ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJECT REPORT

BACHELOR OF COMPUTER APPLICATION

SEMESTER – II

STATISTICS AND DATASCIENCE

Credit Card Fraud Detection using Synthetic Dataset

BY

THUKIVAKAM CHATURYA

2411021240045

DEPARTMENT OF COMPUTER APPLICATION

ALLIANCE UNIVERSITY

CHANDPURA ANEKAL MAIN, ANEKAL

BANGALORE-562106

APRIL – 2025

📊 Credit Card Fraud Detection using Synthetic Dataset

NAME: THUKIVAKAM CHATURYA

Registration no.: 2411021240045

GITHUB LINK: https://github.com/chaturya0229/Credit-Card-Fraud-Detection-using-Synthetic-Dataset

📚 Table of Contents

Welcome to the project notebook! Below is the structured flow of analysis covered in this project.

📘 Project Title and Info

📊 Credit Card Fraud Detection using Synthetic Dataset

📃 **Course:** Introduction to Data Science
Semester: 2nd Semester – BCA
Project Type: Beginner-level Data Science & Machine Learning

Tools Used: Python, Pandas, Seaborn, Scikit-learn
Dataset: Bank Churners Dataset

🔰 Project Overview

This analysis aims to understand customer transaction patterns and detect fraudulent activity by:

- **Cleaning and transforming the dataset**
- **Performing exploratory data analysis (EDA)**
- **Using hypothesis testing**
- **Building classification models (Random Forest, Logistic Regression)**

🎯 Project Goal

- **Detect fraudulent transactions from historical data.**
- **Identify which features best distinguish fraud from non-fraud.**
- **Build and evaluate classification models to predict fraud accurately.**

❗ Challenges Faced

- **Class Imbalance: 251 fraud cases vs 4749 non-fraud.**
- **Irrelevant or redundant columns (e.g., Transaction_ID initially string).**
- **Skewed distributions in transaction values.**

📦 Import Libraries & Load Dataset

Used Python libraries: pandas, numpy, seaborn, matplotlib, sklearn, sci

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
# Machine Learning libraries
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv(r"C:\Users\thuki\Downloads\credit_card_fraud_synthetic
(1).csv")
df
```

```
     Transaction_ID  Transaction_Amount            Timestamp Location  \
0          TID00001             1873.33  2023-01-01 00:00:00   City_C
1          TID00002             4753.62  2023-01-01 01:00:00   City_D
2          TID00003             3660.24  2023-01-01 02:00:00   City_C
3          TID00004             2993.69  2023-01-01 03:00:00   City_D
4          TID00005              780.94  2023-01-01 04:00:00   City_D
...             ...                 ...                  ...      ...
4995       TID04996             4487.09  2023-07-28 03:00:00   City_A
4996       TID04997              597.79  2023-07-28 04:00:00   City_D
4997       TID04998             1639.89  2023-07-28 05:00:00   City_A
4998       TID04999             4078.91  2023-07-28 06:00:00   City_D
4999       TID05000             2986.96  2023-07-28 07:00:00   City_C

     Transaction_Type  Is_Fraudulent  Transaction_Frequency  \
0            Transfer              0                      6
1            Purchase              1                     14
2            Purchase              1                     12
3            Purchase              0                     27
4            Purchase              0                     26
...               ...            ...                    ...
4995         Transfer              1                      1
4996         Purchase              0                      4
4997         Transfer              0                      7
4998         Transfer              0                     11
4999         Transfer              0                     14

      Average_Transaction_Amount  Time_Since_Last_Transaction  \
0                         627.79                           82
1                         518.60                           26
2                        1294.19                           42
3                         787.40                           16
4                        1271.41                           22
...                          ...                          ...
4995                      360.88                           33
4996                     1287.41                           43
4997                      888.82                           12
4998                       86.25                           93
4999                     1102.54                           85

      Deviation_From_Avg
0                1245.54
```

```
1                4235.02
2                2366.05
3                2206.29
4                -490.47
...                  ...
4995             4126.21
4996             -689.62
4997              751.07
4998             3992.66
4999             1884.42

[5000 rows x 10 columns]
```

🖌️ Data Preprocessing

- Checked for missing values (none found).
- Encoded categorical columns using LabelEncoder.
- Converted columns like Transaction_Amount to integers.
- Standardized features using StandardScaler.

```
# Basic info about the dataset
print("Dataset Shape:", df.shape)
print("\nColumn Types:\n", df.dtypes)
```

```
Dataset Shape: (5000, 10)

Column Types:
 Transaction_ID                object
Transaction_Amount           float64
Timestamp                     object
Location                      object
Transaction_Type              object
Is_Fraudulent                  int64
Transaction_Frequency          int64
Average_Transaction_Amount   float64
Time_Since_Last_Transaction    int64
Deviation_From_Avg           float64
dtype: object
```

```
# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())
```

```
Missing Values:
```

```
 Transaction_ID                 0
Transaction_Amount              0
Timestamp                       0
Location                        0
Transaction_Type                0
Is_Fraudulent                   0
Transaction_Frequency           0
Average_Transaction_Amount      0
Time_Since_Last_Transaction     0
Deviation_From_Avg              0
dtype: int64
```

```
# Check class balance
print("\nClass Distribution:\n", df['Is_Fraudulent'].value_counts())
sns.countplot(x='Is_Fraudulent', data=df,hue="Is_Fraudulent")
plt.title("Fraud vs Non-Fraud Class Distribution")
plt.show()
```

```
Class Distribution:
 Is_Fraudulent
0    4749
1     251
Name: count, dtype: int64
```

```
# Change the dataType
df['Transaction_Amount'] = df['Transaction_Amount'].astype(int)
df['Average_Transaction_Amount'] =
df['Average_Transaction_Amount'].astype(int)
df['Deviation_From_Avg']=df['Deviation_From_Avg'].astype(int)

# encoding the categorical data into numeric data
lable=LabelEncoder()
df["Transaction_ID"]=lable.fit_transform(df["Transaction_ID"])
df["Timestamp"]=lable.fit_transform(df["Timestamp"])
df["Location"]=lable.fit_transform(df["Location"])
df["Transaction_Type"]=lable.fit_transform(df["Transaction_Type"])

# Separate features and target
X = df.drop(columns='Is_Fraudulent')
y = df['Is_Fraudulent']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

📊 Exploratory Data Analysis (EDA)

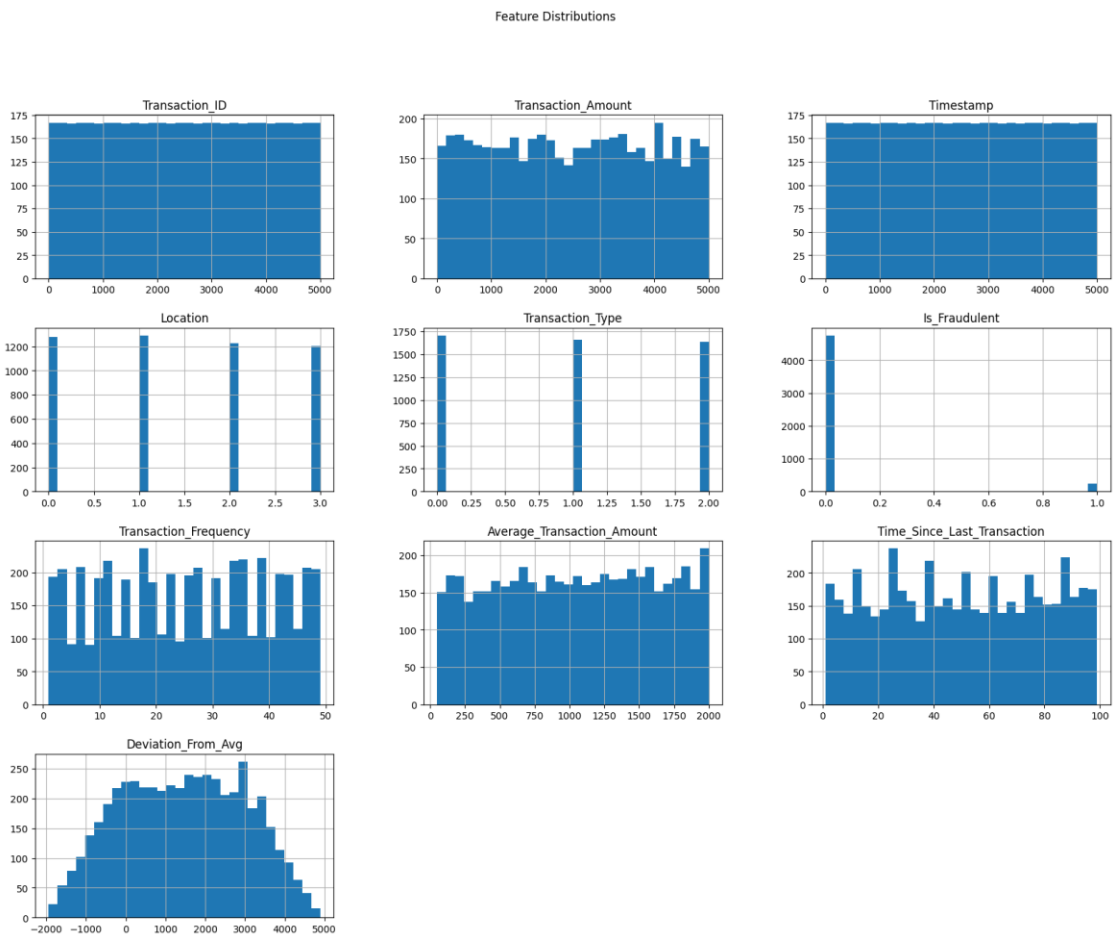
- Used describe() to summarize distributions.
- Plotted:
- Histograms of all features
- Correlation heatmap
- Boxplots to compare features against fraud labels
- Pairplots of selected features
- Found that features like Deviation_From_Avg and Transaction_Amount had wide ranges, while fraud cases were rare.

```
# Summary statistics
df.describe()
```

```
       Transaction_ID  Transaction_Amount    Timestamp     Location  \
count     5000.000000         5000.000000  5000.000000  5000.000000
mean      2499.500000         2484.168000  2499.500000     1.470800
std       1443.520003         1447.882434  1443.520003     1.114359
min          0.000000            1.000000     0.000000     0.000000
25%       1249.750000         1219.500000  1249.750000     0.000000
50%       2499.500000         2500.000000  2499.500000     1.000000
75%       3749.250000         3740.250000  3749.250000     2.000000
max       4999.000000         4998.000000  4999.000000     3.000000
```

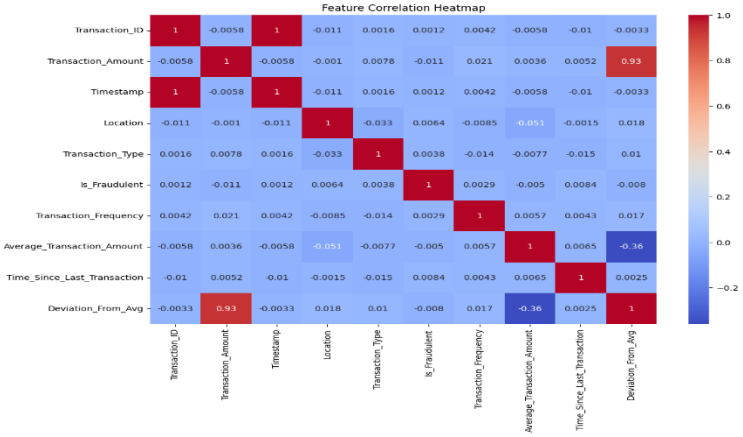
```
       Transaction_Type  Is_Fraudulent  Transaction_Frequency  \
count       5000.000000    5000.000000            5000.000000
mean           0.986400       0.050200              25.253200
std            0.817281       0.218379              14.105165
min            0.000000       0.000000               1.000000
25%            0.000000       0.000000              13.000000
50%            1.000000       0.000000              25.000000
75%            2.000000       0.000000              37.000000
max            2.000000       1.000000              49.000000

       Average_Transaction_Amount  Time_Since_Last_Transaction  \
count                 5000.000000                  5000.000000
mean                  1044.532800                    50.307000
std                    563.794668                    28.576089
min                     50.000000                     1.000000
25%                    563.000000                    26.000000
50%                   1048.000000                    50.000000
75%                   1530.000000                    76.000000
max                   1999.000000                    99.000000

       Deviation_From_Avg
count         5000.000000
mean          1439.355800
std           1551.589304
min          -1940.000000
25%            178.000000
50%           1459.000000
75%           2701.250000
max           4895.000000
```

```python
# Histogram of a few features
df.hist(bins=30, figsize=(20, 15))
plt.suptitle("Feature Distributions")
plt.show()
```



🔥 Heatmap - Correlation Matrix

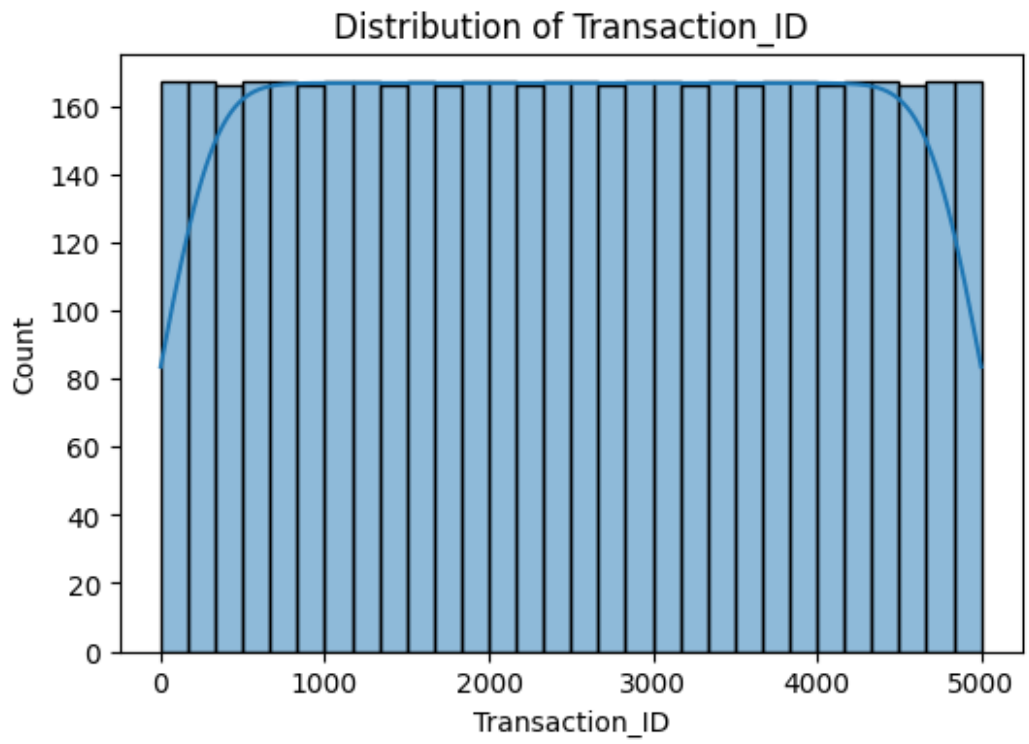
```
# Correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), cmap='coolwarm', annot=True)
plt.title("Feature Correlation Heatmap")
plt.show()
```

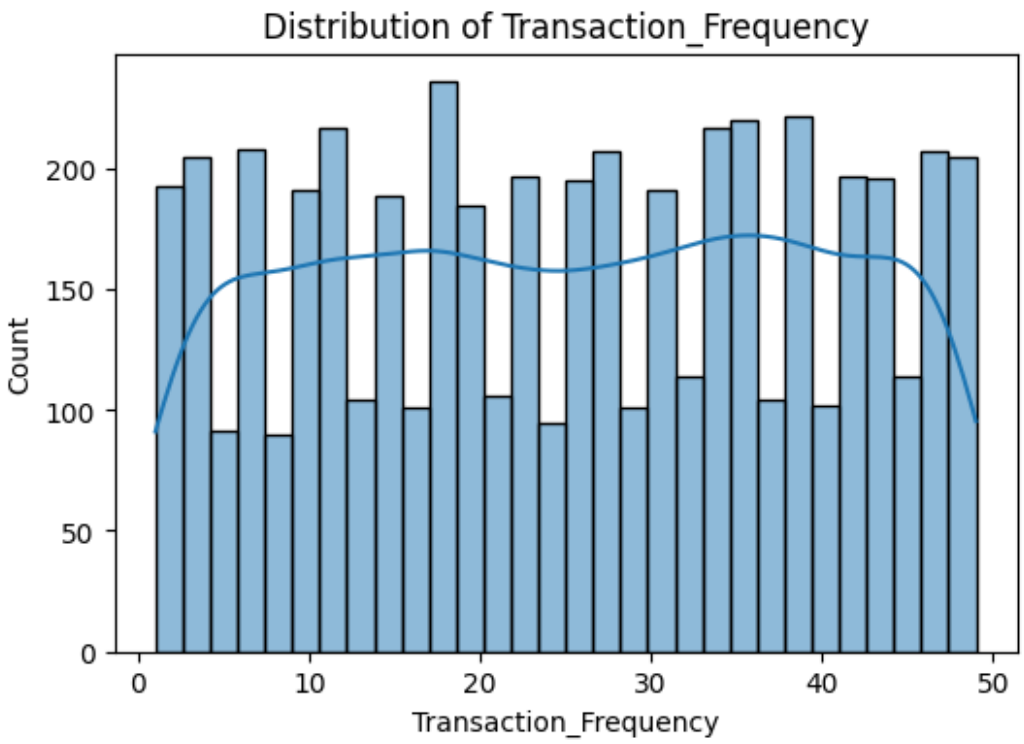


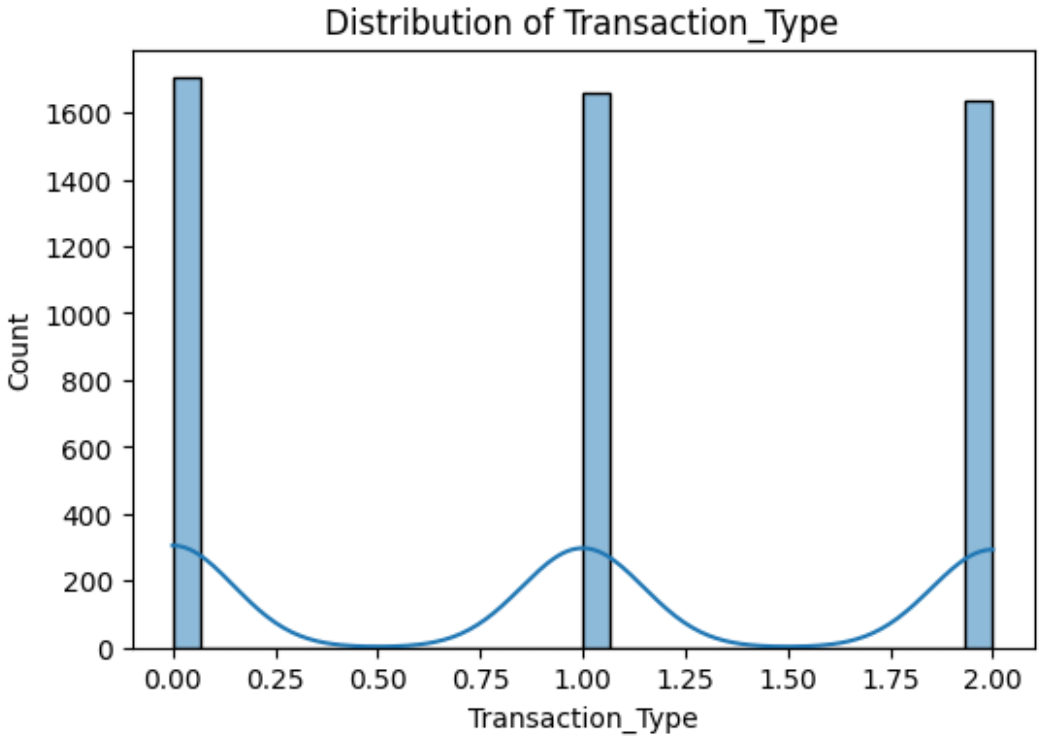
☝ Correlation matrix helps us to find the most target variable and it shows the high relationship between variables in positive or negative ranges which is useful for predictation, and it detect multicollinearity features.

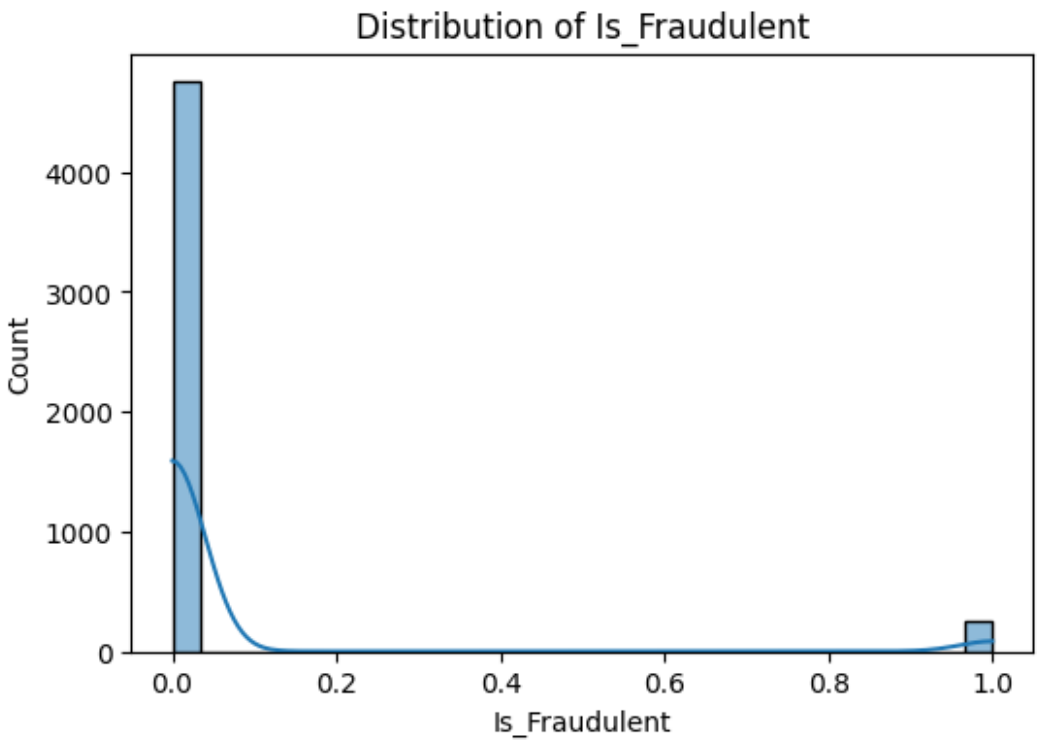
- +1 = Perfect positive correlation
- -1 = Perfect negative correlation
- 0 = No linear relationship

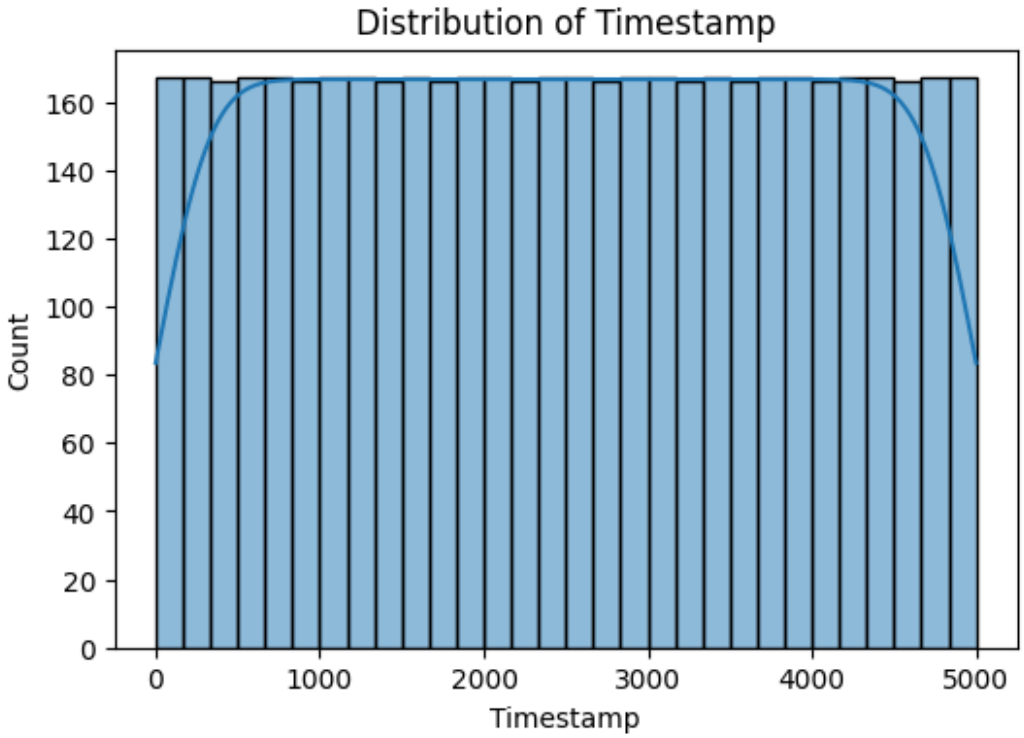
```
# Distribution of a few individual features
import random
features = random.sample(list(df.columns[:-1]), 5)  # random 5 features

for feature in features:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[feature], bins=30, kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()
```




Distribution of Transaction_Frequency
Count
Transaction_Frequency
0
50
100
150
200
0
10
20
30
40
50

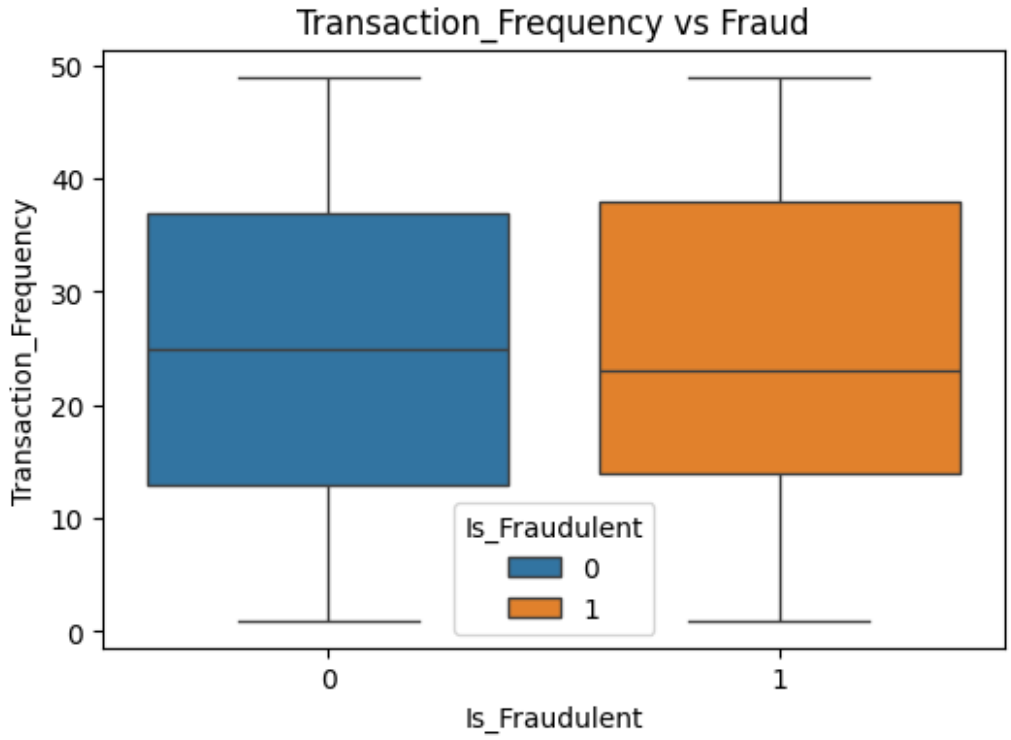

Distribution of Transaction_Type
Count
Transaction_Type
0
200
400
600
800
1000
1200
1400
1600
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

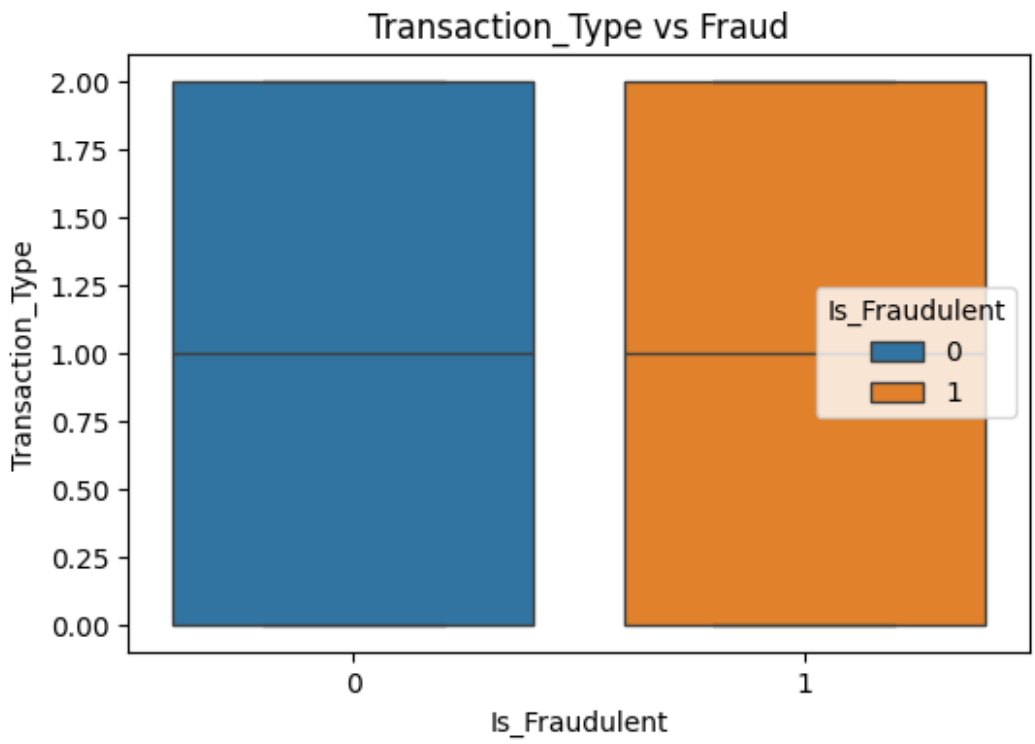


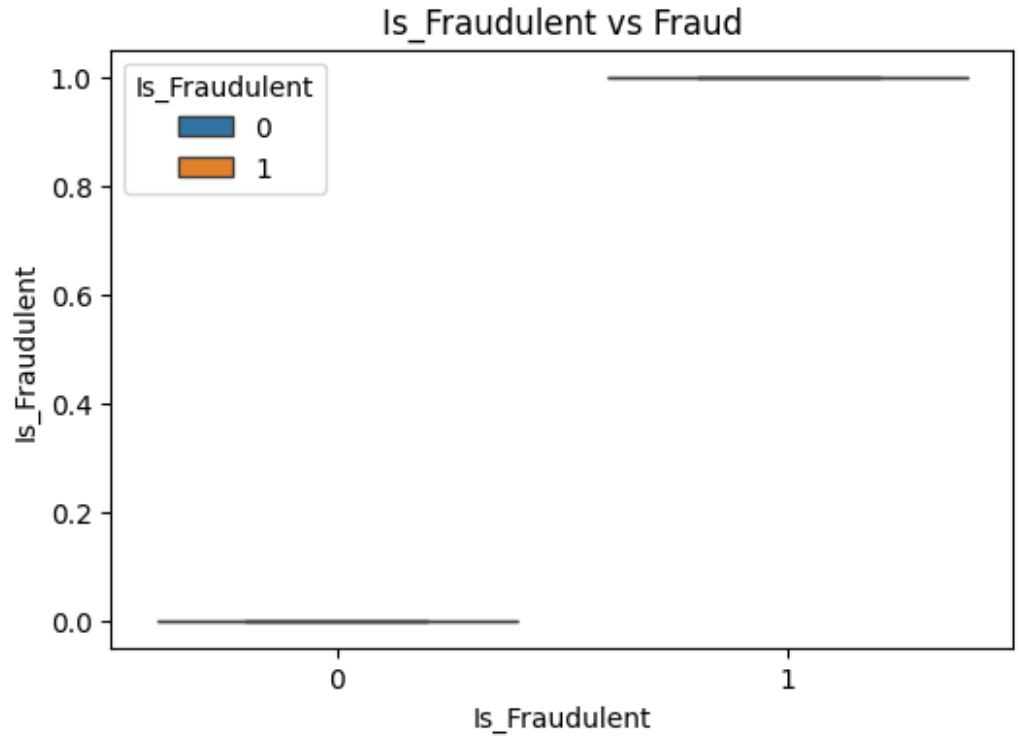



```
 # Boxplots of selected features vs fraud
for feature in features:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x='Is_Fraudulent', y=feature, data=df,hue="Is_Fraudulent")
```

```
plt.title(f'{feature} vs Fraud')
plt.show()
```




Transaction_Type vs Fraud
Transaction_Type
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
Is_Fraudulent
Is_Fraudulent
0
1


Is_Fraudulent vs Fraud
Is_Fraudulent
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Is_Fraudulent
Is_Fraudulent
0
1

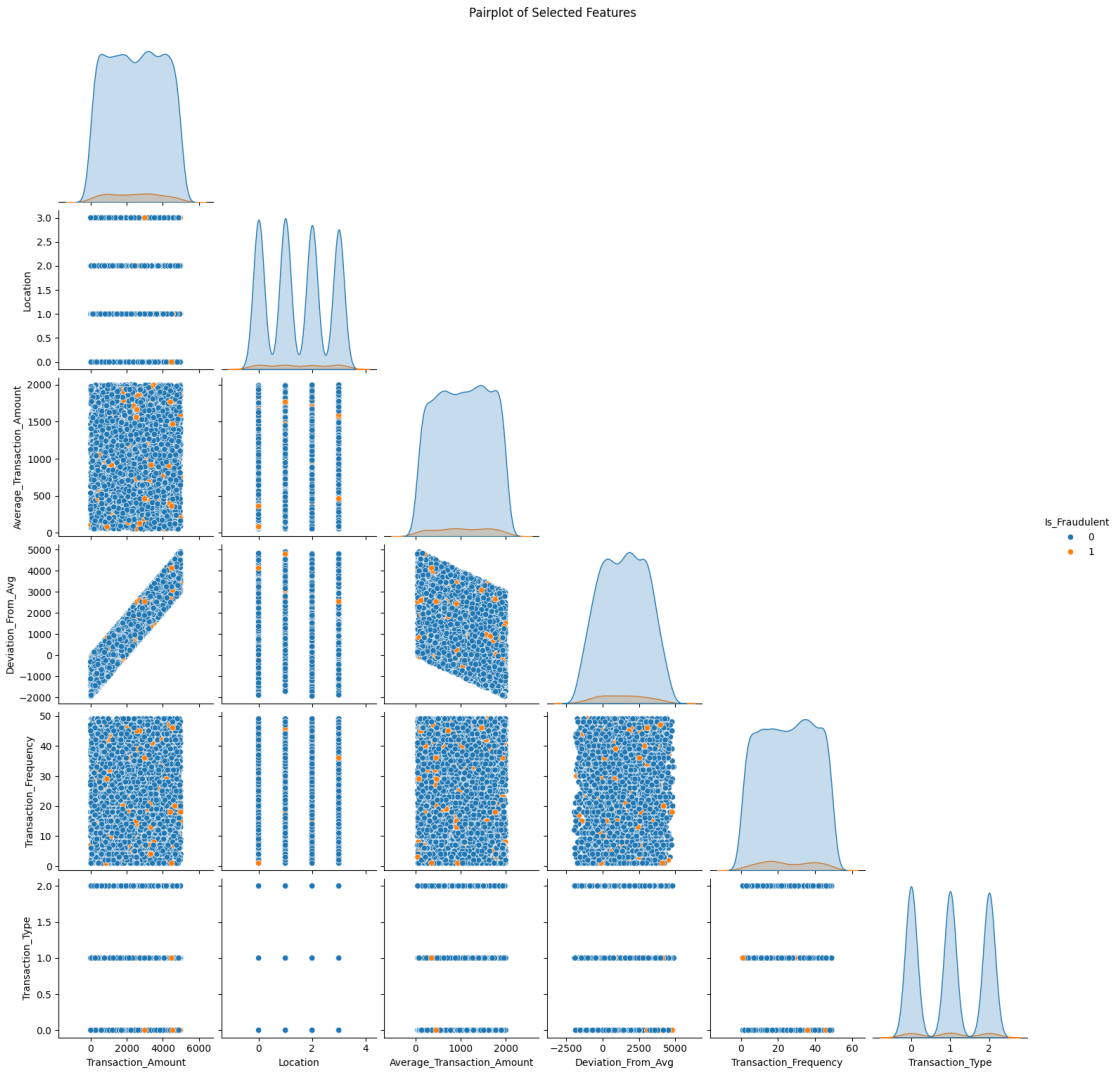

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 10 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Transaction_ID                5000 non-null   int32
 1   Transaction_Amount            5000 non-null   int32
 2   Timestamp                     5000 non-null   int32
 3   Location                      5000 non-null   int32
 4   Transaction_Type              5000 non-null   int32
 5   Is_Fraudulent                 5000 non-null   int64
 6   Transaction_Frequency         5000 non-null   int64
 7   Average_Transaction_Amount    5000 non-null   int32
 8   Time_Since_Last_Transaction   5000 non-null   int64
 9   Deviation_From_Avg            5000 non-null   int32
dtypes: int32(7), int64(3)
memory usage: 254.0 KB
```

```
# Pair plot for a small subset (if dataset isn't too large)
plt.figure(figsize=(8,6))
sns.pairplot(df[['Transaction_Amount', 'Location',
'Average_Transaction_Amount',
'Deviation_From_Avg','Transaction_Frequency','Transaction_Type','Is_Fraudulen
t']], hue='Is_Fraudulent', diag_kind='kde', corner=True)
```

```
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



👆 Pair plot is used to Create scatter plots between every pair of numerical features.

🎲 Probability & Hypothesis Testing

Here, We:

- Shapiro Test for normality
- T-test or ANOVA: Compare session duration by event type
- Chi-Square Test: Relationship between event type and product Category

```
from scipy.stats import ttest_ind
```

```python
# Let's test a few features
test_features = random.sample(list(df.columns[:-1]), 5)

for feature in test_features:
    fraud_vals = df[df['Is_Fraudulent'] == 1][feature]
    nonfraud_vals = df[df['Is_Fraudulent'] == 0][feature]

    t_stat, p_val = ttest_ind(fraud_vals, nonfraud_vals, equal_var=False)

    print(f"T-Test for {feature}:")
    print(f"  t-statistic = {t_stat:.4f}, p-value = {p_val:.4f}")
    if p_val < 0.05:
        print("  --> Statistically significant difference between fraud and
non-fraud.\n")
    else:
        print("  --> No significant difference.\n")
```

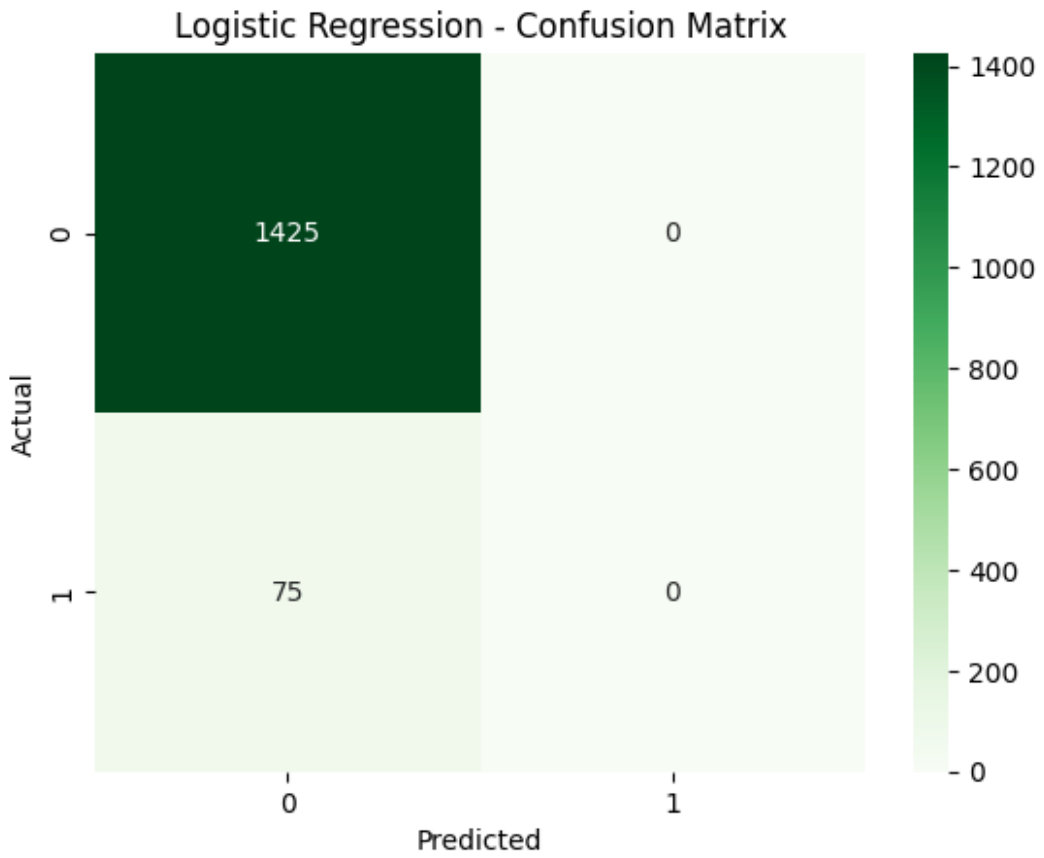
```
T-Test for Transaction_Amount:
  t-statistic = -0.7592, p-value = 0.4484
  --> No significant difference.

T-Test for Location:
  t-statistic = 0.4467, p-value = 0.6554
  --> No significant difference.

T-Test for Transaction_Frequency:
  t-statistic = 0.2049, p-value = 0.8378
  --> No significant difference.

T-Test for Transaction_Type:
  t-statistic = 0.2744, p-value = 0.7840
  --> No significant difference.

T-Test for Is_Fraudulent:
  t-statistic = inf, p-value = 0.0000
  --> Statistically significant difference between fraud and non-fraud.
```

```python
from sklearn.linear_model import LogisticRegression

# Train-test split (repeating in case you skipped previous step)
X = df.drop(columns='Is_Fraudulent')
y = df['Is_Fraudulent']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Scale features
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Logistic Regression
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train_scaled, y_train)

# Predictions
y_pred = logreg.predict(X_test_scaled)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))
# Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.95
Classification Report:
               precision    recall  f1-score   support

           0       0.95      1.00      0.97      1425
           1       0.00      0.00      0.00        75

    accuracy                           0.95      1500
   macro avg       0.47      0.50      0.49      1500
weighted avg       0.90      0.95      0.93      1500
```

```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Greens")
plt.title("Logistic Regression - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



📌 Final Conclusion

This project demonstrated how to:

- Clean and prepare a credit card transaction dataset
- Explore patterns and distributions using visualizations
- Build and evaluate a basic fraud detection model

📍 Key Insights:

- Fraud detection requires more than accuracy — focus on precision, recall, F1-score.
- Need to address class imbalance for better model performance.
- Features like Deviation_From_Avg could be useful but need further exploration.

References:

https://www.kaggle.com/datasets/saigeethasb/credit-card-fraud