentially. We will evaluate our model against AdaBoost / LightBGM and pick the one with the best evaluation criteria.

Reference:

https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

https://lightgbm.readthedocs.io/en/latest/

Benchmark

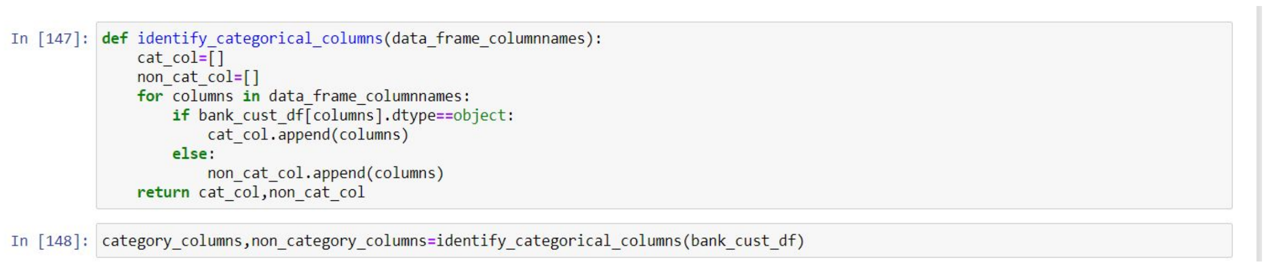
A benchmark for this problem is a classifying model that has a higher recall score. Model with a higher Recall rate is preferred since non-churning customers predicted as churned would not harm the business.

III. Methodology

Data Preprocessing

As previously mentioned, the given dataset had already been well-prepared and processed, the only pre-processing work is on the categorical features(5 Features) which have been dropped since they hold Negative correlation with Target Variable. *Please refer to Fig 1.0 and Fig 2.0 for dropping Categorical features from the dataset.*

I have also classified Features into Categorical and Non-Categorical for Feature Selection and Feature Importance. A method named identify_categorical_columns accepts data_frame_columns as input and returns categorical/non-categorical features.

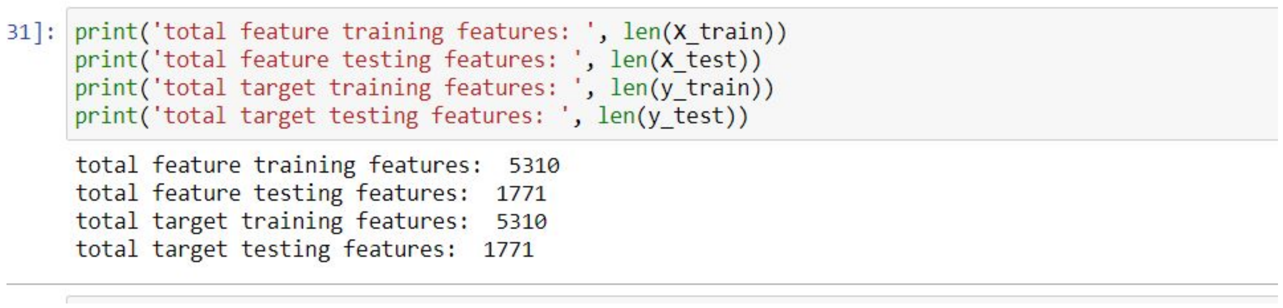


```
In [147]: def identify_categorical_columns(data_frame_columnnames):
              cat_col=[]
              non_cat_col=[]
              for columns in data_frame_columnnames:
                  if bank_cust_df[columns].dtype==object:
                      cat_col.append(columns)
                  else:
                      non_cat_col.append(columns)
              return cat_col,non_cat_col

In [148]: category_columns,non_category_columns=identify_categorical_columns(bank_cust_df)
```

Data conversion

Before we feed the Model , There are few conversions required in the Dataframe , Attrition_Flag holds values as Existing and Attrition Customer as Categorical values , they would be replaced into 0 and 1 . 0- Existing Customer and 1- Attrition Customer.

We would also use Cross-Validation techniques from Sklearn to split the Dataset into Training and Test Dataset.This is essential to know our model behaviour against the actual known data and unknown data .

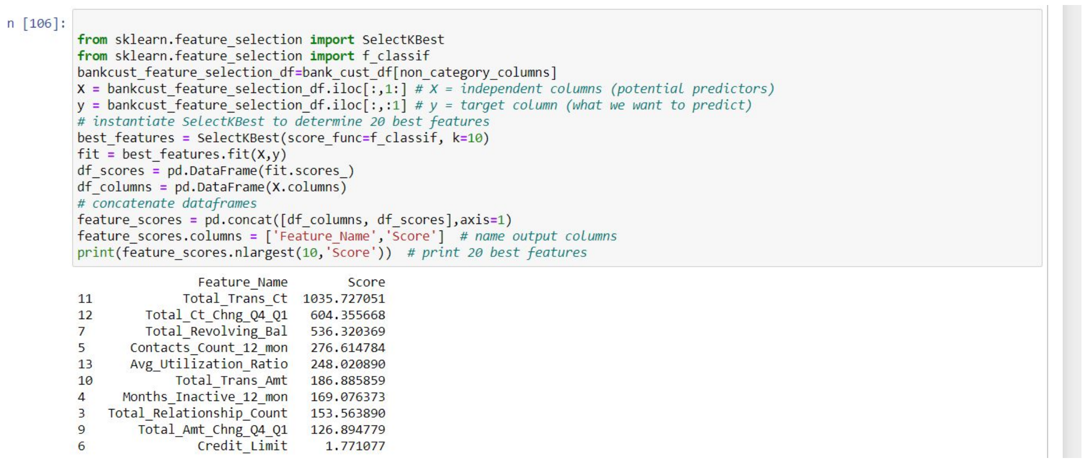


```
31]: print('total feature training features: ', len(X_train))
     print('total feature testing features: ', len(X_test))
     print('total target training features: ', len(y_train))
     print('total target testing features: ', len(y_test))

     total feature training features:  5310
     total feature testing features:  1771
     total target training features:  5310
     total target testing features:  1771
```

Feature Selection and Feature Importance

Sklearn Classification and LightGBM is used for determining Feature selection and Feature Importance.

Feature Selection using **sklearn.f_classif**

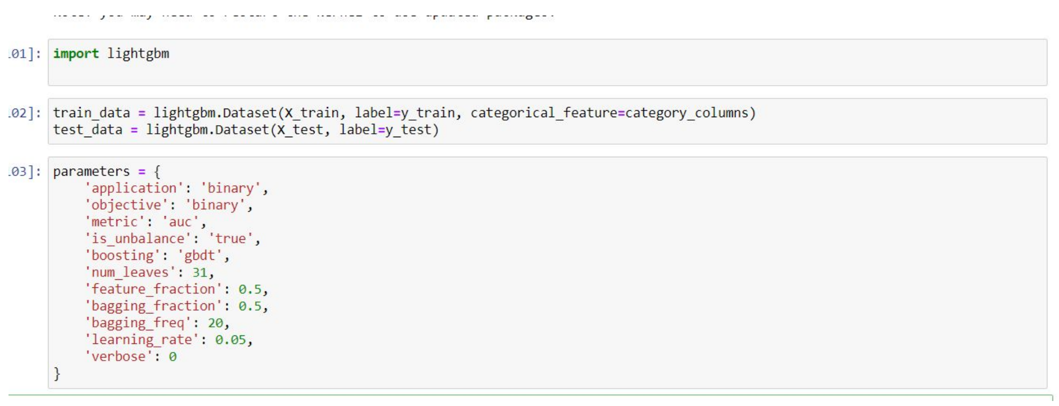


```
n [106]:
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
bankcust_feature_selection_df=bank_cust_df[non_category_columns]
X = bankcust_feature_selection_df.iloc[:,1:] # X = independent columns (potential predictors)
y = bankcust_feature_selection_df.iloc[:,:1] # y = target column (what we want to predict)
# instantiate SelectKBest to determine 20 best features
best_features = SelectKBest(score_func=f_classif, k=10)
fit = best_features.fit(X,y)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(X.columns)
# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(10,'Score'))  # print 20 best features
```

```
               Feature_Name        Score
11            Total_Trans_Ct  1035.727051
12       Total_Ct_Chng_Q4_Q1   604.355668
7        Total_Revolving_Bal   536.320369
5      Contacts_Count_12_mon   276.614784
13     Avg_Utilization_Ratio   248.020890
10           Total_Trans_Amt   186.885859
4     Months_Inactive_12_mon   169.076373
3   Total_Relationship_Count   153.563890
9        Total_Amt_Chng_Q4_Q1   126.894779
6               Credit_Limit     1.771077
```

Fig 1.0-Feature selection using sklearn

Feature Importance using LightGBM includes Categorical / Non-Categorical Features :



```
.01]: import lightgbm

.02]: train_data = lightgbm.Dataset(X_train, label=y_train, categorical_feature=category_columns)
      test_data = lightgbm.Dataset(X_test, label=y_test)

.03]: parameters = {
          'application': 'binary',
          'objective': 'binary',
          'metric': 'auc',
          'is_unbalance': 'true',
          'boosting': 'gbdt',
          'num_leaves': 31,
          'feature_fraction': 0.5,
          'bagging_fraction': 0.5,
          'bagging_freq': 20,
          'learning_rate': 0.05,
          'verbose': 0
      }
```

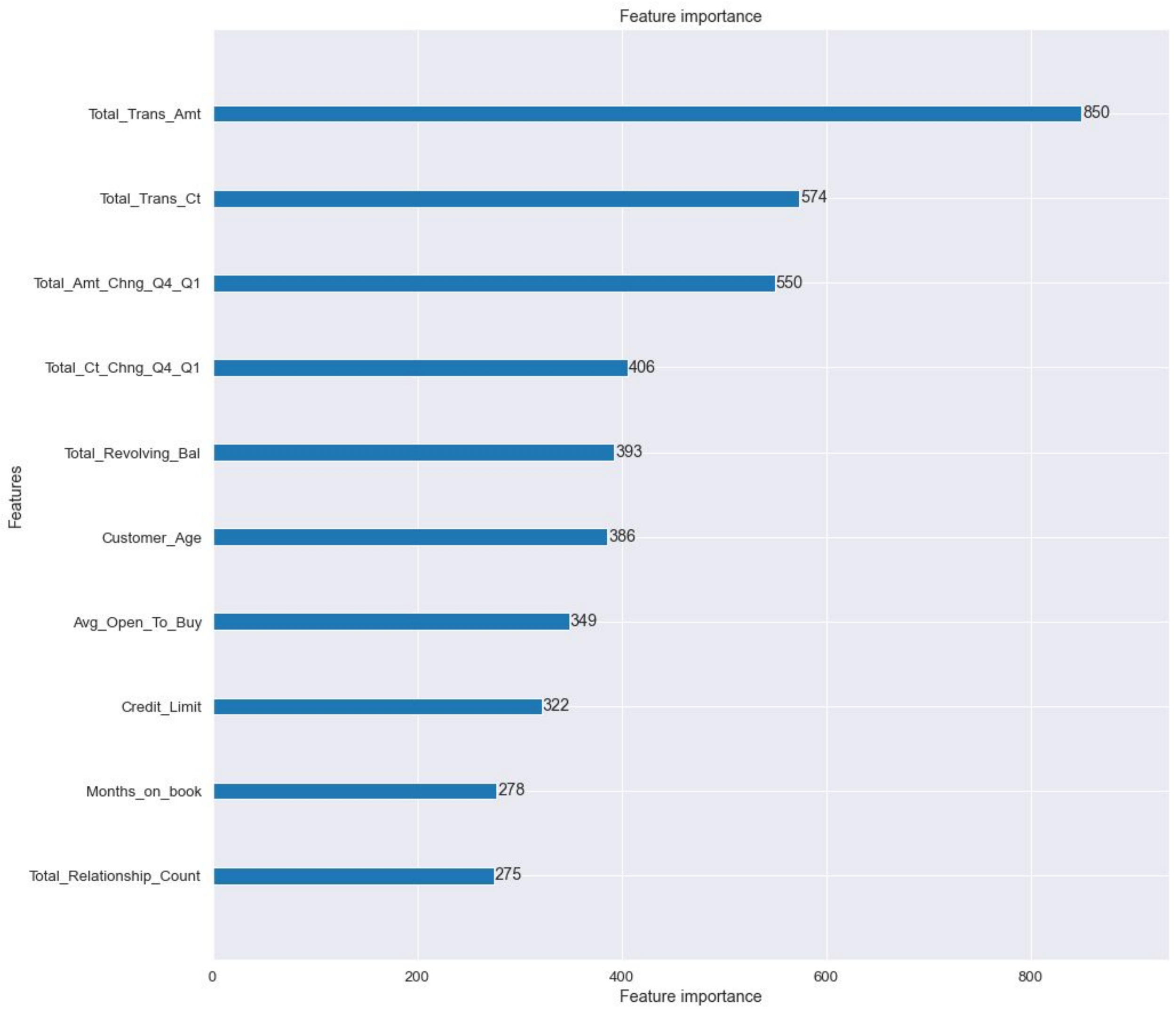
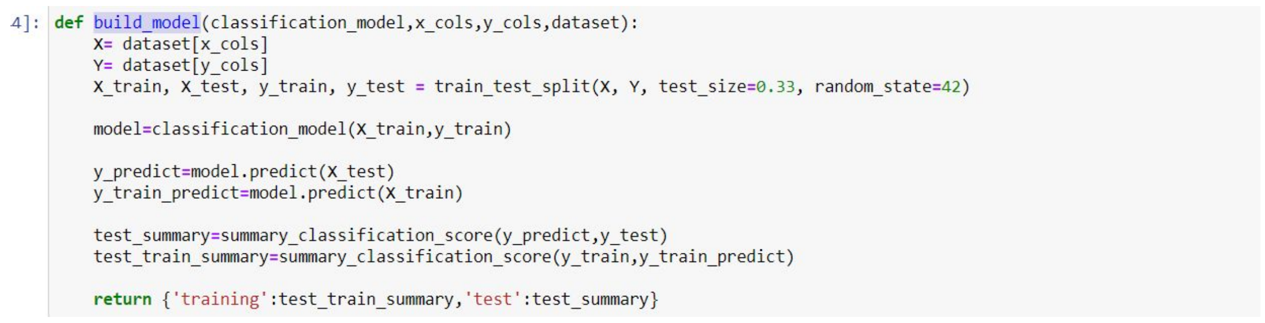


Fig 2.0-Feature Importance using LGBM

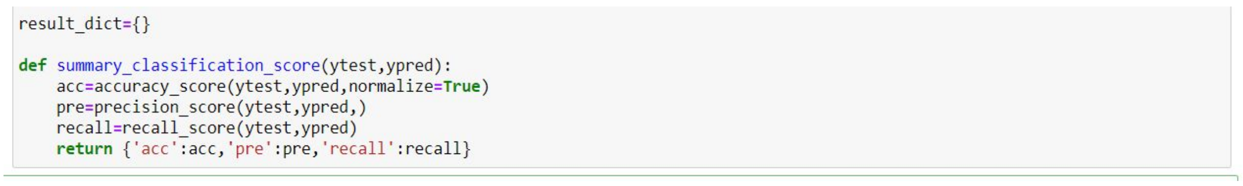
Inference of Feature Selection: Non-Categorical features do not hold any importance and feature selection from sklearn and LightGBM has given evidence ,hence we can safely drop them from the dataframe.

Classification Model Method :

A method named **build_model** that accepts Model,Features,Target,dataframe has been designed that splits the data using Cross- Validate technique of sklearn. Accuracy , Precision,Recall of Model is also evaluated to best determine the suitable Model selection.



```
4]: def build_model(classification_model,x_cols,y_cols,dataset):
        X= dataset[x_cols]
        Y= dataset[y_cols]
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)

        model=classification_model(X_train,y_train)

        y_predict=model.predict(X_test)
        y_train_predict=model.predict(X_train)

        test_summary=summary_classification_score(y_predict,y_test)
        test_train_summary=summary_classification_score(y_train,y_train_predict)

        return {'training':test_train_summary,'test':test_summary}
```



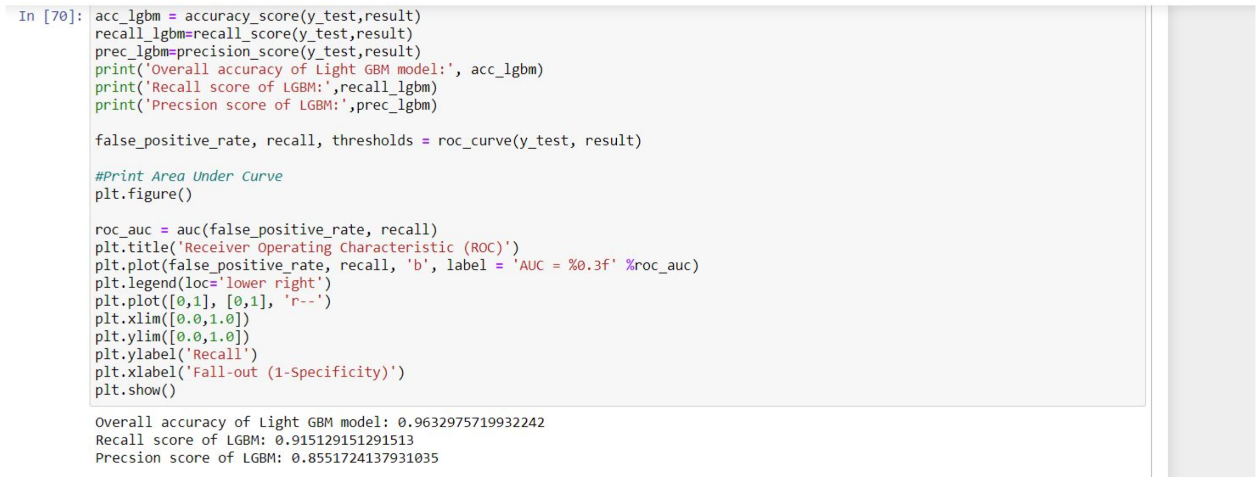
```
result_dict={}

def summary_classification_score(ytest,ypred):
    acc=accuracy_score(ytest,ypred,normalize=True)
    pre=precision_score(ytest,ypred,)
    recall=recall_score(ytest,ypred)
    return {'acc':acc,'pre':pre,'recall':recall}
```

Classification Algorithm used and their scores across Train and Test data are listed below

Model	**Train Evaluation**			**Test Evaluation**		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Logistic Regression	0.8572934233	0.6294964029	0.2336448598	0.8596491228	0.2527472527	0.6216216216
SVM-Linear	0.872470489	0.6493775934	0.4178905207	0.8818998716	0.4368131868	0.6913043478
KNN	0.9190556492	0.7892234548	0.6648865154	0.893025246	0.5906593407	0.6803797468
Random Forest	0.8908094435	0.9529411765	0.3244325768	0.8887462559	0.293956044	0.9727272727
AdaBoost	0.9645868465	0.9040333797	0.867823765	0.9507916132	0.8214285714	0.8567335244

From the above result ,there is clear evidence that Boosting and Tree-Based Decision classifiers could yield better results.

Evaluation with LightBGM



```
In [70]: acc_lgbm = accuracy_score(y_test,result)
         recall_lgbm=recall_score(y_test,result)
         prec_lgbm=precision_score(y_test,result)
         print('Overall accuracy of Light GBM model:', acc_lgbm)
         print('Recall score of LGBM:',recall_lgbm)
         print('Precsion score of LGBM:',prec_lgbm)

         false_positive_rate, recall, thresholds = roc_curve(y_test, result)

         #Print Area Under Curve
         plt.figure()

         roc_auc = auc(false_positive_rate, recall)
         plt.title('Receiver Operating Characteristic (ROC)')
         plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
         plt.legend(loc='lower right')
         plt.plot([0,1], [0,1], 'r--')
         plt.xlim([0.0,1.0])
         plt.ylim([0.0,1.0])
         plt.ylabel('Recall')
         plt.xlabel('Fall-out (1-Specificity)')
         plt.show()

         Overall accuracy of Light GBM model: 0.9632975719932242
         Recall score of LGBM: 0.915129151291513
         Precsion score of LGBM: 0.8551724137931035
```

We could see our Dataset performs best with **LightGBM with a Recall score of 92% ,Accuracy of 96% Approximately.**

Hyperparameters used with LightGBM :

'application': 'binary' - Classification is based on binary
'objective': 'binary', -Classification is based on binary
'metric': 'auc', - Area under the curve
'is_unbalance': 'true', - Determines the imbalance set
'boosting': 'gbdt'- Default boost
'num_leaves': 31,- Leaf nodes
'feature_fraction': 0.5,- Bagging/Boosting fraction
'bagging_fraction': 0.5,- Bagging/Boosting fraction
'bagging_freq': 20,
'learning_rate': 0.05,- Learn rate
'verbose': 0

IV. Results

Model Evaluation and Validation

We have subjected our Dataset to a variety of classification algorithms(Listed below are Evaluation Score) and it gives us clear evidence that Boosting Algorithms and Decision Tree based algorithms outperforms the other models due to im-balance in Data.

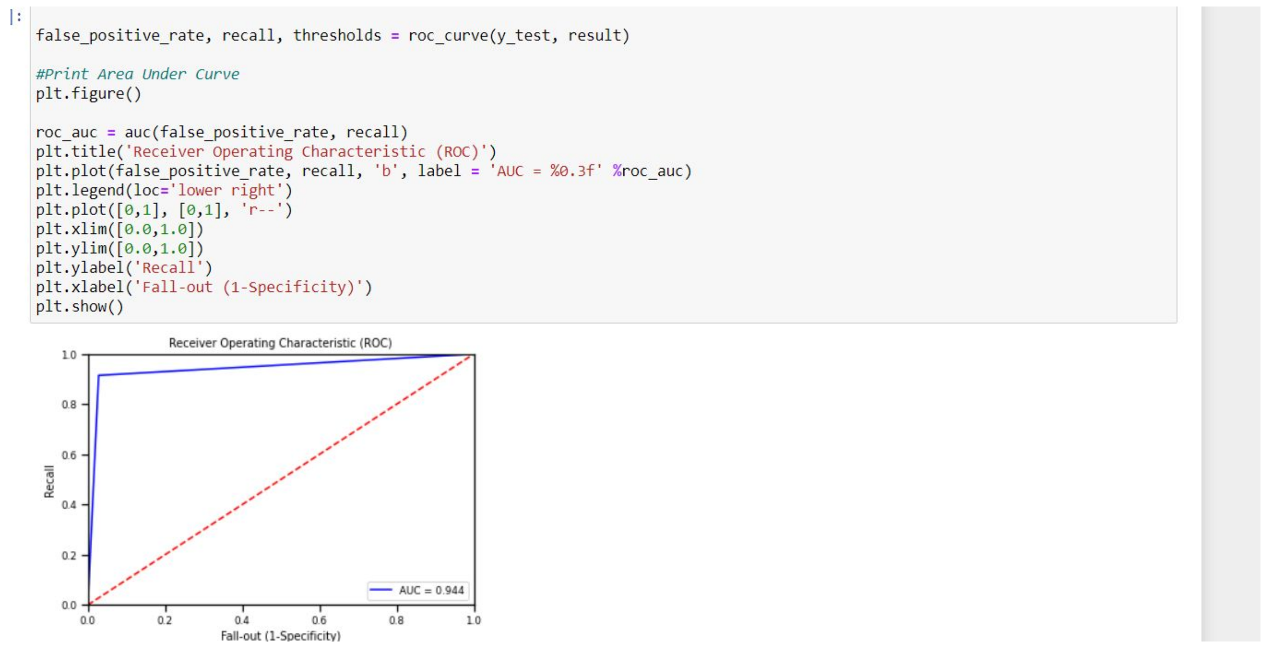
Model	**Train Evaluation**			**Test Evaluation**		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Logistic Regression	0.857293	0.629496	0.233644	0.859649	0.252747	0.621621

	4233	4029	8598	1228	2527	6216
SVM-Linear	0.872470489	0.6493775934	0.4178905207	0.8818998716	0.4368131868	0.6913043478
KNN	0.9190556492	0.7892234548	0.6648865154	0.893025246	0.5906593407	0.6803797468
Random Forest	0.8908094435	0.9529411765	0.3244325768	0.8887462559	0.293956044	0.9727272727
AdaBoost	0.9645868465	0.9040333797	0.867823765	0.9507916132	0.8214285714	0.8567335244

When the Dataset is subject to LightGBM with appropriate Hyperparmeters ,it outperforms the rest of the model and has a good Recall score of 92% with Test and Train Dataset and I would recommend using **LightGBM Binary Classifier as appropriate Model for Credit Card Churn Prediction** .

Justification

Based upon the evaluation Decision Based Tree Algorithms and Boosting techniques had helped our Model to achieve comparably a pretty good Recall Score , With Evidences shared above LGBM - Classifier outperform the other models with Recall score of 92% . RCO Curve is attached for reference with Model accuracy of 94%.



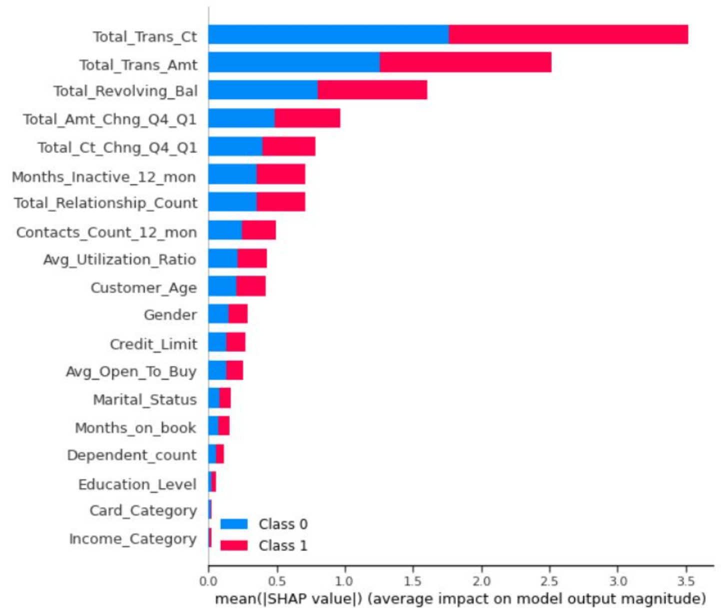
```
false_positive_rate, recall, thresholds = roc_curve(y_test, result)

#Print Area Under Curve
plt.figure()

roc_auc = auc(false_positive_rate, recall)
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(false_positive_rate, recall, 'b', label = 'AUC = %0.3f' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.ylabel('Recall')
plt.xlabel('Fall-out (1-Specificity)')
plt.show()
```

V. Conclusion

Free-Form Visualization

I have used Shap Tree Explainer that emphasises Feature Magnitude against the Target. This shows us the clear evidence and Justification of Features that has been provided as Input for Training and Prediction.

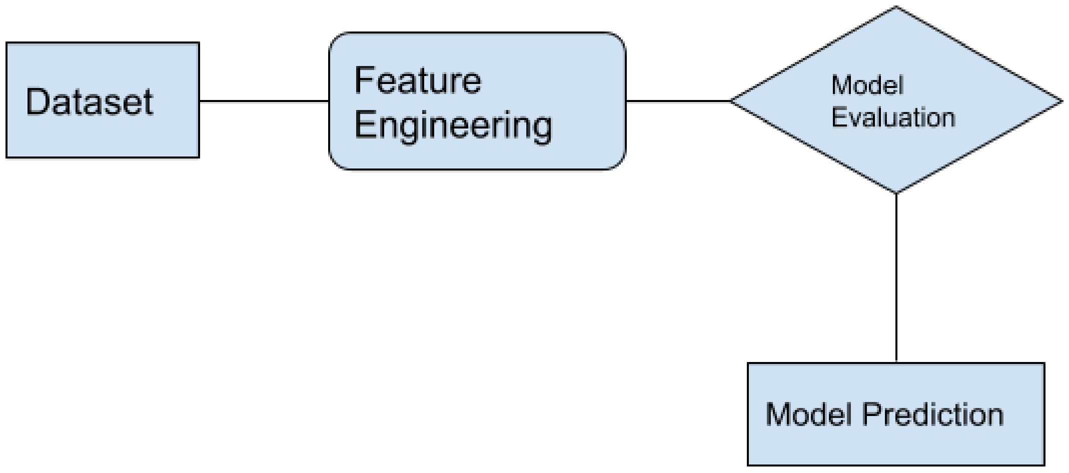
```
In [251]: shap.summary_plot(shap_values, X_test)
```



Reflection

End-to-end problem solution

Goal of the project was to develop a Churn Predictor system for Credit Cards and the challenge was with an Imbalance dataset. To summarize the end to end flow would look like below



I worked on Dataset,FeatureSelection,EDA and the Interesting part was to balance the dataset and adopt boosting techniques. I also evaluated Dataset with Different Classification algorithms and finally with benchmarking ,I chose LightGBM Binary classifier as the perfect model that solves the problem of predicting churn customers.

Improvement

I would like to see the behavior of model on large scale dataset , i.e we have a 10k customer dataset. It would be nice to see how the model would behave when it comes to handling large dataset, say few GBs.