Fraud Detection Case Study

PRN: 230340325044
Name: Ritika R Patil

Data frame:

There are 6362620 records in the data.

There are 11 columns in the data

df.info()

From above we can infer that the data frame has 11 columns

- 3 columns have the datatype int64
- 3 columns have the datatype object
- 5 columns have datatype float64

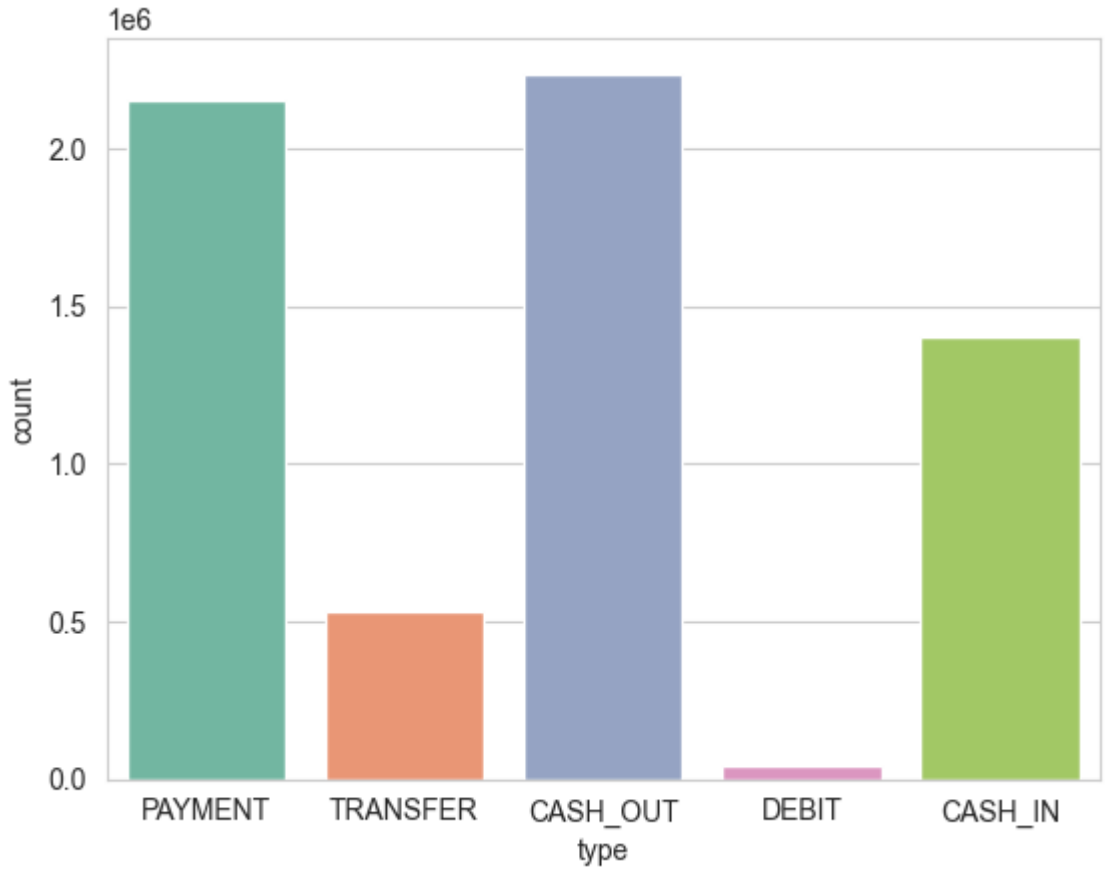
df.describe()

- The describe command gives the statistical analysis of the data.
- We can get the information like Mean, Standard Deviation and range.

Also it can be observed that there are no null values in the data.

```
df.isnull().mean()*100
✓ 5.9s
step              0.0
type              0.0
amount            0.0
nameOrig          0.0
oldbalanceOrg     0.0
newbalanceOrig    0.0
nameDest          0.0
oldbalanceDest    0.0
newbalanceDest    0.0
isFraud           0.0
isFlaggedFraud    0.0
dtype: float64
```

Analysis of Type column



- Above plot we can infer that high percentage of data is there for PYMENT and CASH_OUT type.

- There is very less data for transaction type debit card.

Analysis of isFraud columns

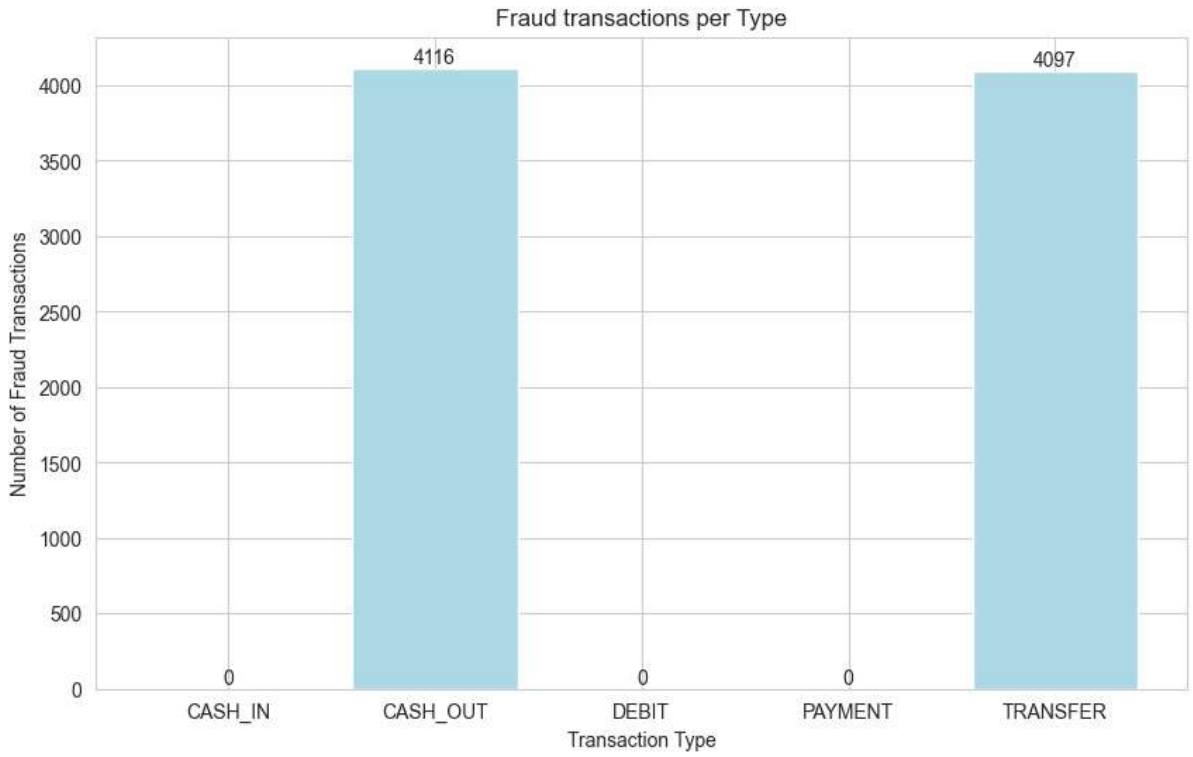
```
df['isFraud'].value_counts()
✓ 0.1s

0    6354407
1       8213
Name: isFraud, dtype: int64
```

Considering **0 → not Fraud** and **1 → Fraud** , it can be observed that the number of fraud transactions is less than number of not fraud transaction.

- From above we can infer that data is highly imbalanced.
- 99 percent of data is of No fraud category.
- Ratio of records is 99:1 of No-fraud with fraud

Analysis of type with Fraud transaction



- From above we can infer that transactions done by payment mode TRANSFER or CASH_OUT are prone to be fraud.
- The counts of frauds in Cash_out and Transfer payment methods are 4116 and 4097

respectively

Feature Selection:

- The name of the person who started the transaction i.e. nameOrig and the person to whom's account the transfer of money is done i.e. nameDest are just the names.
- So, names will not contribute to the prediction whether the transaction is fraud or not.
- Hence, we can drop the columns nameOrig and nameDest

- Now we have only one categorical column left which is TYPE which has the payment_modes of transaction.

- For model building , the categorical column TYPE shall be converted to numerical.

- Hence we used the LabelEncoder to convert TYPE to numerical attribute.

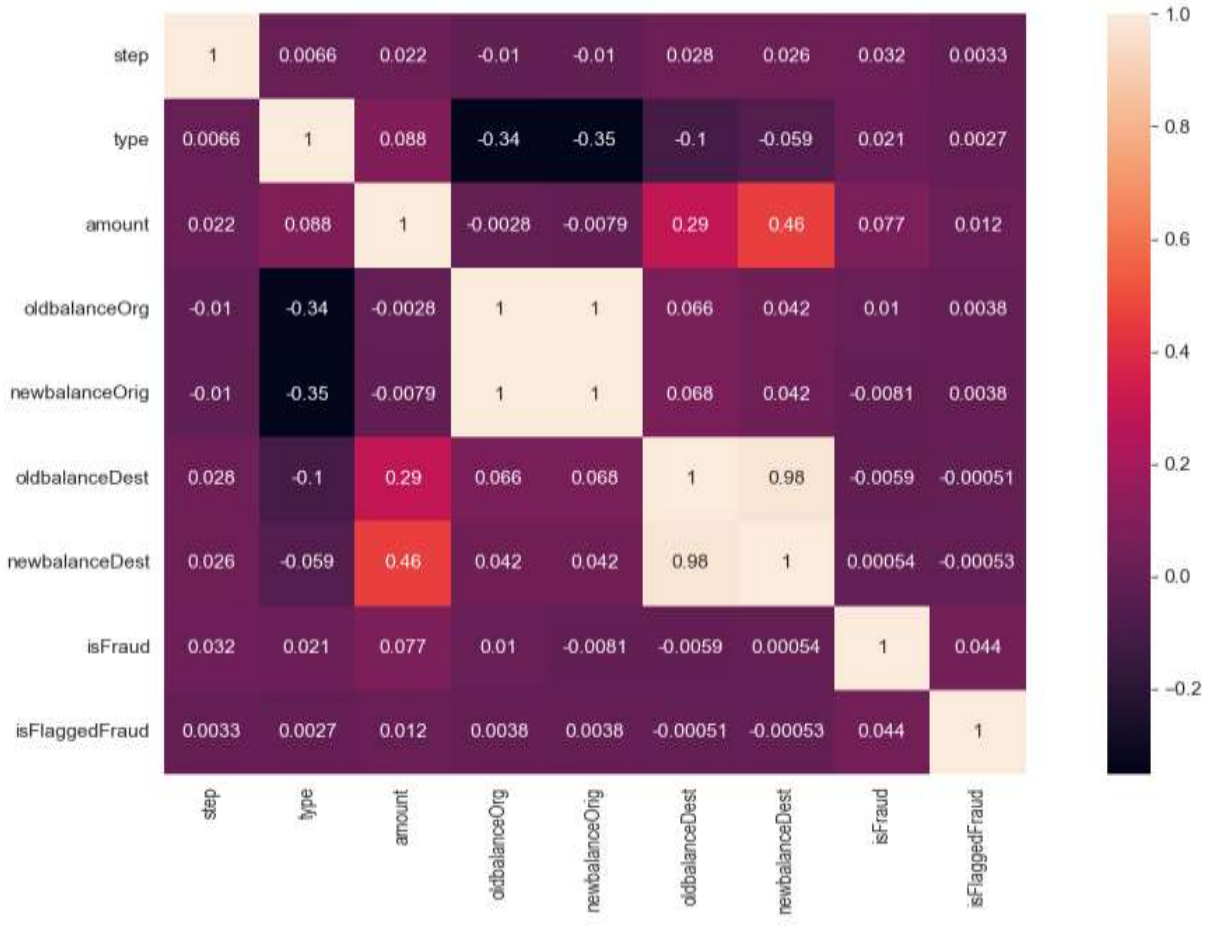
MULTICOLLINEARITY

All predictors should be independent of each other. So for checking that we have two methods:

- Heatmap
- VIF

If attributes are dependent then that affects model accuracy, So we have to check correlation before model training.

Heatmap :



VIF:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

print('Variance Inflation Factor')
for i in range(df.shape[1]):
    print(df.columns[i]," : ",end=' ')
    print(variance_inflation_factor(df.values,i))
```

✓ 1m 28.1s

```
Variance Inflation Factor
step  :  2.250165911116425
type  :  2.085347534546198
amount  :  4.083183557325778
oldbalanceOrg  :  564.4392937717903
newbalanceOrig  :  568.748454118756
oldbalanceDest  :  73.12767121326486
newbalanceDest  :  84.82046715938228
isFraud  :  1.1941601835290432
isFlaggedFraud  :  1.0025645203718174
```

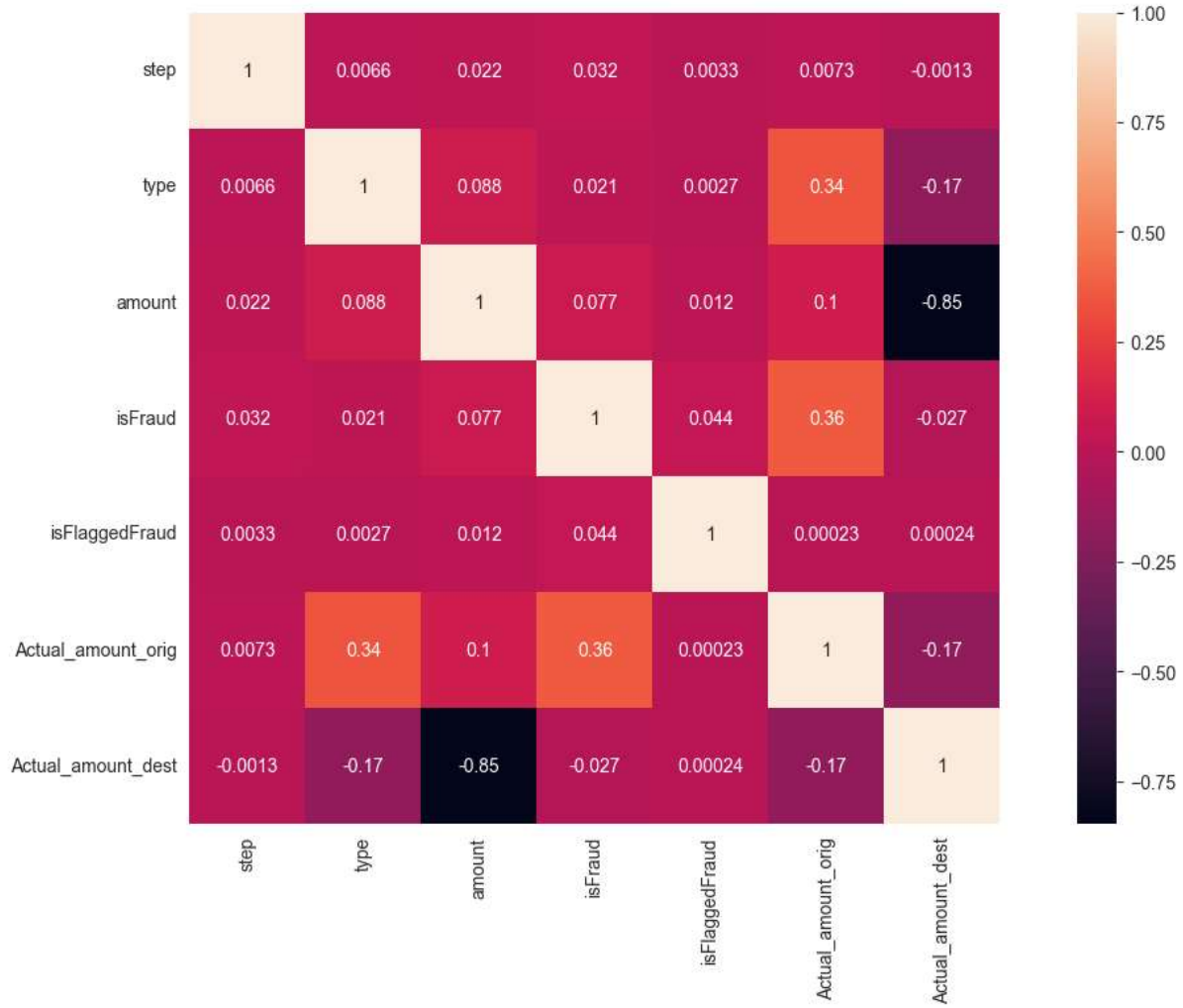
We can see that oldbalanceOrg and newbalanceOrig have too high VIF thus they are highly correlated. Similarly oldbalanceDest and newbalanceDest.

- Normally, we drop the columns that are correlated. But this factors are important for prediction. We have to handle that.
- Hence we use the above factors and calculate introduce two new factors, Actual_amount_orig and Actual_amount_dest

Which can be given as follows :

- Actual_amount_orig = oldbalanceOrg – newbalanceOrig
- Actual_amount_dest = oldbalanceDest – newbalanceDest

After introducing the 2 new columns and dropping oldbalanceOrg and newbalanceOrig , oldbalanceDest and newbalanceDest.



VIF:

```
    print('Variance Inflation Factor')
    for i in range(df.shape[1]):
        print(df.columns[i]," : ",end=' ')
        print(variance_inflation_factor(df.values,i))
```

[207] ✓ 27.2s

```
Variance Inflation Factor
step  :  2.089916143614993
type  :  2.03932831737454
amount  :  3.8833826140180028
isFraud  :  1.1879094829388495
isFlaggedFraud  :  1.0025351175865196
Actual_amount_orig  :  1.2933061926835898
Actual_amount_dest  :  3.7740300466347065
```

As we can see now there is no high correlation between two columns

Scalling :

As we can see above, we need to normalize the range of features in a dataset before applying any machine learning algorithm on them.

As we have outliers in our dataset we will go for Robust Scaler as it is less prone to outliers.

Robust Scaler is less prone to outliers. It works with the interquartile range

Model Observations:

Algorithm	Accuracy Score	Precision	Recall
Decision Tree	0.999464843	0.79	0.79
Random Forest	0.999646372	0.95	0.76
KNN	0.999504920	0.89	0.71
Logistic Regression	0.999222018	0.88	0.46
AdaBoost	0.999281742	0.89	0.51

- All models have equal accuracy.
- Precision of Random Forest is highest among all model
- Decision tree have little high recall than Random Forest.

In fraud detection precision holds significant role. As our primary objective if to predict fraud transaction while minimizing false positive for genuine transactions. If any one of goal is not achieve then innocent customer can get predicted as guilty. Also AUC of random forest is 88% and difference between recall of random forest and decision tree is very less we can select **RANDOM FOREST** as preferred over others.

Prevention Measures to take by customers

1. Browse only on secured websites.

2. Download any applications from official app stores.

3. Use secured internet connections and avoid doing any banking transaction on public/free networks.

4. Keep your mobile and laptop security updated.

5. Don't respond to unsolicited calls/SMS/E-mails.

6. If customer feel like that they have been tricked or security compromised, contact bank at earliest.