Project:
Santander Customer Transaction Prediction

Submitted By: Divyanshu
Date: 26/02/2020

Contents

1. Introduction

At Santander, 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 Data

The given dataset is split into train and test data. The dataset we are provided with has an anonymized numeric feature variables, the binary target column, and a string ID_code column. The task is to predict the value of target column in the test set by training our train data on various Machine Learning Algorithms.

1.3 Goal

Aim of the project is to build Classification modal The process of building the model start from basic use of basic algorithms like Logistic Regression and after that we used four different algorithms which are Decision trees, Random Forest, KNN(K Nearest Neighbour) and Naïve Bayes to develop our model and predict target variable. For Classification the Accuracy metrics we considered are AUC, Precision & Recall.

2. Methodology

2.1 Exploratory Data Analysis

- Set working directory: To import and export all data
  os.chdir

- Load Datatraining data as: santander_train
  santander_train = pd.read_csv('train.csv')

- Checking the shape of Data:
  santander_train.shape

- Checking the info of the Data set and type of each variable in the dataset:
  santander_train.info()

- Check the head of the data set to know the first five entries of the dataset:
  santander_train.head()

- Apply describe() method to calculate the mean, variance, and standard deviation etc of the dataset:
  santander_train.describe()

2.2 Pre-Processing

2.2.1 Missing Value Analysis:

Missing value pertains to condition when one or more values are absent from one or moreindependentvariables of the dataset. The reason for missing values could be many. Some commonly included reasons are Human error, Refuse to answer during survey, optional box questionnaire etc. Missing valuesin a dataset may be imputed or that particular variable may be dropped depending upon the percentage of missing values present in the column.

There are various methods to impute missing values some of which are:

- Mean
- Mode
- Median
- KNN imputation
- Prediction method

Checking the missing value in the Dataset: Our Dataset does not contain any missing value.

```
missing_val = pd.DataFrame(santander_train.isnull().sum())
missing_val[0].sum()
```

2.2.2 Outlier Analysis:

Observations inconsistent with the rest of the dataset is called an outlier. Causes of outliers are poor or contaminated data, low quality measurements, malfunctioning equipments, correct but exceptional data.

- Plot boxplot to visualize Outliers:

```
plt.boxplot(santander_train['var_0'])
```

- Calculating outliers and inliers in each variable:

```
for i in range(2,202):
#print(i)
q75 = np.percentile(santander_train.iloc[:,i],75)
q25 = np.percentile(santander_train.iloc[:,i],25)
iqr = q75 -q25
```

- Calculating inter quartile range:

```
min = q25 -(iqr*1.5)
max = q75 + (iqr*1.5)
```

- Drop those observations which are beyond outliers and inliers:
  santander_train=santander_train.drop(santander_train[santander_train.iloc[:,i] < min].index)
  santander_train=santander_train.drop(santander_train[santander_train.iloc[:,i] > max].index)
- Check again the shape of Dataset after removing outliers.

2.2.3 Feature selection:

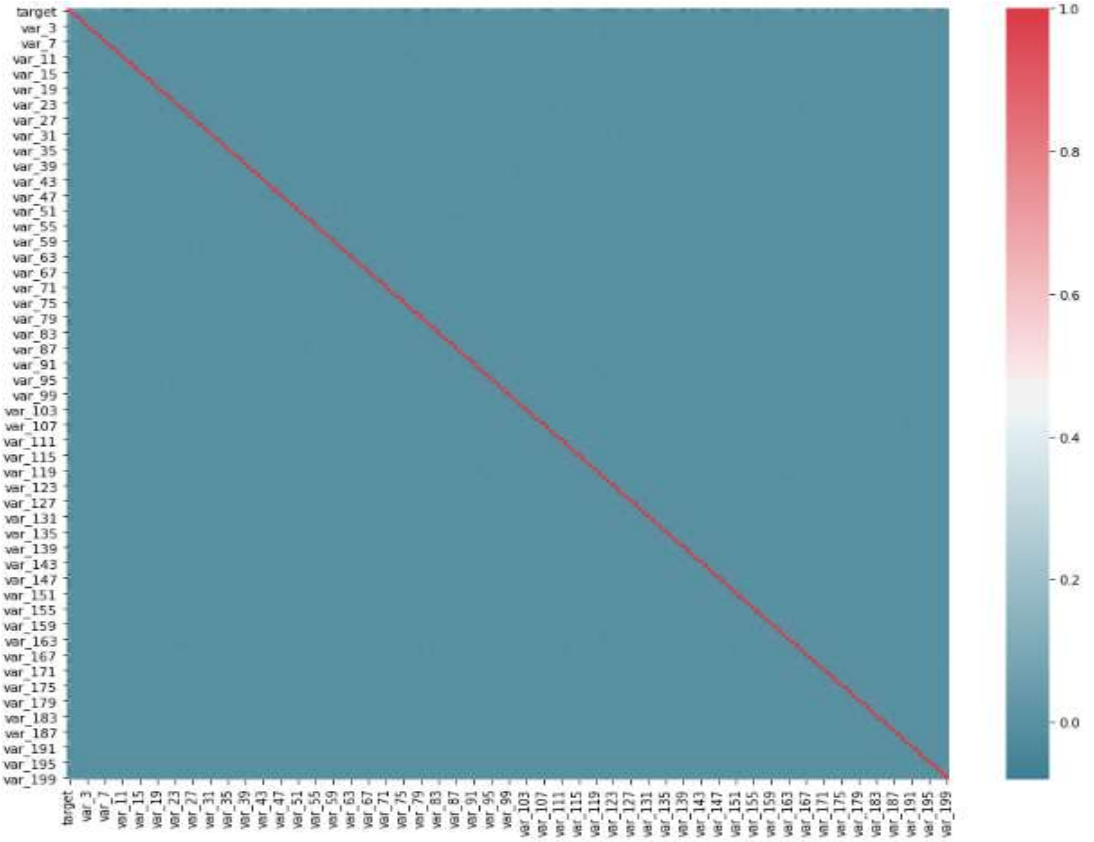
Feature selection refers to selecting a subset of relevant features for use in model development. It refers to a subset of learning algorithms input variable upon which it should focus attention while ignoring the rest. It also reduces the dimensionality of the dataset.

The following set of steps are followed for feature selection

- Generate Correlation Matrix:
  corr = santander_train.corr()9

- Plot heatmap using seaborn library to identify correlation between the various independent variables:

  sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(220, 10, as_cmap=True),square=True, ax=ax)



- As can be seen from the above correlation plot that there is no correlation between the continuous independent variables so there is no need to drop any variable from the Dataset.
- Since our Datasetdoes not contain any categorical variable so there is no need to do Chi square test for feature selection.

2.2.4 Feature scaling:

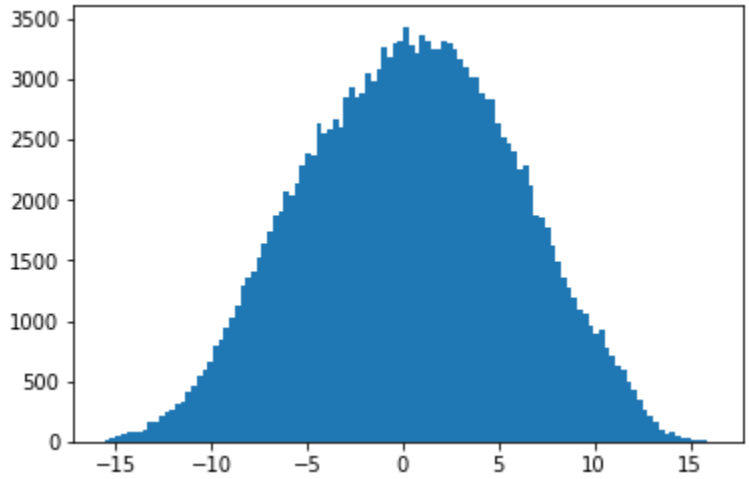
Feature scaling refers to the task of converting the various values of the column within the specified limits. Feature scaling is done to bring all the variable values within the limits so that no single variable has more influence over other variables over other variables while developing the model. It is the process of reducing the variations either within or between the variables. It brings all the variables into proper proportions with one another.
Feature scaling commonly uses two methods to compute.

- Normalization:

```
for i in range(2,202):
#print(i)
santander_train.iloc[:,i]=(santander_train.iloc[:,i]-
np.min(santander_train.iloc[:,i]))/(np.max(santander_train.iloc[:,i])-
np.min(santander_train.iloc[:,i]))
```

- Standardisation:

```
for i in range(3,202):
print(i)
santander_train.iloc[:,i]=(santander_train.iloc[:,i]-
np.mean(santander_train[:,i])/np.std(santander_train[:,i])
```



Any one of the above methodscan be adopted while doing a Feature scaling on dataset depending upon whether different independent variables are normally distributed or not.

3. Model Development

Machine learning is programming computers to optimize a performance criterion using example data or past experience. During the development of this model we applied various machine learning algorithms.

- Replace target variable categories with Yes or No from 1/0:

```
santander_train['target'] = santander_train['target'].replace(0, 'No')
santander_train['target'] = santander_train['target'].replace(1, 'Yes')
```

- Divide data into train and test for model developmentand to check accuracy and deduce which model is best:

```
X = santander_train.values[:, 2:202]
Y = santander_train.values[:,1]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

3.1 LOGISTIC REGRESSION:

- It is a Classification Model
- Input can be continuous or categorical
- Possible outcomes could be class and probabilities
- Predicts the probability of particular outcomes
- Use logistic function to estimate the prediction
- Can be binomial, ordinal or multinomial

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

- Above Formula shows Logistic Function

1. **Import Libraries:**

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

2. **Build and train Logistic Regression model using X_train and y_train:**

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

3. **Predict model outcome using test data X_test:**

```
predictions = logmodel.predict(X_test)
```

4. **Print classification report using y_ test and predicted values from Logistic Regression model:**

print(classification_report(y_test,predictions))

	precision	recall	f1-score	support
No	0.92	0.99	0.95	31510
Yes	0.69	0.26	0.37	3505
accuracy			0.91	35015
macro avg	0.81	0.62	0.66	35015
weighted avg	0.90	0.91	0.90	35015

5. **Build confusion matrix as CM to check accuracy:**

CM = pd.crosstab(y_test,predictions)

col_0	No	Yes
row_0		
No	31113	397
Yes	2607	898

6. **Compute AUC:**

metrics.roc_auc_score(y_test , predictions)

7. **Save TP, TN, FP, FN:**

TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

8. **check accuracy of model:**

((TP+TN)*100)/(TP+TN+FP+FN) = 91.54%

3.2DECISION TREE:

Decision Tree is a predictive model based on a branching series of Boolean tests. It can be used for classification and regression both type of target variables. There are number of different types of decision trees that can be used in Machine learning algorithms. Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. They are extremely easy to understand by the business users. Decision trees build some intuitions about your customer base. E.g. “are customers with different family sizes truly different?”

1. **Import Libraries:**

from sklearn import tree
fromsklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score15

2. **Model Development:**
   C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)

3. **Predict new test cases:**
   C50_Predictions = C50_model.predict(X_test)

4. **Build confusion matrix:**
   from sklearn.metrics import confusion_matrix
   CM = pd.crosstab(y_test, C50_Predictions)

col_0	No	Yes
row_0		
No	28716	2791
Yes	2815	693

5. **AUC:**
   metrics.roc_auc_score(y_test, C50_Predictions)

6. **Save TP, TN, FP, FN:**
   TN = CM.iloc[0,0]
   FN = CM.iloc[1,0]
   TP = CM.iloc[1,1]
   FP = CM.iloc[0,1]

7. **Build Classification Report:**
   print(classification_report(y_test,C50_Predictions))

	precision	recall	f1-score	support
No	0.91	0.91	0.91	31626
Yes	0.18	0.19	0.19	3389
accuracy			0.84	35015
macro avg	0.55	0.55	0.55	35015
weighted avg	0.84	0.84	0.84	35015

8. **Check accuracy of model:**
   ((TP+TN)*100)/(TP+TN+FP+FN) = 83.82%

3.3 RANDOM FOREST:

Random forest is an ensemble that consists of many decision trees. The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995.Thismethod combines Breiman's "bagging" idea and the random selection of features. It outputs the class that is the mode of the class's output by individual trees. It calculates results by evaluating mean for regression and mode for classification. This algorithm can be used for both classification and regression.

1. **Import Libraries:**
   from sklearn.ensemble import RandomForestClassifier

2. **Develop and train random forest modelon train data:**
   RF_model = RandomForestClassifier(n_estimators = 10).fit(X_train, y_train)
3. **Predict new test cases:**
   RF_Predictions = RF_model.predict(X_test)
4. **Build confusion matrix:**
   CM = pd.crosstab(y_test, RF_Predictions)17

col_0	No	Yes
row_0		
No	31471	39
Yes	3451	54

5. **Print Classification report:**
   print(classification_report(y_test,RF_Predictions))

	precision	recall	f1-score	support
No	0.90	1.00	0.95	31510
Yes	0.58	0.02	0.03	3505
accuracy			0.90	35015
macro avg	0.74	0.51	0.49	35015
weighted avg	0.87	0.90	0.86	35015

6. **AUC:**
   metrics.roc_auc_score(y_test , RF_Predictions)

7. **save TP, TN, FP, FN:**

   TN = CM.iloc[0,0]
   FN = CM.iloc[1,0]
   TP = CM.iloc[1,1]
   FP = CM.iloc[0,1]

8. **check accuracy of model:**
   ((TP+TN)*100)/(TP+TN+FP+FN) = 90.03%

3.4 NAÏVE BAYES:

Naïve Bayes is a Classification Algorithm. It is one of the most practical Machine Learning methods. It performs Probabilistic classification. It works on Bayes theorem of probability to predict the class of unknown data set. Bayes' theorem with strong (naive) independence assumptions between the features. Attribute values are conditionally independent given the target value. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:
P(c|x)is the posterior probability of class (c,target) given predictor (x,attributes).
P(c) is the prior probability of class.
P(x|c) is the likelihood which is the probability of predictor given class.
P(x) is the prior probability of predictor.

1. **Import Libraries:**
   from sklearn.naive_bayes import GaussianNB
   from sklearn.metrics import classification_report19

2. **Naive Bayes implementation:**
   NB_model = GaussianNB().fit(X_train, y_train)

3. **Predict test cases:**
   NB_Predictions = NB_model.predict(X_test)

4. **Print Classification report:**
   print(classification_report(y_test,NB_Predictions))

	precision	recall	f1-score	support
No	0.93	0.98	0.96	31510
Yes	0.72	0.36	0.48	3505
accuracy			0.92	35015
macro avg	0.83	0.67	0.72	35015
weighted avg	0.91	0.92	0.91	35015

5. **Build confusion matrix in tabular form:**
   pd.crosstab(y_test, NB_Predictions)

col_0	No	Yes
row_0		
No	31027	483
Yes	2259	1246

6. **AUC:**

metrics.roc_auc_score(y_test , C50_Predictions)

7. **save TP, TN, FP, FN:**

   TN = CM.iloc[0,0]
   FN = CM.iloc[1,0]
   TP = CM.iloc[1,1]
   FP = CM.iloc[0,1]

8. **Check accuracy of model:**
   ((TP+TN)*100)/(TP+TN+FP+FN) = 92.18%

4. SUMMARY OF MODELS

- **PRECISION CRITERIA IS AS FOLLOWS:**
  Random forest> Naive Bayes> Logistic Regression> Decision tree

- **RECALL CRITERIA IS AS FOLLOWS:**
  Naive Bayes> Logistic Regression> Decision tree> Random forest

- **AUC CRITERIA IS AS FOLLOWS:**
  Naive Bayes> Logistic Regression> Decision tree> Random forest

- **ERRORS:**

1. **LOGISTIC REGRESSION:**
   Recall = 26.3435
   Precision = 69.0573
   AUC = 62.53%
   Accuracy = 92%

2. **DECISION TREE:**
   Recall = 19.53
   Precision = 19.1721
   AUC = 54.89%
   Accuracy = 84%

3. **RANDOM FOREST:**
   n_estimators = 100
   Recall = 0.0195
   Precision = 100.0
   AUC = 50.00%
   Accuracy = 90%

4. **NAIVE BAYES:**
   Recall = 35.8608
   Precision = 72.0455
   AUC = 67.08%
   Accuracy = 92%

To get the most accurate model out of various models the value of recall, precision, AUC should be high". As per the directions of our project we have to predict the results based on recall, precision and accuracy of all machine learning algorithm. Out of all the above developed Machine Learning algorithms we can deduce that "Naive Bayes" is giving all the quantities highest among all other algorithms. Hence "Naive Bayes" is selected to predict target variable from our given test data.

5. Predicting Test Data

After forming the machine learning algorithm and test edit for test data we have fetch out from train data set the most accurate method. Now we are ready to predict any external data given to us. Now in the external data named as test.csv is given to us contain all variables of train data except “target” variable. Our task is to predict the target value in form of 0 and 1 where using the most appropriate algorithm we have formed in previous section. As we have developed in our previous section that NAÏVE BAYES gives most accurate model so we will use NAÏVE BAYES to predict the target value.
Steps followed for predicting target variable in test data.

- **Load Data testing data:**
  santander_test = pd.read_csv('test.csv')

- **Checking the shape of the test dataset:**
  santander_test.shape()

- **Since this test data it does not contain target variable which we have to predict:**
  santander_test.head(5)

- **Drop the ID_code column from the Dataset as our model is not trained for it:**
  santander_test = santander_test.drop(['ID_code'], axis=1)

- **Now we apply our model to Naive Bayes to predict our target variable:**
  santander_test['target_pred'] = NB_model.predict(santander_test)
- **Our 'target_pred' column gets added to as the last column of the dataset**

- **Now writing this new dataset wit predicted variable as a CSV file:**
  santander_test.to_csv('santander_test_predict.csv',index=False)

- **The CSV file of the target variable is also included.**