Santander Customer Transaction Prediction

Michael Albert D'Souza
6th November 2019

1 Contents

Chapter 1

1. Introduction

1.1. Problem Statement

The aim of the project is to find out the number of people who will make a transaction in the future, so that it would be helpful for the Santander bank to target those particular set of customers for loan schemes etc. Basically 2 datasets are provided (i.e. : train set and test set) We have to predict the value of target column in the test set.

1.2. Data

The goal of this project is to build suitable classification models as the target variable in train set is a categorical variable, which will help predict the target column. Below is the sample of data for both train and test set.

Table 1.1: Train set Sample Data (Columns 1-8)

ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5
train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834
train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433
train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837
train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361
train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486

var_193	var_194	var_195	var_196	var_197	var_198	var_199
1.6910	18.5227	-2.3978	7.8784	8.5635	12.7803	-1.0914
10.9516	15.4305	2.0339	8.1267	8.7889	18.3560	1.9518
1.6858	21.6042	3.1417	-6.5213	8.2675	14.7222	0.3965

Table 1.2: Train set Sample Data (Columns 193-199)

ID_code	var_0	var_1	var_2	var_3	var_4	var_5
test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805
test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117
test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052
test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744

Table 1.3: Test set Sample Data (Columns 1-7)

var_193	var_194	var_195	var_196	var_197	var_198	var_199
2.4508	13.7112	2.4669	4.3654	10.7200	15.4722	-8.7197
10.1282	15.5765	0.4773	-1.4852	9.8714	19.1293	-20.9760
2.1800	12.9813	2.1281	-7.1086	7.0618	19.8956	-23.1794

Table 1.4: Test set Sample Data (Columns 193-199)

I was provided with an anonymized dataset containing numeric feature variables, the binary target column, and a string ID_code column.

There are total 200000 observations and 202 variables in train set.
There are total 200000 observations and 201 variables in test set.

Chapter 2

2. Methodology

2.1. Pre-processing

- Data preprocessing is a data mining technique that involves transforming raw data into an understandable format.
- Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
- To overcome such complexities we perform data pre-processing, where we handle missing values, outliers, etc.
- Pre-processing of the data should be done to avoid any kind of complexity in the model.
- In this case one technique was performed for data pre-processing – Missing Value Analysis.
- The data preprocessing was done in train dataset as the target variable was present in train dataset.

2.1.1. Missing Value Analysis

- When dealing with data there are various techniques to clean and transform the data.
- Missing Value Analysis is one of the techniques.
- Missing Value Analysis means values which are not present in a dataset.
- Missing Value Analysis describes the pattern of missing data.

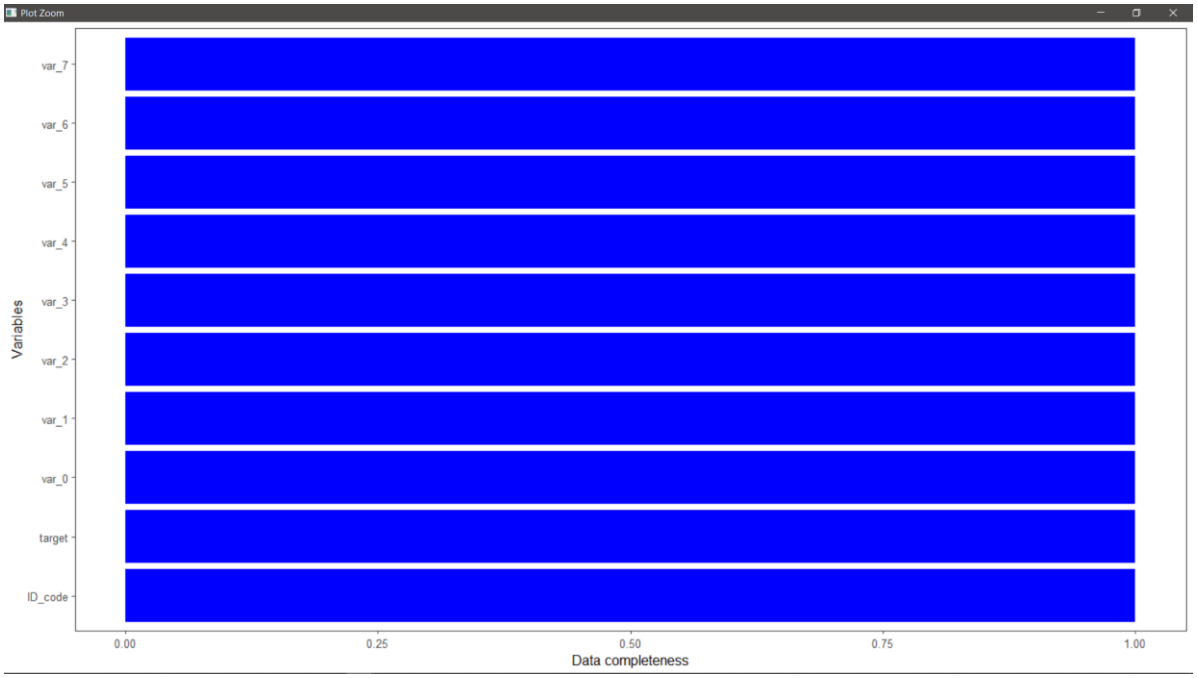

Fig 1.1 : Missing Value Analysis

As you can see in the above figure, no missing values are present in first few columns. Similarly, there were no missing values in the other columns too.

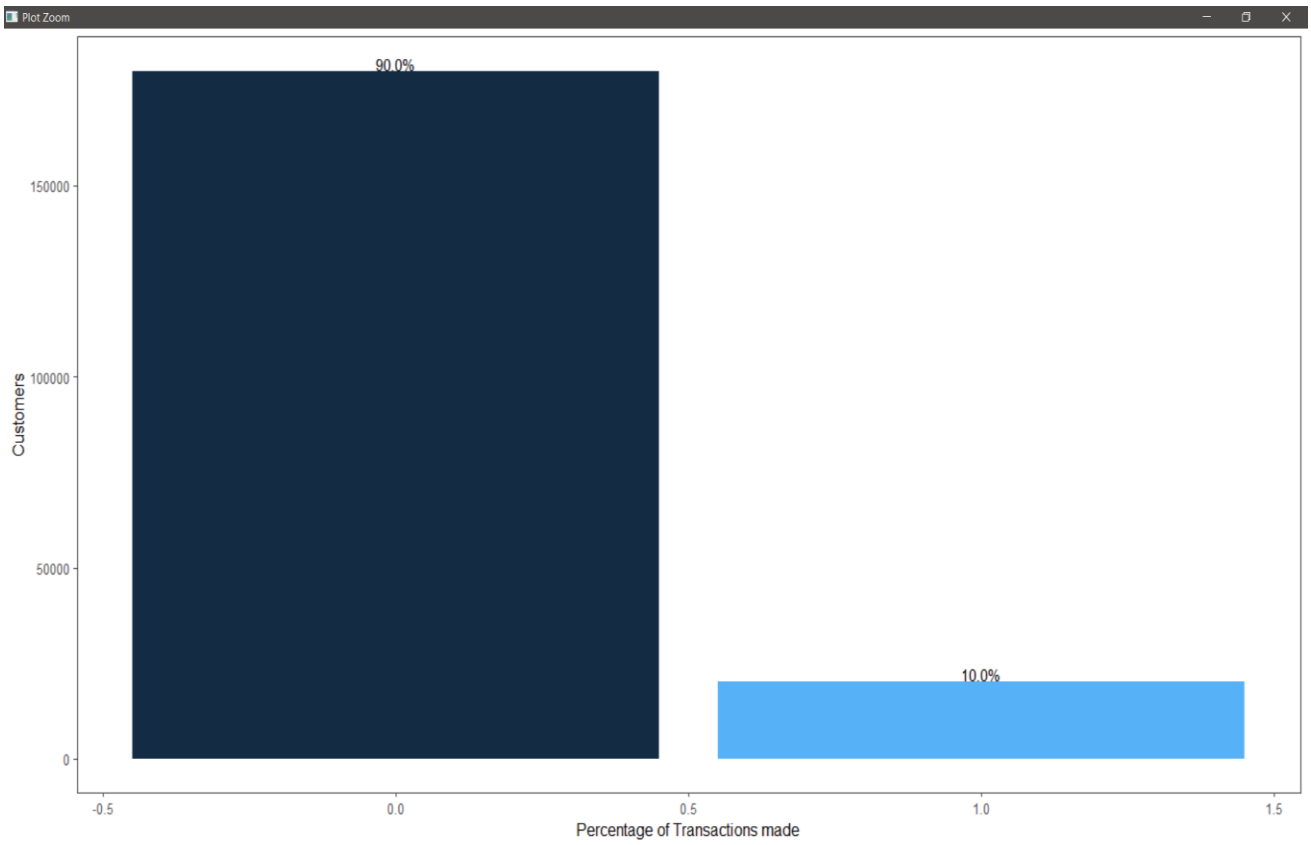

Fig 1.2 : Distribution of target variable

Chapter 3

3. Modeling

3.1. Model Selection

- Once we processed the data, the next stage was model selection.
- As we could clearly see the target variable was categorical variable, so we had to choose the model accordingly.
- To choose the best model we tried 3 models, the model with the better performance will be selected.
- As you could see that the distribution of target variable was 90-10. This means we had to choose it properly.
- However, we should not select the model on the basis of accuracy itself , we should also consider the error metric to finalize the model.
- The three models are :
  - Decision Tree
  - Logistic Regression
  - Naive Bayes

- Data was then divided into train and test.
- The first model we started with was Decision Tree.

3.1.1. Decision Tree

- Decision tree is the most powerful and popular tool for classification and prediction.
- A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
- Decision tree can be used for both classification and regression.
- In our case, as our target variable is categorical variable, so we used Decision tree Classification model.
- After training the model on train data and then predicting it on test data, we found that the model had an accuracy of 83.34% which was pretty decent.
- Though accuracy is not enough, we also tried some error metrics like precision, recall, F1 score.
- even though the accuracy of this model is pretty decent it did not perform well in F1 score.

3.1.2. Logistic Regression

Logit Regression Results

Dep. Variable:	target	**No. Observations:**	159951
Model:	Logit	**Df Residuals:**	159752
Method:	MLE	**Df Model:**	198
Date:	Wed, 06 Nov 2019	**Pseudo R-squ.:**	0.2898
Time:	19:18:09	**Log-Likelihood:**	-37068.
converged:	True	**LL-Null:**	-52197.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
var_0	0.0548	0.003	17.499	0.000	0.049	0.061
var_1	0.0389	0.002	16.340	0.000	0.034	0.044
var_2	0.0688	0.004	19.098	0.000	0.062	0.076
var_3	0.0176	0.005	3.712	0.000	0.008	0.027

Fig 1.3 : Logistic Regression summary

- Logistic Regression is another statistical model which is used for classification.
- It follows the same idea of calculating the regression coefficient of linear regression.
- Output from Logistic Regression can be class or probability.
- As you can see in the above figure, the p-values for almost every variable is less than 0.5.
- The accuracy that logistic model gave was 91.24% which was good, but it shows that the model did not predict well.

3.1.3. Naive Bayes

- Naive Bayes algorithm is only used for classification purpose.
- It is one of the supervised machine learning algorithms based on Bayes theorem.
- It is also called probabilistic algorithm for classifier.
- The accuracy that the Naive Bayes produced was 92.23 % which was good and also it produced the best F1 score amongst all three.

Chapter 4

4. Conclusion

- Now that we have a few models for predicting the target variable, we need to decide which one to choose.
- There are several metrics that exist for evaluating and comparing models. We can compare the models using any of the following classification metrics.
- They are :
    - Misclassification error
    - Specificity
    - Recall
    - False Positive rate
    - AUC
    - Precision
    - F1 score

- For our evaluation purpose we will use the F1 score metric, as the target variable is kind of imbalanced.
- When the data is imbalanced F1 score turns out to be a good metric.
- It considers both recall and precision, hence giving the best possible result.

4.1. F1 SCORE

- When the data is imbalanced the F1 score comes out to be a good metric.
- F1 score is the harmonic mean of Recall and Precision.
- Harmonic mean punishes extreme values more.

- F1 score is calculated using following formula :

    F1 score = 2 * (Precision*Recall)/(Precision+Recall)

- F1 scores obtained for all the three models were as follows :
    - F1 score for Decision Tree : 19.4
    - F1 score for Logistic Regression : 37.6
    - F1 score for Naive Bayes : 48.2

We can see that all three models had a good accuracy but F1 scores predicted that the naive Bayes predicted the best amongst all three. So, for our prediction of target column in the test set we finalize Naive Bayes model.

Chapter 5

5. Appendix

5.1. Figures

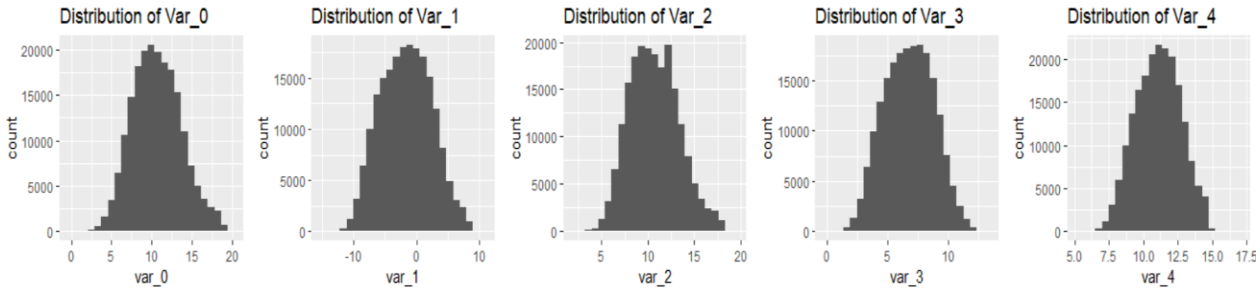

Fig 5.1: Distribution of first five continuous variables

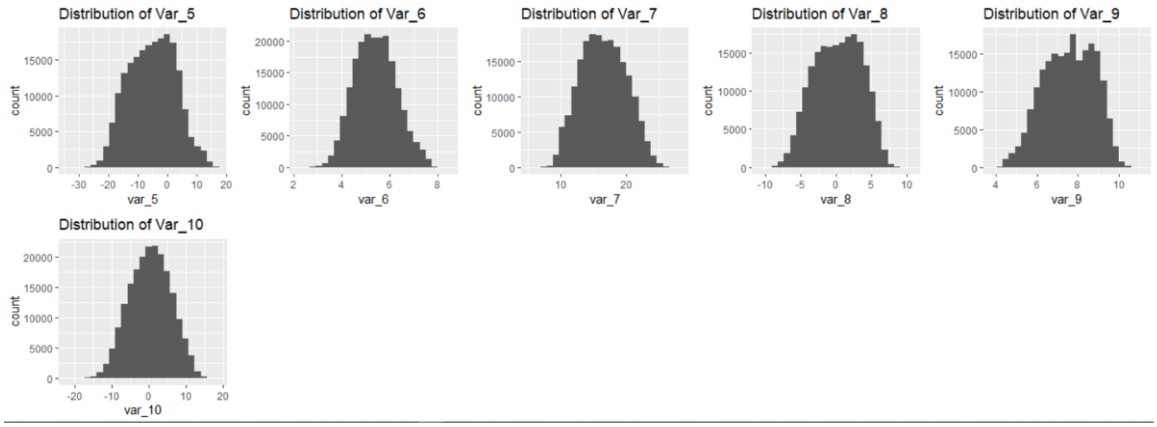

Fig 5.2 : Distribution of next six continuous variables

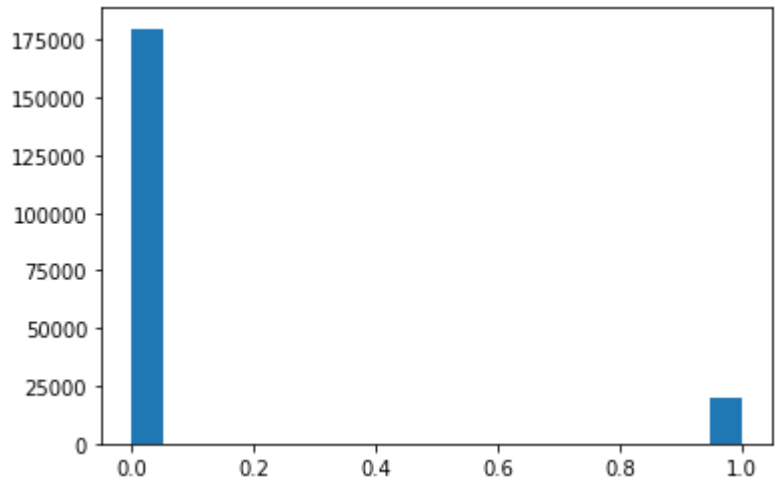

Fig 5.3: Distribution of target variable

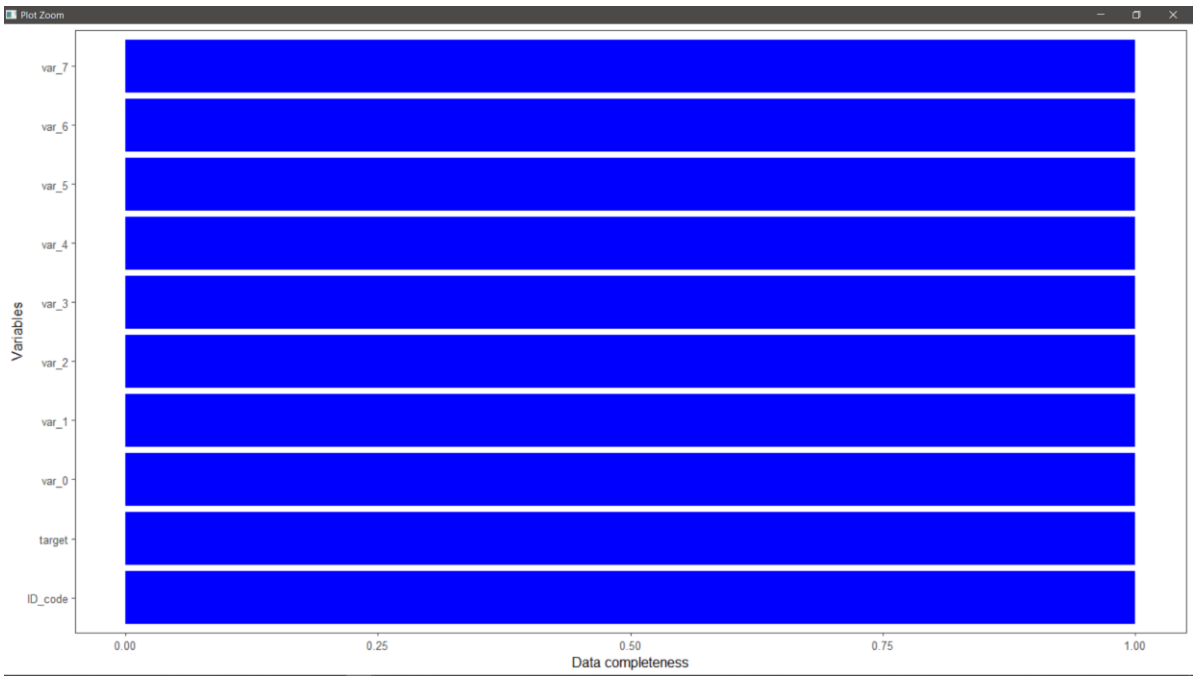

Fig 5.3:Missing Value Analysis

5.2. Code in ‘Python’ Language

```
import os
import pandas as pd
import numpy as np
import matplotlib as mlt
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import statsmodels.api as sm
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB

os.chdir("C:/Users/michael/Desktop/edWisor/PROJECT/santander project")
os.getcwd()

train = pd.read_csv("train.csv")
train.shape
train.head(5)


#missing value analysis
missing_values = pd.DataFrame(train.isnull().sum())
missing_values

#distribution of target variable
plt.hist(train['target'], bins = 'auto')


train = train.drop(['ID_code'], axis = 1)


########################## MODEL SELECTION ################################

## DecisionTree ##
#Accuracy = 83.34 %
#Precision = 19.09
#Recall = 20.09
#F1 score = 19.4

X = train.values[:, 1:200]
Y = train.values[:, 0]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)

model1 = tree.DecisionTreeClassifier(criterion = 'entropy').fit(X_train,
Y_train)

model1_pred = model1.predict(X_test)

con_mat1 = confusion_matrix(Y_test, model1_pred)
con_mat1 = pd.crosstab(Y_test, model1_pred)
TN = con_mat1.iloc[0,0]
FN = con_mat1.iloc[1,0]
TP = con_mat1.iloc[1,1]
```

```
FP = con_mat1.iloc[0,1]

accuracy = accuracy_score(Y_test, model1_pred)*100
precision = (TP)/(TP+FP)
recall = (TP)/(TP+FN)
f1_score = 2*(0.097)


## Logistic Regression ##
#Accuracy = 91.24 %
#Precision = 66.83
#Recall = 26.28
#F1 score = 37.6

sample = np.random.rand(len(train))<0.8
train_data = train[sample]
test_data = train[~sample]
train_cols = train.columns[1:200]


model2 = sm.Logit(train_data['target'], train_data[train_cols]).fit()
model2.summary()
model2_pred = model2.predict(test_data[train_cols])

test_data['Actual_values'] = 1
test_data.loc[model2_pred<0.5, 'Actual_values'] = 0
con_mat2 = pd.crosstab(test_data['target'], test_data['Actual_values'])
TN = con_mat2.iloc[0,0]
FN = con_mat2.iloc[1,0]
TP = con_mat2.iloc[1,1]
FP = con_mat2.iloc[0,1]


model2_accuracy = (TP+TN)/(TP+TN+FP+FN)
model2_precision = (TP)/(TP+FP)
model2_recall = (TP)/(TP+FN)
model2_f1score = 2*(0.188)




## Naive Bayes ##
#Accuracy = 92.23 %
#Precision = 70.80
#Recall = 36.68
#F1 score = 48.2


model3 = GaussianNB().fit(X_train, Y_train)
model3_pred = model3.predict(X_test)

con_mat3 = pd.crosstab(Y_test, model3_pred)
TN = con_mat3.iloc[0,0]
FN = con_mat3.iloc[1,0]
TP = con_mat3.iloc[1,1]
FP = con_mat3.iloc[0,1]


model3_accuracy = accuracy_score(Y_test, model3_pred)*100
model3_precision = (TP)/(TP+FP)
```

```
model3_recall = (TP)/(TP+FN)
model3_f1score = 2*(0.241)


test = pd.read_csv("test.csv")
test.head(5)

final = pd.DataFrame({"ID_code" : test.ID_code.values})
final["target"] = model3_pred
```