PROJECT REPORT

Santander Customer Transaction Prediction

Submitted By

Devesh Acharya

INDEX

1. Introduction

At Santander, our mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals.

Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

1.1 Problem Statement

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

1.2 Dataset

There are two datasets **train.csv** and **test.csv**. Train dataset contains the target variable along with **199** independent variables so that the model can be trained using that, whereas test dataset does not contain target variable.

2. Methodology

2.1 EDA

2.1.1 Missing Value Analysis

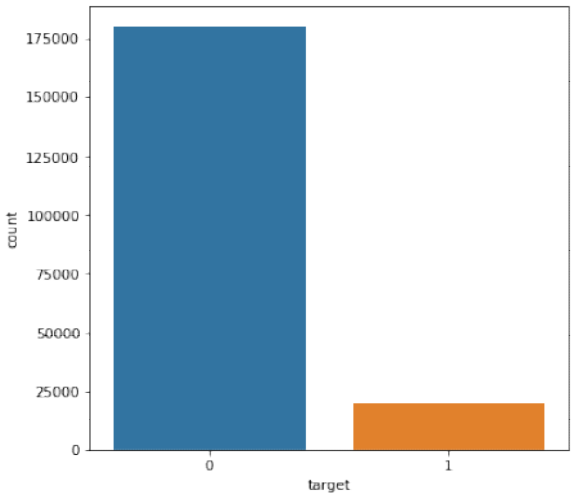
Both train and test datasets do not have any missing values in them.

2.1.2 Outliers

An outlier is an observation which differs from other observations in a significant amount. It may be due to `some human error or it is also possible that the data consists of such observations. One example of natural outliers could be Salary of different employees working in the same organization.

In the respective datasets, there are outliers in almost every variable. We will impute the outliers with mean of the respective variable's column.

2.1.3 Target Value Distribution



From the above graph we can deduce that,

- There is a high class imbalance in target variable.
- The customers who will not make the transaction are much more than those which will.
- Class 0 has around 90% of the total observations and Cass 1 has 10%.

2.1.4 Treating Class Imbalance

Such class imbalance makes the model performance poor. It makes the model bias towards the majority class. To prevent that from happening we can use the following techniques:

- **Random Undersampling:** In this technique observations from the majority class are removed until the majority and minority classes are equal. A drawback of this technique is that we are removing important information that may be valuable and it might affect the performance and the output of the model.
- **Random Oversampling:** As the name suggests, duplicate entries of the minority class is added to the data until both classes are balanced out. This may cause the data to be generalized and overfitted.

- **Synthetic Data:** By using the functions for generating synthetic data, the class imbalance can be treated.

The technique we will be using to deal with class imbalance is SMOTE in python.

2.1.5 Data Distribution

Using the function displot() the distribution of different variables could be visualized. From looking at the displots it was confirmed that almost all of the variables were normally distributed.

Also test data had the same distribution as train data which means both datasets are similar in terms of distribution.

2.1.6 Feature Selection

This is process of subsetting or filtering the relevant features/ variables from the dataset in order to use them in constructing the model. The data chosen to be used to train the ML model can have huge impact on the performance of the model.

Correlation shows that there is no collinearity in the dataset. So we will not drop any variable as the target variable is dependent on each variable.

3. Model Selection

In the following project the three algorithms were used and the one with best performance was chosen. As the data has class imbalance, let us take a look at the three algorithms used in both imbalanced and balanced datasets.

3.1.1 Logistic Regression: Class Imbalance (LMCL)

Logistic Regression is a Supervised Machine Learning method which is used to find the probability of a target/ variable(depenedent).

It uses L*ogit Functions* that derive a relationship between dependent variable(target) and the independent variables.

Code

```
model_lm=LogisticRegression(max_iter=100,random_state=21).fit(X_train,y_train)
```

Parameters used

- max_iter: Maximum number of iterations taken for the solvers to converge.

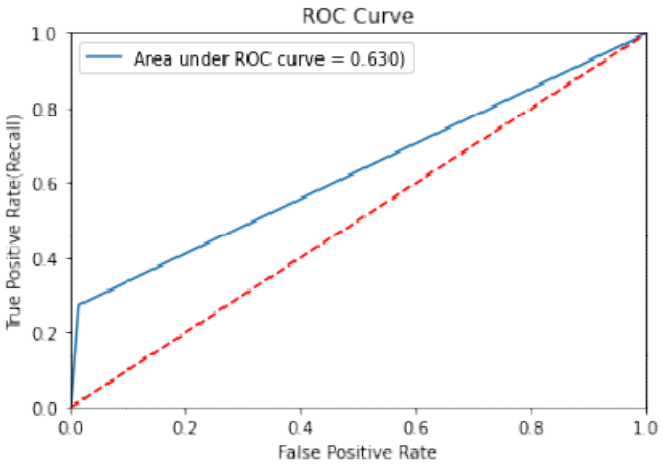
The accuracy score is 0.915.

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.93	0.99	0.95	54059
1	0.68	0.27	0.39	5941

Precision and Recall for Class 0 is great but for Class 1 it is very poor to go ahead with this model.

ROC Curve and AUC



Area under the curve (AUC): 0.629

3.1.2 Logistic Regression: Class Imbalance Treated (LMCIT)

Code

Model_lm_smLogisticRegression(max_iter=5000,random_state=21).fit(X_train_sm,y_train_sm)

Parameters used

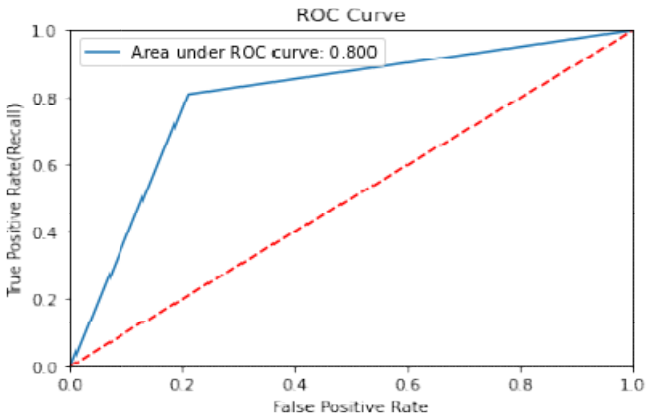
- max_iter: Maximum number of iterations taken for the solvers to converge.

The accuracy score is 0.799

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.81	0.79	0.80	54059
1	0.79	0.81	0.70	5941

ROC Curve and AUC



Area under the curve (AUC): 0.8

Area under the curve (AUC) is 0.8 which is a good enough value but as Precision and Recall are not good enough Logistic Regression Model is not good for this problem.

3.2.1 Random Forest Classifier: Class Imbalance (RFCI)

Random forest is a supervised learning algorithm. The forest built in a Random Forest is made by an ensemble of Decision Trees. It can be used for both classification and regression problems.

Code

model_rfc = RandomForestClassifier(random_state=21,n_estimators=10).fit(X_train, y_train)

Parameters used

- *n_estimators: The number of trees in the forest.*

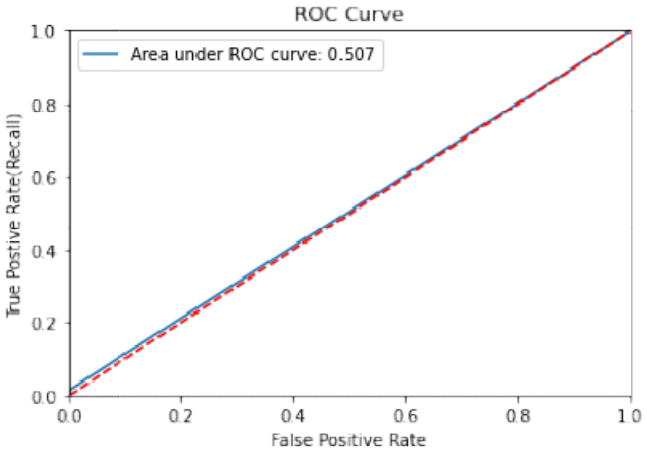
The accuracy score is 0.901

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.95	54059
1	0.55	0.01	0.03	54059

Recall for Class 1 is practically 0.

ROC Curve and AUC



Area under the curve (AUC): 0.507

3.2.2 Random Forest Classifier: Class Imbalance Treated (RFCCIT)

Code

model_rfc_sm = RandomForestClassifier(random_state=21, n_estimators=30).fit(X_train_sm, y_train_sm)

Parameters used

- n_estimators: The number of trees in the forest.

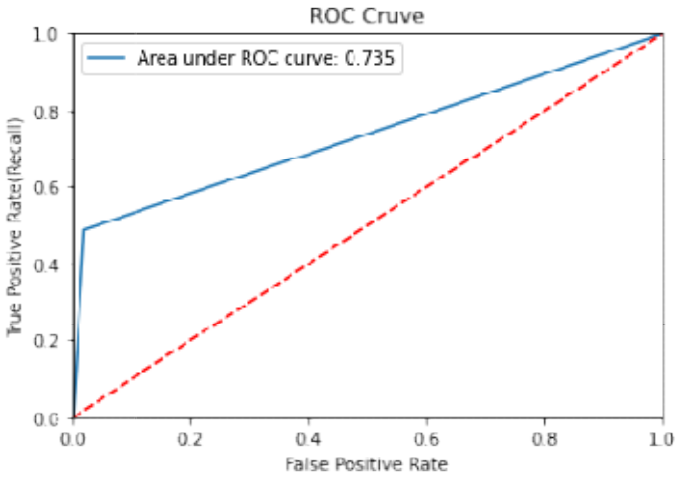
The accuracy score is 0.734

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.66	0.98	0.79	54059
1	0.96	0.49	0.65	54059

Recall for Class 1 is still too low to continue with this model.

ROC Curve and AUC



Area under the curve (AUC): 0.735

3.3.1 Naïve Bayes: Class Imbalance (NBCI)

Code

model_gnb = gnb.fit(X_train, y_train)

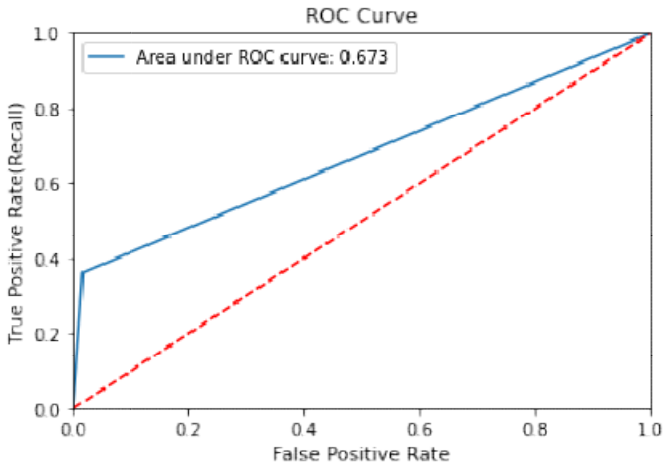
The accuracy score is 0.922

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.93	0.98	0.96	54059
1	0.71	0.36	0.48	5941

Recall is still too low for Class 1.

ROC Curve and AUC



Area under the curve (AUC): 0.672

3.3.2 Naïve Bayes: Class Imbalance Treated (NBCIT)

Code

model_gnb_sm = gnb.fit(X_train_sm, y_train_sm)

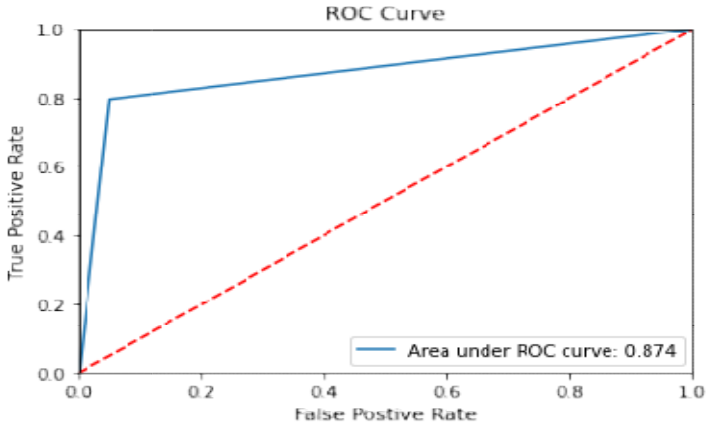
The accuracy score is 0.873

Precision, Recall and F1-Score

Class	Precision	Recall	F1-Score	Support
0	0.82	0.95	0.88	54059
1	0.94	0.80	0.86	54059

Precision and Recall for both Class 1 and Class 0 are much better than the rest of the models.

ROC Curve and AUC



Area under the curve (AUC): 0.874

4. Model Evaluation

Although accuracy of most models shows the model to be performing well, it is not a good evaluation metric especially for imbalanced data as it does not show the correct predictions for different classes. If the data is imbalanced the model will predict the majority class accurately

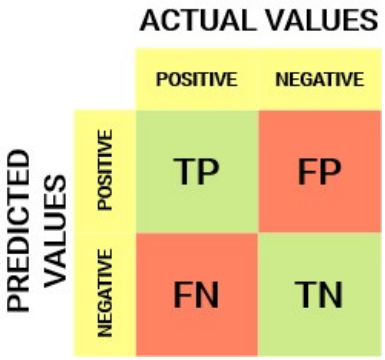
due to biased towards majority class. As in current case, the accuracy of logistic Regression in Imbalanced Dataset is more than 90% because the 90% of target variable consists of only Class 0.

In order to get correct evaluation of the model we will use some metrics which will give correct information about the prediction of the model. The metrics used are **Precision, Recall, Area under the Curve (AUC) and F1-Score**

4.1 Performance Metrics

4.1.1 Confusion Matrix

A confusion matrix is a matrix of size NxN that is used to evaluate the performance of a classification model, where N is the number of target classes.



Confusion Matrix

The matrix provides us the values which allow us to compare the actual values with the values predicted by the machine learning model. It does not gives an actual value to compare the model directly, but it gives value in the form of ***TP, FP, TN and FN*** which gives us an idea of how the model is predicting the target. And also these values are further used to calculate the performance metrics which are used to further check performance of the model.

4.1.2 Precision

Precision gives the value of correctly predicted positives out of all predicted positives. It can be calculated from the given relation,

Precision = True Positives(TP)/(True Positives(TP) + False Positives(FP))

4.1.3 Recall

Recall also known as Sensitivity or True Positive Rate is the ratio of correctly predicted positives by total number of positives,

Recall = True Positives(TP)/(True Positives(TP) + False Negatives(FN))

4.1.4 ROC-AUC Score

ROC is a probability curve and AUC gives us the measure of separability. It tells us how much a model can differentiate between different classes. ROC curve is a graph between True Positive Rate on y-axis and False Positive Rate on x-axis. So the more area under the ROC curve the better the model will perform.

4.1.5 F1-Score

F1-Score is the harmonic mean of Precision and Recall. It is calculated using the relation,

F1-Score = 2*(Precision * Recall)/(Precision + Recall)

5. Conclusion

Let us compare the models using the values of the aforementioned evaluation metrics.

Performance Metric	**Logistic Regression**				**Random Forest Classifier**				**Naïve Bayes**			
	LRCI		**LRCIT**		**RFCCI**		**RFCCIT**		**NBCI**		**NBCIT**	
	Class 0	*Class 1*	*Class 0*	*Class 1*	*Class 0*	*Class 1*	*Class 0*	*Class 1*	*Class 0*	*Class 1*	*Class 0*	*Class 1*
Precision	*0.93*	*0.68*	*0.81*	*0.79*	*0.90*	*0.55*	*0.66*	*0.96*	*0.93*	*0.71*	*0.82*	*0.94*
Recall	*0.99*	*0.27*	*0.79*	*0.81*	*1.00*	*0.01*	*0.98*	*0.49*	*0.98*	*0.36*	*0.95*	*0.80*
F1-Score	*0.95*	*0.39*	*0.80*	*0.70*	*0.95*	*0.03*	*0.79*	*0.65*	*0.96*	*0.48*	*0.88*	*0.86*
ROC-AUC Score	*0.633*		*0.8*		*0.507*		*0.696*		*0.675*		*0.874*	

As we can see in the above *Performance Metric Table* the model that has great scores in all the metrics for both classes is ***NAÏVE BAYES***. Therefore, we will proceed with the Naive Bayes model with class imbalanced treated.