Rohitkumar Keswani

Data Science Intern Task

Exploratory Data Analysis

- First I began with importing all the necessary libraries and loaded the dataset in the datagram using pandas.

- Then I started with basic operations of calling the data frame and inspecting it.
- For example: info() method which gives summarized information about the data frame like column datatype, no-null content and memory usage.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12500 entries, 0 to 12499
Data columns (total 4 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   bank_transaction_id           12500 non-null  int64
 1   bank_transaction_description  12369 non-null  object
 2   bank_transaction_amount       12500 non-null  float64
 3   bank_transaction_type         12500 non-null  object
dtypes: float64(1), int64(1), object(2)
memory usage: 390.8+ KB
```

- describe() method which will give statistical information about the data like mean, standard deviation, percentile, min max values.

	bank_transaction_id	bank_transaction_amount
count	1.250000e+04	12500.000000
mean	2.226077e+07	-19.613017
std	9.391952e+05	15.060147
min	2.178620e+07	-102.590000
25%	2.178932e+07	-28.022500
50%	2.179244e+07	-19.040000
75%	2.179557e+07	-4.687500
max	2.414033e+07	-0.320000

- isnull().sum() will give the total number of missing data in the dataframe.
- Here we have kept the missing values as there can be missing values in the real world data also.

```
bank_transaction_id                0
bank_transaction_description     131
bank_transaction_amount            0
bank_transaction_type              0
dtype: int64
```

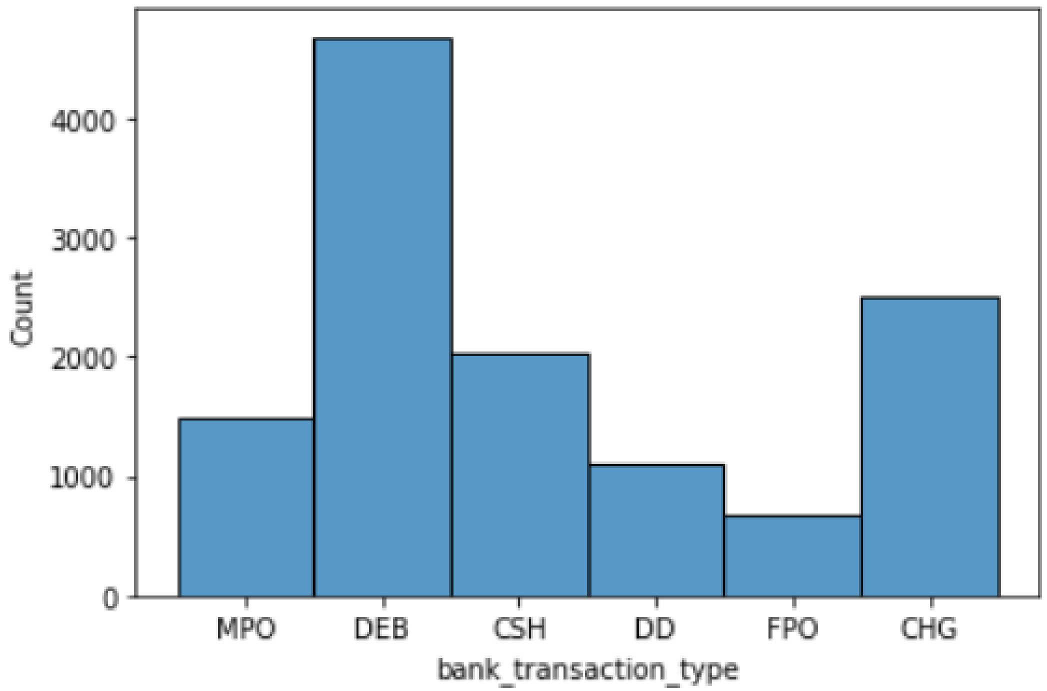
Is there any difference between the number of transactions and their types, amounts and descriptions in each category?

bank_transaction_type	bank_transaction_id	bank_transaction_description	bank_transaction_amount
CHG	2510	1543	211
CSH	2041	1686	1619
DD	1108	926	827
DEB	4678	3899	3013
FPO	679	618	577
MPO	1484	1370	1260

- We can see from the above table that there are more transactions in the DEB transaction type comparing with other transaction types.
- The least number of transaction is from the FPO transaction type.
- We can also conclude that CHG bank transaction type has lowest bank transaction amount

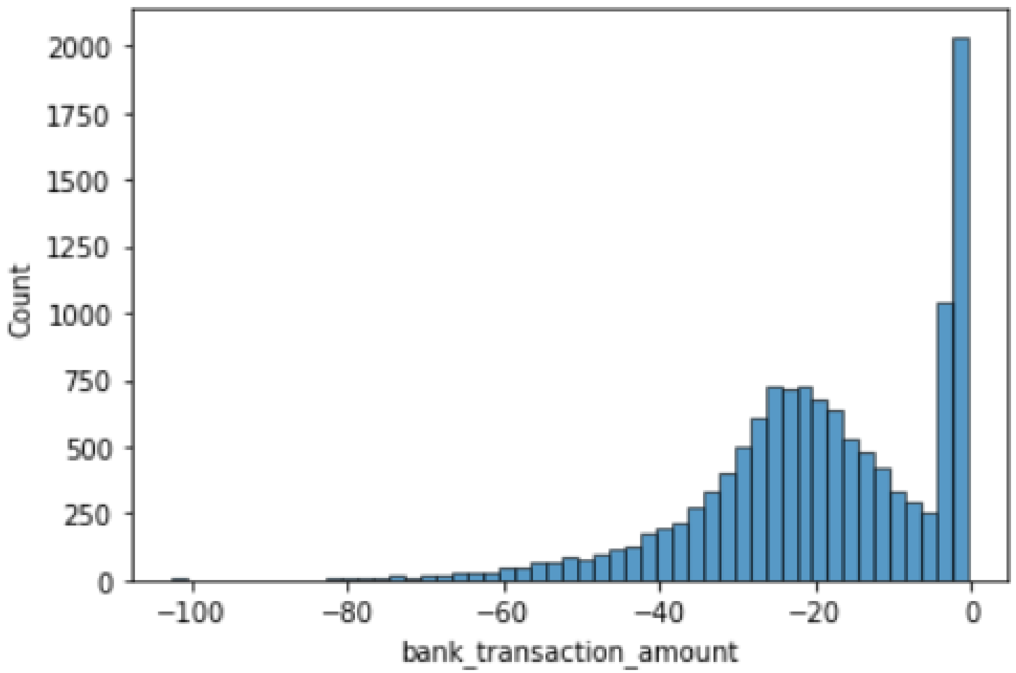
- A histogram with bank transaction type on the x-axis and bank transaction amount on the y axis.

```
<AxesSubplot:xlabel='bank_transaction_type', ylabel='Count'>
```



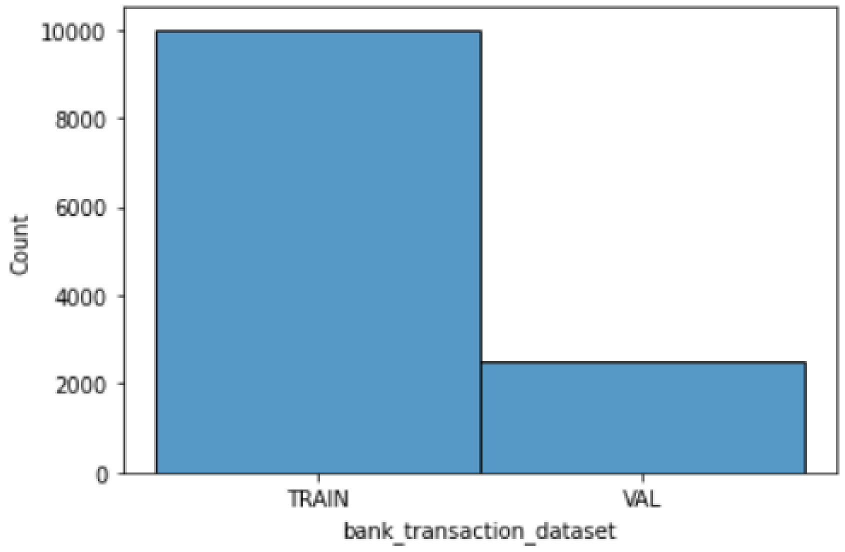
- A histogram of bank transaction amount

```
<AxesSubplot:xlabel='bank_transaction_amount', ylabel='Count'>
```



How does the validation data compare with the training data?

```
<AxesSubplot:xlabel='bank_transaction_dataset', ylabel='Count'>
```

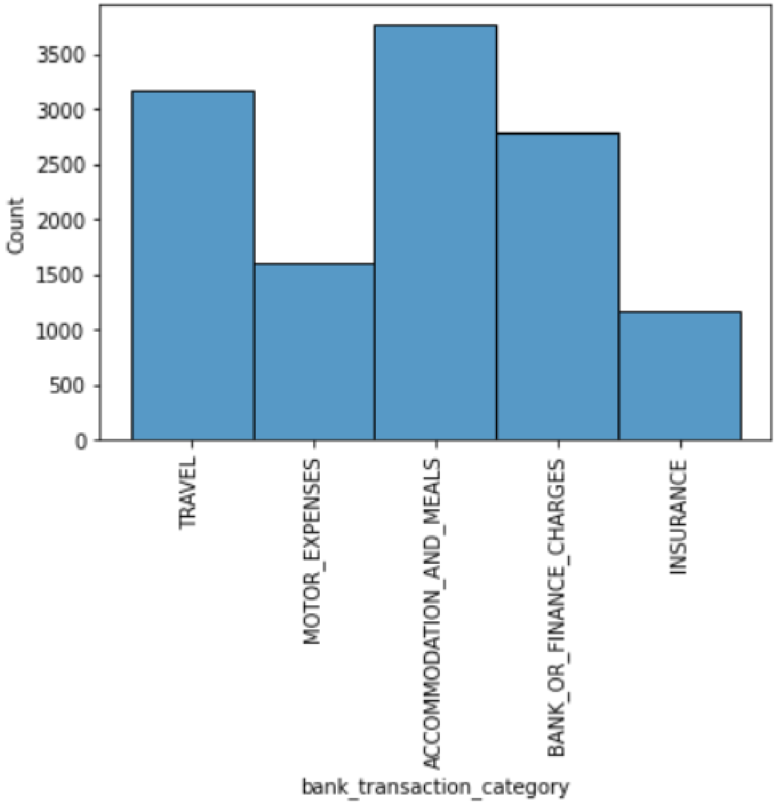


- Here we can see that training data has 10,000 rows or whereas validation data has 2500 rows

- Checking for any null values in the labels dataset

```
bank_transaction_id          0
bank_transaction_category    0
bank_transaction_dataset     0
dtype: int64
```

- We can see that there are no null values in the labels dataset.



- We can observe from above visualization the number of transactions in each category.
- Accommodation and Meals with highest number of transactions and Insurance with least number of transactions.

bank_transaction_category	bank_transaction_id	bank_transaction_dataset
ACCOMMODATION_AND_MEALS	3765	2
BANK_OR_FINANCE_CHARGES	2790	2
INSURANCE	1170	2
MOTOR_EXPENSES	1609	2
TRAVEL	3166	2

- We can see that most of the transaction are of Accommodation and Meals category and the least number of transactions are from Insurance category.

- Now we will take the bank_transaction_description column into another data frame on which the preprocessing will be done as this transaction description will be the input for our model.

- We will start by lowering the text and removing all the special characters using regex, and we will change the datatype to string.

- Then as the input to our model cannot be text so we will convert the text into bag of words using count vectorizer from sklearn library.

- After converting our text into vectors we will divide our data into train and validation as the first 10,000 data points are for training and remaining for validation.

- Similarly we will divide our transaction category column into training and validation.

- Now it is a multi classification problem with more than two class to classify we will use label encoder to label different categories as numbers which will be the target variable to predict.

- After that I have used Support Vector Machine and have fit the model with the training data and then have predicted on the validation data.

- Next I have used many different metrics to validate the model such as accuracy, precision, recall and f1-score.
- Precision calculates:

$$P = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Precision is a good measure when the cost of False positive is high.

- Recall calculates:

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Recall is a good measure when there is a high cost associated with False negative

- F1- score calculates:

$$\text{f1-score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1-score might be a better approach to use if we need to seek a balance between Precision and Recall.

- We got the accuracy of around 94.5% on our training data and around 91.7% on our validation data which proves that our model is not overfitting
- As this is a multi classification problem there can be a particular category which can be misclassified which can be found out from confusion matrix.
- Confusion Matrix on training Data:

```
array([[2887,    0,    0,    4,  126],
       [   0, 2248,    0,    0,    2],
       [   1,    0,  880,   21,   18],
       [  82,   11,   41, 1043,   58],
       [  64,   45,    0,   77, 2393]], dtype=int64)
```

- The diagonal values in the above matrix are the ones which are correctly classified and the zeros in the above matrix are the values which are not miss classified.

- Confusion matrix on validation data

```
array([[707,   1,   0,   0,  40],
       [  1, 536,   0,   0,   3],
       [  1,   2, 234,   3,  10],
       [ 19,   1,  21, 308,  25],
       [ 29,  15,   0,  34, 508]], dtype=int64)
```

- I got Precision as 91.5% , Recall as 91.2% and F1-score as 91.3% for our validation data.

- The values in the diagonal [707, 536, 234, 308, 508] are correctly classified values and the remaining values are the ones which are misclassified.

- Next I tried by using different model which is the logistic regression.
- I got the training accuracy of around 93.8% which is slightly less than the SVM model.

- Next I tried to fit the model on the validation data and got accuracy of 91.67 %, Precision of 91.3%, Recall of 91.1% and F1 score of 91.1%, which are quite good but slightly poor than the SVM model.

- And the confusion matrix for the logistic regression model on validation data is

```
array([[706,   4,   0,   0,  38],
       [  0, 539,   0,   0,   1],
       [  0,   6, 236,   5,   3],
       [ 18,   3,  22, 302,  29],
       [ 26,  22,   0,  31, 507]], dtype=int64)
```

- We can see that there are more zeros in the above matrix than the SVM model before which means that there more categories correctly classified than previous model.

- The model can be further improved by adding more features in it like the transaction_amount or using the bank_transaction_type data into consideration and using the features to improve the model.

- The validation data can be altered by adding some missing values in the data as in the real world scenario the data is not perfect and contains more missing values.