

In [1]: `import pandas as pd
import numpy as np`

In [2]: `data = pd.read_csv("Credit_crad_data.csv")
print(data.head())`

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

In [3]: `##Check for any NULLs in Data`

In [4]: `print(data.isnull().sum())`

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
	0	0	0	0	0	0	0	0	0	0	0

dtype: int64

Data set doesn't have any NULLs

In [5]: `#Check transaction Types`

In [6]: `print(data.type.value_counts())`

type	count
CASH_OUT	2237500
PAYMENT	2151495
CASH_IN	1399284
TRANSFER	532909
DEBIT	41432

Name: count, dtype: int64

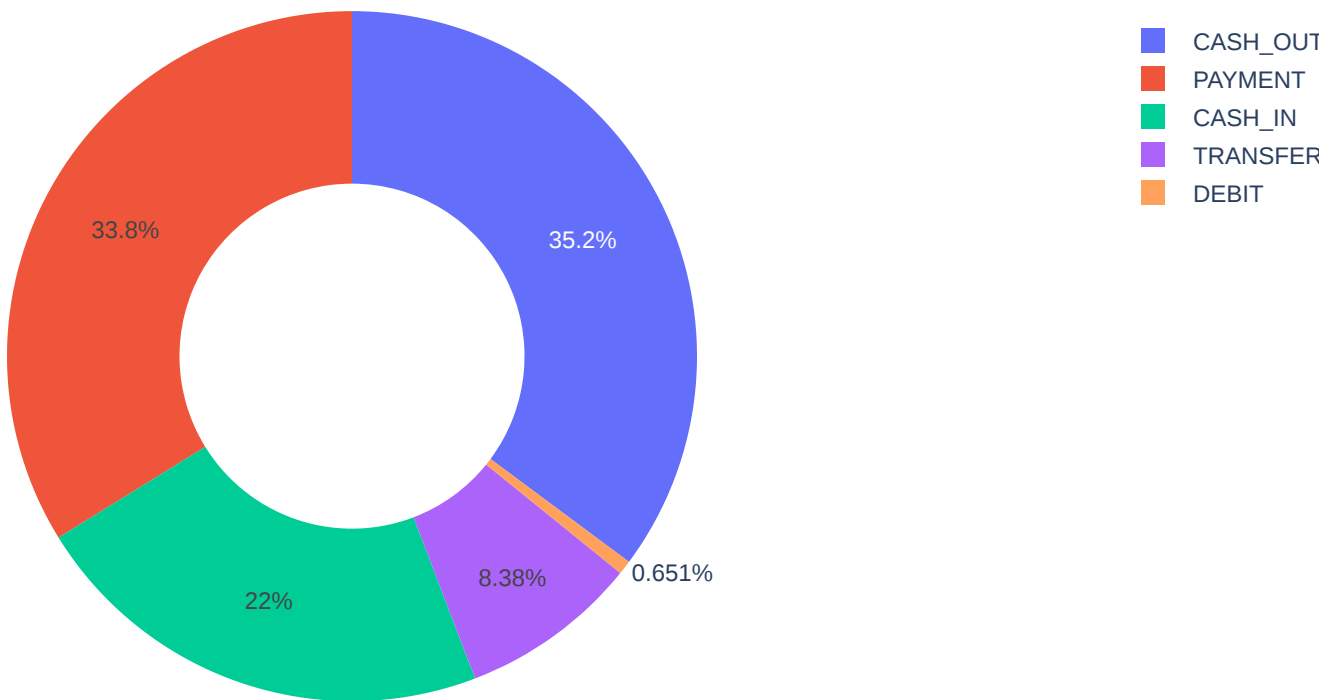
In [7]: `#check Distribution of Transaction types.`

In [8]: `type = data["type"].value_counts()
transactions = type.index
quantity = type.values

import plotly.express as px
figure = px.pie(data,
 values=quantity,
 names=transactions,hole = 0.5,
 title="Distribution of Transaction Type")
figure.show()`



Distribution of Transaction Type



In [9]: `# Drop type & object columns to Check correlation
data_num = data.drop(columns=["type"])
data_num = data_num.select_dtypes(exclude=['object'])`

In [10]: `#Describe after dropping columns
print(data_num.dtypes)`

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
dtype:	int64	float64	float64	float64	float64	float64	int64	int64

In [11]: `#Checking correlation
correlation = data_num.corr()
print(correlation["isFraud"].sort_values(ascending=False))`

	isFraud
isFraud	1.000000
amount	0.076688
isFlaggedFraud	0.044109
step	0.031578
oldbalanceOrg	0.010154
newbalanceDest	0.000535
oldbalanceDest	-0.005885
newbalanceOrig	-0.008148

Name: isFraud, dtype: float64

In summary IsFruad has Week possitive correraltion with Amount,isFlaggedFraud and Step , Very week to negligeble positive correlation with rest of the features.

Convert categorical features into numerical

In [12]: `data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2,
 "CASH_IN": 3, "TRANSFER": 4,
 "DEBIT": 5})
print(data.head())`

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	2	9839.64	C1231006815	170136.0	160296.36	
1	1	2	1864.28	C1666544295	21249.0	19384.72	
2	1	4	181.00	C1305486145	181.0	0.00	
3	1	1	181.00	C840083671	181.0	0.00	
4	1	2	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

In [13]: `data["isFraud"] = data["isFraud"].map({0: "No Fraud", 1: "Fraud"})
print(data.head())`

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	2	9839.64	C1231006815	170136.0	160296.36	
1	1	2	1864.28	C1666544295	21249.0	19384.72	
2	1	4	181.00	C1305486145	181.0	0.00	
3	1	1	181.00	C840083671	181.0	0.00	
4	1	2	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	No Fraud	0
1	M2044282225	0.0	0.0	No Fraud	0
2	C553264065	0.0	0.0	Fraud	0
3	C38997010	21182.0	0.0	Fraud	0
4	M1230701703	0.0	0.0	No Fraud	0

Fraud Detection Model

In [14]: `#Split the data into Training and test`

In [15]: `from sklearn.model_selection import train_test_split
x = np.array(data[["type", "amount", "oldbalanceOrg", "newbalanceOrig"]])
y = np.array(data[["isFraud"]])`

In [16]: `# ML Model Training - Decision tree classifier`

In [17]: `from sklearn.tree import DecisionTreeClassifier
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.20, random_state=42)
model = DecisionTreeClassifier()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))`

0.999704524236871

In [18]: `# Model cross validation , Hyperparameter tuning`

In [19]: `from sklearn.model_selection import cross_val_score, GridSearchCV

Cross-validation model evaluation
cv_scores = cross_val_score(model, x, y, cv=5) # Perform 5-fold cross-validation
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())

Hyperparameter tuning
param_grid = {'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10]}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(x, y)

print("Best hyperparameters:", grid_search.best_params_)
print("Best cross-validation score:", grid_search.best_score_)`

Cross-validation scores: [0.99964559 0.99970217 0.99969981 0.99973989 0.99969588]
Mean cross-validation score: 0.999696658389154
Best hyperparameters: {'max_depth': 30, 'min_samples_split': 2}
Best cross-validation score: 0.9996960371670791

In [22]: `import numpy as np

distinct_values = np.unique(ytest)
print("Distinct values in ytest:", distinct_values)`

Distinct values in ytest: ['Fraud' 'No Fraud']

In [25]: `from sklearn.metrics import precision_score, recall_score, f1_score

Assuming y_pred is the predicted labels and y_true is the true labels
y_pred = model.predict(xtest)
distinct_values = np.unique(y_pred)
print("Distinct values in ytest:", distinct_values)`

Distinct values in ytest: ['Fraud' 'No Fraud']

In [39]: `from sklearn.metrics import precision_score, recall_score, f1_score

Calculate precision, recall, and F1 score with pos_label='Fraud'
precision = precision_score(ytest, y_pred, pos_label='Fraud')
recall = recall_score(ytest, y_pred, pos_label='Fraud')
f1 = f1_score(ytest, y_pred, pos_label='Fraud')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)`

Precision: 0.8868159203980099
Recall: 0.8802469135802469
F1 Score: 0.8835192069392813

In []: