Library importing ¶

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

```

In [4]:
```
df=pd.read_csv('fraud_detection.csv')
```

In [3]:
```
df = df.sample(n=500000, random_state=42) # Command Not RUN
```

In [5]:
```
df.head()
```

Out[5]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDe
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [7]:
```
df.shape
```

Out[7]:
```
(6362620, 11)
```

In [8]:
```
df.type.value_counts()
```

Out[8]:
```
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

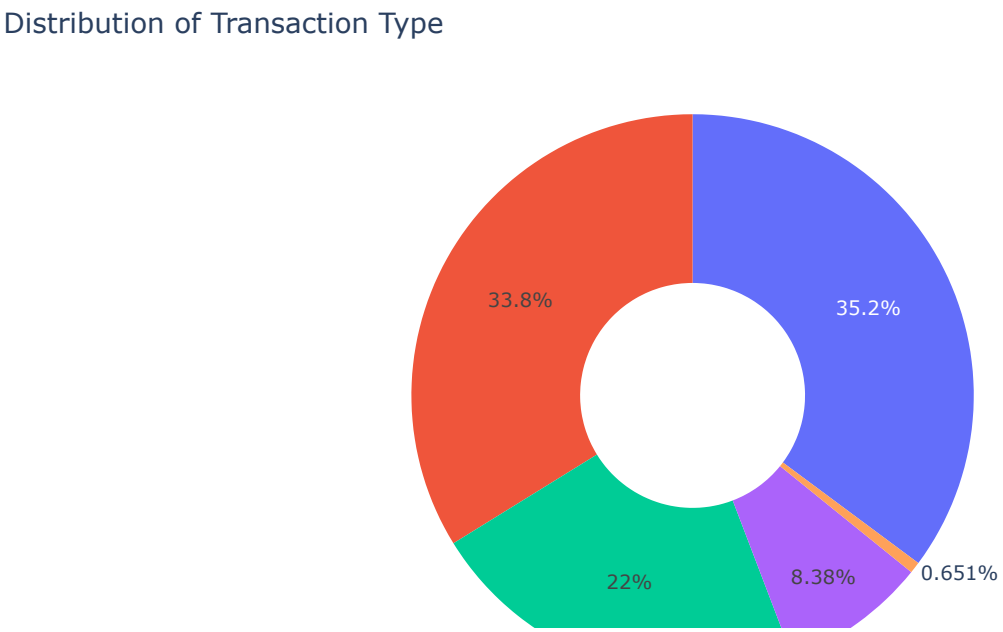
In [9]:
```
type=df['type'].value_counts()
```

In [10]:
```
transactions=type.index
```

In [11]:
```
quantity=type.values
```

Visualization -- Pie Plot

In [12]:
```

import plotly.express as px
px.pie(df,values=quantity,names=transactions,hole=0.4,title="Distribution of Transaction Type"
```



Dropping Null Rows

In [13]:
```
df=df.dropna()
```

In [14]:
```
df
```

Out[14]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbala
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	3
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	63
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	

Replacing 0's and 1's to Labels

In [15]:
```
df['isFraud']=df['isFraud'].map({0:'No Fraud',1:'Fraud'})
```

In [16]:
```
df
```

Out[16]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
...	...	...	...	...	...	...	...	...	...
62615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339
62616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	
62617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379
62618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	
62619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360

62620 rows × 11 columns

In [17]:
```
df['type'].unique()
```

Out[17]:
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [18]:
```
1 df['type'].value_counts()
```

Out[18]:
```
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

In [19]:
```
1 df
```

Out[19]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbala
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
...	...	...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	3
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	63

PAYMENT = 1, TRANSFER = 4, CASH_OUT = 2, DEBIT= 5, CASH_IN= 3

In [20]:
```
1 df['type']=df['type'].map({'PAYMENT':1, 'TRANSFER':4, 'CASH_OUT':2, 'DEBIT':5, 'CASH_IN':3})
```

In [21]:
```
1 df['type'].value_counts()
```

Out[21]:
```
type
2    2237500
1    2151495
3    1399284
4     532909
5      41432
Name: count, dtype: int64
```

In [22]:
```
1 df
```

Out[22]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalance
0	1	1	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	1	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	4	181.00	C1305486145	181.00	0.00	C553264065	0.00	
3	1	2	181.00	C840083671	181.00	0.00	C38997010	21182.00	
4	1	1	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
...	...	...	...	...	...	...	...	...	
6362615	743	2	339682.13	C786484425	339682.13	0.00	C776919290	0.00	33968
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	
6362617	743	2	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	637989
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	
6362619	743	2	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	736010

6362620 rows × 11 columns

In [23]:
```
1 df['type'].unique()
```

Out[23]:
```
array([1, 4, 2, 5, 3], dtype=int64)
```

In [24]:
```
1 df['type'].value_counts()
```

Out[24]:
```
type
2    2237500
1    2151495
3    1399284
4     532909
5      41432
Name: count, dtype: int64
```

Prediction Criteria

In [25]:
```
1 x=df[['type','amount','oldbalanceOrg','newbalanceOrig']]
```

In [26]:
```
1 y=df.iloc[:,-2]
```

In [27]:
```
1 y
```

Out[27]:
```
0          No Fraud
1          No Fraud
2             Fraud
3             Fraud
4          No Fraud
               ...
6362615       Fraud
6362616       Fraud
6362617       Fraud
6362618       Fraud
6362619       Fraud
Name: isFraud, Length: 6362620, dtype: object
```

MODEL TRAINING

In [28]:
```
1 xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=42)
```

In [29]:
```
1 from sklearn.tree import DecisionTreeClassifier
```

In [30]:
```
1 model=DecisionTreeClassifier()
```

In [31]:
```
1 history = model.fit(xtrain,ytrain)
```

Model Accuracy

In [54]:
```
1 model.score(xtest,ytest)
```

Out[54]: 0.9997147401542132

In [96]:
```
1 model.predict([[1,10000,30000,20000]])
```

```
C:\Users\pc\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:

X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
```

Out[96]: array(['No Fraud'], dtype=object)

In [34]:
```
1 x
```

Out[34]:

	type	amount	oldbalanceOrg	newbalanceOrig
0	1	9839.64	170136.00	160296.36
1	1	1864.28	21249.00	19384.72
2	4	181.00	181.00	0.00
3	2	181.00	181.00	0.00
4	1	11668.14	41554.00	29885.86
...	...	...	...	...
6362615	2	339682.13	339682.13	0.00
6362616	4	6311409.28	6311409.28	0.00
6362617	2	6311409.28	6311409.28	0.00
6362618	4	850002.52	850002.52	0.00
6362619	2	850002.52	850002.52	0.00

6362620 rows × 4 columns

Saving a Model using Pickle Library

In [38]:
```
import pickle

with open('Model.pkl', 'wb') as f:
    pickle.dump(model, f)

```

Different Models that can be used for this problem with Accuracy

In [34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [35]:
```
models=[]
results=[]
names=[]

```

In [36]:
```
models.append(('LR',LogisticRegression()))
models.append(('LDA',LinearDiscriminantAnalysis()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('CART',DecisionTreeClassifier()))
models.append(('NB',GaussianNB()))
models.append(('SVM',SVC()))


```

In [37]:
```
models
```

Out[37]:
```
[('LR', LogisticRegression()),
 ('LDA', LinearDiscriminantAnalysis()),
 ('KNN', KNeighborsClassifier()),
 ('CART', DecisionTreeClassifier()),
 ('NB', GaussianNB()),
 ('SVM', SVC())]
```

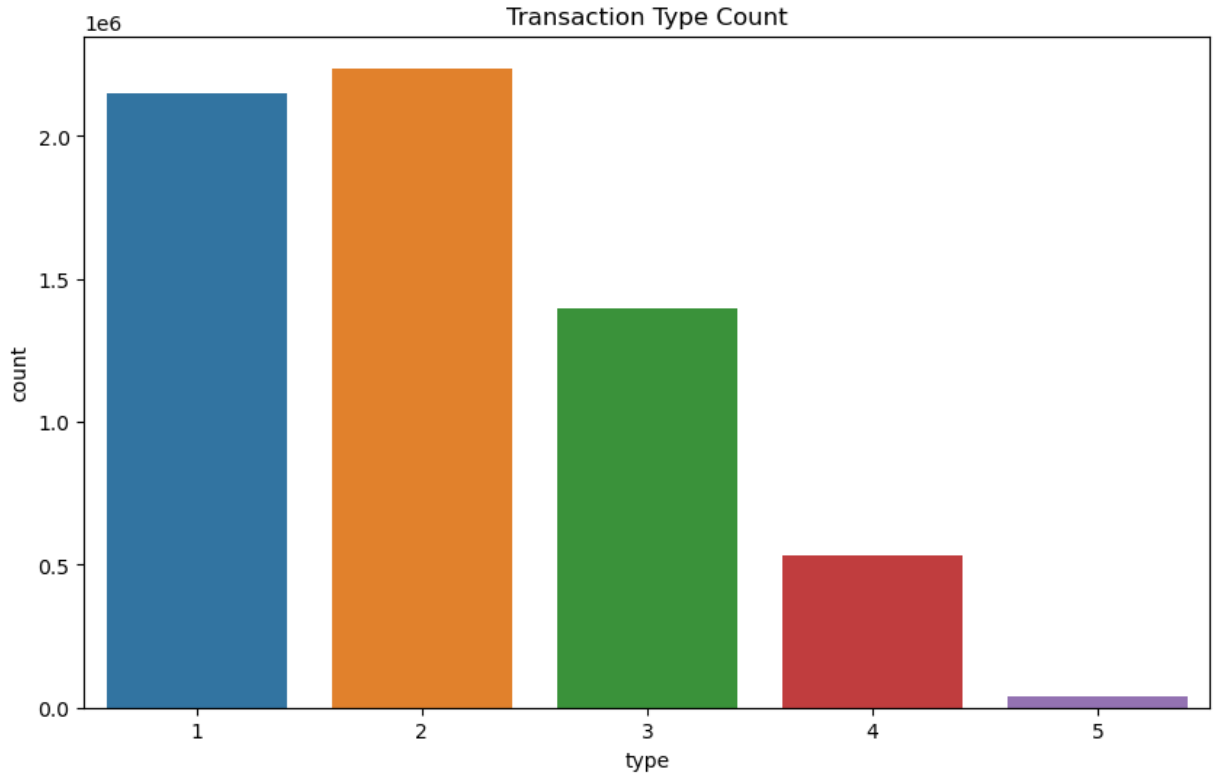
In [ ]:
```
from sklearn import model_selection
for name,model in models:
    kfold=model_selection.KFold(n_splits=10,random_state=7,shuffle=True)
    cv_results=model_selection.cross_val_score(model,x,y,cv=kfold,scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg="%s: %f (%f)" %(name,cv_results.mean(),cv_results.std())
    print(msg)

```

```
LR: 0.999366 (0.000151)
LDA: 0.999038 (0.000148)
KNN: 0.999292 (0.000153)
CART: 0.999366 (0.000093)
NB: 0.993880 (0.000373)
```

Transaction Type Visualization

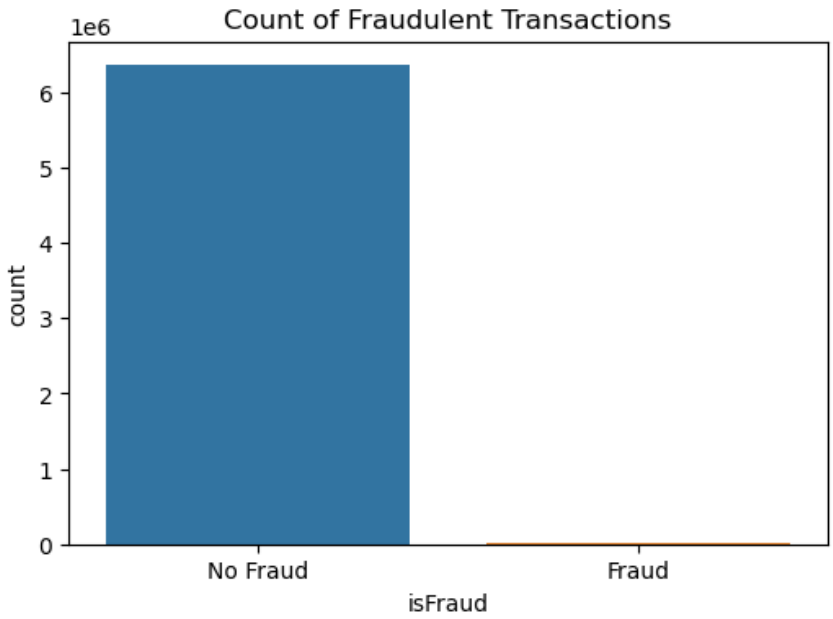
In [36]:

```
# Visualizing the transaction type count
plt.figure(figsize=(10, 6))
sns.countplot(x='type', data=df)
plt.title('Transaction Type Count')
plt.show()
```



Fraudulent Transactions Visualization

In [38]:
```
# Visualizing the count of fraudulent transactions
plt.figure(figsize=(6, 4))
sns.countplot(x='isFraud', data=df)
plt.title('Count of Fraudulent Transactions')
plt.show()
```



Classification Report

In [50]:
```
labels = ["Fraud", "No Fraud"]

from sklearn.metrics import classification_report

print(classification_report ((ytest), prediction, target_names=labels))
```

```
              precision    recall  f1-score   support

       Fraud       0.89      0.89      0.89      1620
    No Fraud       1.00      1.00      1.00   1270904

    accuracy                           1.00   1272524
   macro avg       0.94      0.94      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

In [107]:
```
model.predict([[0,6271,63000,72837]])
```

```
C:\Users\pc\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:

X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
```

Out[107]:
```
array(['No Fraud'], dtype=object)
```

In [81]:
```
1 model.predict([[0,6271,12461,0]])
```

```
C:\Users\pc\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:

X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
```

Out[81]:
```
array(['Fraud'], dtype=object)
```

In [45]:
```
1 prediction = model.predict(xtest)
```

In [46]:
```
1 prediction
```

Out[46]:
```
array(['No Fraud', 'No Fraud', 'No Fraud', ..., 'No Fraud', 'No Fraud',
       'No Fraud'], dtype=object)
```