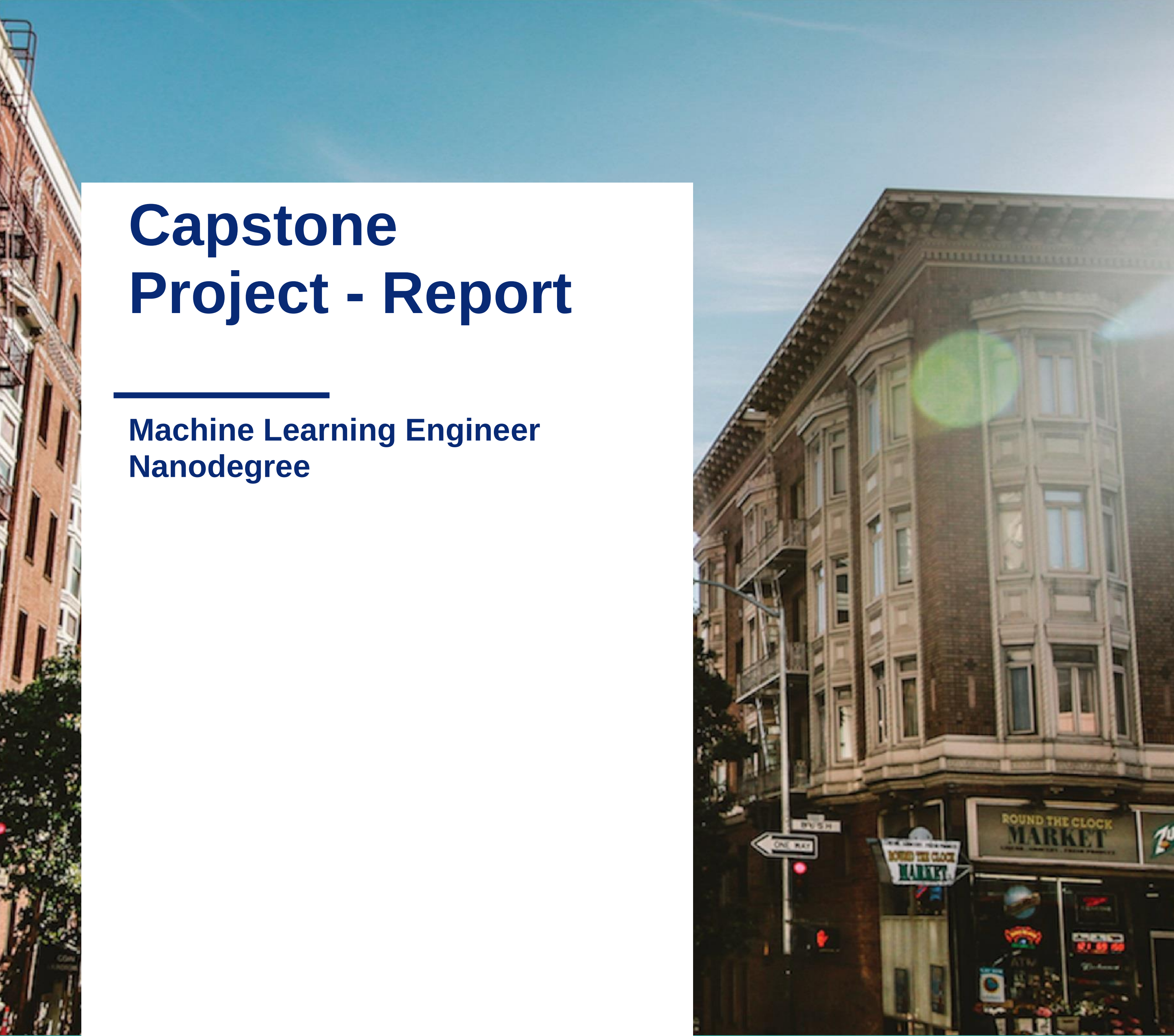


Capstone Project - Report

Machine Learning Engineer Nanodegree

1. Project Overview

1.1 Domain background

This project is mainly focused on Banking and Financial industry Customer satisfaction. As like other industries, Customer satisfaction is very important for any bank to success with given technology revolutions and change in ways of banking.
Unhappy customers rarely voice their dissatisfaction before leaving. So, it's important to understand the customer satisfaction levels in bank and update the products/services accordingly to maintain good market share.

This project will be mainly focused on Santander bank, please refer the following link to know more about the bank **Banco Santander - Wikipedia**

1.2 Problem Statement

The objective of this project is to identify dissatisfied customers early in their relationship with use of machine learning algorithms. Doing so would allow Santander to take proactive steps to improve a customer's happiness before it's too late. Solutions from this project will be feed into Kaggle Competition hosted by Santander

1.3 Datasets and Inputs

Santander customer data points would be used to predict customer satisfaction as highlighted in the competition, details as follows

Two datasets are provided with Target Variable as 0/1 - It equals one for unsatisfied customers and 0 for satisfied customers.

- train.csv - the training set including the target
- test.csv - the test set without the target

Data will be sourced from Kaggle - **Santander Customer Satisfaction | Kaggle**

1.4 Solution Statement

The sophisticated Machine model will be built to predict the customer satisfaction and also the drivers of unsatisfaction through EDA on the data points provided. This will help the bank to take right proactive steps

1.5 Benchmark Model

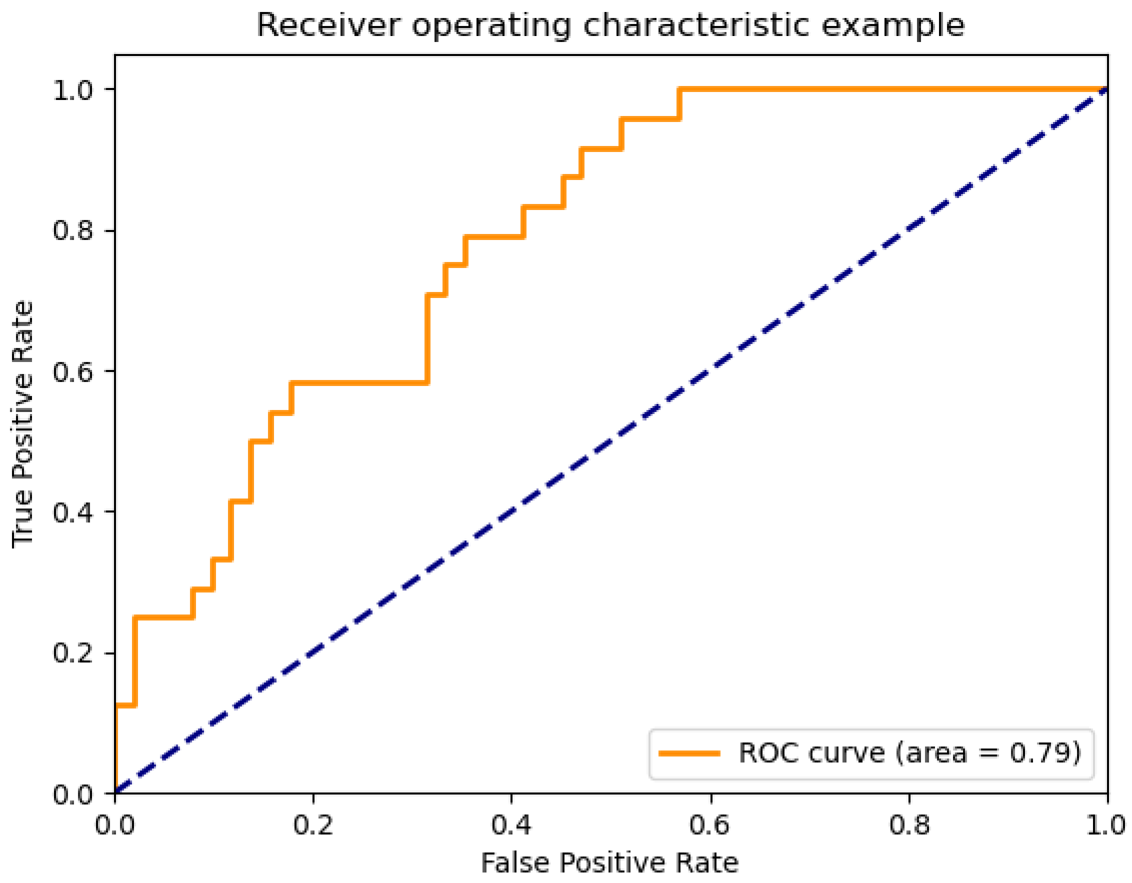
We would be using logistic regression model as a bench mark model with AUC of at-least 50%.

1.6 Evaluation Metrics

Evaluation metrics for the outcome on this project is area under the ROC curve (AUC).

AUC is calculated from a graphical plot curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one.

Example –



2. Analysis – report

2.1 Data exploration and visualization

Since this project is based on competition hosted in Kaggle by Santander, we will be using the datasets provided as part of competition which are

i. train.csv - the training set including the target
ii. test.csv - the test set without the target
iii. sample_submission.csv - a sample submission file in the correct format

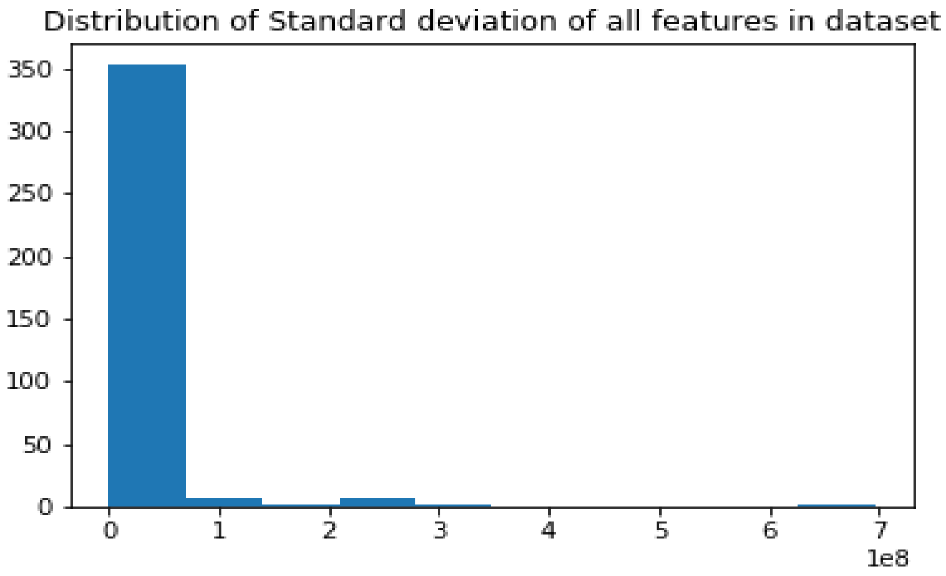
Uploaded this data to notebook instance for the purpose of this project, but datasets can be downloaded from Kaggle URL - **Santander Customer Satisfaction | Kaggle**

There are 370 features in total which are all anonymized (not including the 'TARGET' feature). We will try to understand features based on their distribution

But since there are many features in the dataset, following methods are applied to reduce the dimensions in dataset

2.1.1 - Drop features which are having zero standard deviation

If features have constant values, will drop them from our dataset as these features has zero variance which doesn't add any value to the model predictability power



We can see few features with zero variance from above graph

Code snippet to create a list of features with Zero STD to drop
drop_columns = summary_df.iloc[2][summary_df.iloc[2] == 0].index.to_list()

Identified columns to drop – 34 features

['ind_var2_0', 'ind_var2', 'ind_var27_0', 'ind_var28_0', 'ind_var28', 'ind_var27', 'ind_var41', 'ind_var46_0', 'ind_var46', 'num_var27_0', 'num_var28_0', 'num_var28', 'num_var27', 'num_var41', 'num_var46_0', 'num_var46', 'saldo_var28', 'saldo_var27', 'saldo_var41', 'saldo_var46', 'imp_amort_var18_hace3', 'imp_amort_var34_hace3', 'imp_reemb_var13_hace3', 'imp_reemb_var33_hace3', 'imp_trasp_var17_out_hace3', 'imp_trasp_var33_out_hace3', 'num_var2_0_ult1', 'num_var2_ult1', 'num_reemb_var13_hace3', 'num_reemb_var33_hace3', 'num_trasp_var17_out_hace3', 'num_trasp_var33_out_hace3', 'saldo_var2_ult1', 'saldo_medio_var13_medio_hace3']

We have 337 features in our train dataset after dropping the above columns

2.1.2 - Drop duplicate columns

We often find that there would be duplicate columns in datasets provided in Kaggle competitions

Function to find out duplicates in our dataset

```
def dropDuplicates(df):
    '''Function to find duplicate columns in dataframe
       df: dataset'''
    # Declare a list to append the results
    duplicate_columns = []
    # Temp Variable - to store number of columns
    tot_cols = df.shape[1]
    # will reiterate through sequential findings
    for base_col in range(tot_cols):
        col_sel = df.iloc[:,base_col]
        # iterate through all columns
        for compare_with in range(base_col+1, df.shape[1]):
            compare_col = df.iloc[:,compare_with]
            if col_sel.equals(compare_col):
                duplicate_columns.append(df.columns.values[compare_with])
    return duplicate_columns
```

There are 29 duplicate features found in dataset which are

['ind_var29_0', 'ind_var29', 'ind_var13_medio', 'ind_var18', 'ind_var26', 'ind_var25', 'ind_var32', 'ind_var34', 'ind_var37', 'ind_var39', 'num_var29_0', 'num_var29', 'num_var13_medio', 'num_var18', 'num_var26', 'num_var25', 'num_var32', 'num_var34', 'num_var37', 'num_var39', 'saldo_var29', 'saldo_medio_var13_medio_ult1', 'delta_num_reemb_var13_1y3', 'delta_num_reemb_var17_1y3', 'delta_num_reemb_var33_1y3', 'delta_num_trasp_var17_in_1y3', 'delta_num_trasp_var17_out_1y3', 'delta_num_trasp_var33_in_1y3', 'delta_num_trasp_var33_out_1y3']

We have 308 features in our train dataset after dropping duplicate columns

2.1.3 – Check if any features have null values

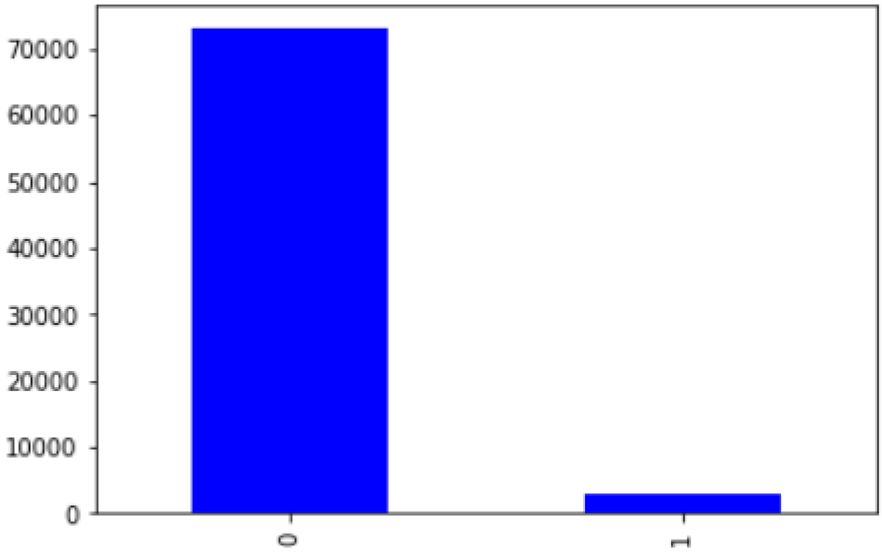
If there are null values in dataset, we either populate the features with reasonable representation such as mean, median..etc or drop features if null value percentage is high

```
# Chekc if any variables has null values
# Store in temporary dataset, as we can't view all columns with Jupyter limitations
Ds = train_df.isnull().sum()/train_df.shape[0]
# Filter only variables with atlest one null value
Ds[Ds > 1]
```

Output is null, so there are no null values for any features in the dataset

2.1.4 – Distribution of target variable

Satisfied Customers percentage – **96.04%**
Unsatisfied Customers percentage – **3.96%**



Our target variable is biased, have only 3.96% of unsatisfied customers in our dataset. Hence, we should factor this in our model development

Since there are c300 features in our training dataset and that too all are anonymized, will explore the variables further after dropping less important variables

2.2 – Dimensionality reduction

As there c300 features in our training data, Dimensionality reduction methods play an important role here as training models with c300 features would be associated with lot of cost.

There are many dimensionality reduction methods such as PCA, SelectKbest, Chi..etc which help us to capture the variance in our training data with less features compare to the original set of features.

We would be using Extra Trees Classifier in our project to find out top important features in our dataset as this algorithm is highly efficient in classification problems and associates with less cost

2.2.1 Overview of ExtraTreesClassifier

Extremely Randomized Trees Classifier (Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output its classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. (source - geeksforgeeks.org)

2.2.2 application of ExtraTreesClassifier

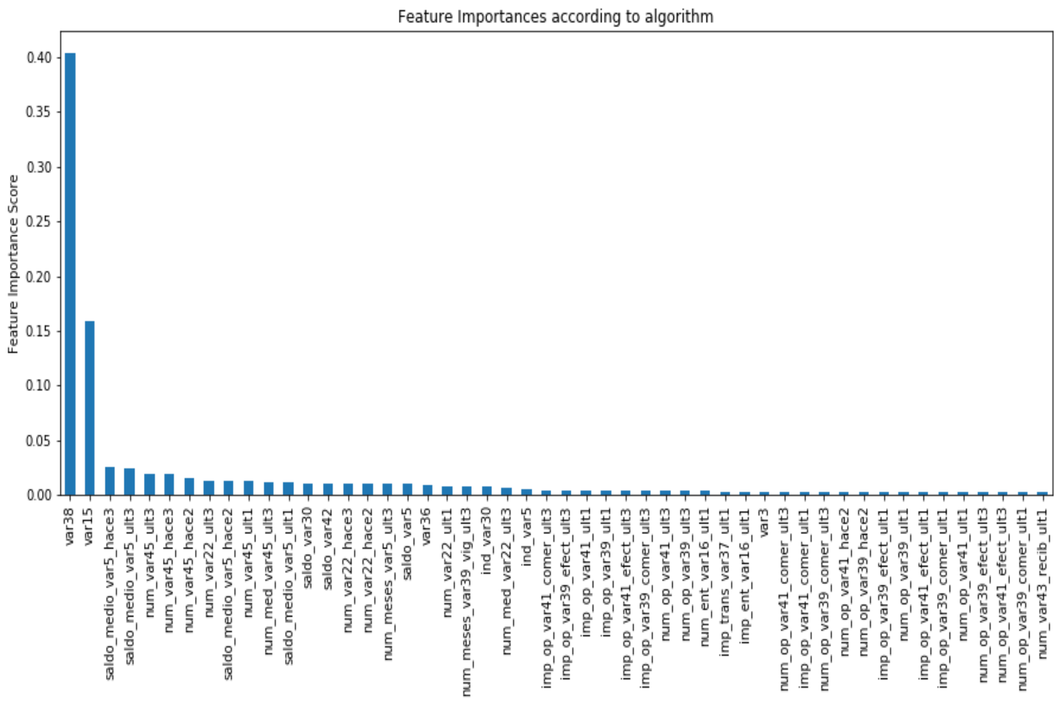
We have referred Sckit-learn package to use this method in our project, more detail at [Scikit-learn - Extra trees classifier]

Code Snippet

```
## # Feature selection with ExtraTreesClassifier
clf = ExtraTreesClassifier(random_state = 69)
selector = clf.fit(train_df.drop(['TARGET'],axis=1),train_df['TARGET'])

# plot most important features
feat_imp = pd.Series(clf.feature_importances_, index = train_df.drop(['TARGET'],axis=1).columns.values).sort_values(ascending = False)
# Visualise Top 50 features
feat_imp[:50].plot(kind = 'bar', title = 'Feature Importances according to algorithm',
                figsize = (15, 8))
plt.ylabel('Feature Importance Score')
plt.subplots_adjust(bottom = 0.3)
plt.show()
```

Top 50 features based on importance score returned by Extra Trees Classifier



Let’s explore few top important variables to build an understanding on data

2.2.3 Exploration of top 2 variables based on feature importance score

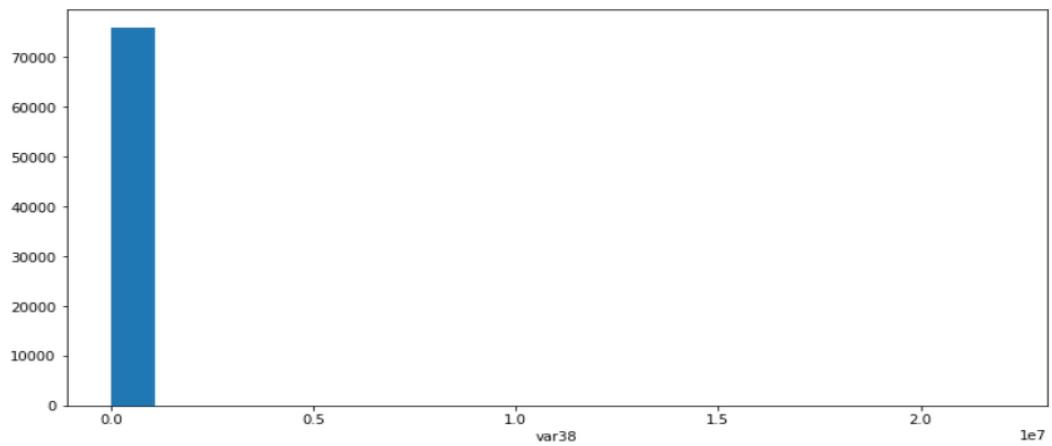
2.2.3.1 Var38

This feature has Standard deviation 182,665 which represents the data is highly distributed.

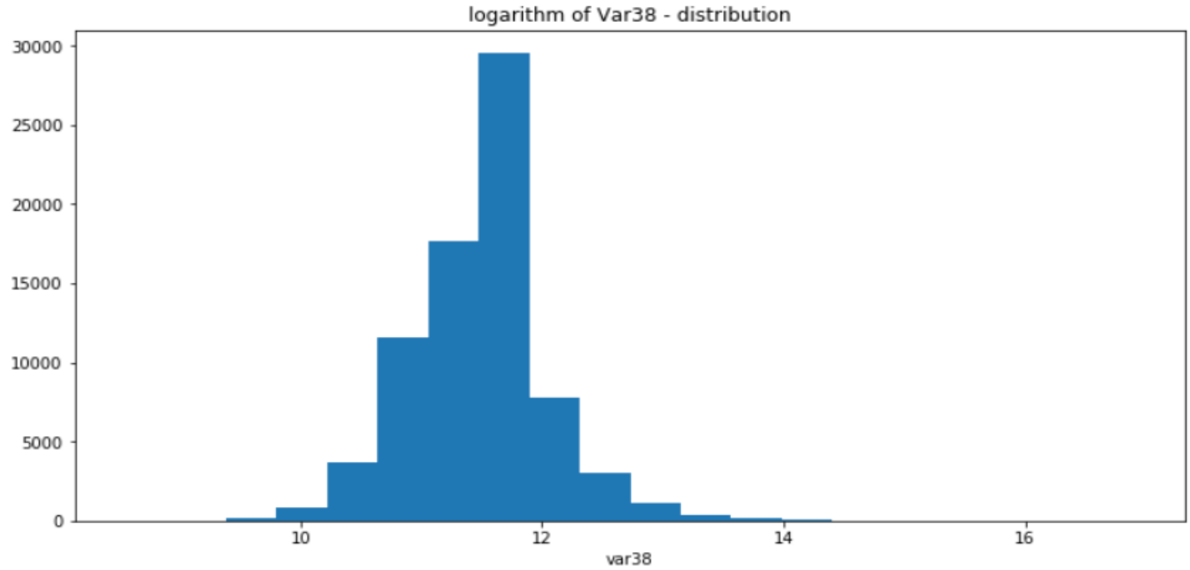
Here is the distribution of Var38

Minimum of Var38: 5,163.75

Maximum of Var38: 22,034,738.76



Since there are few big values for Var38, we can't view the distribution properly in histogram.
Let's apply logarithm on this feature for better visualization of distribution

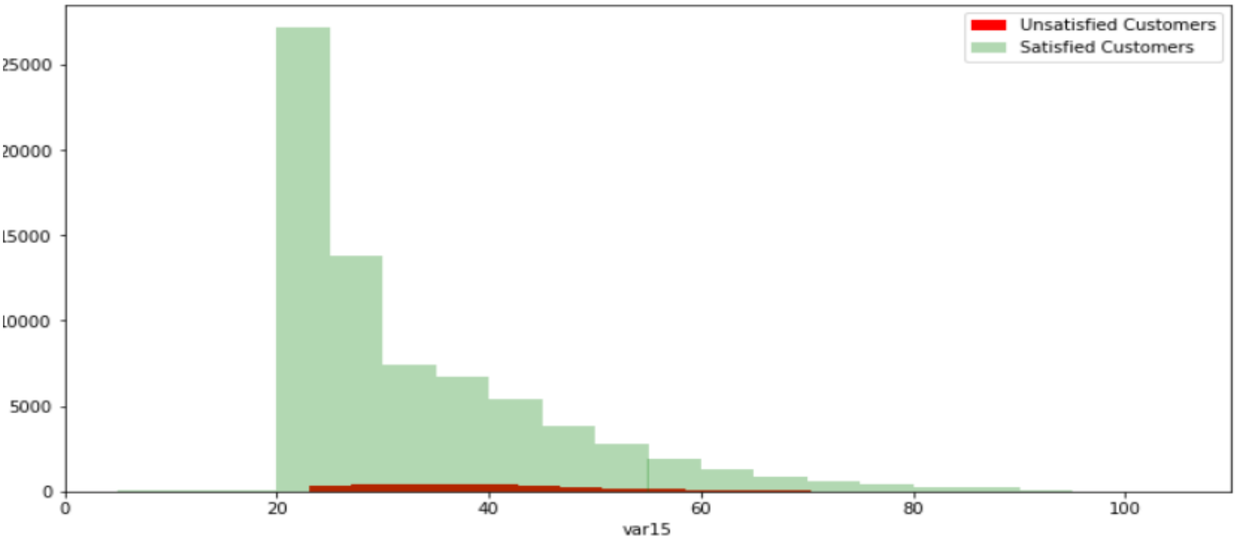


From this distribution and by applying business knowledge, we can understand this variable represents the balance of any banking product such as savings or mortgage...etc

2.2.3.1 Var15

Distribution metrics of this variable

Minimum of var15: 5
Maximum of Var38: 105
Standard deviation of var15: 12.96



Based on distribution, this variable might be representing the age of customer.
With that assumption, we can see less un-satisfaction in age old customers compare to middle age

2.2.3 Model dataset preparation

Based on feature importance score, we would be using top 40 features for model building as we see the importance score is very less/ going down after that.
The top 40 number is decided based on many experimentations results I.e., AUC

```
# Select top 40 features for model development , as these features collectively explain the decent variance in target variable;
Top_Features = feat_imp[:40].index.to_list()
```

Features finalized for model building

['var38', 'var15', 'saldo_medio_var5_hace3', 'saldo_medio_var5_ult3', 'num_var45_ult3', 'num_var45_hace3', 'num_var45_hace2', 'num_var22_ult3', 'saldo_medio_var5_hace2', 'num_var45_ult1', 'num_med_var45_ult3', 'saldo_medio_var5_ult1', 'saldo_var30', 'saldo_var42', 'num_var22_hace3', 'num_var22_hace2', 'num_meses_var5_ult3', 'saldo_var5', 'var36', 'num_var22_ult1', 'num_meses_var39_vig_ult3', 'ind_var30', 'num_med_var22_ult3', 'ind_var5', 'imp_op_var41_comer_ult3', 'imp_op_var39_efect_ult3', 'imp_op_var41_ult1', 'imp_op_var39_ult1', 'imp_op_var41_efect_ult3', 'imp_op_var39_comer_ult3', 'num_op_var41_ult3', 'num_op_var39_ult3', 'num_ent_var16_ult1', 'imp_trans_var37_ult1', 'imp_ent_var16_ult1', 'var3', 'num_op_var41_comer_ult3', 'imp_op_var41_comer_ult1', 'num_op_var39_comer_ult3', 'num_op_var41_hace2']

2.3 – Model Development and Results

2.3.1 Splitting data for Validation and Testing

Since our data is biased, we have used stratified split of our train dataset into Validation and test dataset

We have used the following composition of data in our model

- Training – 60% of data
- Validation – 20 % data
- Testing – 20 % data

2.3.2 XGBoost

We have used two different modelling techniques to predict the probability of unsatisfied customer, details as follows

- Linear learner (Binary- Logistic)
- XGboost

XGboost modelling results as follows

2.3.2.1 Model training

XGBoost model instance used in our project

```
# Create XGboost model
xgb = Estimator(container,
                role,
                instance_count=1,
                instance_type='ml.m4.xlarge',
                output_path='s3://{}/{}/output'.format(bucket_name, prefix))

xgb.set_hyperparameters( max_depth=5,
                        min_child_weight=1,
                        gamma= 0,
                        subsample= 0.8,
                        colsample_bytree= 0.8,
                        objective= 'binary:logistic',
                        nthread=4,
                        eval_metric ='auc',
                        scale_pos_weight=0.396,
                        seed=4242,
                        num_round=30)
# scale_pos_weight - 0.396 represents % of unsatisfied customers in our train dataset
# after few experimentations - decided with this hyperparameters set
```

```
# Fit XGboost model
xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

Hyperparameters are decided based on experimentation

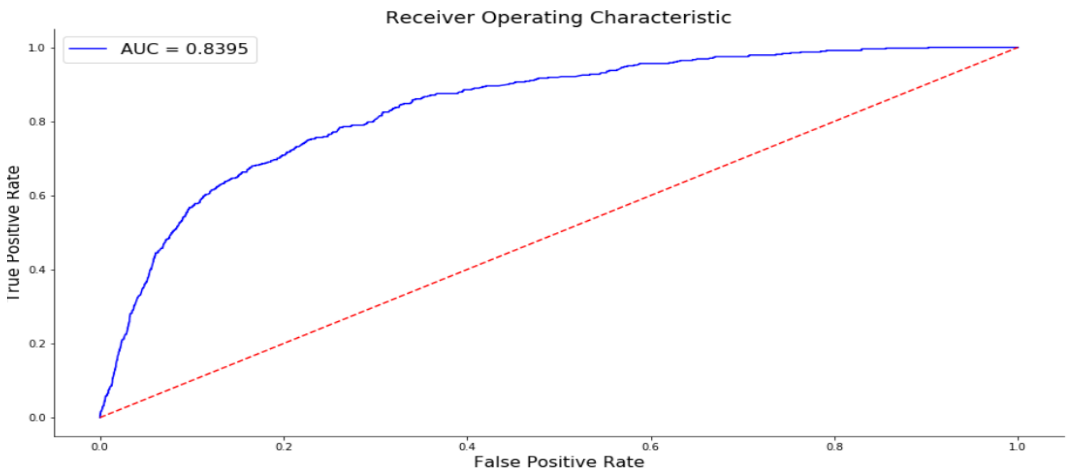
2.3.2.1 Model deployment and results

Once the model is trained, we have deployed the model to create an active endpoint.

```
# Deploy XGboost model
xgb_predictor = xgb.deploy(initial_instance_count=1,
                           instance_type='ml.m4.xlarge')
```

We have supplied our test dataset to endpoint in batches of 500 records to predict the probability of unsatisfied customer

Results – 83.95% AUC



2.3.3 Linear learner

2.3.3.1 Model training

Linear learner model instance used in our project

```
linear = LinearLearner(role=role,
                               train_instance_count=1,
                               train_instance_type='ml.c4.xlarge',
                               predictor_type='binary_classifier',
                               output_path=output_path,
                               sagemaker_session=session,
                               epochs=15,
                               binary_classifier_model_selection_criteria='f_beta',
                               positive_example_weight_mult='auto')
```

We have used 70% of training and 30% test data composition for this model

```
# Convert data into numpy;
train_x_np = X_train_ls.values.astype('float32')
train_y_np = y_train_ls.values.astype('float32')
# create RecordSet
formatted_train_data = linear.record_set(train_x_np, labels=train_y_np)
```

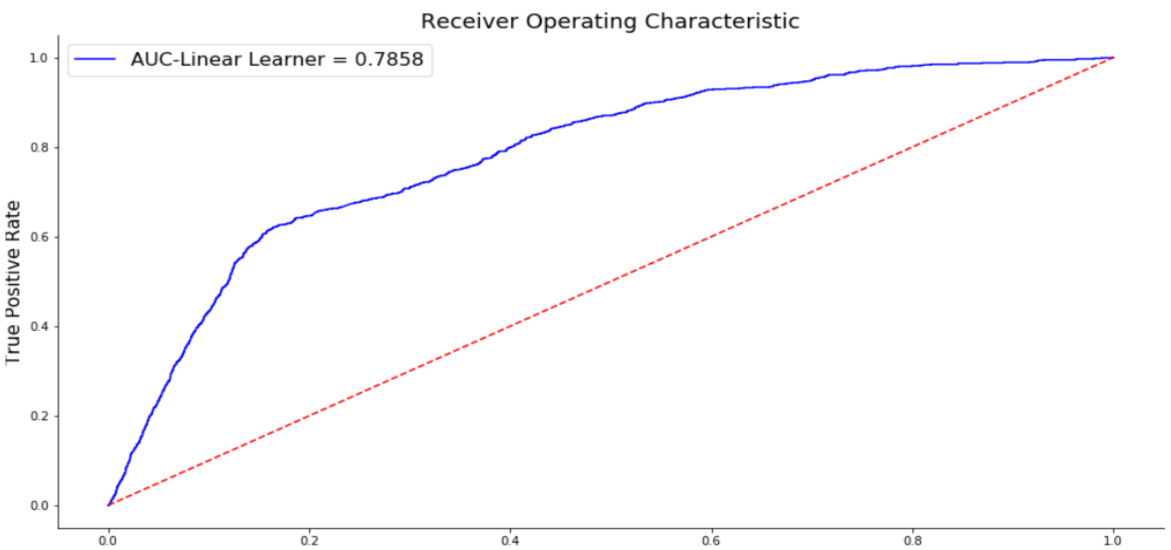
2.3.3.1 Model deployment and results

Once the model is trained, we have deployed the model to create an active endpoint.

```
%%time
# train the estimator on formatted training data
linear.fit(formatted_train_data)
```

We have supplied our test dataset to endpoint in batches of 100 records to predict the probability of unsatisfied customer

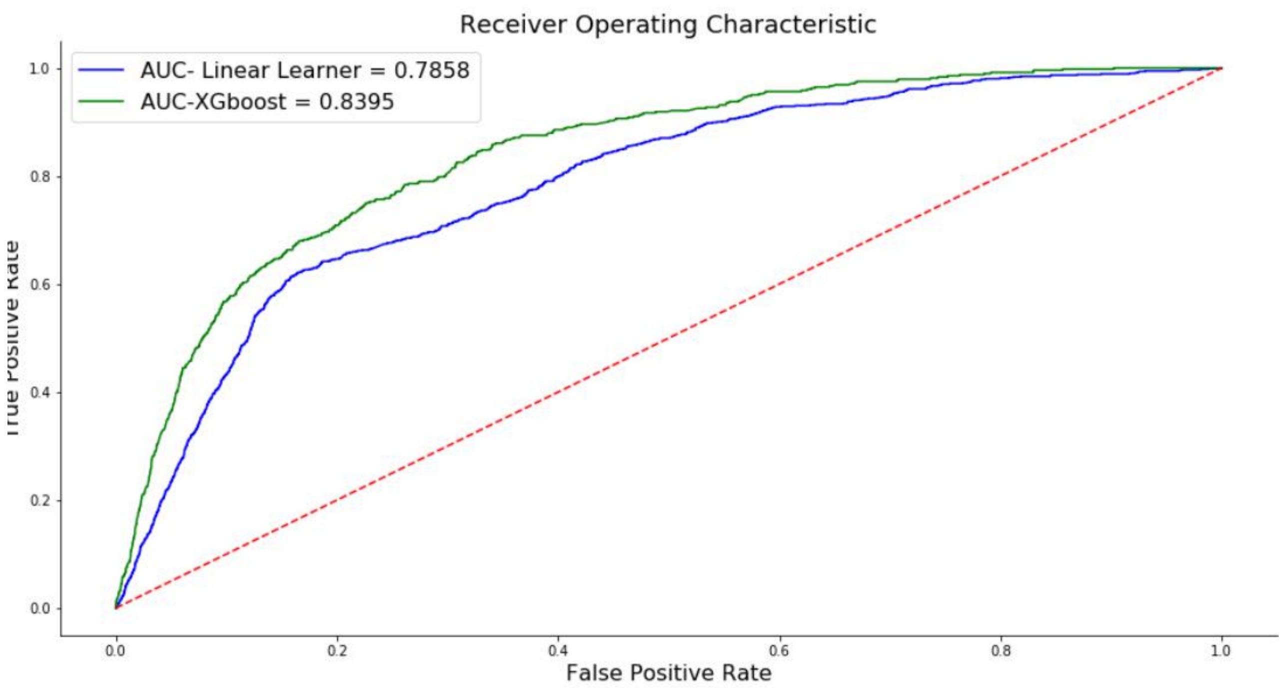
Results – 83.95% AUC



2.3.4 Comparison of models

Since XGboost is delivering highest AUC over Linear learner, we have picked XGBoost as final model for our project.

Here is the comparison of both models interms of our objective metric (i.e. AUC)



2.3.5 Submission to Kaggle

Submitted the three solutions to Kaggle with variations on XGboost based on test dataset provided by Kaggle

2 submissions for somprasad theegala   Sort by Private Score

All   Successful   Selected

Submission and Description	Private Score	Public Score	Use for Final Score
submission (1).csv 4 hours ago by somprasad theegala add submission details	0.82034	0.83081	☐
submission (2).csv 4 hours ago by somprasad theegala XGboost - Hyperparameters tuned	0.81934	0.83233	☐
submission.csv a day ago by somprasad theegala XGboost with feature selection	0.80897	0.81904	☐

2.3.5 Improvements and Next steps

We may attain better accuracy with more powerful hyperparameters tuning on XGboost or application of ensemble techniques on this data.

If this would have been real data and project, we would create proper API to return the probability scores from our applications

References

1. Sagemaker Examples
   amazon-sagemaker-examples/introduction_to_applying_machine_learning at master · aws/amazon-sagemaker-examples (github.com)

2. Extra trees Classifier
   sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.24.1 documentation (scikit-learn.org)