Santander Customer Transaction Prediction

Final Project Report for DATA 1030, Fall 2021 at Brown University

Supervised by Prof. Andras Zsom

(https://github.com/Ramyahkay/Santander-Customer-Transaction-Prediction-)

Ramya Harikrishnan

I. INTRODUCTION

A. Description

Santander is looking to predict and identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted, to help understand their financial health and identify which products and services might help them achieve their monetary goals. Challenges like will a customer buy this product? Or is the customer satisfied? Or can a customer pay this loan can be tackled with this prediction.

B. Dataset

The available dataset is provided by the Santander Bank, N. A., formerly Sovereign Bank, a wholly owned subsidiary of Spanish Santander Group in the Santander Transaction Prediction competition from Kaggle. It has the same structure as the real data however, what is interesting is that the data is completely anonymous with no customer details revealed. We are provided with two datasets, train and test. The brief characteristics of the data can be stated as follows: (a) 198 anonymized, independent numeric features and 200,000 records (b) primary key column which marks each transaction (c) Binary response target variable where 0 implies 'No Transaction' and 1 implies 'Yes Transaction'. Our task is to build a binary classification model to predict the value of the target column in the test set.

II. EXPLORATORY DATA ANALYSIS

In this phase, it was discovered that the properties of the dataset have datatypes taking up large memory space. To handle this data more effectively, the data types have been changed and reduced by 50% without any loss in information, hence saving space and increasing computation power. As shown in Fig 1., it was observed that the dataset is unbalanced. Only 10% of the records have response 1. It was also interpreted that the numerical features follow a normal distribution as shown in Fig 2. Further analysis of the datasets showed that from their mean, both datasets are almost identical. And their standard distributions are large. This can be seen in Fig 3.

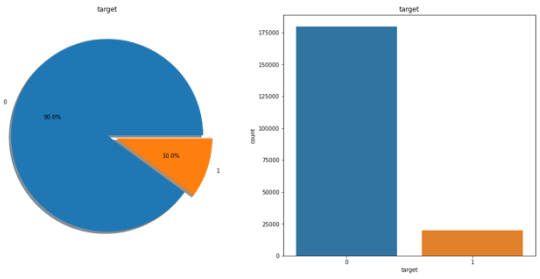


Fig 1. Distribution of Target Class Column

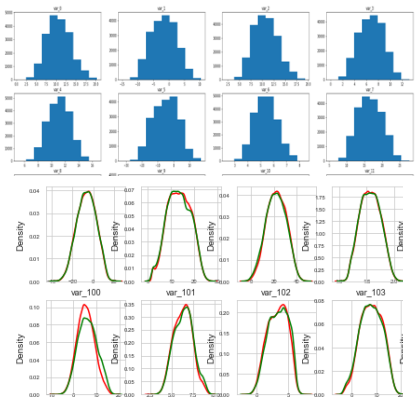


Fig 2. Distribution of train data

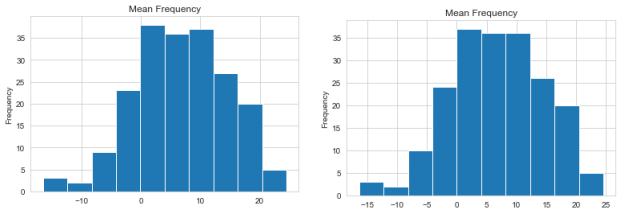


Fig 3.1 Mean distribution of train and test

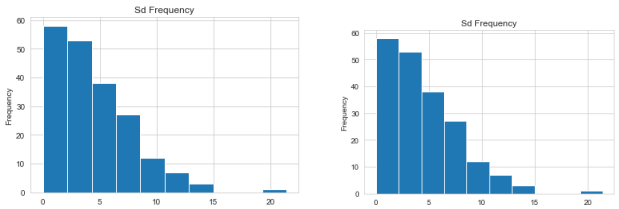


Fig 3.2 Standard Deviation of train and test

During analyzing the correlation between the features, it was found that there is no strong correlation between any of the features. From Fig 4. and Fig 5., we can see using the scale on the right that shows the correlation level and color that the correlation is almost zero.

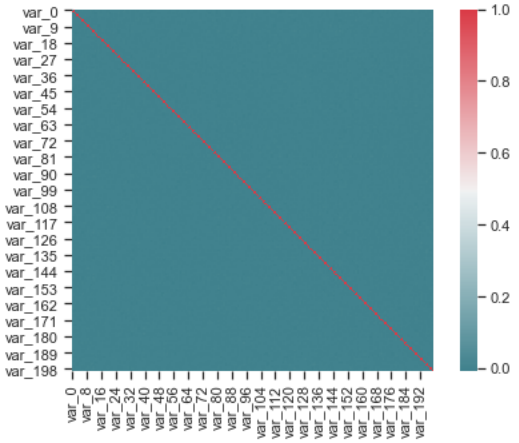


Fig 4. Correlation of train data

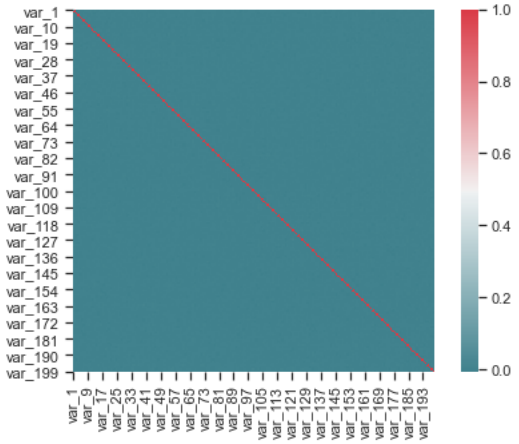


Fig 4. Correlation of test data

III. DATA PREPROCESSING

To meet our requirements, we transform data accordingly. There were no missing values in this dataset. The primary key column is removed as it bears no importance in identifying the transaction. We then remove the column and split the data into 80:20 ratio for training and testing respectively using StratifiedKFold technique with nfold set as 5 to give a better accuracy. As the dataset was imbalanced, it was appropriate to implement stratified splitting of data. Standard Scaling was performed on the 198 numerical features as they were continuous data. This dataset can

be interpreted as an IID data as all the 198 features are independent to each other.

IV. METHODOLOGY

In this project, four machine learning models have been used to estimate the predictions of the target variable. The models used are Logistic Regression, Decision Tree, Naïve Bayes and gradient boosting algorithm - XGBoost. Parameters used in these models have been optimized through several test runs giving best optimal results.

- *Evaluation Metrics:* To better understand and predict actual performance of the classifiers, area under the (AUC) and confusion matrix have been used. The confusion matrix shows the following results:
  - True positives (TP)
  - False positives (FP)
  - True negatives (TN)
  - False negatives (FN)

Based on these outcomes, further effectiveness of the model can be checked using recall, precision and f1 scores. The below equations can be used to calculate the mentioned scores.

$$Recall = \text{TP TP+FN} \quad (1)$$
$$Precision = \text{TP TP+FP} \quad (2)$$

To find optimum combination of precision and recall, we use f1.

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision+Recall}} \quad (3)$$

The results of the binary classification models can be illustrated using the receiver operating characteristics and area under the curve. There are representations of true positive rate (TPR) and false positive rate (FPR) as a function of the model's threshold. The confusion matrix calculates the actual and predicted labels from a classification in a matrix format.

- *Hyperparameter Tuning:*

Once the baseline scores that were produced, for each particular model, to further improve the accuracy, it was decided to select best parameters of the model to most benefit the test and obtain higher accuracy. Specific elaboration on which parameters were tuned can be seen in the Results section below.

V. RESULTS

Based on the specifications from the previ ous sections, all the four models were impl emented and results were obtained from ea ch model. Also, each model was referred t o the baseline score (0.89951), using Zero Rate classifier, to compare its abilities to p redict both 0's and 1's.

i. Logistic Regression

As this is a binary classification dataset, it was appropriate to fit the model using logi stic regression that serves as a baseline mo del for further analysis. The following assu mptions were made (i) FNR and FPR score s were high (ii) ROC_AUC needs more ma ximization. After further hyperparameter t uning, i.e., setting the parameters to {'pena lty': 'l2', 'max_iter': 5, 'class_weight': 'balan ced', 'C': 0.0001}, the accuracy decreased, below the baseline score.

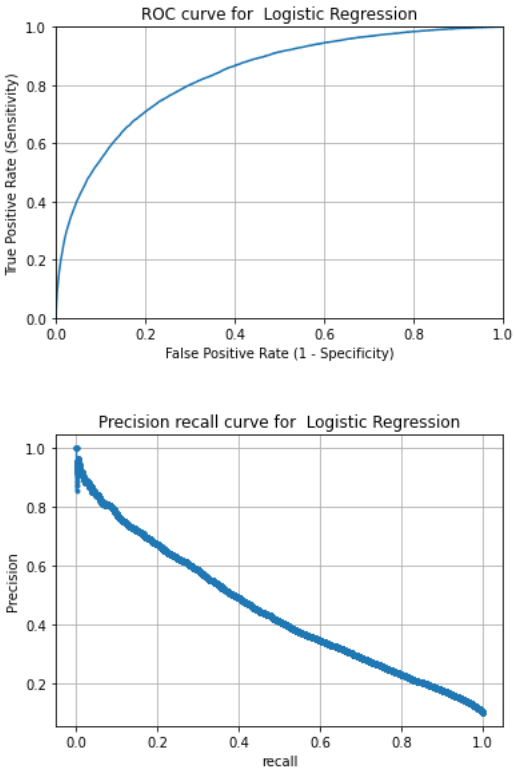


Fig 5. Logistic Regression
a) ROC curve b) PR curve

ii. Decision Tree

The model was also fit using decision tree. Similar to logistic regression, the roc-auc needed more maximization, and FNR and FPR should be decreased. After hyperparameter tuning the model to the best parameters obtained, that are {class_weight='balanced' ,random_state =5 ,min_samples_split= 3, max_depth=5, criterion='entropy'}, however, here were no major changes in the accuracy, but is below the baseline score.

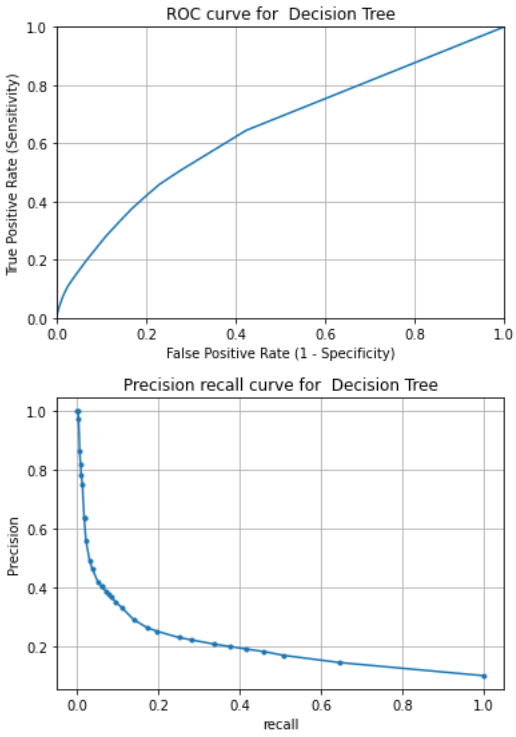
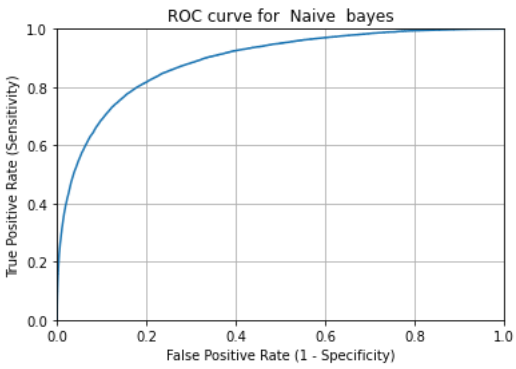


Fig 6. Decision Tree
a) ROC curve b) PR curve

iii. Naïve Bayes

When implementing Naïve Bayes model, a higher accuracy score was produced, but F NR and FPR scores needed to be reduced m ore. After hyperparameter tuning with the b est parameters: {'var_smoothing': 6.951927 961775605e-07}, there was a slight decrea se in the FNR and FPR scores but accuracy was the same, higher than the baseline scor es.



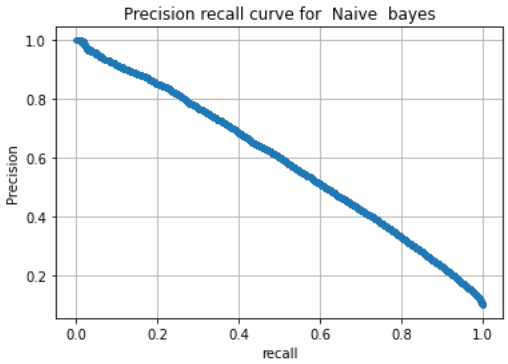


Fig 7. Naïve Bayes
a) ROC curve b) PR curve

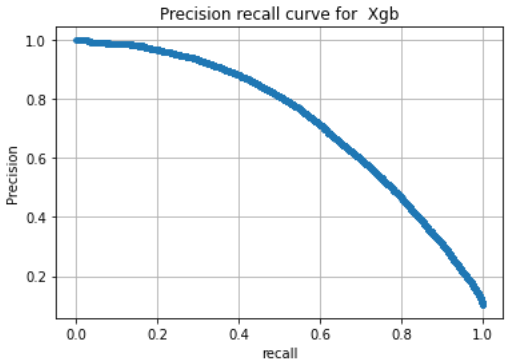
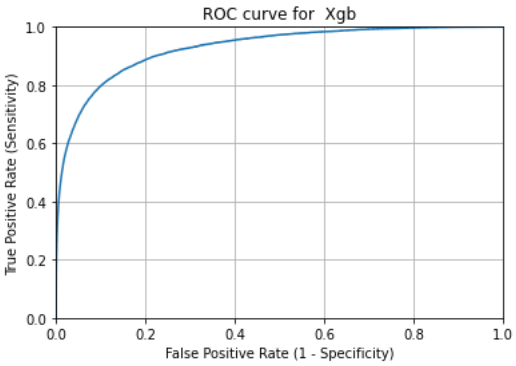


Fig 8. XGBoost
a) ROC curve b) PR curve

iv. XGBOOST

Gradient boosting algorithm – XGBoost worked really well when fitted with the dataset. Precision was almost 100%, accuracy was 93% but FNR score was too high. After hyperparameter tuning, setting to {learning_rate =0.1, n_estimators=800, max_depth=5,min_child_weight=1,gamma =0,subsample=0.8,colsample_bytree=0.8,objective= 'binary:logistic',nthread=4, seed=27, scale_pos_weight=2), FNR scores reduced and accuracy shot up to 98%, higher than the baseline score.



Check Fig 9. to view all the base scores and optimal scores of accuracy, precision, recall, f1, fpr, fnr and specificity from all the models.

	precision	accuracy	recall	specificity	fpr	fnr	f1
xgb_opti	0.969562	0.985305	0.881097	0.996917	0.003083	0.118903	0.923215
logit_base	0.277469	0.776062	0.768973	0.776853	0.223147	0.231027	0.407793
logit_opti	0.255981	0.755344	0.755337	0.755344	0.244656	0.244663	0.382376
dt_base	0.164981	0.676203	0.548933	0.690386	0.309614	0.451067	0.253709
dt_opti	0.162462	0.676031	0.536933	0.691532	0.308468	0.463067	0.249448
nb_opti	0.714825	0.921695	0.367300	0.983630	0.016370	0.632700	0.485259
nb_base	0.716664	0.921898	0.365591	0.983893	0.016107	0.634409	0.484186
xgb_base	0.647934	0.914500	0.322503	0.980472	0.019528	0.677497	0.430652

Fig 9. Goodness of Fit Comparison

Overall, by far XGBoost produced the highest roc_auc (98.5%), low FNR (11.1%) and FPR (0.3%) and a high f1 score (92.3%). So, the final model is XGBoost as this would best predict instances where a customer would make the transaction were not.

VI. FEATURE IMPORTANCE

Feature importance metrics for global and local have been implemented.

1. Global Feature Importance

Global Feature Importance is calculated using Permutation Importance. Weights of each feature were calculated and observed as shown in Fig 10. The importance of features in descending order. The green color indicates that the feature has a positive impact on prediction, while white means no impact on predictions. We can see that var_81 is the most important feature.

Weight	Feature
0.0039 ± 0.0003	var_81
0.0027 ± 0.0001	var_139
0.0026 ± 0.0002	var_12
0.0026 ± 0.0004	var_6
0.0024 ± 0.0003	var_26
0.0023 ± 0.0002	var_53
0.0021 ± 0.0003	var_21
0.0020 ± 0.0001	var_22
0.0020 ± 0.0001	var_80
0.0020 ± 0.0002	var_76
0.0019 ± 0.0004	var_146
0.0019 ± 0.0002	var_110
0.0018 ± 0.0002	var_174
0.0018 ± 0.0002	var_148
0.0017 ± 0.0001	var_99
(…)	
0.0002 ± 0.0001	var_119
0.0002 ± 0.0000	var_52
0.0002 ± 0.0000	var_49
0.0002 ± 0.0000	var_130
0.0002 ± 0.0001	var_105
0.0002 ± 0.0001	var_186
0.0002 ± 0.0001	var_112
0.0002 ± 0.0000	var_150
0.0002 ± 0.0000	var_114
0.0002 ± 0.0001	var_135
0.0002 ± 0.0001	var_141
0.0002 ± 0.0001	var_43
0.0001 ± 0.0001	var_128
0.0001 ± 0.0001	var_70
0.0001 ± 0.0001	var_24
0.0001 ± 0.0001	var_23
0.0001 ± 0.0001	var_199
0.0001 ± 0.0001	var_74
… 100 more …	

Fig 10. Weights of Important Feature

Using feature importance scores, we can p erform feature selection. While evaluating XGBoost model, we select three importanc e features of XGBoost; weight, gain and co ver. The importance value of each of these features is observed for the variables under various threshold values. The best values f or weight, gain and cover were 27, 15.8, and 925.9715395548783 respectively.

2. Local Feature Importance

Local feature importance metrics can be calculated using SHAP. A few observations, we can see from Fig 11. that two most important features var_81 and var_139 where as v_198 and v_190 were least important. We can also see that the low values of both these features are packed densely forming a pink blob. Features like var_53 and var_6 also have low values forming clusters but of blue blobs.

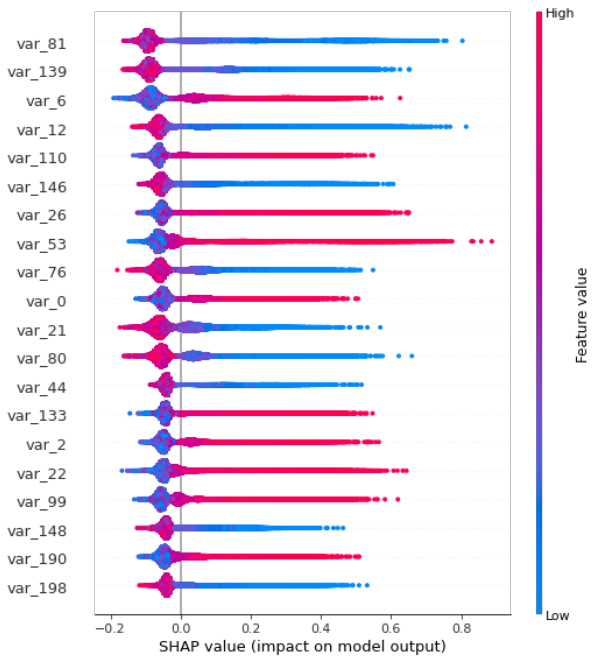


Fig 11. Summary Plot

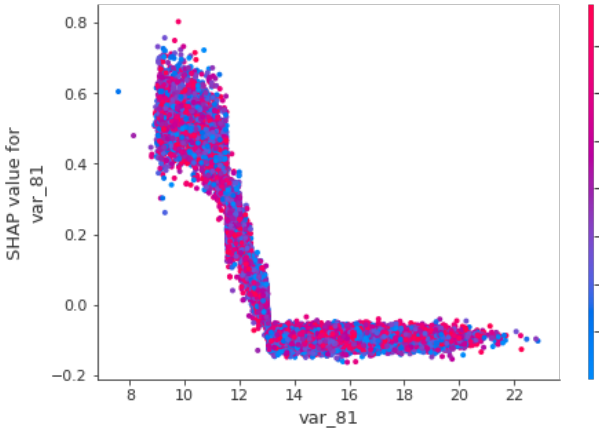


Fig 12. Dependence Plot for var_81

From Fig 12., we can see those low values of var_81 highly influences the model output more significantly for observations.

VII. OUTLOOK

So far only four models were considered, but a further detailed analysis can be done using models like Random Forest, SVMs which may perform better.

Additionally, due to the heavy imbalanced dataset, it is not efficient to rely on only one model to predict the accuracy. We can use ensemble methods (bagging, boosting, stacking) for a better prediction of classes to avoid over-fitting and for a high performance.

VIII. REFERENCES

[1] Kaggle, "Santander Customer Transaction Prediction - Can you identify who will make a transaction?," 2019.
[2] Scikit-learn, "Scikit-learn," 2019.
[3] Yang, Y., & Padmanabhan, B. (2003, November). Segmenting customer transactions using a pattern-based clustering approach. In Third IEEE International Conference on Data Mining (pp. 411-418). IEEE
[4] Tatman, R. (n.d.). Machine Learning with XGBoost (in R). Retrieved from https://www.kaggle.com/rtatman/machine-learning-with-xgboost-in-r
[5]https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.Booster.get_score
[6] https://towardsdatascience.com/be-careful-when-interpreting-your-features-importance-in-xgboost-6e16132588e7