Santander Customer Transaction Prediction Project

Submitted By:
UMANG

Contents

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:

Number of attributes:

We are provided with an anonymized dataset containing 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.

Let's take a quick view of a sample of the given train data:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405

2. Methodology:

In the first step, all the required libraries were installed and imported i.e. numpy, pandas, matplotlib, seasborn etc. to run the code.

2.1 Data Pre-processing:

After importing all the required libraries, data preprocessing is the first step to do. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. It involves transforming data into a basic form that makes it easy to work with. Data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen.

Further, in this stage, we'll deal with outliers, missing values and some unwanted data that we don't want in our model.

2.2 Missing value analysis:

After analysing, we can see that there is no missing value in the data set.

```
(train.isnull().sum()).sum()
```

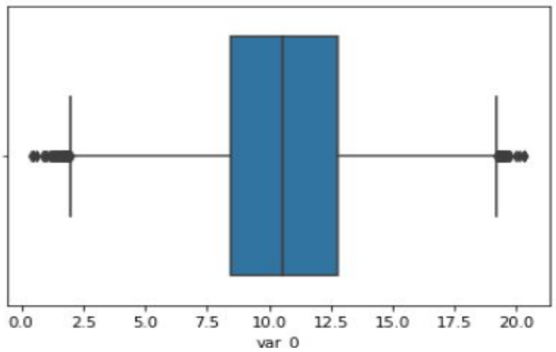
```
0
```

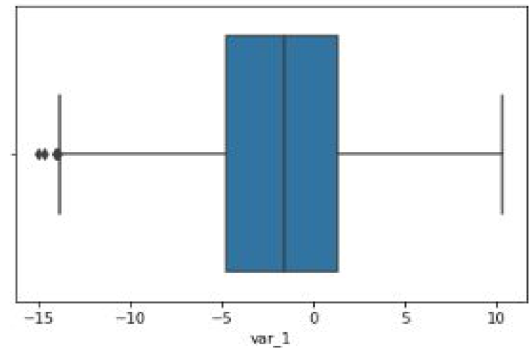
2.3 Outlier analysis:

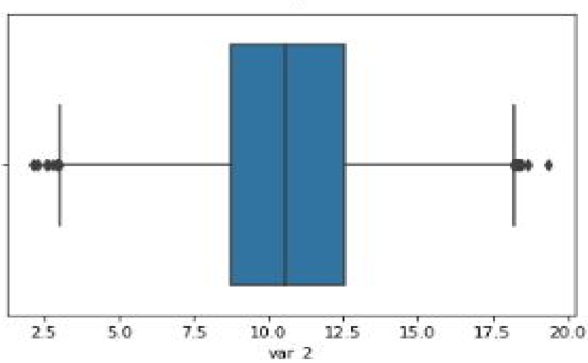
After analysing the missing values, the next step performed is outlier analysis. An outlier is a data point that differs significantly from other observations. We can deal with outliers by using various methods:

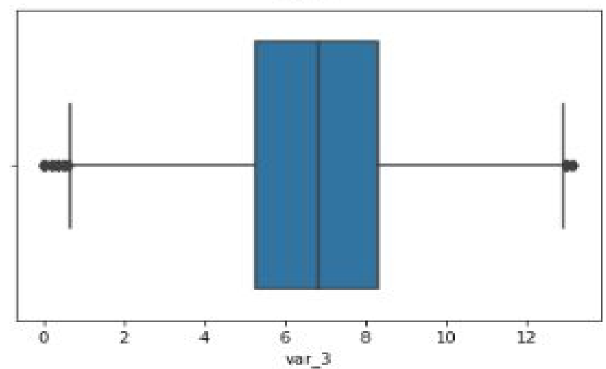
- ➢ Remove the entire rows which have outliers
- ➢ Replace outliers with NA and then apply missing value analysis.
- ➢ Replace (capping) outliers with maximum or minimum values (according to the scenario).

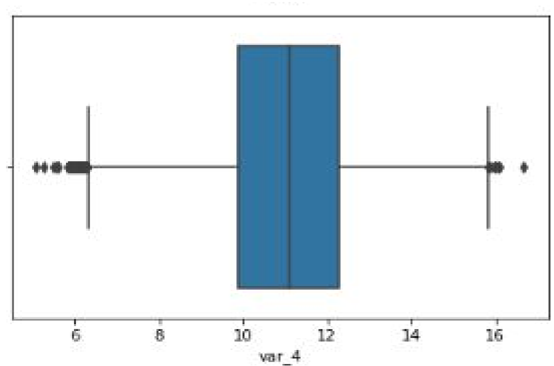
We can see the outliers by plotting the boxplot graph. I have pasted the images of boxplot graph of different variables (before removing outliers).

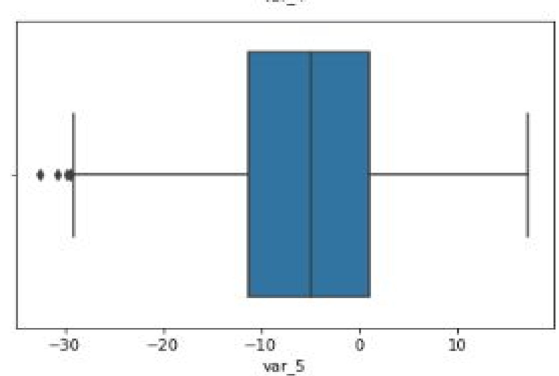












Above images show the distribution of outliers in different variables. I have taken only 6 variables (var_0, var_1, var_2, var_3, var_4 and var_5) to show the outliers.

I have replaced the outliers by using upper fence and lower fence. There is a formula to calculate the interquartile range.

Maximum **=** *Q75 + 1.5*IQR*

*Minimum = Q25 -1.5*IQR*

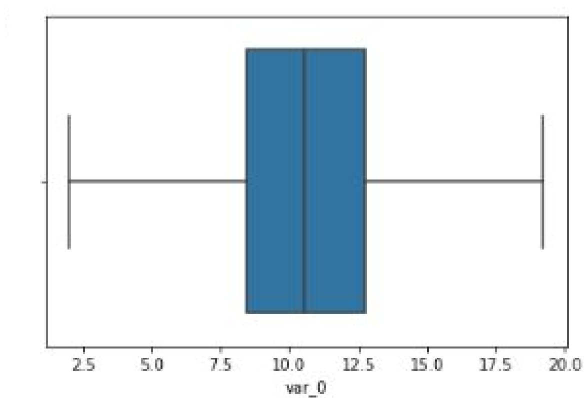
first quartile (Q25 /25th Percentile): The middle number between the smallest number (not the “minimum”) and the median of the dataset.

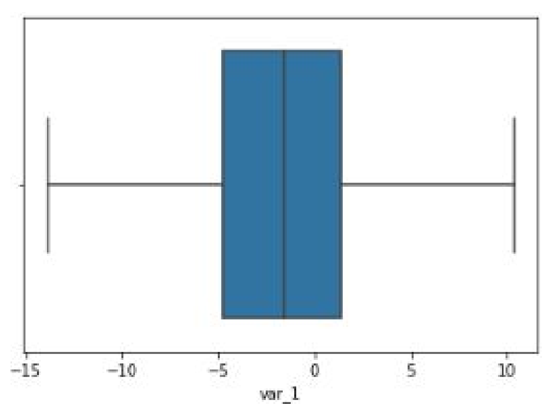
third quartile (Q75 /75th Percentile): The middle value between the median and the highest value (not the “maximum”) of the dataset.

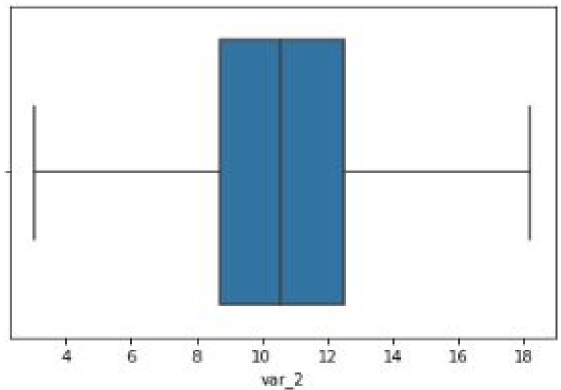
interquartile range (IQR): 25th to the 75th percentile.

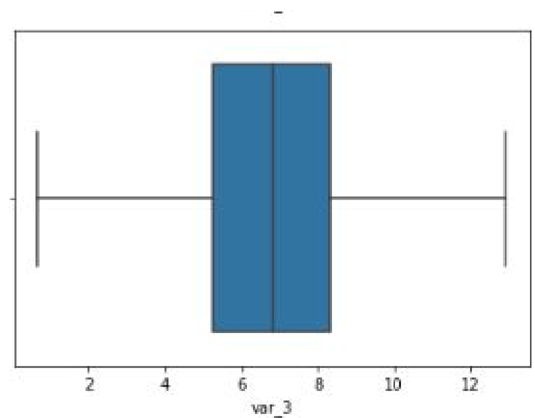
The values which fall beyond “maximum” and “minimum” are treated as outliers. We have almost 13.26% outliers in the whole dataset, if we remove them, we might lose some important information so I decided to replace these values by capping them with maximum or minimum values i.e. if a value falls beyond the upper fence then I had made it equal to the “maximum” similarly, if a value falls beyond the lower fence then I had made it equal to the “minimum”.

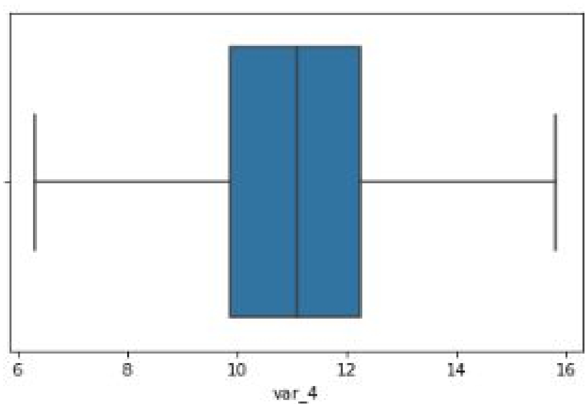
Now, I am pasting the images of boxplot graph of variables after removing the outliers.

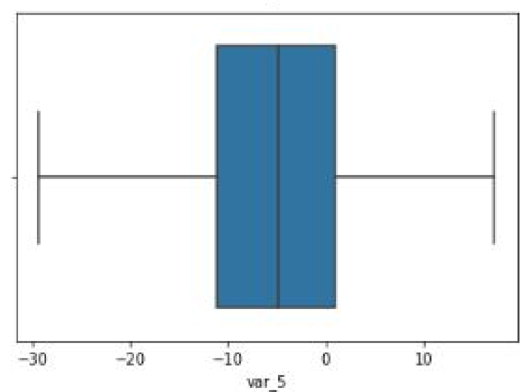










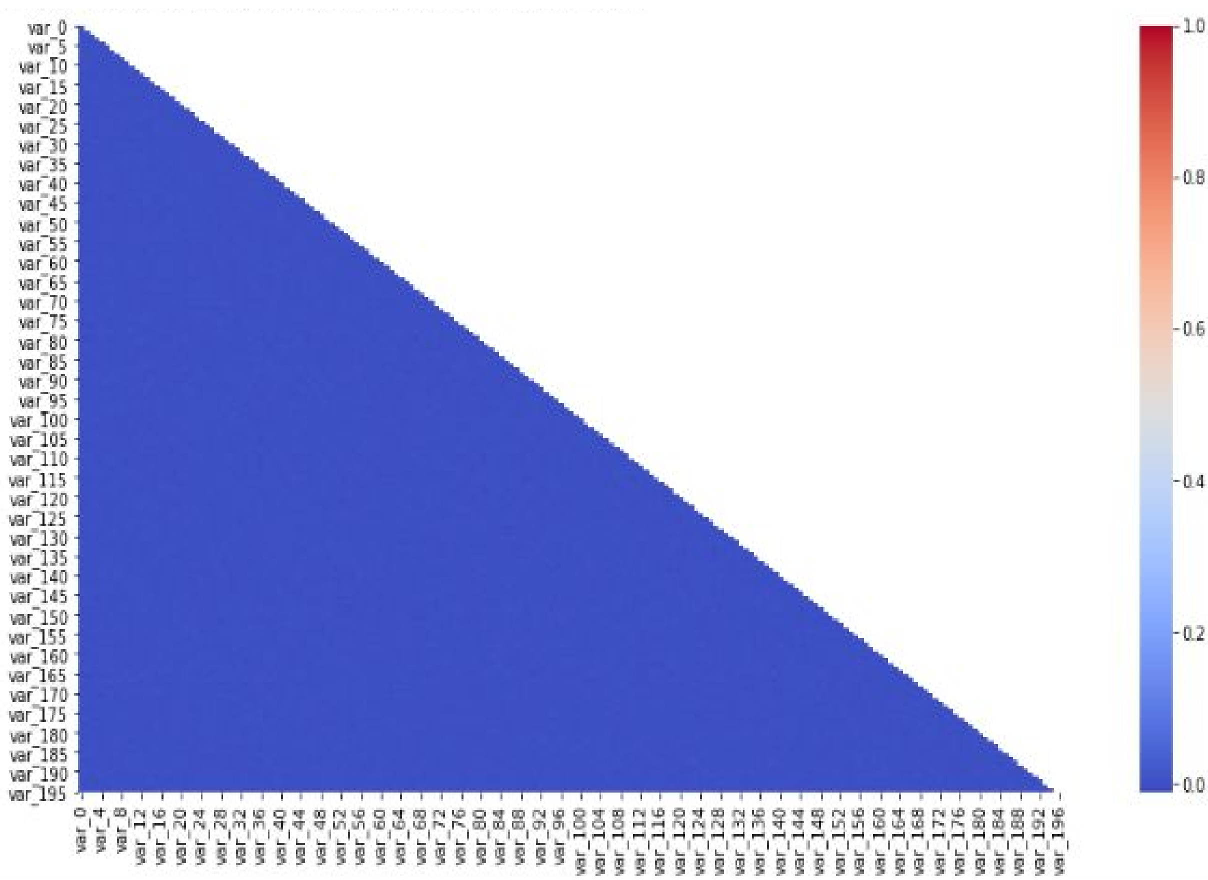


Above figures are plotted after removal of outliers and these figures show that the outliers in different variables have been removed/replaced.

3. Feature Selection:

Feature selection is the process where we select those features which contribute most to our prediction variable. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features. We can remove the irrelevant features by using some statistical techniques:

3.1 Correlation analysis: It compares two numerical variables in a contingency table to see if they are related or not.



Correlation plot of numeric variable

- After observing the above graph, we can see that there is no dependency between numeric variables.

3.2 ANOVA test:

In both Python and R, I have done ANOVA test, it compares categorical variables with numerical variable.

Null hypothesis: No relationship exists on the categorical variables. The two variables are independent of each other.
Alternate hypothesis: The two variables are not independent.

- If p-value < 0.05 then we reject the null hypothesis, saying that these two variables are independent. If p-value > 0.05 then we can't reject the null hypothesis, saying that these two variables are not independent.
- If p-value > 0.05, remove the variable from our data set and if p-value < 0.05, keep the variable.

```
               sum_sq        df           F         PR(>F)
var_0       49.646959       1.0  550.749071  1.264733e-121
Residual 18028.705021  199998.0         NaN            NaN
               sum_sq        df           F         PR(>F)
var_1       45.817526       1.0  508.160051  2.203099e-112
Residual 18032.534454  199998.0         NaN            NaN
               sum_sq        df           F         PR(>F)
var_2       56.437211       1.0  626.311328  5.179204e-138
Residual 18021.914769  199998.0         NaN            NaN
               sum_sq        df          F        PR(>F)
var_3        2.209272       1.0  24.443822  7.657022e-07
Residual 18076.142708  199998.0        NaN           NaN
               sum_sq        df          F    PR(>F)
var_4        2.155092       1.0  23.844295  0.000001
Residual 18076.196888  199998.0        NaN       NaN
               sum_sq        df           F        PR(>F)
var_5       17.348524       1.0  192.108384  1.154409e-43
Residual 18061.003456  199998.0         NaN           NaN
               sum_sq        df          F         PR(>F)
var_6       80.539507       1.0  894.98323  3.287946e-196
Residual 17997.812473  199998.0        NaN            NaN
               sum_sq        df         F    PR(>F)
var_7        0.165327       1.0  1.829002  0.176247
Residual 18078.186653  199998.0       NaN       NaN
               sum_sq        df          F        PR(>F)
var_8        6.933731       1.0  76.736221  1.968674e-18
Residual 18071.418249  199998.0        NaN           NaN
              sum_sq        df           F        PR(>F)
var_9       33.12516       1.0  367.131193  9.339174e-82
Residual 18045.22682  199998.0         NaN           NaN
               sum_sq        df        F    PR(>F)
var_10       0.088433       1.0  0.97833  0.322613
Residual 18078.263547  199998.0      NaN       NaN
```

Table of ANOVA test (there are 200 observations but I have pasted only 11)

- By observing the above table, we have found that there are 19 columns which have p-value is greater than 0.05, hence removed.
- The variable which were removed after performing ANOVA test are mentioned below:

```
for i in cnames:
    mod = ols("target" + '~' + i, data = df_train_final).fit()
    aov_table = sm.stats.anova_lm(mod, typ = 2)
    #print(aov_table)
    if aov_table["PR(>F)"][0] > 0.05:
      del df_train_final[i]
      print(i)
```

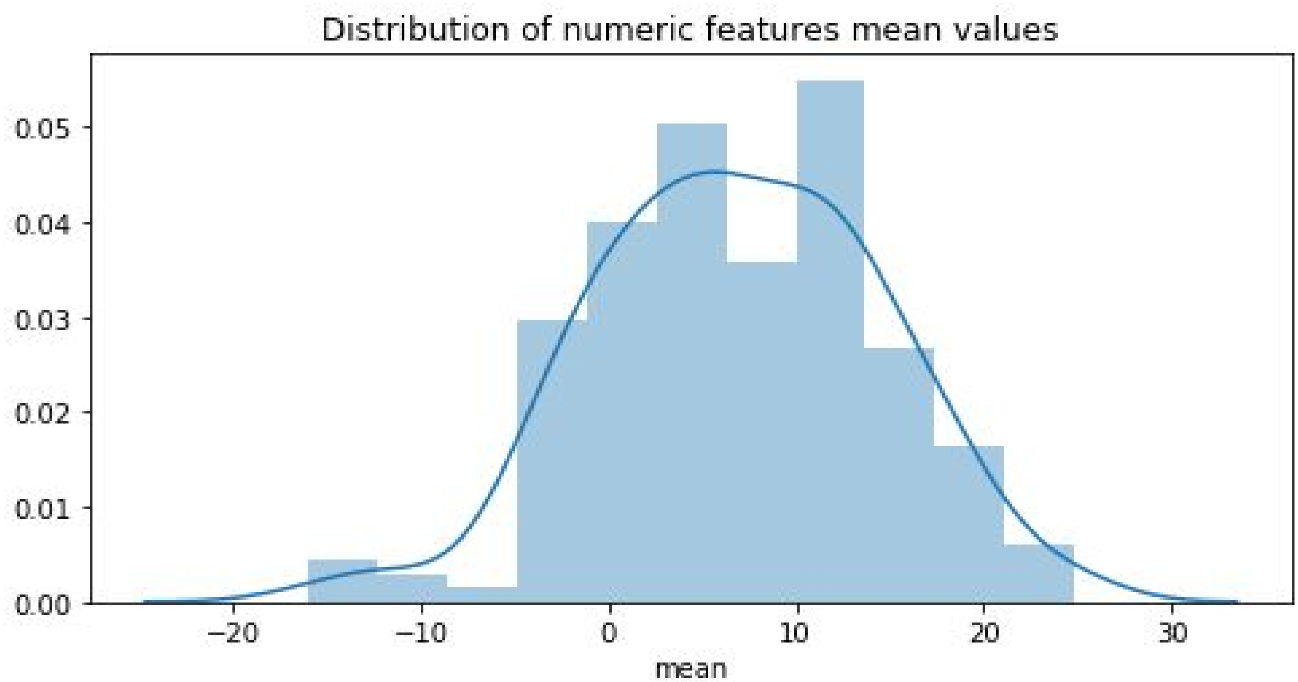
```
var_7
var_10
var_17
var_27
var_30
var_38
var_39
var_41
var_96
var_98
var_100
var_103
var_117
var_124
var_126
var_136
var_158
var_161
var_185
```

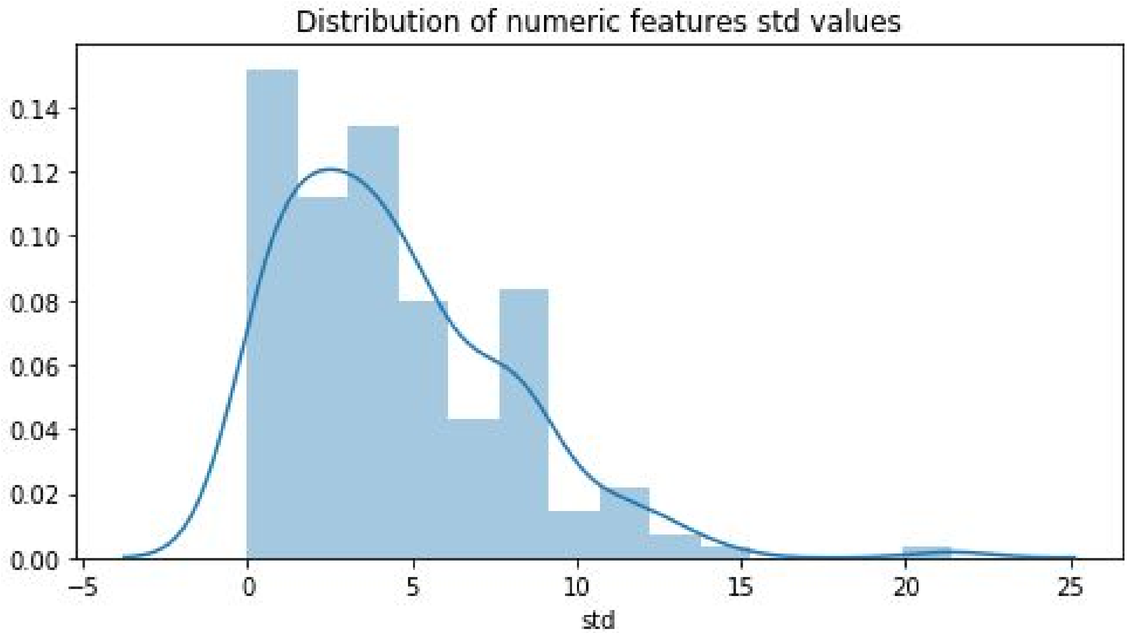
4 Feature Scaling:

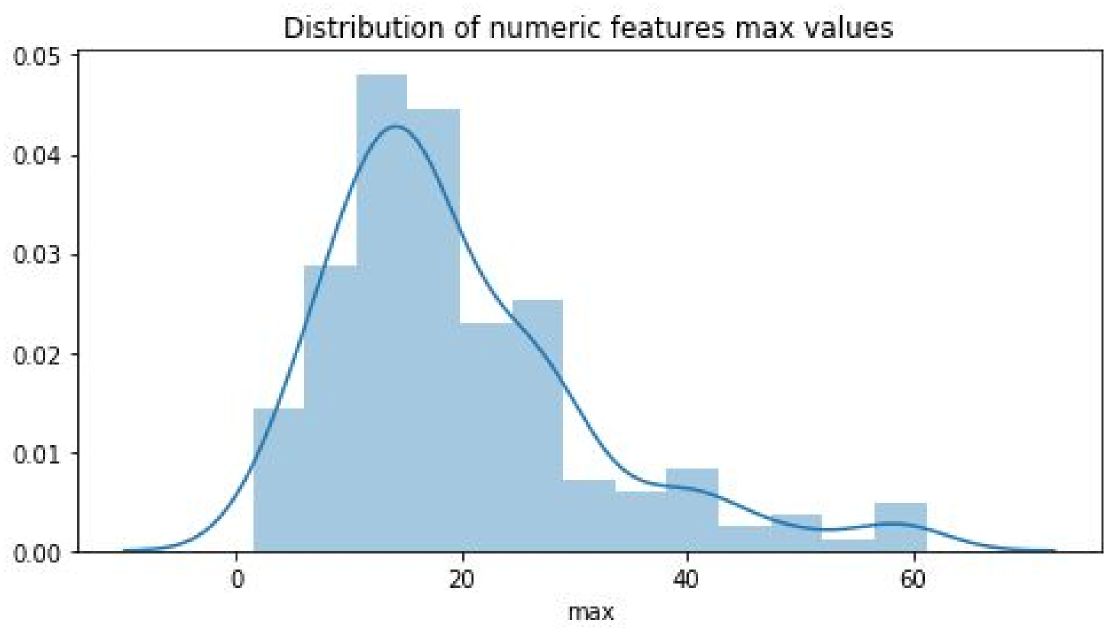
It is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization. Sometimes, it also helps in speeding up the calculations in an algorithm. I have done normalisation in both python and R.

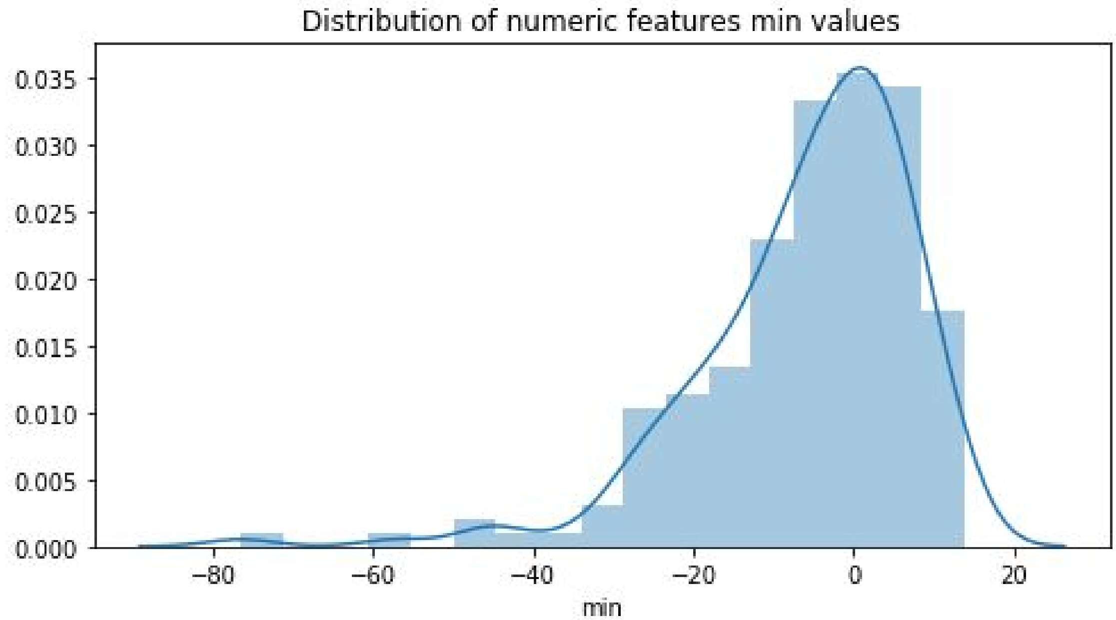
4.1 Normalisation:

Normalisation is a technique often applied as part of data preparation for machine learning. The goal of normalisation is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.
In normalisation, we normalise the numeric variable in the same range of (0,1).



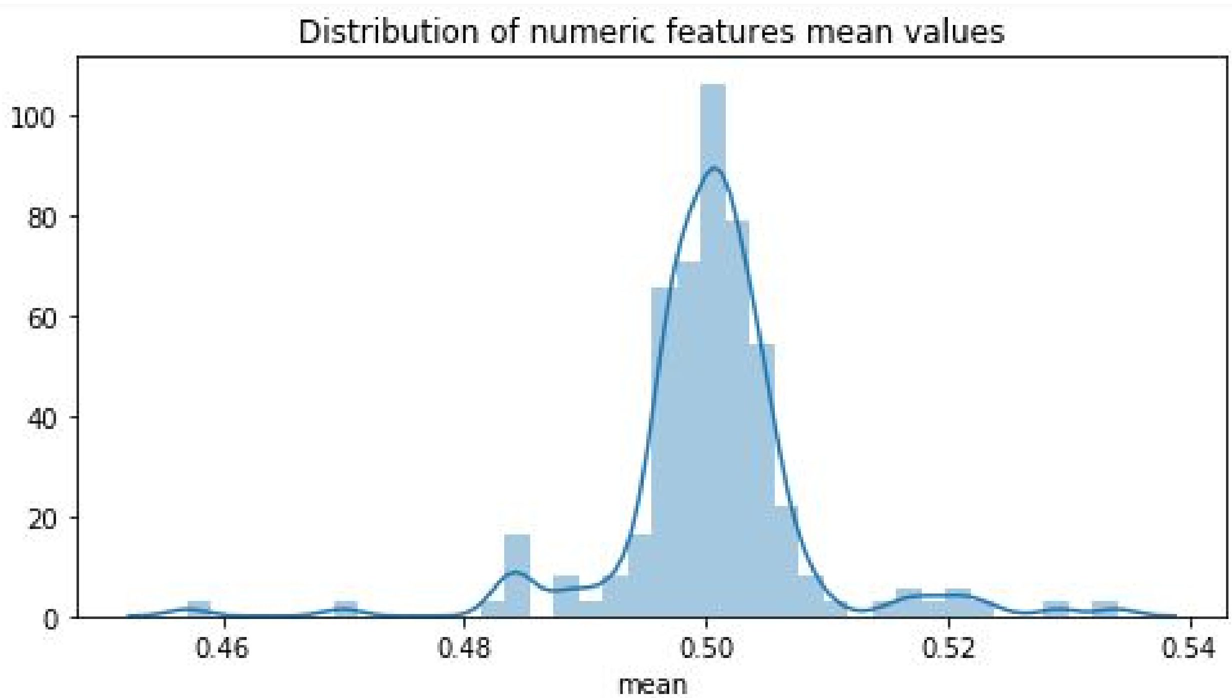

Distribution of numeric features std values
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−5
0
5
10
15
20
25
std

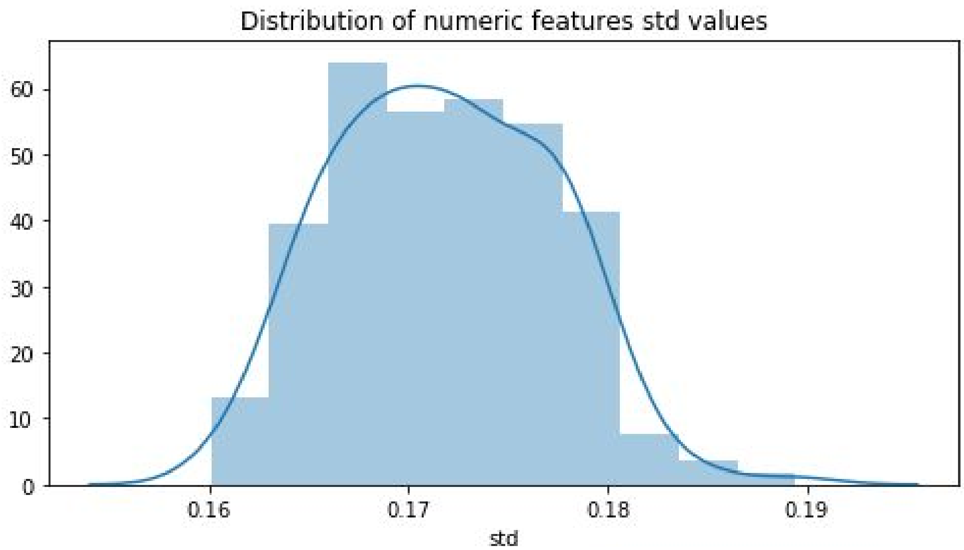

Distribution of numeric features max values
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
max

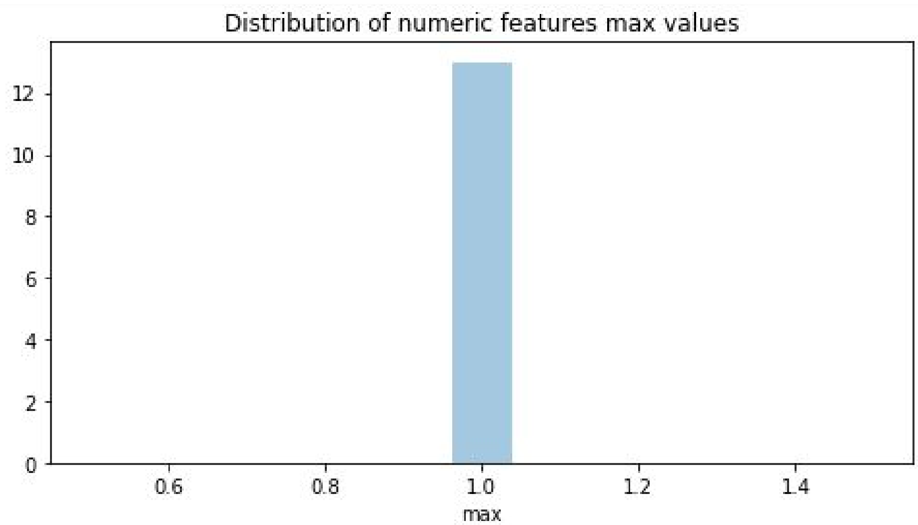


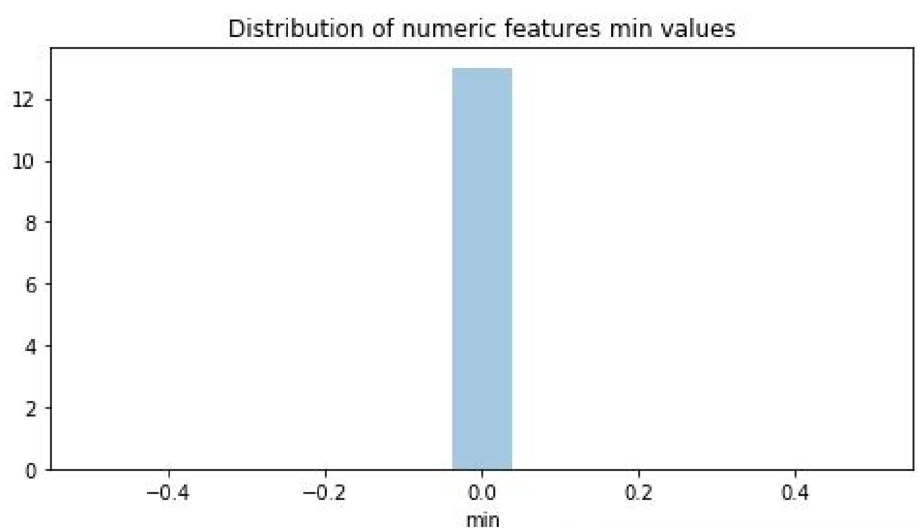
- As we can clearly see in the above graphs that our data is skewed, I have normalised all the numeric variable in the same range of (0,1).

<u>Graphs after normalisation:</u>




Distribution of numeric features std values
0
10
20
30
40
50
60
0.16
0.17
0.18
0.19
std


Distribution of numeric features max values
0
2
4
6
8
10
12
0.6
0.8
1.0
1.2
1.4
max


Distribution of numeric features min values
0
2
4
6
8
10
12
−0.4
−0.2
0.0
0.2
0.4
min

5. Sampling:

5.1 Checking Dataset (balanced or imbalanced):

Since it is a binary classification problem so that after doing outlier analysis, the type data set (whether it is balanced or imbalanced) is to be checked. Imbalanced classes put “accuracy” out of business. This is a common problem in machine learning (specifically in classification), occurring in datasets with a disproportionate ratio of observations in each class.

To do this, we have created a table of target variable in which the number of “1” and “0” is to be counted.

```
df_train["target"].value_counts()
```

```
0    179902
1     20098
Name: target, dtype: int64
```

```
sns.countplot(x= "target", data = df_train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd41699940>
```

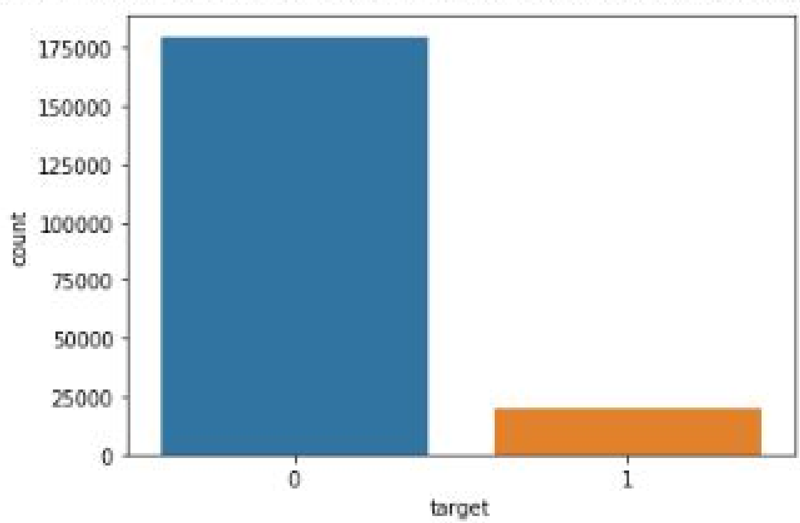


Fig: Distribution of target variable before sampling

We can clearly see in the graph that our data set is imbalanced i.e. number of “0s” are more than that of number of “1s”.

To overcome this imbalance problem we have to apply various sampling techniques. I have applied two sampling techniques on the data set.

1. *__Down-sampling:__* Down-sampling involves randomly removing observations from the majority class to prevent its signal from dominating the learning algorithm.

   The most common heuristic for doing so is resampling without replacement.

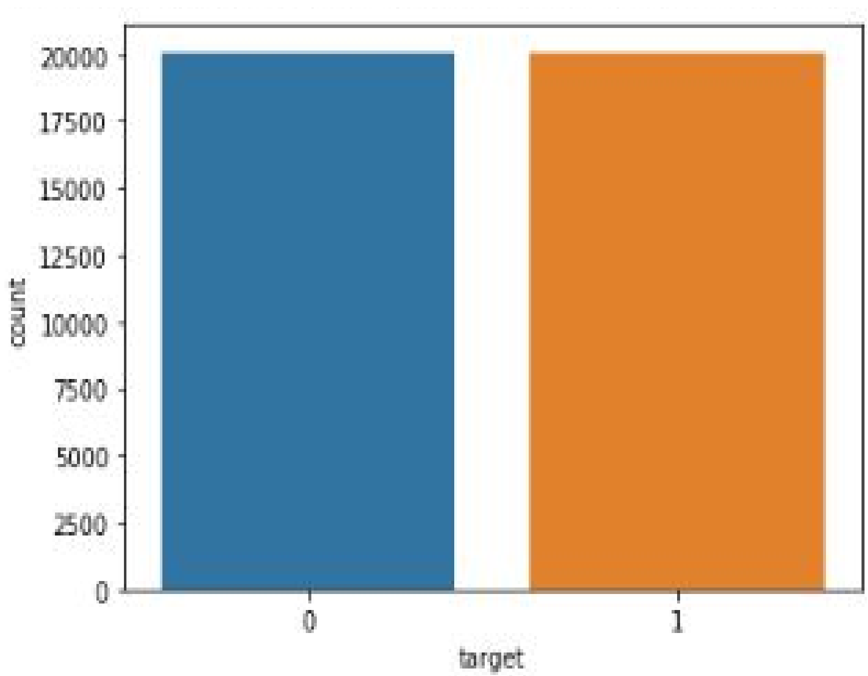


Fig: Distribution of target variable after down-sampling

2. *__Over-sampling:__* Over-sampling is the process of randomly duplicating observations from the minority class in order to reinforce its signal or we can do it by using SMOTE (Synthetic Minority Over-sampling Technique) or Random over sampling in python and ROSE in R. It is a well-known way to potentially improve models trained on imbalanced data.

The right way to do over-sample the data is that we should do it after splitting the data into test and train and oversampling on only the training data.

Upsampling using SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique. This is a statistical technique for increasing the number of cases in your dataset in a balanced way. The module works by generating new instances from existing minority cases that you supply as input. This implementation of SMOTE does not change the number of majority cases. We have oversampled only training dataset, SMOTE is not applied on test dataset.

```
[107] sm = SMOTE(random_state=42, ratio =1)
      X_train_smote, y_train_smote = sm.fit_sample(X_train, y_train)
```

```
unique, count = np.unique(y_train_smote, return_counts= True)
y_train_value_count = {k:v for (k,v) in zip(unique,count)}
y_train_value_count
```

```
{0: 143922, 1: 143922}
```

Fig: Distribution of target variable after up-sampling

6. Model development and result:

After preprocessing of data we must proceed with model development. Firstly, we split the clean data into test & train and then develop and apply different models. I have applied four different models for up sampling and downsampling both and after comparing all the models, the best is chosen.

(a) *Logistic Regression:* Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.

Result for downsampling:

```
Confusion Matrix
     Predicted     0     1
Actual
0              3156   882
1               868  3134

ROC AUC Score:
 0.782341741462107

Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.78      0.78      4038
           1       0.78      0.78      0.78      4002

    accuracy                           0.78      8040
   macro avg       0.78      0.78      0.78      8040
weighted avg       0.78      0.78      0.78      8040
```

Result for upsampling:

```
Confusion Matrix
     Predicted      0     1
Actual
0              28510  7470
1                954  3065

ROC AUC Score:
 0.7775060887134084

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.79      0.87     35980
           1       0.29      0.76      0.42      4019

    accuracy                           0.79     39999
   macro avg       0.63      0.78      0.65     39999
weighted avg       0.90      0.79      0.83     39999
```

(b) *XGBoost classifier:* XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

Result for downsampling:

```
Confusion Matrix
     Predicted     0     1
Actual
0              3139   899
1              1092  2910

ROC AUC Score:
 0.7522507319891317

Classification Report:
               precision    recall  f1-score   support

           0       0.74      0.78      0.76      4038
           1       0.76      0.73      0.75      4002

    accuracy                           0.75      8040
   macro avg       0.75      0.75      0.75      8040
weighted avg       0.75      0.75      0.75      8040
```

Result for upsampling:

```
Confusion Matrix
     Predicted      0     1
Actual
0              30293  5687
1               2479  1540

ROC AUC Score:
 0.6125599310722649

Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.84      0.88     35980
           1       0.21      0.38      0.27      4019

    accuracy                           0.80     39999
   macro avg       0.57      0.61      0.58     39999
weighted avg       0.85      0.80      0.82     39999
```

(c) ***DecisionTree classifier:*** Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Result for downsampling:

```
Confusion Matrix
     Predicted     0     1
Actual
0             2430  1608
1             1644  2358

ROC AUC Score:
 0.5954942291112988

Classification Report:
               precision    recall  f1-score   support

           0       0.60      0.60      0.60      4038
           1       0.59      0.59      0.59      4002

    accuracy                           0.60      8040
   macro avg       0.60      0.60      0.60      8040
weighted avg       0.60      0.60      0.60      8040
```

Result for upsampling:

```
Confusion Matrix
     Predicted      0     1
Actual
0             26897  9083
1              2701  1318

ROC AUC Score:
 0.5377482354867741

Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.75      0.82     35980
           1       0.13      0.33      0.18      4019

    accuracy                           0.71     39999
   macro avg       0.52      0.54      0.50     39999
weighted avg       0.83      0.71      0.76     39999
```

(d) *Random Forest classifier:* Random Forest is an ensemble that consists of many decision trees. It can be used in both types of problem statements i.e. Regression and Classification.

Result for downsampling:

```
Confusion Matrix
     Predicted     0     1
Actual
0              2926  1112
1               828  3174

ROC AUC Score:
 0.7588597974415467

Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.72      0.75      4038
           1       0.74      0.79      0.77      4002

    accuracy                           0.76      8040
   macro avg       0.76      0.76      0.76      8040
weighted avg       0.76      0.76      0.76      8040
```

Result for upsampling:

```
Confusion Matrix
     Predicted      0    1
Actual
0              35450  530
1               3797  222

ROC AUC Score:
 0.5202536077589206

Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.99      0.94     35980
           1       0.30      0.06      0.09      4019

    accuracy                           0.89     39999
   macro avg       0.60      0.52      0.52     39999
weighted avg       0.84      0.89      0.86     39999
```

7. Conclusion:

As it can be clearly seen from the result ,Logistic Regression model is performing good for Santander Customer Transaction dataset (for all the metrics i.e. f1_score, recall and ROC_AUC). So, I have selected Logistic Regression model for the prediction of test dataset.

*F1 score :*The F1 Score is the 2*((precision*recall)/(precision+recall)). The F1 score conveys the balance between precision and recall.

Recall : It literally is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found.

ROC_AUC (Receiver Operating Characteristic Area Under Curve) Score: It indicates how well the probabilities from the positive classes are separated from the negative classes.