**Santander Customer Transaction**

**Prediction**– Data Science Project

# Abstract: Santander Bank is a wholly owned subsidiary of Spanish Santander Group. It is headquartered at Boston, Massachusetts, United States. We are given an anonymized dataset containing numeric feature variables. Since, it is confidential financial data, variable names are completely masked. We need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted. Various ML models were implemented and tuned to achieve high performance in terms of the evaluation metric “AUC-ROC score” for this classification problem.

# Project Plan:

CRISP-DM (Cross Industry Standard Process for Data Mining) methodology will be followed to tackle this **Santander Customer Transaction Prediction Project**.

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment

**Business/Data Understanding:**

Business and Data understanding for this particular dataset was very challenging because of the anonymized data without feature names.

From the problem statement, we could infer that the classification is for customers who make a specific transaction. We could observe class 0 in the target would be customers who didn’t make the transaction and class 1 is the minority class of customers who did make the transaction.

Since there is a target class imbalance, accuracy wouldn’t be a good evaluation metric. Precision and Recall for the minority class should be considered. Area under the ROC curve would be more ideal than accuracy as an evaluation metric.

Exploratory Data Analysis to be done to obtain even better Data understanding

**Data Preparation:**

There are 200K observations in both train.csv and test.csv.

Data preparation steps should include the following:

Sampling, Missing value analysis, Outlier Analysis, Feature Engineering, Feature Scaling

**Modelling**:

5 classification models were implemented namely:

* Logistic Regression
* Decision Trees
* Naive Bayes
* Random Forest
* XGBoost

Model development was carried out from simple to complex starting with a basic Logistic Regression.

**Evaluation:**

The models were evaluated based on their AUC-ROC score.

**Deployment:**

Deployable R script and .py script were prepared using the finalized model.

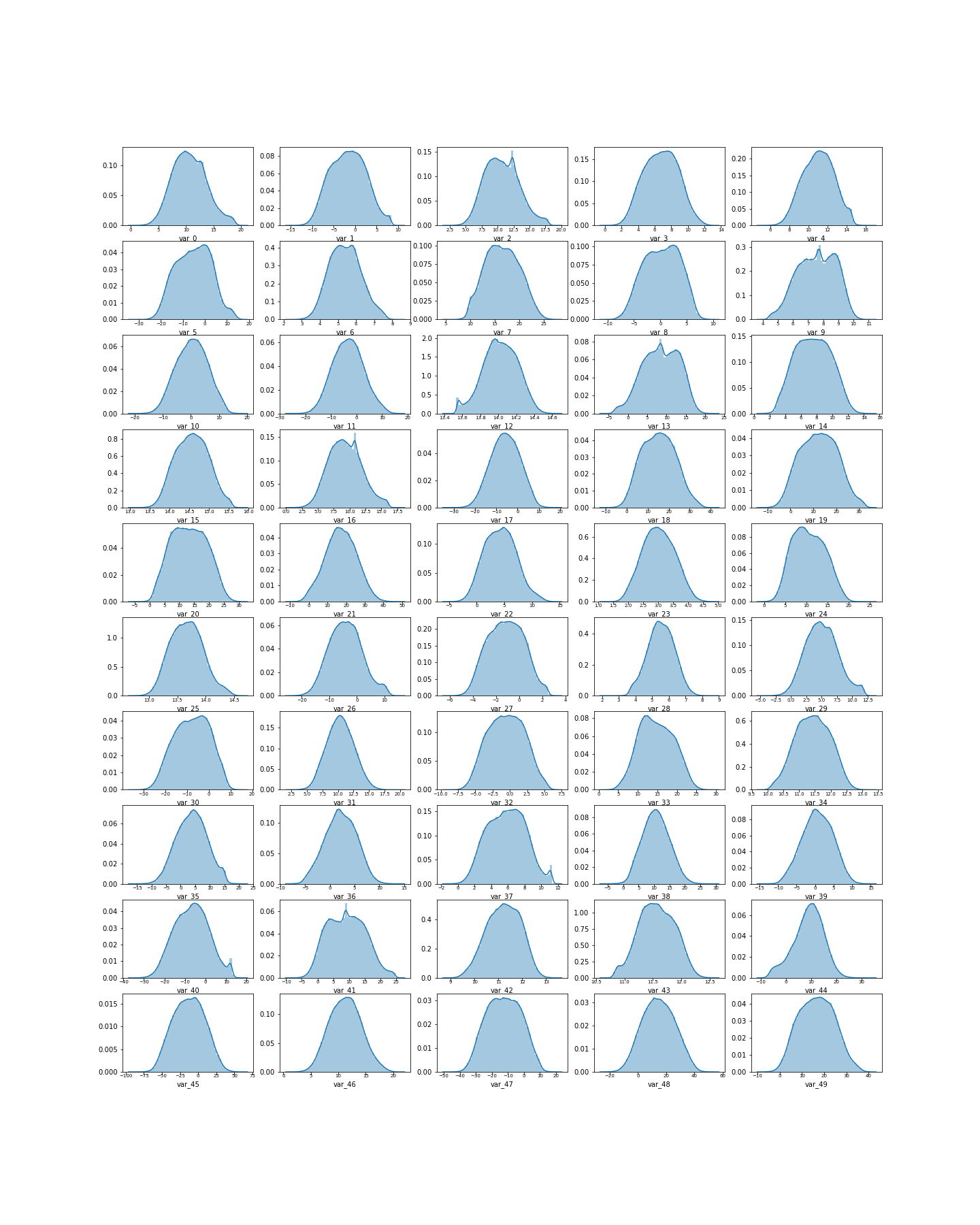
This would take test.csv as input and make predictions for the observations and return a output .csv file with just customer Id and prediction columns to avoid utilizing unnecessary space as both train.csv and test.csv size are 300 MB approximately.

# Exploratory Data Analysis:

**Missing Values**: No missing values were present in either train or test datasets.

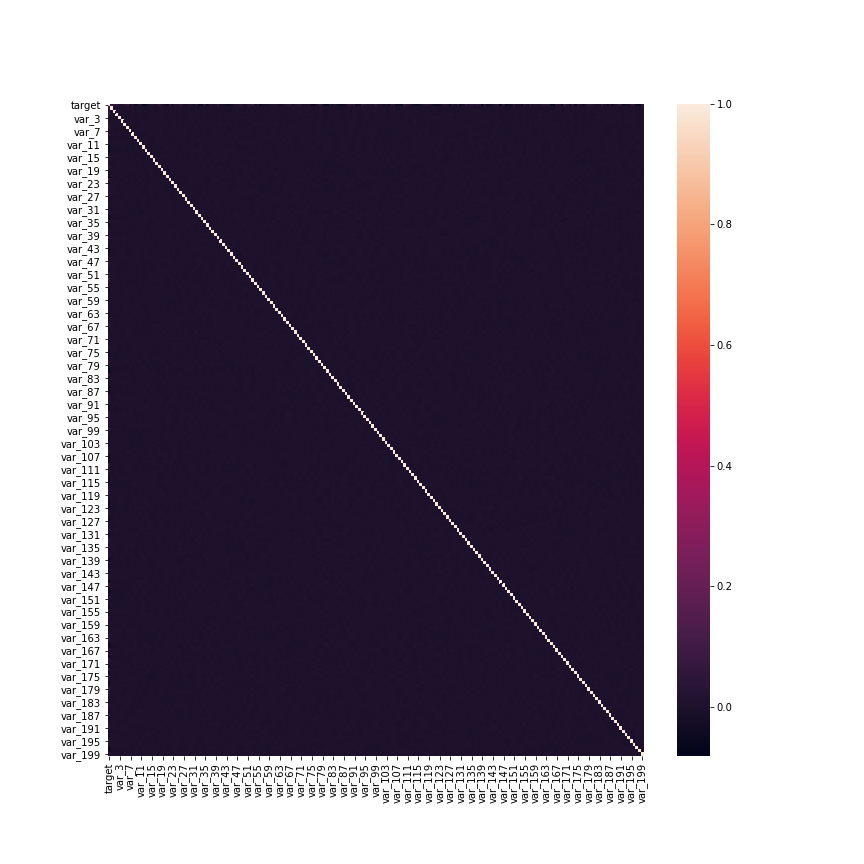
**Distribution of Numerical features:**

From analysis of the distribution plots, we can see almost all variables seem to follow a normal distribution.

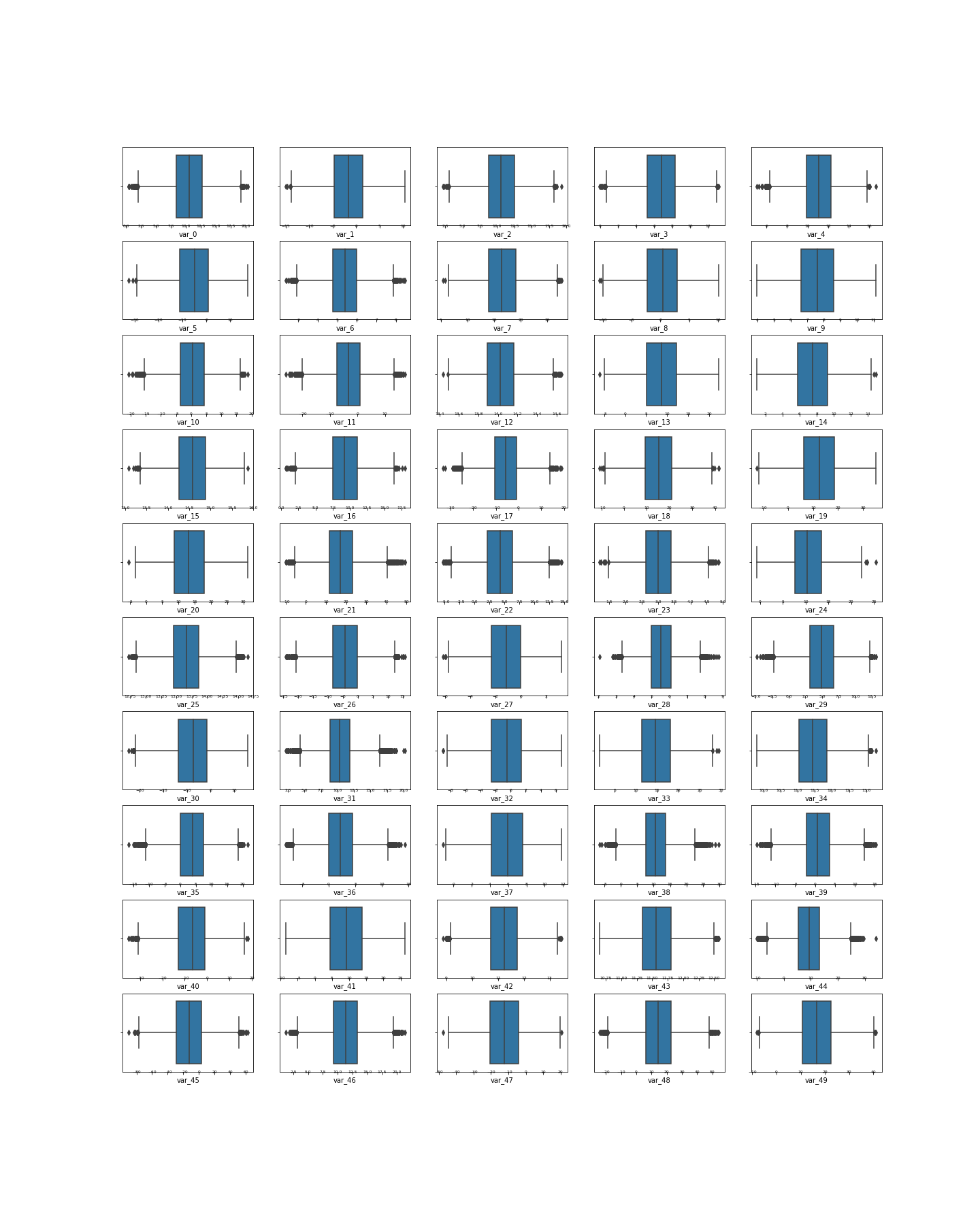
****

**Correlation Analysis:**

Upon observing the correlation matrix, there seems to be absolutely no multicollinearity in this dataset. Absolute value of correlation is less than 0.1 for all elements in the correlation matrix except for the correlation between the variable and itself which is 1. Even the features are not much correlated with the target variable.

Feature selection will be extremely difficult, so we can maybe hope to do feature extraction through dimensionality reduction technique like PCA/LDA to see if it helps during model building.

**Outlier Analysis:**

On analyzing boxplots for the first 50 features, we see that almost all features seem to have few outliers.

# Data Preprocessing/Preparation/Cleaning:

**Dealing with Target Class Imbalance:**

In the given training dataset, we have 2 classes namely 0 and 1 in the target variable in the ratio approximately 9:1. We have 9 class 0 observations for every class 1 observation. Machine learning algorithms tend to get biased towards predicting the majority class in case of severely unbalanced datasets. If the minority class is less than 5 percent then it is called a rare class. Achieving good recall on rare class is a big problem for most algorithms.

The ideal solution to deal with Target class imbalance is to collect additional data on the minority class. In most situations, since this is not possible, we will explore other options.

Resampling Techniques:

* Under-Sampling the majority class
* Over-Sampling the minority class
* SMOTE/ROSE

Cost Sensitive Learning:

* Penalize algorithms more for misclassifying the minority class than for misclassifying the majority class.

Apart from the above, generally tree based algorithms using bagging and boosting techniques perform well with imbalanced datasets.

I had experimented with all the above techniques. For this dataset, Resampling methods to balance the dataset didn’t work effectively. I used a hybridized method by doing doing optimum under sampling and over sampling to decrease the imbalance slightly by 4 to 5 percent so as to not result in information loss due to under sampling and not result in over fitting due to over sampling.

ROSE: Random Over-Sampling Examples resulted in poor performance of the algorithms. SMOTE for this high dimensional dataset required too much computational resources. Due to hardware limitations, I couldn’t experiment by applying SMOTE techniques.

Data preprocessing steps include the following:

1. Missing value handling.
2. Outlier Removal/Treatment.
3. Feature Extraction and Selection.
4. Feature Scaling.

**Missing Value Handling:** No missing values in the given datasets

**Outlier Removal/Treatment:**

Firstly, I tried removing the outliers with Z score method as well as Inter-Quartile range method. Both methods yielded in results which further increases the target class imbalance.

Furthermore, from missing value analysis and correlation analysis, we can infer that the given datasets seem to be preprocessed to a certain extent. So, the possibility of outliers being human error during data collection is very low. So, if the outliers represent valid observations, then we cannot afford to remove them because they might contain useful information.

Therefore we decide to keep the outliers.

**Feature Engineering:**

Feature Selection:

Since this is an anonymized dataset, we don’t know what each feature represents. So, we can’t use domain knowledge to perform feature selection. Furthermore, the correlation analysis didn’t show any correlation between any features to help in feature selection.

Dimensionality Reduction:

PCA for dimensionality reduction didn’t give useful results as well. The first 3 principal components barely explained 2 percent of variance in the data. Even with 150 principal components, we were unable to reach explained variance threshold of 80 percent. Performing PCA will only negatively affect the performance of our model in case of this dataset.

Feature Scaling:

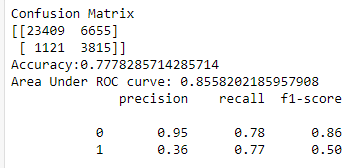
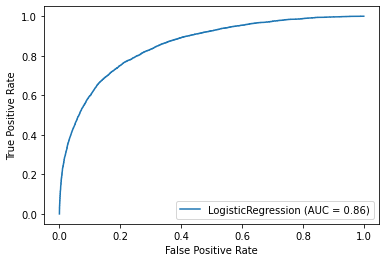
Both normalization and standardization were performed separately and one didn’t outperform the other significantly during model evaluation. So, choice of scaling is up to us to choose either normalization or standardization. Since the features were found to be normality distributed during EDA, we can choose standardization as the scaling method.

Note: Tree based algorithms like Random Forest, XGBoost doesn’t require scaling. Distance based algorithms like KNN, KMeans, SVM and algorithms using gradient descent needs scaling mandatorily to perform well.

# Modelling:

Model 1: **Logistic Regression**

* Basic classifier belonging to the family of Generalized Linear Models (GLM).
* Fits an S shaped curve probability curve to our data based on the logistic function.
* Uses log of odds and maximum likelihood to select the best fit curve.

Evaluation:

Reason of Acceptance/Rejection: **Rejected.**

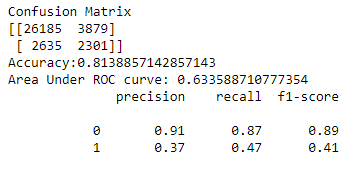
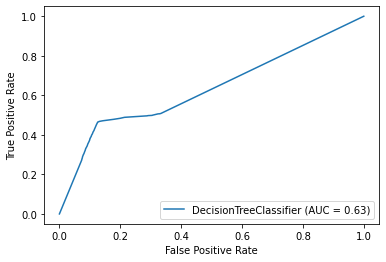
Even though this basic logistic regression model gave us Area under the ROC curve score of 0.86 which is not poor, the precision 0.36 for the positive class is too low to be taken in to consideration for the final model.

Furthermore, there are still sophisticated models yet to be explored which are well-known to outperform a basic logistic regression model.

Model 2: **Decision Tree**

* Splits our dataset using gini index or entropy until homogeneity of classes is achieved in a branch.
* Generally over fitting models and weak learners.

Evaluation:



Reason of Acceptance/Rejection: **Rejected**

We obviously have a poor AUC\_ROC score. Both precision and recall for the positive class is low.

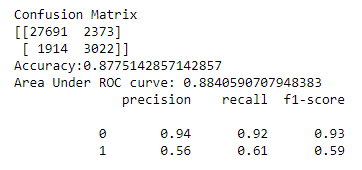
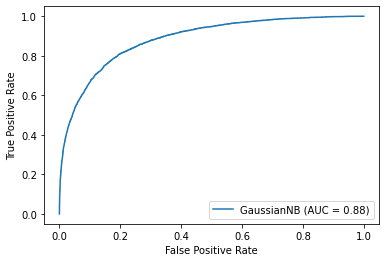
Decision trees are weak learners in general. Since we have 200 features influencing the target variables almost equally, choosing a single feature out of these as a root node will obviously be messy. One decision tree can’t be expected to model this dataset. So, we expected this poor performance.

Bagging many weak learners (Decision Trees) with Random Forest could improve performance. We’ll need to explore that.

Model 3: **Naive Bayes**

* Using Bayes theorem of probability to make classifications. Assumes feature independence.

Evaluation:



Reason of Acceptance/Rejection: **Partially Accepted (Preferred 2nd Choice).**

Naïve Bayes gave a reasonable precision and recall for positive class. Based on AUC\_ROC score it clearly outperformed both Random Forest and XGBoost (Before hyper parameter tuning).

Though XGBoost after hyper parameter tuning beat the AUC\_ROC score of Naïve Bayes, it took a long time for training around 2 hours. For just 20 iterations using randomized search CV for finding the best hyper parameters, XGBoost took more than 10 hours.

On the other hand, Naïve Bayes is extremely fast and robust and performed reasonably well. Since Naïve Bayes just calculates the posterior probability for predictions, it is computationally less expensive and fast.

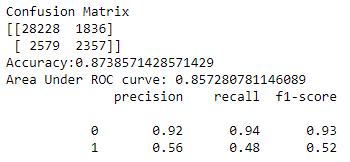
Furthermore, this dataset is high dimensional with 200 features and generally Naïve Bayes is known to perform well for high dimensional datasets apart from used for text classification.

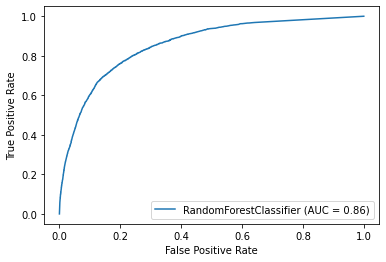
Also this dataset seems to conform to the naïve independence assumption between the features made by Naïve Bayes. During correlation analysis we noticed that no features were even slightly correlated. Also, all features were following a Gaussian/Normal distribution which also plays to the advantage of Naïve Bayes.

There were the reasons that enable Naïve Bayes to perform well on this dataset. In real time, if I am asked to come up with a reasonably good model for this huge dataset in short time, I would go for Naïve Bayes.

Model 4: **Random Forest**

* Combines many weak learning decision trees to build a strong learner by method of bagging.

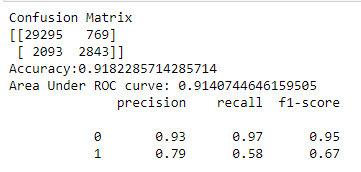
Evaluation:



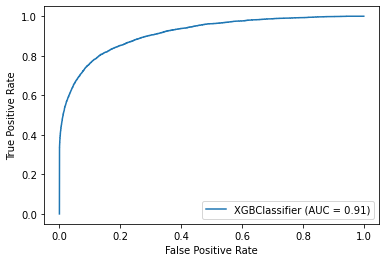
Reason of Acceptance/Rejection: **Rejected.**

Random forest didn’t give a good recall for positive class and also the AUC\_ROC score didn’t beat Naïve Bayes. It also takes a long time to train because of the size of the dataset. Hyper parameter tuning might improve performance but could be computationally expensive and even after tuning it could fall short compared to XGBoost which has shown to outperform most algorithms.

Model 5: **XGBoost**

* Uses gradient boosting techniques to build trees. Each subsequent tree is build based on taking in to consideration the errors made by the previous tree.
* Trees are modeled to predict the residuals.
* The influence of each tree is scaled by the learning rate.

Evaluation:



Reason of Acceptance/Rejection: **Accepted**

Recall is on par with the naïve Bayes model. Whereas in precision, AUC\_ROC score, accuracy it beat all other models by a great margin. scale\_pos\_weight parameter can be used to implement cost sensitive learning to deal with the target class imbalance in the dataset.

With better hardware and sufficient computational resources, XGBoost will be the ideal pick. Even though it is slow compared to Naïve Bayes, it is well-known to be the most fast and efficient among all other gradient boosting algorithms.

# Deployment:

Deployment scripts for both python and R are prepared for both Naïve Bayes and XGBoost algorithms. The scripts will take the train.csv and test.csv as inputs and automatically generate a prediction file for test dataset with the ID\_code column and the probability score and class prediction.

These scripts can be run from the DOS command prompt.

# Conclusion: This model can be utilized by the client to help predict the potential customers who would make a particular transaction in the future and device business strategies based on the knowledge. The threshold probability for classification which is 0.5 by default can be adjusted to get a better recall or precision according to the client needs.