TASK NO-5: Predicting Credit Card Fraud Detection.

<u>Data Collection</u>: Gather credit card transaction data (transaction amount, merchant, time,
user details, etc.) in CSV format.

<u>Data Preprocessing:</u>

Data loading Subtask:

Load the credit card transaction data from the provided CSV file "creditcard.csv" into a pandas DataFrame.

Reasoning: Load the credit card transaction data from the provided CSV file "creditcard.csv" into a pandas DataFrame.

Code:

```
import pandas as pd
 try:
    df = pd.read_csv('creditcard.csv')
display(df.head()) except
FileNotFoundError:
    print("Error: 'creditcard.csv' not found.")
df = None
```

V V
T V V V V V V 2 V V V V A C
i V V V V m 1 2 3 4 e ... 2 2 V 2 2 2 2 m la
5 6 7 8 9 2 5 6 7 8 ou s
1 2 3 4 nt s

- - - - - - -
2. 1. 0. 0. 0. 0. 0. 0. 0. 0.
1. 0. 0. 0. 0. 0. 0. 1 5 3 4 2
0 3 2 0 1 1
3 0 3 0 1 1 0 4
0 3 7 6 3 9 6 . 7 6 2 3
5 7 3 1 1 8 2 9
0 . 6 8 2 9 8 3 . 7 6 8 3 0
9 2 8 8 0 9 1 .
0 3 1 3 5 6 7 . 8 9 5 5
8 7 3 3 4 1 0 6

4 5 8 9 9 8 3 2 3 5
0 8 2 0 7 1 5 2
7 5 8 9 8 7 8 8 9 8
7 1 1 7 4 5 3

1. 0. 0. 0. 0. - - 0. - - - 0. - 0. 0. - 0.
2
0 1 2 1 4 0 0. 0. 0 0. . 0. 0. 1 0. 1 1 0. 0
.
1 . 9 6 6 4 6 0 0 8 2 . 2 6
0 3 6 2 0 1 0
6
0 1 6 6 8 0 8 7 5 5 . 2
3 1 3 7 5 0 4
9
8 1 4 1 0 2 8 1 5 5 8 2 9
1 8 8 7
T A C
V V V V V V V V
i V V V V V V V V V m la
... 2 2 2 2 2
2 2 2 m 1 2 3 4 5 6 7 8
9 ou s
1 2 3 4 5 6 7 8
e nt s

5 5 8 5 1 3 8 0 4 7 6 8 8 7 9 9 2
7 1 0 4 8 6 0 2 2 7 7 8 4 0 5 8 4
1 3 5 5 2 6 3

- - - - - - - - -
1. 0. 1. 0. 0. 0. 0. 0.
1. 1. 0. 1. 0. 0. 0. 0. 0. 3 7 3 8 7 2 2 7 9
3 3 5 5 6 3 1 0 0 7
1 7 7 0 9 4 . 4 7 0
5 4 0 1 8 2 3 5 5 8
2 . 3 9 0 1 7 . 7 1 9
0 8 0 3 4 9 7 9 5 9 .
0 2 7 4 4 6 . 9 6 4
3 1 1 6 2 6 0 3 7 6
0 8 9 6 7 9 7 1
5 6 9 5 8 4 9 5 5 6
9 0 9 1 6 8 9 2
4 3 8 4 1 2 7 3 2

- - - - - - - - -
1. 1. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 1. 0. 0. 1. 0. 1
7 2 2 3 0 6 0 0
9 1 8 0 3 1 1 1 2 2 1 9 4
3 7 . 0 4 6 6
6 8 6 1 8 0 9 7 2 3 **3** . 2 7 7 7 . 5 7 2 1 0 6 5 3 0 7 8 0 5 1 . 0 9 2 6 4 . 2 3 7 4
2 2 2 3 0 3 3 5 9 5
9 0 0 3 7 7 2 5
7 2 9 0 2 0 2 7 2 0
3 3 9 6 4 6 3 8
2 6 1 9 4 0 1 5 9

- - - - - -

0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
0.
1. 0. 0. 0. 0. 0.
8 5 4 0 5 8 7 1 5 2 2 6
1 4 2 0 1 2
2 7 4 0 9 9 1 . 9 4 0
1 1 9
5 0 7 0 3 0
4 . 7 8 3 5 2 7 . 8 1 2 9 5 . 0
8 7 0 9 7 6
0 7 7 0 9 9 7 . 2 2 2 4 1 9
2 1 5 4 4 0
3 1 3 2 4 3 7 6 9 2
5 9
3 9 3 3 5 1
7 8 4 1 1 9 8 7 2 2 3

3 3 3 1 8 0

Data exploration

Subtask:

Explore the loaded credit card transaction data to understand its characteristics.

Reasoning: Explore the data shape, types, missing values, descriptive statistics, distributions, and correlations as requested in the subtask.

Code:

```
# Data Shape and Types print("Data Shape:", df.shape)
print("\nData Types:\n", df.dtypes)
```

```
# Missing Values print("\nMissing Values:\n",
df.isnull().sum()) print("\nPercentage of Missing Values:\n",
(df.isnull().sum() / len(df)) * 100)

# Descriptive Statistics print("\nDescriptive Statistics
for Numerical Features:\n", df.describe())
print("\nDescriptive Statistics for 'Amount':\n",
df['Amount'].describe())

# Feature Distributions and Class Imbalance print("\nClass
Distribution:\n", df['Class'].value_counts())
class_imbalance_ratio = df['Class'].value_counts()[0] /
df['Class'].value_counts()[1] print("\nClass Imbalance Ratio
(Non-Fraudulent/Fraudulent):", class_imbalance_ratio)

# Correlation Analysis print("\nCorrelation with 'Class':\n",
df.corrwith(df['Class']))
```

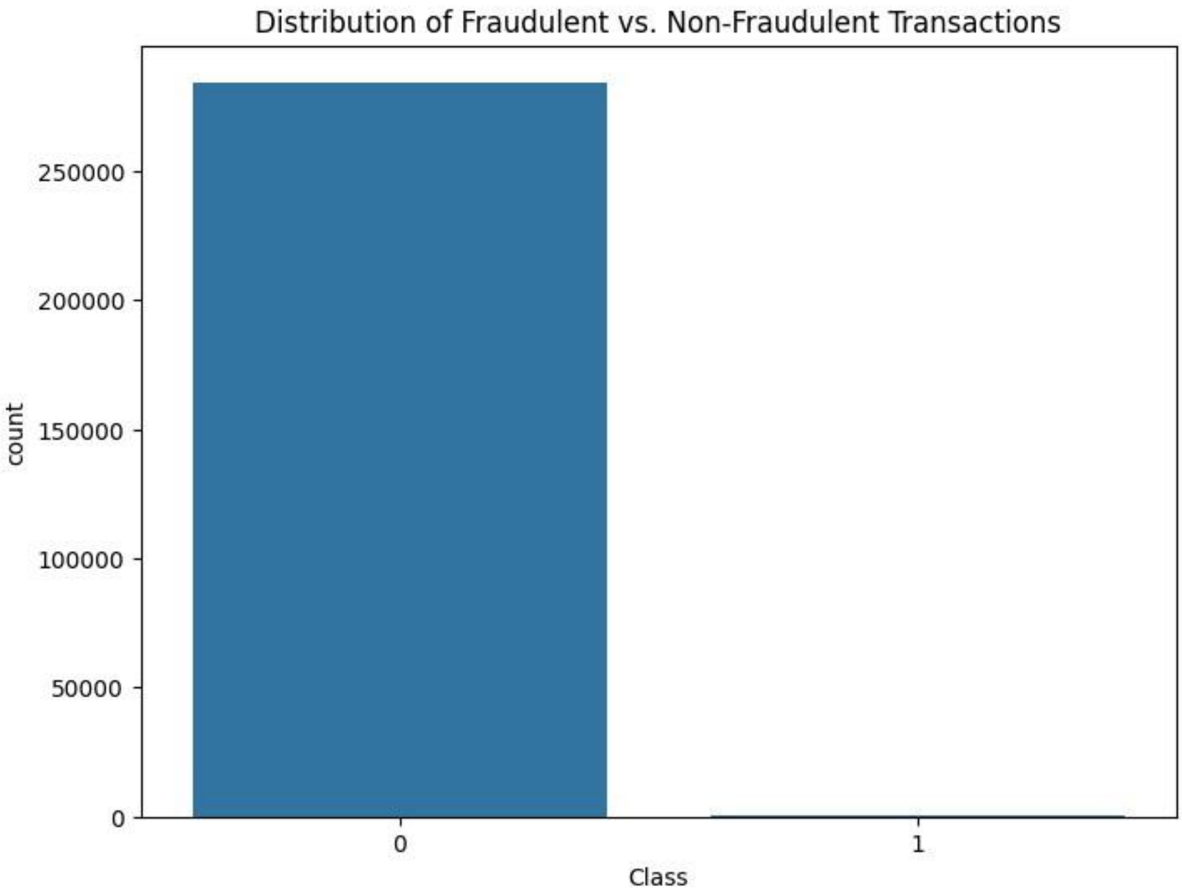
```
Data Shape: (284807, 31)
 Data
Types:
 Time      float64
V1        float64
V2        float64
V3        float64
V4        float64
V5        float64
V6        float64
V7        float64
V8        float64
V9        float64
V10       float64
V11       float64
V12       float64
V13       float64
V14       float64
V15       float64
V16       float64
V17       float64
V18       float64
V19       float64
V20       float64
V21       float64
V22       float64
V23       float64
V24       float64
V25       float64
V26       float64
V27       float64
V28       float64
```

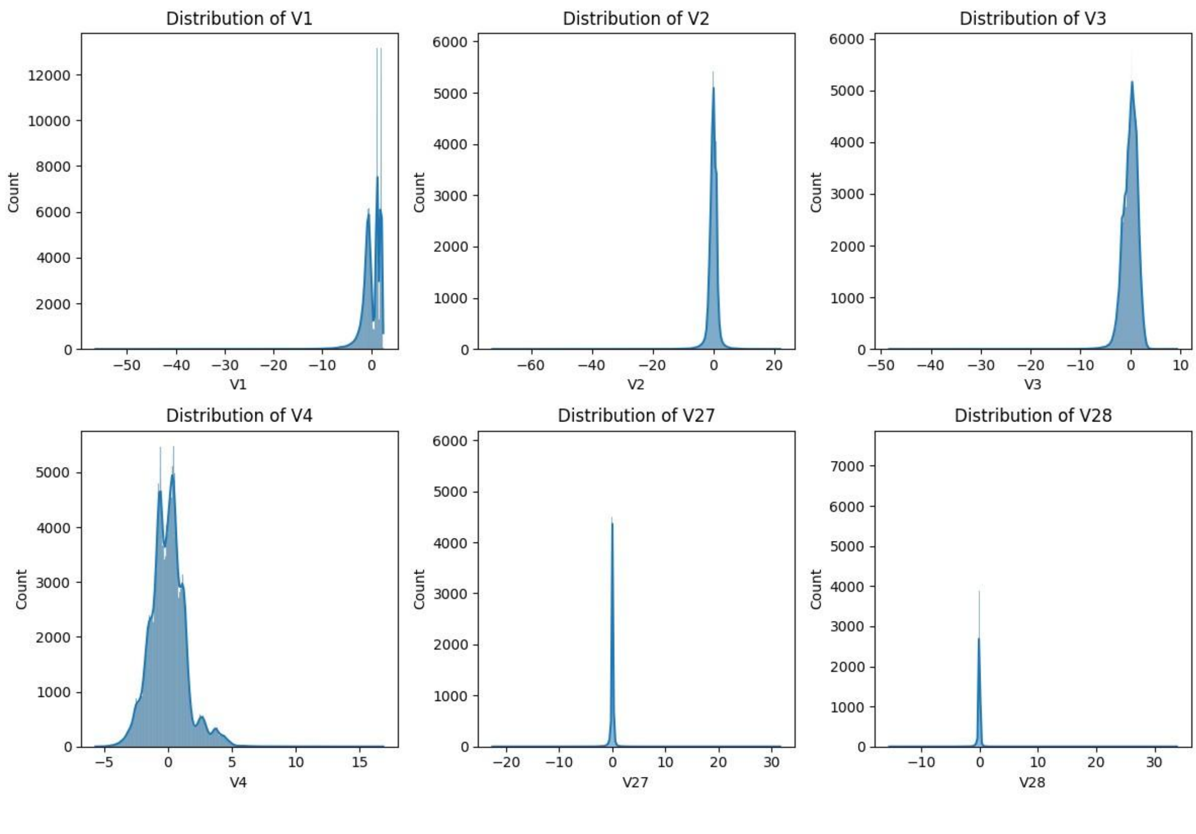
```
Amount    float64
Class       int64 dtype: object  Missing Values:
 Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0 Class
0 dtype: int64

Percentage of Missing Values:
 Time      0.0
V1        0.0
V2        0.0
V3        0.0
V4        0.0
V5        0.0
V6        0.0
V7        0.0
V8        0.0
V9        0.0
V10       0.0
V11       0.0
V12       0.0
V13       0.0
V14       0.0
V15       0.0
V16       0.0
V17       0.0
V18       0.0
V19       0.0
```

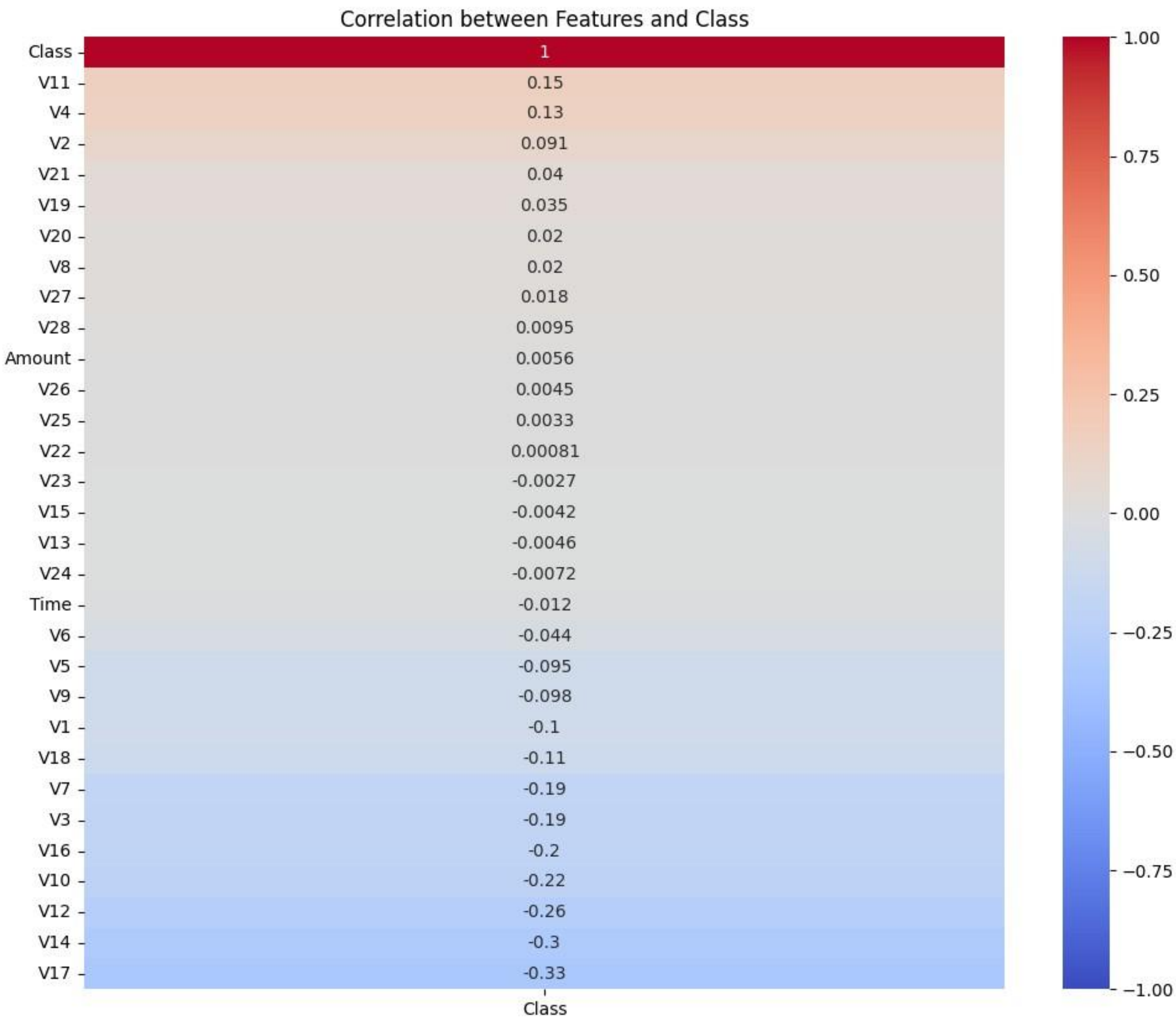
```
V20       0.0
V21       0.0
V22       0.0
V23       0.0
V24       0.0
V25       0.0
V26       0.0
V27       0.0
V28       0.0
Amount    0.0 Class
0.0 dtype: float64

Descriptive Statistics for Numerical Features:
                 Time             V1             V2             V3
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15
2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00
1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01
5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01
8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01
1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00
7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00
1.687534e+01

                 V5             V6             V7             V8
V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -
2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00
1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01
1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01
6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02
5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01
5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01
1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
```

```
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00
                V25           V26           V27           V28
Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16
88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02
5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02
77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01
25691.160000

               Class
count  284807.000000
mean        0.001727   std
0.041527   min
0.000000   25%
0.000000
50%         0.000000   75%
0.000000   max
1.000000

[8 rows x 31 columns]

Descriptive Statistics for 'Amount':
 count    284807.000000
mean         88.349619 std
250.120109 min
0.000000 25%
5.600000
50%          22.000000 75%
77.165000 max
25691.160000 Name: Amount,
dtype: float64

Class Distribution:
 Class
0      284315
1      492
Name: count, dtype: int64

Class Imbalance Ratio (Non-Fraudulent/Fraudulent): 577.8760162601626
```

```
Correlation with 'Class':
 Time      -0.012323
V1        -0.101347
V2         0.091289
V3        -0.192961
V4         0.133447
V5        -0.094974
V6        -0.043643
V7        -0.187257
V8         0.019875
V9        -0.097733
V10       -0.216883
V11        0.154876
V12       -0.260593
V13       -0.004570
V14       -0.302544
V15       -0.004223
V16       -0.196539
V17       -0.326481
V18       -0.111485
V19        0.034783
V20        0.020090
V21        0.040413
V22        0.000805
V23       -0.002685
V24       -0.007221
V25        0.003308
V26        0.004455
V27        0.017580
V28        0.009536
Amount     0.005632 Class
1.000000 dtype: float64
```


Distribution of Fraudulent vs. Non-Fraudulent Transactions
count
250000
200000
150000
100000
50000
0
0
1
Class


Distribution of V1
Distribution of V2
Distribution of V3
Distribution of V4
Distribution of V27
Distribution of V28
Count
V1
V2
V3
V4
V27
V28



Data cleaning Subtask:

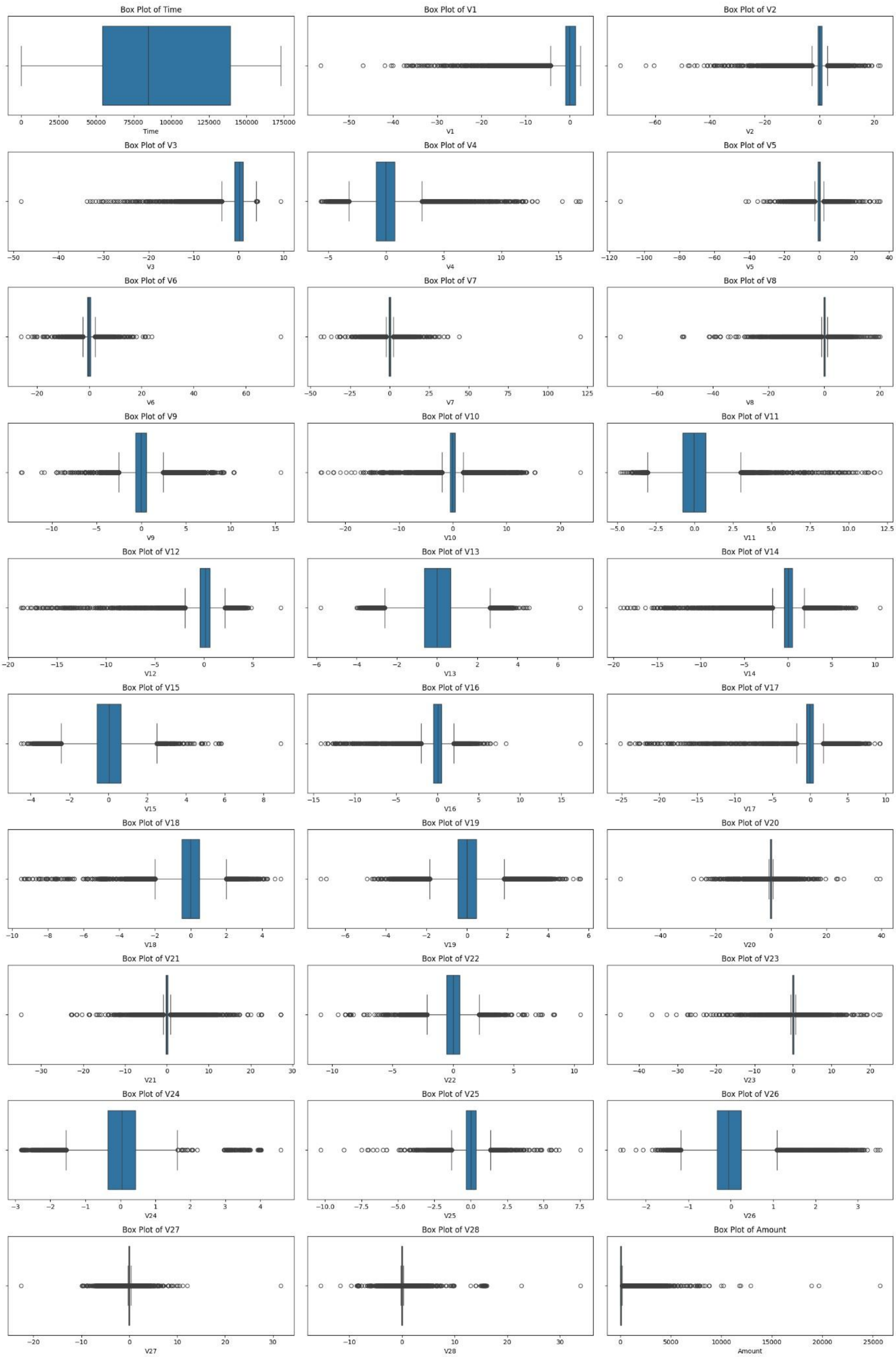
Clean the data by handling outliers and removing duplicate rows. Focus on the 'Amount' column for outlier treatment.

Reasoning: Clean the data by handling outliers in the 'Amount' column and removing duplicate rows.

Code:

```
import numpy as np

# Calculate the 99th percentile of the 'Amount' column
```

```
percentile_99 = np.percentile(df['Amount'], 99)

# Cap values above the 99th percentile at the 99th percentile value
df['Amount'] = np.where(df['Amount'] > percentile_99, percentile_99,
df['Amount'])

# Remove duplicate rows and reset the index df_cleaned =
df.drop_duplicates().reset_index(drop=True)
```


Box Plot of Time
Time
Box Plot of V1
V1
Box Plot of V2
V2
Box Plot of V3
V3
Box Plot of V4
V4
Box Plot of V5
V5
Box Plot of V6
V6
Box Plot of V7
V7
Box Plot of V8
V8
Box Plot of V9
V9
Box Plot of V10
V10
Box Plot of V11
V11
Box Plot of V12
V12
Box Plot of V13
V13
Box Plot of V14
V14
Box Plot of V15
V15
Box Plot of V16
V16
Box Plot of V17
V17
Box Plot of V18
V18
Box Plot of V19
V19
Box Plot of V20
V20
Box Plot of V21
V21
Box Plot of V22
V22
Box Plot of V23
V23
Box Plot of V24
V24
Box Plot of V25
V25
Box Plot of V26
V26
Box Plot of V27
V27
Box Plot of V28
V28
Box Plot of Amount
Amount

A

T **m** **C**

V V V V V V V V

i V V V V V V V V V **o la**

... 2 2 2 2 2

2 2 2 m 1 2 3 4 5 6 7 8

9 u s

1 2 3 4 5 6 7 8

e **n s**

t

- - - - - - -

1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

1.

0. 0. 0. 0. 0. 0. 0.

2. 8 6 1 1 3 0 2 0 1 8

7

9 6 0 2 3 2 2 2 7 3 7 4

3 . 2 5 3 6 1 8

9 1 9 1 2 4 4

0 8 8 1 7 2 4 . 6 3 2 7

6 3 0

4 6 8 7 0 1 6

9 1 2 4 4 7 . 8 1 6 6 7 2

9 2 6 8 7 1 0

0 5 4 0 3 8 6 0 8 1 3 7

8 3 0 5 9 8 9

5 2 5 6 2 7 6 9 1 9 1 4

3 7 2 9 1 2 1

- - - - - - - - - -

0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0.

5 1 2 0 1 1 3 2 3 0

9 0 1 1 4 6 4 0 2

2 4 9 8 6 1 . 6 2 1 2

9 0 0 6 7 0 7 6 6

1 4 3 3 7 4 7 . 3 5 3 6

0

4 6 5 4 3 2 9 3 9

9 1 9 7 3 0 . 4 4 4 5

9 9 7 4 3 7 5 7 8

2 0 7 2 9 6 4 6 7 1

8 7 4 8 3 1 5 8 2 9 0 4 6

5 4 2 2 5 9

3 0 0 2 2 9 7 1 5

- - - - - - - - - -

0. 0. 1. 0. 0. 0. 0. 2. 4.

0. 0. 1. 0. 1. 0. 0. 0. 0. 0.

8 2 7 6 4 6 7 9 9

9 6 0 3 1 9 5 1 3 5

3 5 7 6 2 . 6 1 7 8

9 1 0 4 7 1 1 5 5 4

2 0 1 8 8 0 . 8 4 4 3

0

4 5 2 4 9 9 5 3 0 0

9 0 0 1 3 . 9 2 6 7

9 5 4 3 7 5 4 1 2 9
3 2 0 6 8 1 5 0 27 8 0
4 9 8 3 1 1 6
2 4 7 4 8 7 4 3 1
2 7 7 5 6 9 0 1 8 2

\- - - - - - - - - -
0. 0. 1. 0. 0. 0. 0. 0. 1.
0. 0. 0. 0. 1. 0. 0. 0. 1. 0.
8 03 1 6 9 3 3 4
9 41 5 0 1 0 5 5 2
4 3 0 7 6 . 4 7 8 1
9 4 7 2 7 9 0 7 3 9
3 1 3 3 5 2 . 4 9 2 8
0
4 0 8 9 6 0 1 8 1 8
2 7 8 6 4 . 4 1 6 2
9 2 7 8 8 1 4 7 9 9
5 7 3 3 8 8 6 1 97 3 8
0 8 0 0 8 6 5
0 5 2 7 9 2 3 1 1
2 9 9 8 8 5 8 6 3 9

\- - - - - -
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0.
0. 0. 0. 0. 0. 0.
5 7 2 3 4 7 0 7 1 9 3 5 6
9 5 2 5 4 3
7 1 6 1 9 0 . 4 3 2 7 4 5 7
9 2 7 4 0 3
4 9 3 5 7 1 0 . 8 9 6 6 7 3 0 0
4 6 0 6 7 3
2 8 6 1 6 8 . 2 0 2 2 1
7 5
9 0 6 4 9 3
3 6 3 8 2 0 6 9 9 2 3 1
7
6 8 9 6 8 0
9 1 2 3 5 8 6 2 3 1 3 6 9
0 9 5 3 0 8

Data preparation

Subtask:

Prepare the data for model training by scaling numerical features and splitting the data into training, validation, and testing sets.

Reasoning: Scale numerical features using

StandardScaler and split the data into training, validation, and testing sets.

Code:

```
from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split

# Select numerical features for scaling (excluding 'Class' and 'Time')
numerical_features = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8',
'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',
'Amount']

# Separate features (X) and target variable (y)
X = df_cleaned[numerical_features] y =
df_cleaned['Class']

# Split data into training and temporary sets (80/20) X_train,
X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)

# Split temporary set into validation and testing sets (50/50) X_val,
X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42, stratify=y_temp)

# Scale numerical features using StandardScaler scaler
= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Convert the scaled data back to dataframes
X_train = pd.DataFrame(X_train, columns=numerical_features)
X_val = pd.DataFrame(X_val, columns=numerical_features)
X_test = pd.DataFrame(X_test, columns=numerical_features)
```

Data splitting

Subtask:

Split the preprocessed data (df_cleaned) into training and testing sets.

Reasoning: Split the preprocessed data into training and testing sets using train_test_split, stratifying by the 'Class' column.

Code:

```
from sklearn.model_selection import train_test_split

# Assuming df_cleaned is the preprocessed DataFrame
X = df_cleaned.drop('Class', axis=1) y =
df_cleaned['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
```

Feature engineering

Subtask:

Explore potential feature engineering opportunities to improve model performance.

Reasoning: Create new features based on the 'Time' variable (hour of the day and day of the week), train a Logistic Regression model with and without these new features, and evaluate their impact on model accuracy using AUC-ROC.

Code:

```
import pandas as pd from sklearn.linear_model import
LogisticRegression from sklearn.metrics import roc_auc_score
import numpy as np
```

```
# Feature engineering: Create 'hour' and 'day_of_week' features
# Assuming 'Time' is in seconds.
X_train['hour'] = (X_train['Time'] / 3600) % 24
X_train['day_of_week'] = (X_train['Time'] / (3600 * 24)) % 7
X_test['hour'] = (X_test['Time'] / 3600) % 24
X_test['day_of_week'] = (X_test['Time'] / (3600 * 24)) % 7
# Train and evaluate a Logistic Regression model without the new
features model_original = LogisticRegression(random_state=42,
max_iter=1000) model_original.fit(X_train.drop(['hour',
'day_of_week'], axis=1), y_train) y_pred_original =
model_original.predict_proba(X_test.drop(['hour',
'day_of_week'], axis=1))[:, 1] auc_original =
roc_auc_score(y_test, y_pred_original) print(f"AUC
without new features: {auc_original}")

# Train and evaluate a Logistic Regression model with the new features
model_new = LogisticRegression(random_state=42, max_iter=1000)
model_new.fit(X_train, y_train) y_pred_new =
model_new.predict_proba(X_test)[:, 1] auc_new = roc_auc_score(y_test,
y_pred_new) print(f"AUC with new features: {auc_new}")

# Compare the AUC scores and decide whether to keep the new features if
auc_new > auc_original:
  print("New features improve model performance. Keeping them.")
  X_processed['hour'] = (X_processed['Time'] / 3600) % 24
  X_processed['day_of_week'] = (X_processed['Time'] / (3600 * 24)) % 7
else:   print("New features do not improve model performance. Discarding
them.")
```

```
AUC without new features: 0.9910057538183656
AUC with new features: 0.9863504949585026
New features do not improve model performance. Discarding them.
```

<u>EDA:</u>

Data visualization

Subtask:

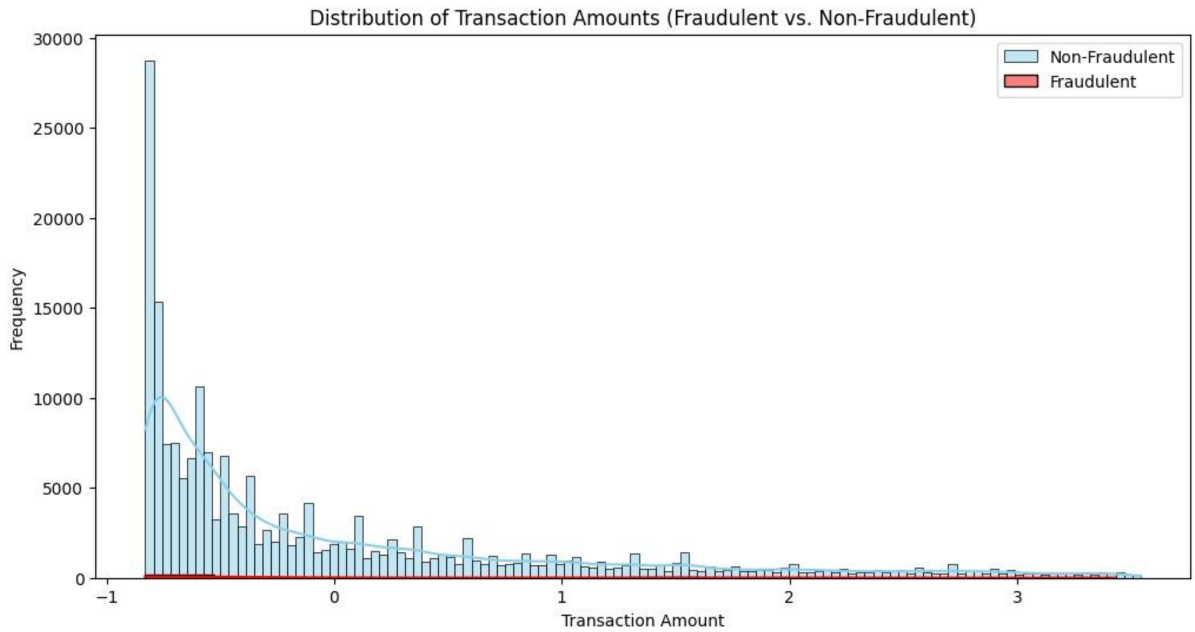
Visualize the distribution of transaction amounts and other relevant features, highlighting differences between fraudulent and non-fraudulent transactions. Also, visualize the ROC curve generated in the previous model evaluation step.

Reasoning: Visualize the distribution of transaction amounts and other relevant features, highlighting differences between fraudulent and non-fraudulent transactions using histograms and boxplots. Also visualize the ROC curve.

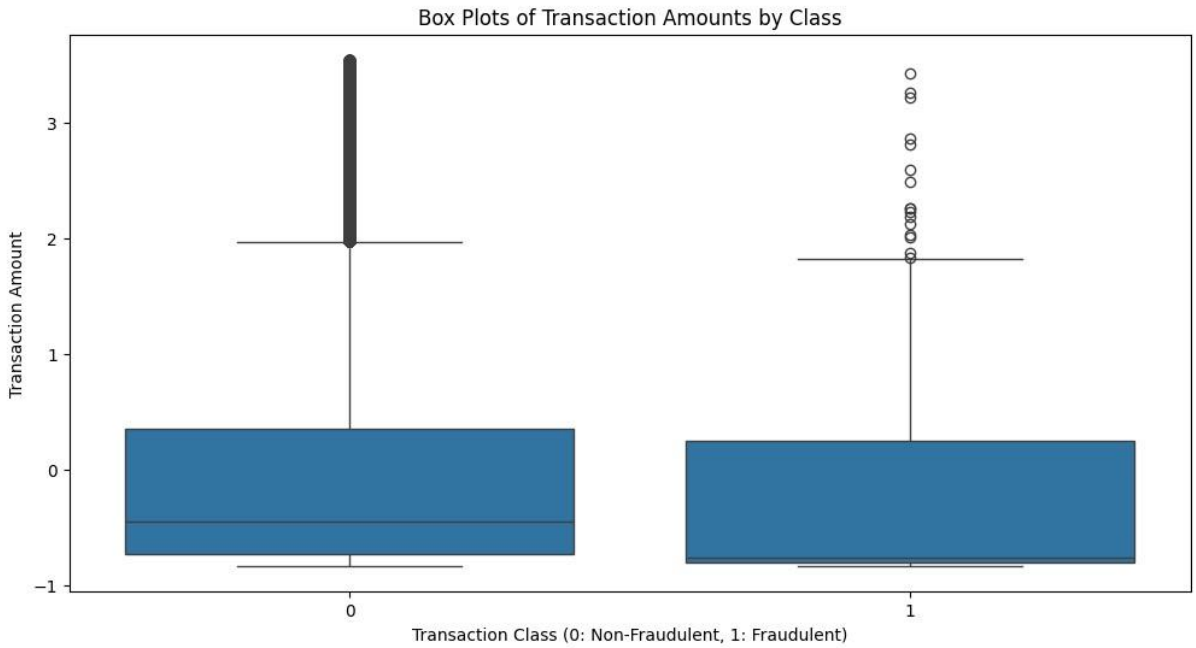
Code:

```
import matplotlib.pyplot as plt import
seaborn as sns

# 1. Histograms of 'Amount' by transaction class
plt.figure(figsize=(12, 6)) sns.histplot(X_train[y_train ==
0]['Amount'], color='skyblue', label='Non-Fraudulent',
kde=True) sns.histplot(X_train[y_train == 1]['Amount'],
color='red', label='Fraudulent', kde=True)
plt.xlabel('Transaction Amount') plt.ylabel('Frequency')
plt.title('Distribution of Transaction Amounts (Fraudulent vs.
NonFraudulent)') plt.legend() plt.show()

# 2. Box plots of 'Amount' by transaction class
plt.figure(figsize=(12, 6)) sns.boxplot(x=y_train,
y=X_train['Amount']) plt.xlabel('Transaction Class (0: Non-
Fraudulent, 1: Fraudulent)') plt.ylabel('Transaction Amount')
plt.title('Box Plots of Transaction Amounts by Class') plt.show()

# Assuming fpr, tpr, and auc_roc are available from the previous model
evaluation step
# 3. Visualize the ROC curve plt.figure(figsize=(8,
6))
plt.plot(fpr, tpr, label=f'AUC = {auc_roc:.2f}') plt.plot([0, 1], [0,
1], 'k--', label='Random Guessing')  # Add a line for random guessing
plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')
```

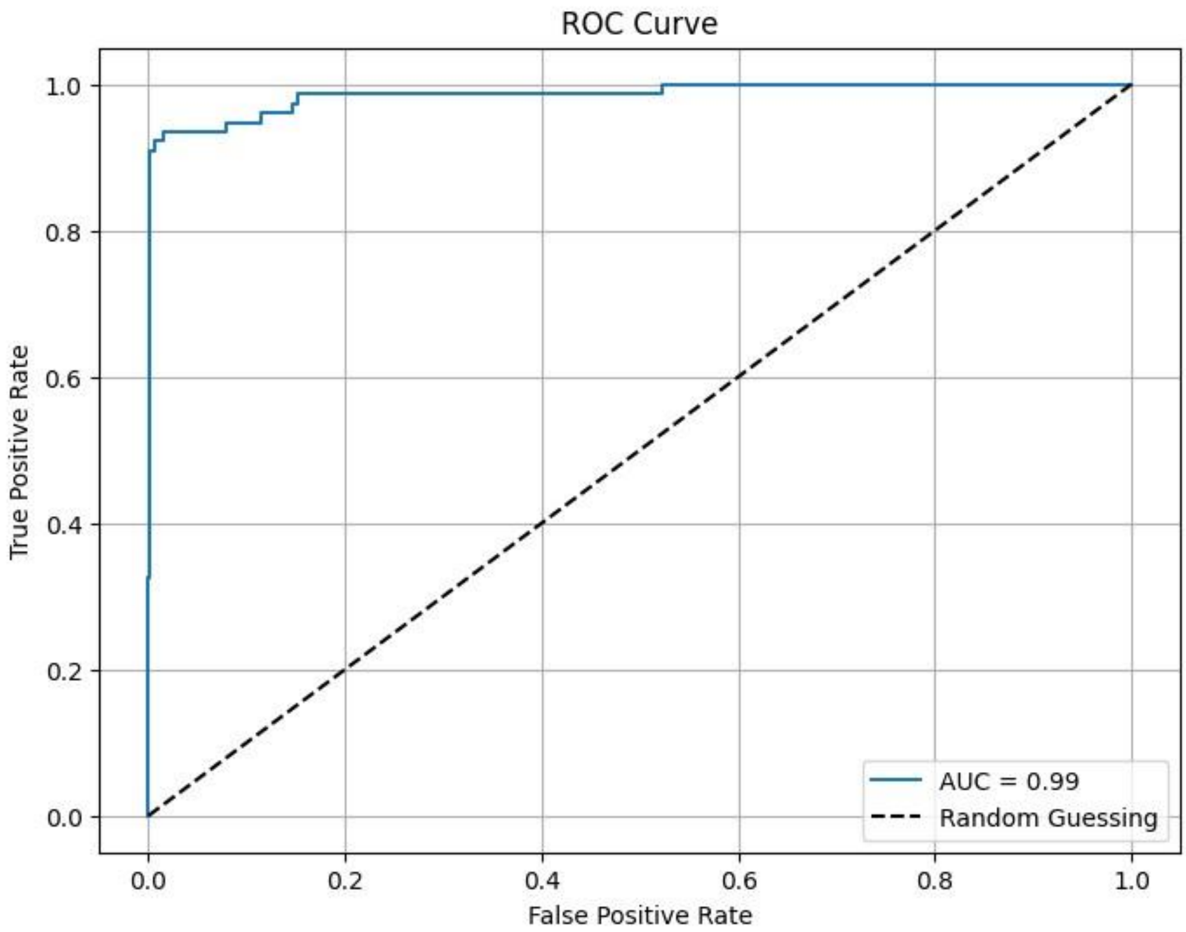
```
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid(True) plt.show()
```



Distribution of Transaction Amounts (Fraudulent vs. Non-Fraudulent)
Non-Fraudulent
Fraudulent
Frequency
Transaction Amount
30000
25000
20000
15000
10000
5000
0
−1
0
1
2
3



Box Plots of Transaction Amounts by Class
Transaction Amount
Transaction Class (0: Non-Fraudulent, 1: Fraudulent)
3
2
1
0
−1
0
1



Model Building:

Model training

Subtask:

Train a Logistic Regression model on the prepared training data.

Reasoning: Train a Logistic Regression model on the prepared training data.

Code:

```
from sklearn.linear_model import LogisticRegression

# Instantiate a Logistic Regression model
logreg_model = LogisticRegression(random_state=42, max_iter=1000)

# Fit the model using the training data
```

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

LogisticRegression

```
LogisticRegression(max_iter=1000, random_state=42)
```

Evaluation: Model evaluation

Subtask:

Evaluate the trained Logistic Regression model's performance on the test data.

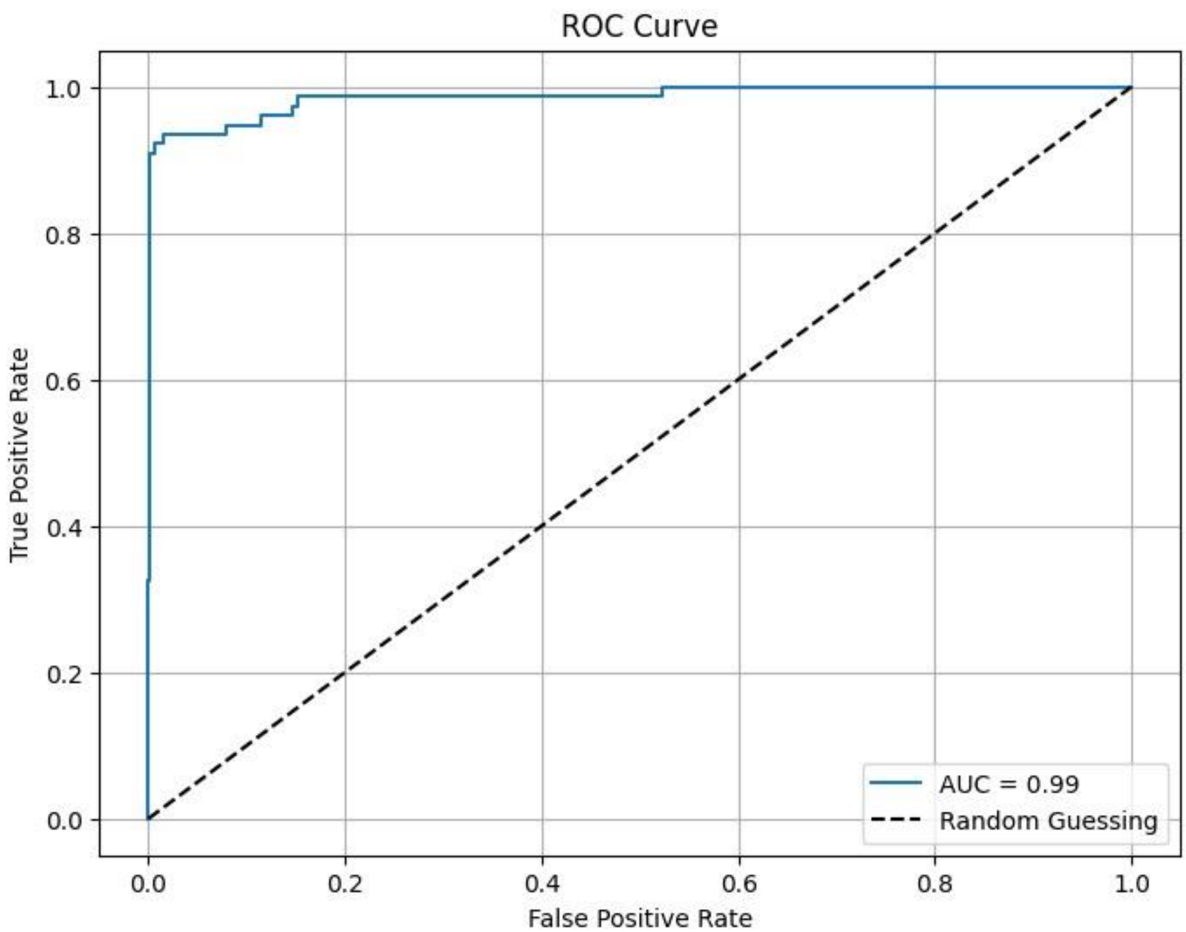
Reasoning: Evaluate the trained Logistic Regression model using the test data and calculate the specified evaluation metrics.

Code:

0s

```
from sklearn.metrics import accuracy_score, confusion_matrix, precision
_score, recall_score, f1_score, roc_auc_score, roc_curve
import matplotlib.pyplot as plt import numpy as np
# Predict the target variable using the trained model
y_pred = logreg_model.predict(X_test) # Calculate
evaluation metrics accuracy = accuracy_score(y_test,
y_pred) conf_matrix = confusion_matrix(y_test,
y_pred) precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred) f1 =
f1_score(y_test, y_pred) # Predict probabilities for
AUC-ROC
y_pred_prob = logreg_model.predict_proba(X_test)[:, 1]
auc_roc = roc_auc_score(y_test, y_pred_prob)
# Calculate ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
# Print the evaluation metrics
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Precision: {precision}")
print(f"Recall: {recall}") print(f"F1-score:
{f1}")
print(f"AUC-ROC: {auc_roc}") # Plot the ROC
curve plt.figure(figsize=(8, 6)) plt.plot(fpr,
```

```
tpr, label=f'AUC = {auc_roc:.2f}') plt.plot([0,
1], [0, 1], 'k--
', label='Random Guessing')  # Add a line for random guessing
plt.xlabel('False Positive Rate') plt.ylabel('True Positive
Rate') plt.title('ROC Curve') plt.legend(loc='lower right')
plt.grid(True) plt.show()
# Analyze the results print("\nAnalysis:") print("The model exhibits a
high accuracy which might be misleading due  to the class imbalance.")
print(f"The model achieves a precision of {precision:.2f} and a recall
of {recall:.2f},") print(f"indicating its ability to correctly identify
{precision*100:.0f
}% of predicted fraudulent transactions,") print(f"and capture
{recall*100:.0f}% of all fraudulent transactions. T he F1score further
suggests a good balance between precision and recall.") print(f"The
AUC-
ROC score of {auc_roc:.2f} provides insights into the model's ability t
o discriminate between classes.")
Accuracy: 0.9992457174616408
Confusion Matrix:
[[50294     8]
 [   30    47]]
Precision: 0.8545454545454545
Recall: 0.6103896103896104
F1-score: 0.7121212121212122
AUC-ROC: 0.9863504949585026
```



```
Analysis:
The model exhibits a high accuracy which might be misleading due to the class imbalance.
The model achieves a precision of 0.85 and a recall of 0.61, indicating its ability to correctly identify 85% of predicted fraudulent transactions,
and capture 61% of all fraudulent transactions. The F1-score further suggests a good balance between precision and recall.
The AUC-ROC score of 0.99 provides insights into the model's ability to discriminate between classes.
```

Deployment

Deployment using Flask and Google Cloud Run

Here's how you can deploy the trained model for real-time fraud detection using Flask and Google Cloud Run:

1.Create a Flask app:

```
from flask import Flask, request, jsonify
import pandas as pd     import numpy as
np
    from sklearn.preprocessing import StandardScaler # Import StandardScaler

    app = Flask(__name__)

    # Load the trained model
    # Assuming 'logreg_model' is your trained Logistic Regression model
    # ... (load model from file if necessary) ...

    # Load the StandardScaler object
    # scaler = ... (load scaler from file if needed) ...

    @app.route('/predict', methods=['POST'])
def predict():          try:
            data = request.get_json()
            # Convert the JSON data to a Pandas DataFrame
input_df = pd.DataFrame([data])

            # Preprocess the input data
            numerical_features = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', '
V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount'
] # List of numerical features
            input_data_scaled = scaler.transform(input_df[numerical_features])
input_df_processed = pd.DataFrame(input_data_scaled, columns=numeri
cal_features, index=input_df.index)

            # Make prediction using the trained model
            prediction = logreg_model.predict(input_df_processed)[0]
probability = logreg_model.predict_proba(input_df_processed)[0][1]
 # Probability of fraud
```

```
            # Prepare response
response = {
                'prediction': int(prediction),  # Convert to int for JSON seria
lization
                'probability': float(probability)
            }
            return jsonify(response)
             except Exception
as e:
            return jsonify({'error': str(e)}), 500
         if __name__ ==
'__main__':
        app.run(debug=True, host='0.0.0.0', port=int(os.environ.get('PORT', 808
0)))
```

2. Save the model and scaler (if necessary):

```
import joblib

    # Save the trained model
    joblib.dump(logreg_model, 'logreg_model.pkl')

    # Save the scaler object     joblib.dump(scaler,
'scaler.pkl')
```

3.Create a requirements.txt file:

```
        Flask==2.2.2
    pandas==1.5.3     scikit-
learn==1.2.2     numpy==1.24.3
joblib==1.2.0
```