A STUDY ON THE ONLINE PAYMENT FRAUD DETECTION USING MACHINE LEARNING

PROJECT REPORT

SUBMITTED BY

M.SWATHI (2021PITEC161)

V.SHIVA SHRUTHI (2021PITEC158)

BACHELOR OF ENGINEERING

IN

DEPARTMENT OF ELECTONICS AND COMMUNICATION ENGINEERING

ABSTRACT:

The rapid evolution of e-commerce and digital banking has significantly increased the volume of online transactions, making the financial sector a prime target for fraudulent activities. As traditional fraud detection mechanisms struggle to cope with sophisticated and constantly evolving fraud techniques, there is an urgent need for advanced solutions that can adapt and respond to new threats in real-time. This study presents a comprehensive approach to online payment fraud detection using machine learning (ML) algorithms, aimed at enhancing the security of digital transactions while minimizing false positives that can disrupt genuine transactions. Our dataset comprises a large number of transaction records, including both fraudulent and legitimate transactions, with features encompassing transaction details, user behaviour, and account information. We preprocess the data to handle missing values, normalize feature scales, and encode categorical variables, ensuring the ML models receive clean and structured input. Our results demonstrate the effectiveness of machine learning algorithms in detecting online payment fraud, with significant improvements in detection rates and reduced false positives compared to traditional rule-based systems.

In conclusion, the integration of machine learning into online payment systems offers a promising avenue for improving the security and integrity of digital transactions. By continuously adapting to new and complex fraud strategies, machine learning algorithms can help protect consumers and businesses alike, fostering a safer and more trustworthy digital financial ecosystem.

INTRODUCTION:

Online payment fraud detection is a critical area of cybersecurity that addresses the escalating challenge of fraudulent transactions in digital commerce. With the proliferation of online shopping and financial services, the potential for fraud has significantly increased, posing a substantial risk to both businesses and consumers. Traditional fraud detection methods, which often rely on rule-based systems, have proven to be inadequate in dealing with the sophistication and evolving nature of modern fraud schemes. This inadequacy has prompted the need for more advanced and adaptable solutions.

The introduction of ML into fraud detection involves several key steps, including data collection, preprocessing, feature selection, model training, and evaluation. Data collection encompasses gathering a comprehensive dataset that includes both legitimate and fraudulent transactions. Preprocessing involves cleaning and normalizing the data, while feature selection focuses on identifying the most relevant attributes that contribute to fraud detection. Model training entails using the processed data to train ML algorithms, and evaluation assesses the model's performance in accurately identifying fraudulent transactions.

The dataset is collected from Kaggle, which contains historical information about fraudulent transactions which can be used to detect fraud in online payments.

The dataset consists of 10 variables:

- step: represents a unit of time where 1 step equals 1 hour
- type: type of online transaction
- amount: the amount of the transaction
- nameOrig: customer starting the transaction
- oldbalanceOrg: balance before the transaction
- newbalanceOrig: balance after the transaction
- nameDest: recipient of the transaction
- oldbalanceDest: initial balance of recipient before the transaction
- newbalanceDest: the new balance of recipient after the transaction
- isFraud: fraud transaction

OBJECTIVE:

The objective of online payment fraud detection using machine learning is to minimize financial losses and maintain trust in digital payment systems by accurately identifying and preventing fraudulent transactions in real-time. This goal encompasses several specific objectives:

1.Accuracy Enhancement: Improve the precision and recall of fraud detection mechanisms to reduce both false positives (legitimate transactions flagged as fraudulent) and false negatives (fraudulent transactions missed by the system). High accuracy ensures that customers experience minimal disruption in their legitimate transactions while effectively catching fraud.

2.Real-time Detection: Achieve the capability to detect and flag fraudulent transactions as they occur, allowing for immediate action to prevent fraud. This is crucial in mitigating losses and preventing the execution of unauthorized transactions.

3.Adaptability and Scalability: Develop systems that can adapt to emerging fraud tactics and scale with the growth in transaction volumes and complexity. Fraudsters continuously evolve their strategies to bypass detection, so the system must learn from new patterns of fraud to stay effective.

4.Cost Efficiency: Reduce the operational costs associated with manual review and investigation of transactions by automating the fraud detection process. Machine learning can handle vast volumes of transactions more efficiently than human teams, freeing up resources for other critical security tasks.

5.User Experience Improvement: Enhance the overall customer experience by minimizing friction during the transaction process. By reducing false positives, customers face fewer unjustified transaction declines, leading to higher satisfaction and trust in the payment system.

EXISTING SYSTEM:

The existing systems for online payment fraud detection using machine learning are diverse and sophisticated, employing various techniques and algorithms to identify and prevent fraudulent transactions. These systems are integral to the security infrastructure of financial institutions, e-commerce platforms, and payment processors. Here's an overview of the key components and approaches used in these systems:

1. Data Collection and Preprocessing

2. Machine Learning Models

- Supervised Learning Models
- Unsupervised Learning Models
- Deep Learning

3. Ensemble Techniques

4. Real-time Analysis and Decision Systems

5. Adaptive Learning

6. Integration with Other Systems

PROPOSED SYSTEM:

The proposed system for online payment fraud detection aims to build upon the foundations of existing systems while addressing their limitations and incorporating the latest advancements in machine learning

(ML) and data analytics. The goal is to create a more dynamic, efficient, and accurate system that not only detects fraudulent transactions in real-time but also adapts to evolving fraud patterns more effectively. Here's an outline of the key components and innovations of the proposed system:

1. Importing Libraries and Datasets
2. Dataset Cleaning and Evaluation
3. Exploratory Data Analysis
4. Tailoring dataset
5. Decision Tree Classifier
6. Final Test Set Evaluation

CODE:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing Libraries and Datasets:

The libraries used are:

- **Pandas:** This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.
- **Seaborn/Matplotlib:** For data visualization.
- **Numpy:** Numpy arrays are very fast and can perform large computations in a very short time.

In 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from plotly.offline import init_notebook_mode, plot, iplot
import cufflinks as cf
init_notebook_mode(connected=True)
cf.go_offline()
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "colab"
pio.templates.default = 'seaborn'
```

In 2:

```
df = pd.read_csv('/content/onlinefraud.csv')
```

Dataset Cleaning and Evaluation:

In 3:

```
df.info()
```

Out 3:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97225 entries, 0 to 97224
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            97225 non-null  int64
 1   type            97225 non-null  object
 2   amount          97225 non-null  float64
 3   nameOrig        97225 non-null  object
 4   oldbalanceOrg   97225 non-null  float64
 5   newbalanceOrig  97225 non-null  float64
 6   nameDest        97225 non-null  object
 7   oldbalanceDest  97225 non-null  float64
 8   newbalanceDest  97224 non-null  float64
 9   isFraud         97224 non-null  float64
 10  isFlaggedFraud  97224 non-null  float64
dtypes: float64(7), int64(1), object(3)
memory usage: 8.2+ MB
```

In 4:

```
df.head(15)
```

Out 4:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0.0	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0.0	0.0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1.0	0.0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1.0	0.0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0.0	0.0
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0.0	0.0
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0.0	0.0
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0.0	0.0
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0.0	0.0
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0.0	0.0
10	1	DEBIT	9644.94	C1900366749	4465.00	0.00	C997608398	10845.0	157982.12	0.0	0.0
11	1	PAYMENT	3099.97	C249177573	20771.00	17671.03	M2096539129	0.0	0.00	0.0	0.0
12	1	PAYMENT	2560.74	C1648232591	5070.00	2509.26	M972865270	0.0	0.00	0.0	0.0
13	1	PAYMENT	11633.76	C1716932897	10127.00	0.00	M801569151	0.0	0.00	0.0	0.0
14	1	PAYMENT	4098.78	C1026483832	503264.00	499165.22	M1635378213	0.0	0.00	0.0	0.0

In 5:

```
df.describe()
```

Out 5:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	97225.000000	9.722500e+04	9.722500e+04	9.722500e+04	9.722500e+04	9.722400e+04	97224.000000	97224.0
mean	8.456817	1.724217e+05	8.793150e+05	8.956148e+05	8.792683e+05	1.182315e+06	0.001173	0.0
std	1.833480	3.419651e+05	2.689865e+06	2.727826e+06	2.403354e+06	2.802840e+06	0.034223	0.0
min	1.000000	3.200000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.0
25%	8.000000	9.893120e+03	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.0
50%	9.000000	5.208135e+04	1.994700e+04	0.000000e+00	2.080800e+04	4.894480e+04	0.000000	0.0
75%	10.000000	2.103607e+05	1.863453e+05	2.107046e+05	5.853365e+05	1.051531e+06	0.000000	0.0
max	10.000000	1.000000e+07	3.379739e+07	3.400874e+07	3.400874e+07	3.397234e+07	1.000000	0.0

In 6:

```
df.isFraud.value_counts()
```

Out 6:

0.0 97110
1.0 114
Name: isFraud, dtype: int64

In 7:

```
df.isFlaggedFraud.value_counts()
```

Out 7:

0.0 97224
Name: isFlaggedFraud, dtype: int64

In 8:

```
merged_df = pd.merge(df[df['isFraud']==1], df[df['isFlaggedFraud']==1],
how='outer', indicator=True)
common_rows = merged_df[merged_df['_merge'] == 'both']
```

In 9:

```
len(common_rows)
```

Out 9:

0

In 10:

```
df.isnull().sum()
```

Out 10:

step 0
type 0
amount 0
nameOrig 0
oldbalanceOrg 0
newbalanceOrig 0

```
nameDest          0
oldbalanceDest    0
newbalanceDest    1
isFraud           1
isFlaggedFraud    1
dtype: int64
```

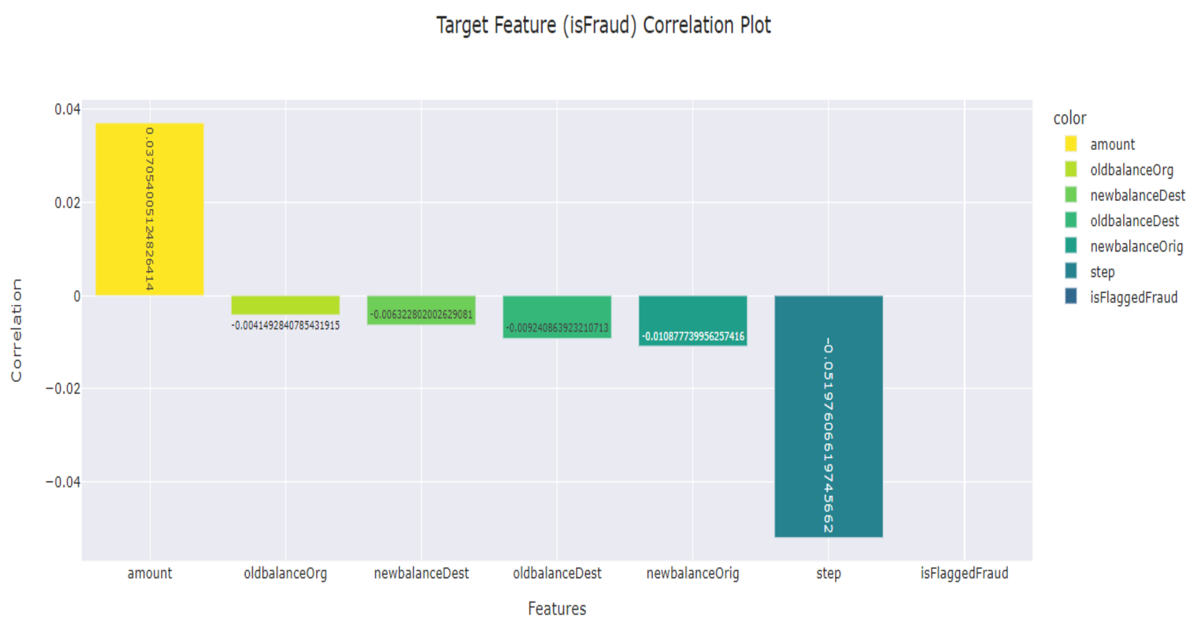
No null values present in the dataset. No cleaning of the dataset needed.

Exploratory Data Analysis:

In 11:

```python
# Correlation between isFraud feature with other features
#
df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)[1:].iplot(kind='bar')
fig1 =
px.bar(x=df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)[1:].index,
y=df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)[1:].values,
              color=df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)[1:].index,
color_discrete_sequence=px.colors.sequential.Viridis_r,
              text=df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)[1:].values, title='Target Feature (isFraud) Correlation Plot')
fig1.update_xaxes(title_text='Features')
fig1.update_yaxes(title_text='Correlation')
```

Out 11:



The correlation plot shows that isFraud feature is most correlated with the amount of transaction made, though it is only 7.6%. isFraud feature is negatively correlated with oldbalanceDest and newbalanceOrig features.

In 12:

```
df.type.value_counts().sort_values()
```

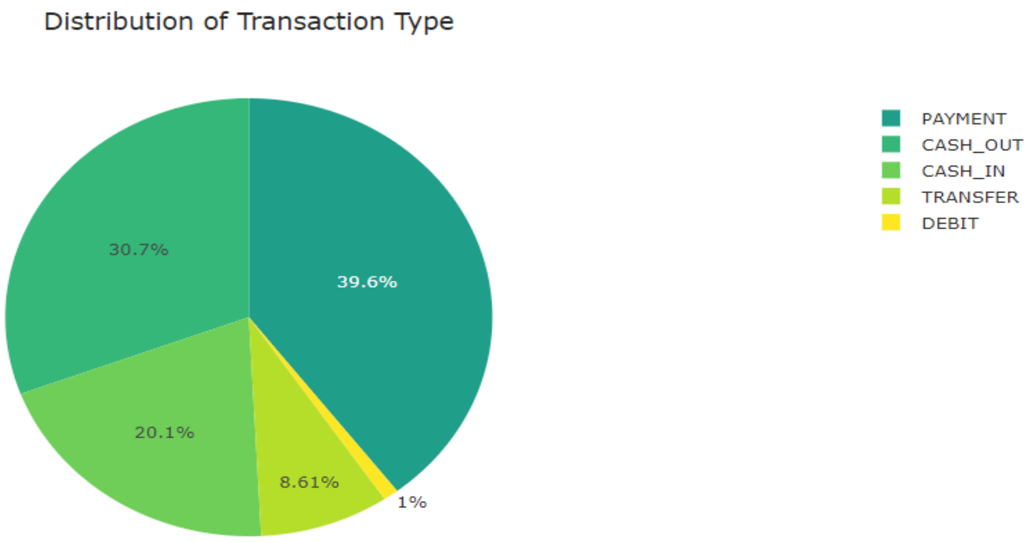
Out 12:

```
DEBIT       976
TRANSFER    8371
CASH_IN    19561
CASH_OUT   29839
PAYMENT    38478
Name: type, dtype: int64
```

In 13:

```
# Transaction Type Distribution
fig2 = px.pie(data_frame=df,
              values=df.type.value_counts().sort_values().values,
              names=df.type.value_counts().sort_values().index,
              title='Distribution of Transaction Type',
color=df.type.value_counts().sort_values().index,
             color_discrete_sequence=px.colors.sequential.Viridis_r,
height=500)
fig2
```

Out 13:



The above transaction type distribution shows that CASH_OUT, PAYMENT and CASH_IN are top 3 most preferred transaction types with TRANSFER and DEBIT being the least preferred.

Tailoring Dataset:

In 14:

```
df['type'] = df['type'].map({'CASH_OUT':1, 'PAYMENT':2, 'CASH_IN': 3,
'TRANSFER': 4, 'DEBIT': 5})
df['isFraud'] = df['isFraud'].map({0: 'No Fraud', 1: 'Fraud'})
# df['isFlaggedFraud'] = df['isFlaggedFraud'].map({0: 'No Fraud', 1:
'Fraud'})
df.drop(['nameOrig', 'nameDest'], axis=1, inplace=True)
```

In 15:

```
df.head()
```

Out 15:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	2	9839.64	170136.0	160296.36	0.0	0.0	No Fraud	0.0
1	1	2	1864.28	21249.0	19384.72	0.0	0.0	No Fraud	0.0
2	1	4	181.00	181.0	0.00	0.0	0.0	Fraud	0.0
3	1	1	181.00	181.0	0.00	21182.0	0.0	Fraud	0.0
4	1	2	11668.14	41554.0	29885.86	0.0	0.0	No Fraud	0.0

In 16:

```
from sklearn.model_selection import train_test_split
```

As the dataset is quite large and it may take very long to train model on such a large dataset. So I will use only a small portion of the dataset to continue with this project.

In 17:

```
current_ratio = df.isFraud.value_counts(normalize=True)
current_ratio
```

Out 17:

No Fraud 0.999164
Fraud 0.000836
Name: isFraud, dtype: float64

In 18:

```
new_df_size = 80000
nofraud_need, fraud_need =
np.ceil(new_df_size*current_ratio).astype(int)
```

In 19:

```
df_nofraud = df[df['isFraud'] == 'No Fraud'].sample(nofraud_need)
df_fraud = df[df['isFraud'] == 'Fraud'].sample(fraud_need)
df_new = pd.concat([df_nofraud, df_fraud], ignore_index=True)
df_new.head()
```

Out 19:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	181	1	161218.39	12926.00	0.00	369868.98	531087.37	No Fraud	0.0
1	163	2	8046.65	51388.00	43341.35	0.00	0.00	No Fraud	0.0
2	260	3	185307.99	1424969.44	1610277.43	2380676.82	2195368.83	No Fraud	0.0
3	253	3	289313.21	10621581.99	10910895.20	348970.75	59657.55	No Fraud	0.0
4	33	1	41522.53	119403.00	77880.47	0.00	41522.53	No Fraud	0.0

In 20:

```
X_new = df_new.drop('isFraud', axis=1)
y_new = df_new.isFraud
```

In 21:

```
# X = df.drop('isFraud', axis=1).iloc[:80000]
# y = df.isFraud.iloc[:80000]
```

In 22:

```
y_new.value_counts()
```

Out 22:

No Fraud 79934
Fraud 67
Name: isFraud, dtype: int64

In 23:

```
X_train, X_out, y_train, y_out = train_test_split(X_new, y_new,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_out, y_out,
test_size=0.5, random_state=42)
```

Decision Tree Classifier:

In 24:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
from sklearn.model_selection import GridSearchCV
param_grid = {'criterion': ['gini', 'entropy', 'log_loss'],
              'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
              'max_features': [2, 3, 8, 'sqrt', 'log2']
              }
from sklearn.metrics import f1_score, make_scorer

# Define a custom scorer
f1_scorer = make_scorer(f1_score, pos_label='Fraud')
# from sklearn.metrics import SCORERS
# print(SCORERS.keys())
dtree_grid = GridSearchCV(estimator=dtree,
                          param_grid=param_grid,
                          n_jobs=-1,
                          cv=5,
                          scoring=f1_scorer,
                          error_score="raise"
                          )
```

In 25:

```
dtree_grid.fit(X_train, y_train)
```

Out 25:

```
▾ GridSearchCV
GridSearchCV(cv=5, error_score='raise', estimator=DecisionTreeClassifier(),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
                         'max_features': [2, 3, 8, 'sqrt', 'log2']},
             scoring=make_scorer(f1_score, pos_label=Fraud))
▾ estimator: DecisionTreeClassifier
DecisionTreeClassifier()
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In 26:

```
dtree_grid.best_estimator_
```

Out 26:

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features='log2')
```

In 27:

```
dtree_grid.best_params_
```

Out 27:

{'criterion': 'entropy', 'max_depth': 9, 'max_features': 3}

In 28:

```
dtree_grid.best_score_
```

Out 28:

0.592382960035134

In 29:

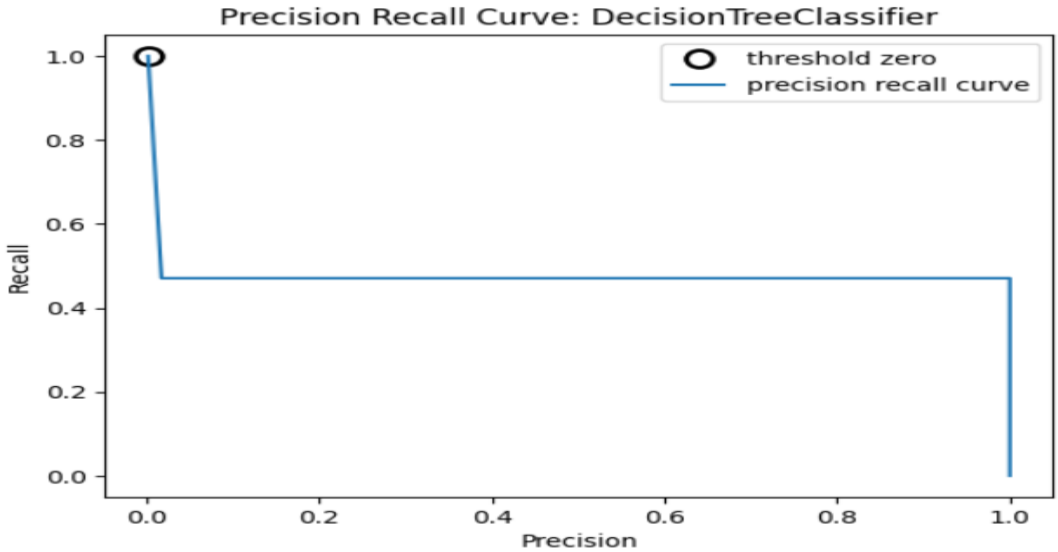
```
dtree_pred = dtree_grid.predict(X_val)
from sklearn.metrics import classification_report,
ConfusionMatrixDisplay, confusion_matrix, precision_recall_curve,
roc_curve, roc_auc_score
```

In 30:

```
precision, recall, thresholds = precision_recall_curve(y_val,
dtree_grid.best_estimator_.predict_proba(X_val)[:, 0],
pos_label='Fraud')
# find threshold closest to zero
close_zero = np.argmin(np.abs(thresholds))
plt.plot(precision[close_zero], recall[close_zero], 'o', markersize=10,
label="threshold zero", fillstyle="none", c='k', mew=2)
plt.plot(precision, recall, label="precision recall curve")
plt.title('Precision Recall Curve: DecisionTreeClassifier')
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.legend(loc='best')
```

Out 30:

<matplotlib.legend.Legend at 0x7a9e8df8f220>

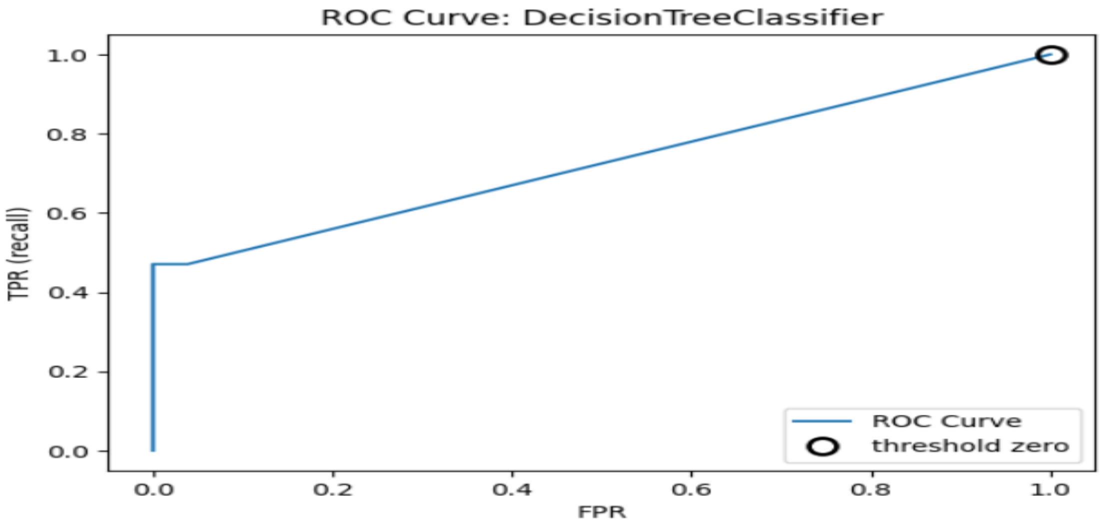


In 31:

```
fpr, tpr, thresholds = roc_curve(y_val,
dtree_grid.best_estimator_.predict_proba(X_val)[:, 0],
pos_label='Fraud')
plt.title('ROC Curve: DecisionTreeClassifier')
plt.plot(fpr, tpr, label="ROC Curve")
plt.xlabel("FPR")
plt.ylabel("TPR (recall)")
# find threshold closest to zero
close_zero = np.argmin(np.abs(thresholds))
plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
label="threshold zero", fillstyle="none", c='k', mew=2)
plt.legend(loc=4)
```

Out 31:

<matplotlib.legend.Legend at 0x7a9e8e04eef0>



In 32:

```
print('AUC of Decision Tree: ', roc_auc_score(y_val,
dtree_grid.best_estimator_.predict_proba(X_val)[:, 1]))
```

Out 32:

AUC of Decision Tree: 0.7250222128407401

In 33:

```
print('Decision Tree Classifier: Confusion Matrix')
ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_val,
dtree_pred), display_labels=dtree_grid.classes_).plot();
```

Out 33:

Decision Tree Classifier: Confusion Matrix

In 34:

```
print(f"Classification Report Decision Tree
Classifier:\n{classification_report(y_val, dtree_pred)}")
```

Out 34:

```
Classification Report Decision Tree Classifier:
              precision    recall  f1-score   support

       Fraud       0.73      0.47      0.57        17
    No Fraud       1.00      1.00      1.00     11983

    accuracy                           1.00     12000
   macro avg       0.86      0.74      0.79     12000
weighted avg       1.00      1.00      1.00     12000
```

Final Test Set Evaluation:

In 35:

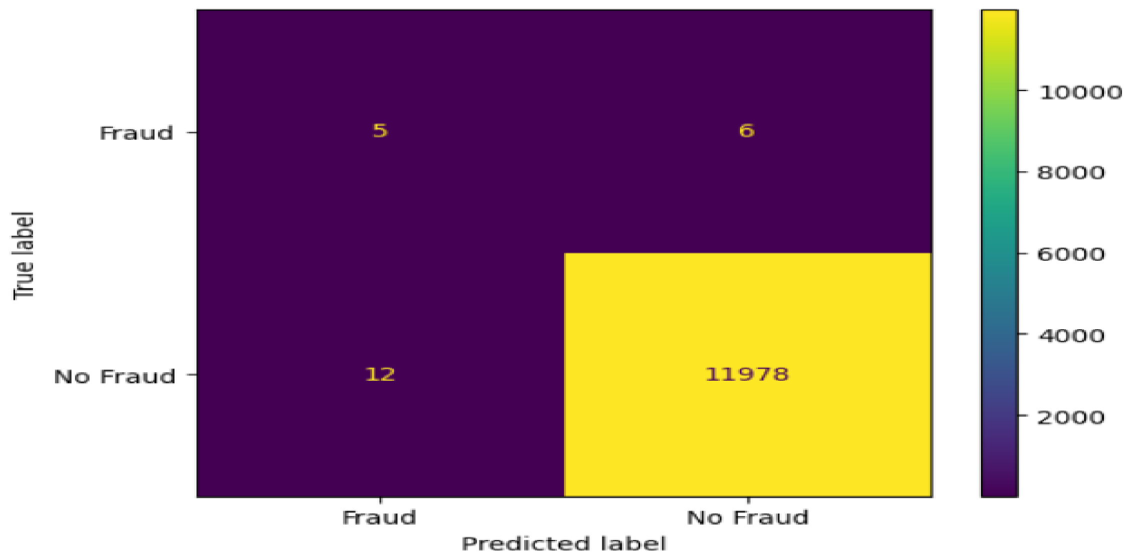
```
test_pred = dtree_grid.predict(X_test)
```

In 36:

```
print('Decision Tree Classifier: Test Set Confusion Matrix')
ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test,
test_pred), display_labels=dtree_grid.classes_).plot();
```

Out 36:

Decision Tree Classifier: Test Set Confusion Matrix



In 37:

```
print(f"Classification Report Decision Tree Classifier Test
Set:\n{classification_report(y_test, test_pred)}")
```

Out 37:

```
Classification Report Decision Tree Classifier Test Set:
              precision    recall  f1-score   support

       Fraud       0.29      0.45      0.36        11
    No Fraud       1.00      1.00      1.00     11990

    accuracy                           1.00     12001
   macro avg       0.65      0.73      0.68     12001
weighted avg       1.00      1.00      1.00     12001
```

In 38:

```
feature_names = ['step', 'type', 'amount', 'oldbalanceOrg',
'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest', 'isFlaggedFraud']
features = np.array([[1, 1, 100000.0, 100000.0, 0.0, 0.0, 0.0, 0]])
features_df = pd.DataFrame(features, columns=feature_names)
print(dtree_grid.predict(features_df))
```

Out 38:

['No Fraud']

CONCLUSION:

- The project on online payment fraud detection using machine learning explored several critical techniques and algorithms to address the challenges inherent in identifying fraudulent transactions.
- So, the decision tree classifier performs well on the final test set correctly classifying most of the fraud payments. It also correctly classified the features_df.