NAME : Lokarapu Vijaya Satya Devi

CLASS : CSE-2A

ID : S180156

SUBJECT : Data Science With Python

(A Real Time Data Science Project on Online Payment Frauds)

ONLINE PAYMENT FRAUD DETECTION

NOTE :- The dataset which has been used in this project from the Kaggle.

TABLE OF CONTENTS

Contents

- Abstract
- Project Aim
- About the Dataset
- Prerequisites
- Data Science Techniques &python Tools used
- Python code and Outcomes
- Predictions
- Conclusion
- Reference

ABSTRACT

Online payments refer to the electronic exchange of currency through the internet. These payments usually consist of the transfer of monetary funds from a customer's bank or debit or credit card account, into the seller's bank account, in exchange for products or services.

Payment fraud occurs when someone steals another person's payment information and uses it to make unauthorized transactions or purchases. The actual cardholder or owner of the payment information then notices their account being used for transactions or purchases they did not authorize, and raises a dispute.

Online payment systems has helped a lot in the ease of payments. But, at the same time, it increased in payment frauds. Online payment frauds can happen with anyone using any

payment system, especially while making payments using a credit card. That is why detecting online payment fraud is very important for credit card companies to ensure that the customers are not getting charged for the products and services they never paid.

PROJECT AIM

The project aims to build a Online Payment Fraud Detection Model. I use the input features and their labels as fraud or No fraud to detect if new transactions made by the customer are fraud are not.

INPUT FEATURES

Type , amount , Org_oldbalance , Dest_newbalance.

TARGET OUTPUT

Fraud or No Fraud

ABOUT THE DATASET

For this task, I collected a dataset from Kaggle, which contains information about fraudulent transactions which can be used to detect fraud in online payments. Below are all the columns from the dataset I'm using here:

1. type: type of online transaction
2. amount: the amount of the transaction
3. nameOrig: customer starting the transaction
4. oldbalanceOrg: balance before the transaction
5. newbalanceOrig: balance after the transaction
6. nameDest: recipient of the transaction
7. oldbalanceDest: initial balance of recipient before the transaction
8. newbalanceDest: the new balance of recipient after the transaction
9. isFraud: fraud transaction

PRE REQUISITES

- ✓ Basic idea on what is data.
- ✓ Basic idea on Online Payments and its Frauds.
- ✓ Must know how to import Datasets and how to work on them using Pandas in Python.
- ✓ Must know how to pre-process the data.
- ✓ Must know Classification and its algorithms for making predictions.
- ✓ Must know how to visualize data using different plotting techniques.

PYTHON TOOLS (LIBRARIES) & DATA SCIENCE TECHNOLOGIES USED

Python Tools (Libraries) used:

- ➢ *Pandas*
- ➢ *Numpy*

- *Matplotlib*
- *Seaborn*
- *Sklearn*

Data Science Techniques Used :

- *Logistic Regression*
- *Decision Tree*
- *K Neighbor's Classifier*

PYTHON CODE & OUTCOMES

1.Importing Libraries:-

```
In [1]: #importing Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn import tree
        from sklearn.tree import DecisionTreeClassifier
```

2.Loading DataSet:-

```
In [2]: #Loading Dataset
        df=pd.read_csv("Desktop/OnlinePaymentFraudDetection2.csv")
        df.head(10)
```

Out[2]:

	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0
1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0
2	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1
3	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1
4	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0
5	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0
6	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0
7	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0
8	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0
9	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0

3.Exploring Dataset:-

Dimension of the dataset:-

```
In [3]: # ndim gives dimension of the dataset
        df.ndim
```

Out[3]: 2

Shape of the dataset:-

```
In [4]: # shape gives no.of cols & rows of the given dataset
        df.shape
```

Out[4]: (1048575, 9)

Size of the Dataset:-

```
In [5]: #size of the dataset
        df.size

Out[5]: 9437175
```

Information about the Dataset:-

```
In [6]: # info gives information about the Dataset
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 9 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   type            1048575 non-null  object
 1   amount          1048575 non-null  float64
 2   nameOrig        1048575 non-null  object
 3   oldbalanceOrg   1048575 non-null  float64
 4   newbalanceOrig  1048575 non-null  float64
 5   nameDest        1048575 non-null  object
 6   oldbalanceDest  1048575 non-null  float64
 7   newbalanceDest  1048575 non-null  float64
 8   isFraud         1048575 non-null  int64
dtypes: float64(5), int64(1), object(3)
memory usage: 72.0+ MB
```

Description about the Dataset:-

```
In [7]: #Describe() gives mathmetical information of the columns of the only type float64
        df.describe()
```

Out[7]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06	1.089097e-03
std	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06	3.298351e-02
min	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05	0.000000e+00
75%	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06	0.000000e+00
max	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07	1.000000e+00

4.Data Cleaning & Processing:-

Checking Null values:-

```
In [9]: #checking the null values
        df.isna().sum()

Out[9]: type            0
        amount          0
        nameOrig        0
        oldbalanceOrg   0
        newbalanceOrig  0
        nameDest        0
        oldbalanceDest  0
        newbalanceDest  0
        isFraud         0
        dtype: int64
```

So this dataset does not have any null values.

5.Data Formating:-

Now transform the categorical features into numerical. Here I will also transform the values of the **isFraud** column into No Fraud and Fraud labels to have a better understanding of the output.

```
In [11]: # converting numerical value to categorical value
         df['isFraud'] = df['isFraud'].map({0:'No_Fraud', 1:'Fraud'})
```

```
In [12]: # converting categorical value to numerical value
         df['type'] = df['type'].map({'CASH_OUT':0, 'PAYMENT':1, 'CASH_IN':2, 'TRANSFER':3, 'DEBIT':4,'others':5})
```

```
In [12]: # Total No.of Frauds and No_frauds in DataSet
         df['isFraud'].value_counts()

Out[12]: No_Fraud    1047433
         Fraud          1142
         Name: isFraud, dtype: int64
```

```
In [13]: # Total No.of Transactions of each type
         type=df.type.value_counts()
         type

Out[13]: CASH_OUT    373641
         PAYMENT     353873
         CASH_IN     227130
         TRANSFER     86753
         DEBIT         7178
         Name: type, dtype: int64
```

Correlation between the features of the data with the **isFraud** column.

```
In [19]: # corrolation is used know the relationship b/w among the columns
         corr=df.corr()
         corr
```

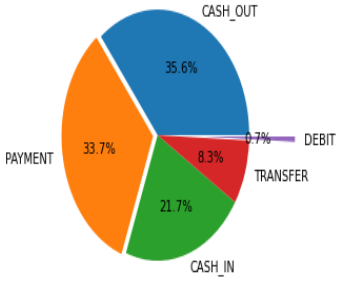
Out[19]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
type	1.000000	0.265263	0.262623	0.272537	0.057842	0.059061	0.014645
amount	0.265263	1.000000	0.004864	-0.001133	0.215558	0.311936	0.128862
oldbalanceOrg	0.262623	0.004864	1.000000	0.999047	0.093305	0.064049	0.003829
newbalanceOrig	0.272537	-0.001133	0.999047	1.000000	0.095182	0.063725	-0.009438
oldbalanceDest	0.057842	0.215558	0.093305	0.095182	1.000000	0.978403	-0.007552
newbalanceDest	0.059061	0.311936	0.064049	0.063725	0.978403	1.000000	-0.000495
isFraud	0.014645	0.128862	0.003829	-0.009438	-0.007552	-0.000495	1.000000

6.Data VisualiZation:-

Pie plot is used for knowing which type of the transaction is more.

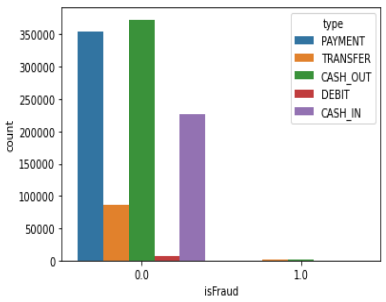
```
In [10]: # pie plot for knowing which type of the trasaction type is more
         plt.pie(type,labels=type.index,autopct="%.1f%%",explode=(0,0.05,0,0,0.5))
         plt.show()
```



Seaborn countplot() method is used to show the counts of observations in each categorical bin using bars

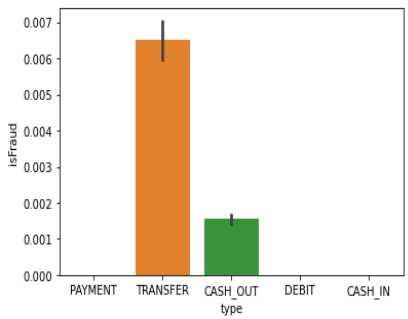
```
In [15]: # Count of frauds and NoFrauds in each type of transaction
         import seaborn as sns
         sns.countplot(x="isFraud",hue="type",data=df)
```

```
Out[15]: <AxesSubplot:xlabel='isFraud', ylabel='count'>
```



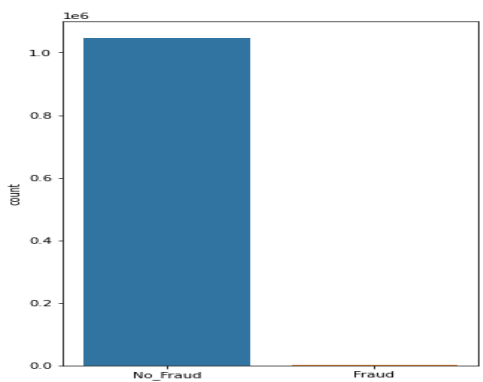
In [11]:
```
# Barplot for identify the frauds in each type transactions
sns.barplot(x=df["type"],y=df["isFraud"])
```

Out[11]: <AxesSubplot:xlabel='type', ylabel='isFraud'>

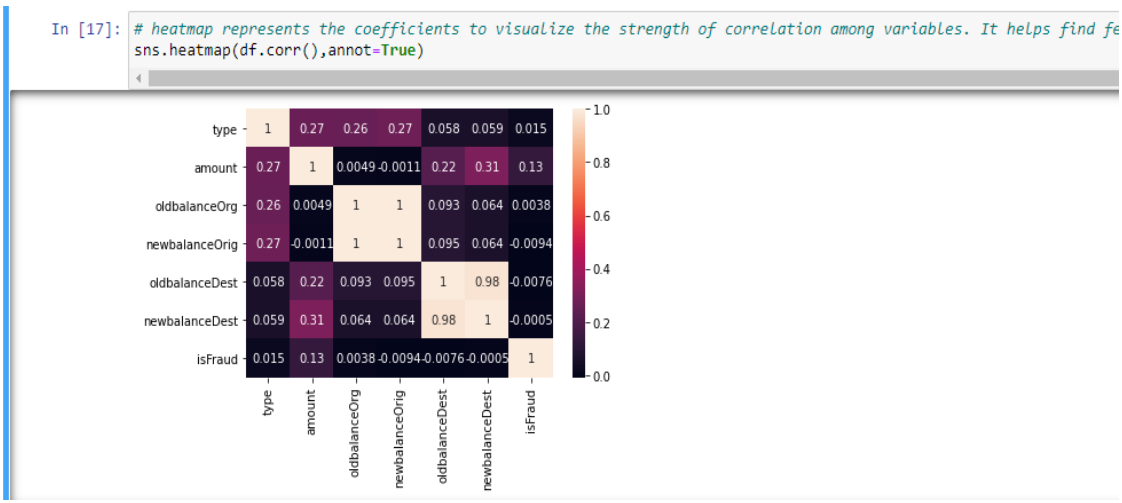


In [39]:
```
# countplot for total no.of frauds and No_frauds
plt.figure(figsize=(5,8))
sns.countplot(df['isFraud'])
```

Out[39]: <AxesSubplot:xlabel='isFraud', ylabel='count'>



Heatmap represents the coefficients to visualize the strength of correlation among variables. It helps find features that are best for model



7.Spliting the dataset into trainng and testing sets:-

Now let's train a classification model to classify fraud and non-fraud transactions. Before training the model, I will split the data into training and test sets.

```
In [26]: # features of the dataset
         X = np.array(df[['type', 'amount', 'oldbalanceOrg','newbalanceDest']])

In [27]: # target of the dataset
         y = np.array(df[['isFraud']])

In [28]: # splitting the dataset into training and testing
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y)

In [29]: print("The shape of the training set=(",X_train.shape,y_train.shape,")")
         print("The Shape of the testing set=(",X_test.shape,y_test.shape,")")

         The shape of the training set=( (786431, 4) (786431, 1) )
         The Shape of the testing set=( (262144, 4) (262144, 1) )
```

8.Model Development :-

1.*Logistic Regression*:-

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

```
In [78]: # Training a model using logisticRegression algorithm
         from sklearn.linear_model import LogisticRegression
         model= LogisticRegression()
         model.fit(X_train, y_train)

         C:\Users\SATYADEVI\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector y w
         sed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           return f(**kwargs)

Out[78]: LogisticRegression()

In [79]: model.score(X_train,y_train)

Out[79]: 0.9992454044775052

In [80]: # checking accuracy of LogisticRegression
         model.score(X_test, y_test)

Out[80]: 0.9993801111031638
```

2.*K-Neighbors Classification* :-

This algorithm is used to solve the classification model problems. K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.

```
In [ ]: # Training a model using KneighborClasification algorithm
        from sklearn.neighbors import KNeighborsClassifier
        model_dt = KNeighborsClassifier()
        model_dt.fit(X_train, y_train)

In [43]: # checking accuracy for KneighborClassifier
         model_dt.score(X_train,y_train)

Out[43]: 0.9990920985874122

In [44]: model_dt.score(X_test, y_test)

Out[44]: 0.9989204406738281
```

3.*DecisionTree Classification*:-

Decision tree algorithm is also a supervised learning. They can be used to solve both regression and classification problems.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.We can represent any boolean function on discrete attributes using the decision tree.

```
In [91]: # Training a model using DecisionTree algorithm
         from sklearn.tree import DecisionTreeClassifier
         model_dt = DecisionTreeClassifier()

In [92]: model_dt.fit(X_train, y_train)
Out[92]: DecisionTreeClassifier()

In [93]: # checking accuracy for DecisionTree
         model_dt.score(X_train,y_train)
Out[93]: 1.0

In [94]: model_dt.score(X_test, y_test)
Out[94]: 0.999494552130272
```

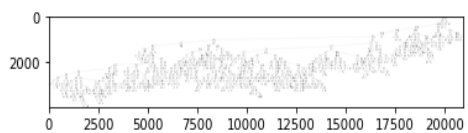
9.Model Evaluation:-

The model is trained by all classification Algorithms.But in all classification algorthims DecisionTree gives best for this model.Because the accuracy of the

DecisionTree is more than Other Classification Algorithms.

These four lines of code is used to plot the decision tree for the Dataset

```
In [88]: data=tree.export_graphviz(model_dt,out_file=None)

In [89]: graph=pydotplus.graph_from_dot_data(data)

In [91]: graph

Out[91]: <pydotplus.graphviz.Dot at 0x2e299103dc0>

In [108]: import graphviz
          graph.write_png("output.png")

Out[108]: True

In [112]: img=pltimg.imread("output.png")

In [113]: imgplot=plt.imshow(img)
          plt.show()
```



10.Prediction:-

Now classify whether a transaction is a fraud or not by feeding about a transaction into the model.

In [34]:
```
# predicted values
y_pred=model_dt.predict(X_test)
y_pred
```

Out[34]:
```
array(['No_Fraud', 'No_Fraud', 'No_Fraud', ..., 'No_Fraud', 'No_Fraud',
       'No_Fraud'], dtype=object)
```

In [72]:
```
# predictiong for random values
data = np.array([[3,1000,2000,3333334444]])
predict=model_dt.predict(data)
print(predict)
```

```
['No_Fraud']
```

In [75]:
```
data1 = np.array([[3,1000,2000,45]])
print(model_dt.predict(data1))
```

```
['Fraud']
```

PREDICTIONS

Taking input from User for predictions

In [52]:
```
Type=int(input("Enter a type Transaction{0:Cash_Out,1:Payment,2:Cash_in,3:Transfer,4:Debit,5:others}:"))
amount=float(input("Enter the amount:"))
Org_oldbalance=float(input("Enter the old balance of sender :"))
Dest_newbalance=float(input("Enter the new balance of receiver:"))
```

```
Enter a type Transaction{0:Cash_Out,1:Payment,2:Cash_in,3:Transfer,4:Debit,5:others}:3
Enter the amount:100
Enter the old balance of sender :2000
Enter the new balance of receiver:1000
```

In [53]:
```
data1 = np.array([[Type,amount,Org_oldbalance,Dest_newbalance]])
print(model_dt.predict(data1))
```

```
['No_Fraud']
```

In [54]:
```
Type=int(input("Enter a type Transaction{0:Cash_Out,1:Payment,2:Cash_in,3:Transfer,4:Debit,5:others}:"))
amount=int(input("Enter the amount:"))
Org_oldbalance=int(input("Enter the old balance :"))
Dest_newbalance=int(input("Enter the new balance :"))
```

```
Enter a type Transaction{0:Cash_Out,1:Payment,2:Cash_in,3:Transfer,4:Debit,5:others}:3
Enter the amount:1000
Enter the old balance :2000
Enter the new balance :0
```

In [55]:
```
data1 = np.array([[Type,amount,Org_oldbalance,Dest_newbalance]])
print(model_dt.predict(data1))
```

```
['Fraud']
```

CONCLUSION

Detecting online payment frauds is one of the applications of data science in finance.So this is how we can detect online payment frauds using Machine learning with Python.

REFERRENCES

https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

https://github.com/devisklm/DataScience/blob/main/project.ipynb