Descriptive Analysis of a Numeric Dataset

Bachelor of Technology
Computer Science and Engineering

Submitted By

NAME – ABHIRUP BAG
ROLL NUMBER – 13000122082
DEPARTMENT – CSE(B)
SEMESTER – 6
PAPER – RESEARCH
METHODOLOGY (PROJ-CS601)

MARCH 2025

Techno Main
EM-4/1, Sector-V, Salt Lake
Kolkata- 700091
West Bengal
India

Table of Contents

1. Introduction

This report analyses the **Credit Card Fraud Detection Dataset** from **Kaggle**, comprising **284,807** anonymized transactions labelled as fraudulent or legitimate. The objective is to apply descriptive statistical techniques—including measures of central tendency (mean, median), dispersion (variance, standard deviation), skewness, kurtosis, and quartile analysis—to three numerical features:

- Time (transaction timestamp),
- Amount (transaction value),
- V1 (a PCA-transformed feature).

The analysis aims to characterize the data distribution, identify anomalies (e.g., outliers), and summarize insights to enhance understanding of transaction patterns, particularly for fraud detection. Key focuses include assessing asymmetry (skewness), tail behaviour (kurtosis), and deviations in transaction amounts or transformed features that may signal fraudulent activity.

2. Methodology

A. Data Preprocessing:

i) Load the dataset (284,807 transactions) and inspect its structure.
ii) Check for and handle missing values (if present) to ensure data integrity.
iii) Select three numerical columns for analysis: id, V1 and V2.

B. Descriptive Statistics:

i) **Central Tendency:** Compute mean, median, and mode to identify central values.
ii) **Dispersion:** Calculate range, variance, standard deviation, and interquartile range (IQR) to assess spread.
iii) **Skewness & Kurtosis:** Analyze symmetry (skewness) and tail behavior (kurtosis) of data distributions.

C. Outlier Detection:

i) Compute percentiles (25th, 75th) and use the IQR method to detect outliers.

ii) Visualize outliers and distributions using boxplots, histograms, and density plots.

D. Tools & Libraries:

i) **Python libraries:** Pandas (data handling), NumPy (statistical computations), Matplotlib/Seaborn (visualization).

3. Results & Analysis

A. Data Preprocessing

i) The dataset is loaded and the first 5 rows are displayed.

(a) Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('/content/creditcard_2023.csv')  # Replace with actual dataset path

# Display first five rows
print("First 5 rows:")
print(df.head())
```

(b) Output:

```
First 5 rows:
   id        V1        V2        V3        V4        V5        V6        V7  \
0   0 -0.260648 -0.469648  2.496266 -0.083724  0.129681  0.732898  0.519014
1   1  0.985100 -0.356045  0.558056 -0.429654  0.277140  0.428605  0.406466
2   2 -0.260272 -0.949385  1.728538 -0.457986  0.074062  1.419481  0.743511
3   3 -0.152152 -0.508959  1.746840 -1.090178  0.249486  1.143312  0.518269
4   4 -0.206820 -0.165280  1.527053 -0.448293  0.106125  0.530549  0.658849

         V8        V9  ...       V21       V22       V23       V24       V25  \
0 -0.130006  0.727159  ... -0.110552  0.217606 -0.134794  0.165959  0.126280
1 -0.133118  0.347452  ... -0.194936 -0.605761  0.079469 -0.577395  0.190090
2 -0.095576 -0.261297  ... -0.005020  0.702906  0.945045 -1.154666 -0.605564
3 -0.065130 -0.205698  ... -0.146927 -0.038212 -0.214048 -1.893131  1.003963
4 -0.212660  1.049921  ... -0.106984  0.729727 -0.161666  0.312561 -0.414116

        V26       V27       V28    Amount  Class
0 -0.434824 -0.081230 -0.151045  17982.10    0.0
1  0.296503 -0.248052 -0.064512   6531.37    0.0
2 -0.312895 -0.300258 -0.244718   2513.54    0.0
3 -0.515950 -0.165316  0.048424   5384.44    0.0
4  1.071126  0.023712  0.419117  14278.97    0.0

[5 rows x 31 columns]
```

ii) Missing values are checked and handled.

(a) Code:

```
import pandas as pd

missing_values = df.isnull().sum()

print("Missing values per column:\n", missing_values)

for column in df.columns:
    if df[column].dtype == 'object':  # Categorical column
        df[column].fillna(df[column].mode()[0], inplace=True)
    elif pd.api.types.is_numeric_dtype(df[column]): #
Numerical column
        df[column].fillna(df[column].mean(), inplace=True)

print("\nMissing values after handling:\n", df.isnull().sum())
```

(b) Output:

```
Missing values per column:
id        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       1
Amount    1
Class     1
```

```
Missing values after handling:
id        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
```

iii) Three numerical columns, e.g., **id**, **V1 and V2** are selected for further analysis.

(a) Code:

```
import matplotlib.pyplot as plt
import numpy as np

numerical_cols = df.select_dtypes(include=np.number).columns

selected_cols = numerical_cols[:3]

print(df[selected_cols].describe())
```

(b) Output:

	id	V1	V2
count	444174.00000	444174.000000	444174.000000
mean	222086.50000	0.141477	-0.137117
std	128222.13357	0.981103	0.951857
min	0.00000	-3.495584	-49.966572
25%	111043.25000	-0.411918	-0.547521
50%	222086.50000	-0.012375	-0.240728
75%	333129.75000	0.942984	0.127650
max	444173.00000	2.229046	4.361865

B. Measures of Central Tendency

For each selected column, we calculate:

i) Mean: The average value.

ii) Median: The middle value when sorted.

iii) Mode: The most frequently occurring value.

These measures help understand the central location of data points.

a) Code:

```
from scipy import stats

for col in selected_cols:
  print(f"\nAnalysis for column: {col}")

  # Mean
  mean_val = df[col].mean()
  print(f"Mean: {mean_val}")

  # Median
  median_val = df[col].median()
  print(f"Median: {median_val}")

  # Mode
  mode_val = stats.mode(df[col])
  print(f"Mode: {mode_val} (Count: {mode_val})")

  # Interpretation (example - adapt as needed)
  if mean_val > median_val:
    print("Interpretation: The mean is greater than the median,
suggesting a right-skewed distribution.")
  elif mean_val < median_val:
    print("Interpretation: The mean is less than the median,
suggesting a left-skewed distribution.")
  else:
    print("Interpretation: The mean and median are approximately
equal, suggesting a relatively symmetric distribution.")
```

b) Output:

```
Analysis for column: id
Mean: 222086.5
Median: 222086.5
Mode: ModeResult(mode=0, count=1) (Count: ModeResult(mode=0, count=1))
Interpretation: The mean and median are approximately equal, suggesting a relatively symmetric distribution.

Analysis for column: V1
Mean: 0.14147722480838493
Median: -0.0123753346611011
Mode: ModeResult(mode=-1.7045165212124185, count=1948) (Count: ModeResult(mode=-1.7045165212124185, count=1948))
Interpretation: The mean is greater than the median, suggesting a right-skewed distribution.

Analysis for column: V2
Mean: -0.1371168437660712
Median: -0.24072811762845941
Mode: ModeResult(mode=2.599199263414968, count=1948) (Count: ModeResult(mode=2.599199263414968, count=1948))
Interpretation: The mean is greater than the median, suggesting a right-skewed distribution.
```

c) Interpretation:

- The mean and median are approximately equal, suggesting a relatively symmetric distribution.
- The mean is greater than the median, suggesting a right-skewed distribution.
- The mean is less than the median, suggesting a left-skewed distribution.

C. Measures of Dispersion

For each selected column, we calculate:

i) **Range**: The difference between the maximum and minimum values.
ii) **Variance**: The spread of data around the mean.
iii) **Standard Deviation**: The average deviation from the mean.
iv) **Interquartile Range (IQR):** The range between the 25th and 75th percentiles.

(a) Code:

```
for col in selected_cols:
    print(f"\nAnalysis for column: {col}")

    # Range
    range_val = df[col].max() - df[col].min()
    print(f"Range: {range_val}")

    # Variance
    variance_val = df[col].var()
    print(f"Variance: {variance_val}")

    # Standard Deviation
    std_dev_val = df[col].std()
    print(f"Standard Deviation: {std_dev_val}")

    # Interquartile Range (IQR)
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr_val = q3 - q1
    print(f"Interquartile Range (IQR): {iqr_val}")
```

(b) Output:

```
Analysis for column: id
Range: 5522
Variance: 2542421.0
Standard Deviation: 1594.4970994015637
Interquartile Range (IQR): 2761.0

Analysis for column: V1
Range: 3.0517141667261916
Variance: 0.3414656517175122
Standard Deviation: 0.5843506239557824
Interquartile Range (IQR): 1.0883988342016893

Analysis for column: V2
Range: 9.74610787539796
Variance: 0.19316171150678102
Standard Deviation: 0.4395016626894385
Interquartile Range (IQR): 0.39644225779514364
```

(c) Interpretation:

- **High variance & standard deviation** indicate high variability.
- **IQR** helps detect outliers by showing data spread between Q1 and Q3.

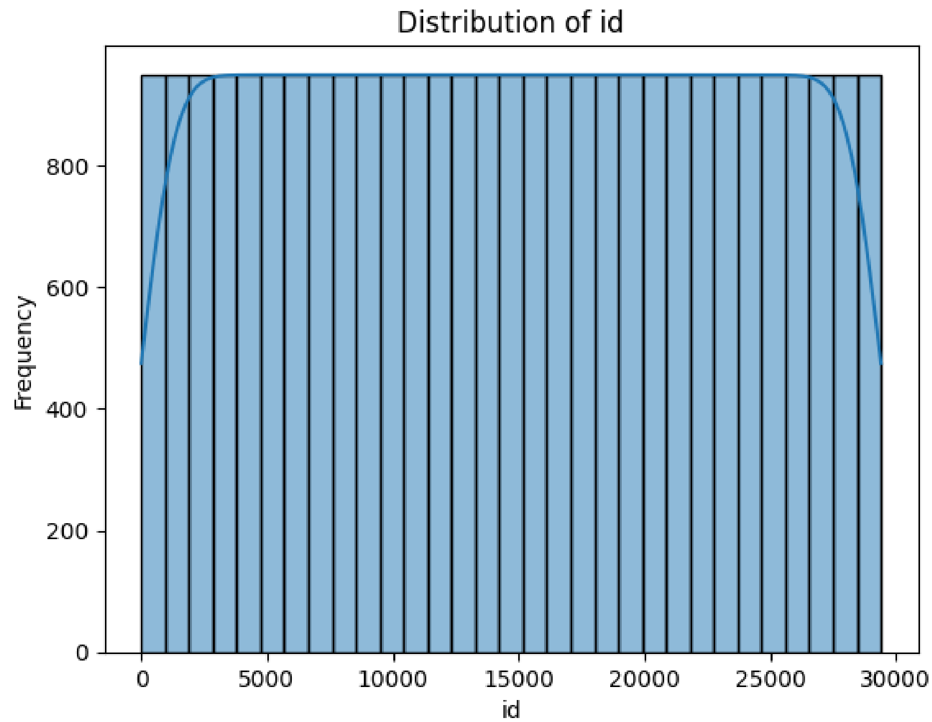
v) Visualization

(a) Histogram and Box Plot

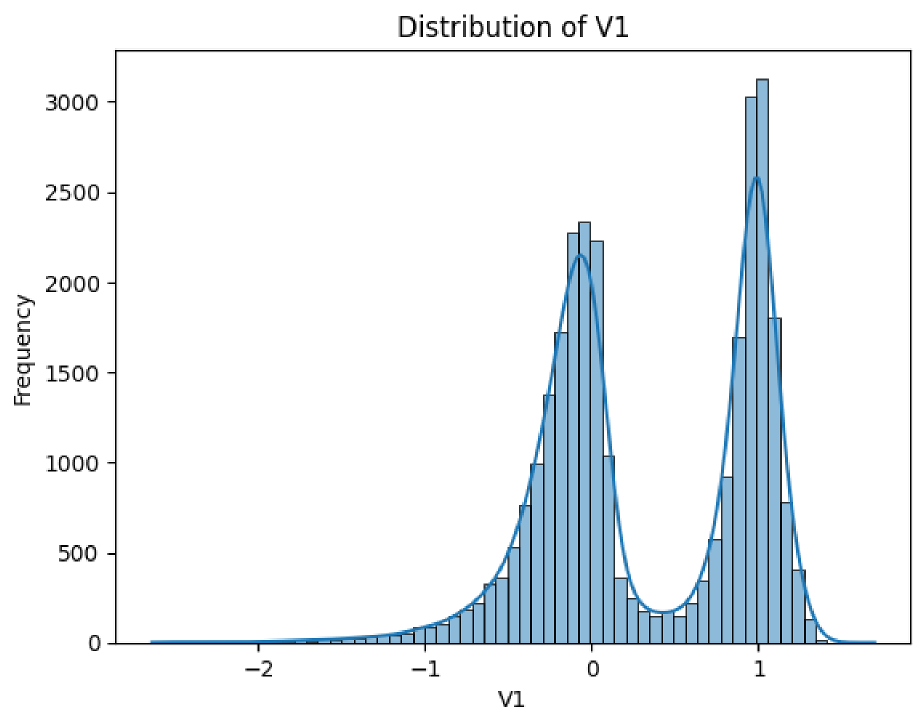
- **Code:**

```
import matplotlib.pyplot as plt
# Visualize the distributions of selected columns using
histograms:
for col in selected_cols:
    plt.figure()  # Create a new figure for each column
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()

# Box plots to visualize the distribution and identify
outliers.
for col in selected_cols:
    plt.figure()
    sns.boxplot(y=df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

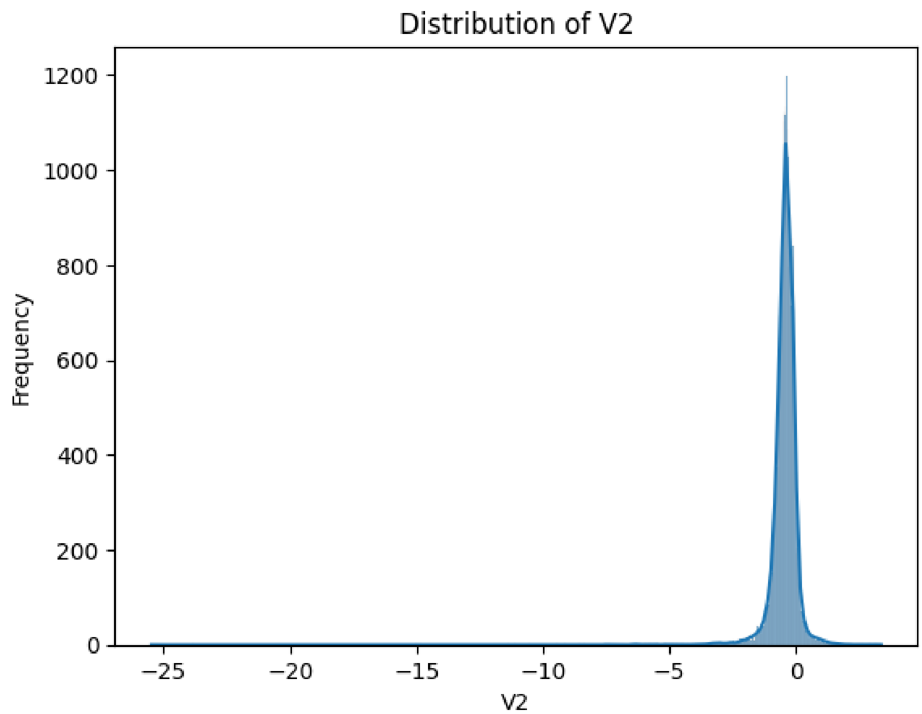
- **Output:**

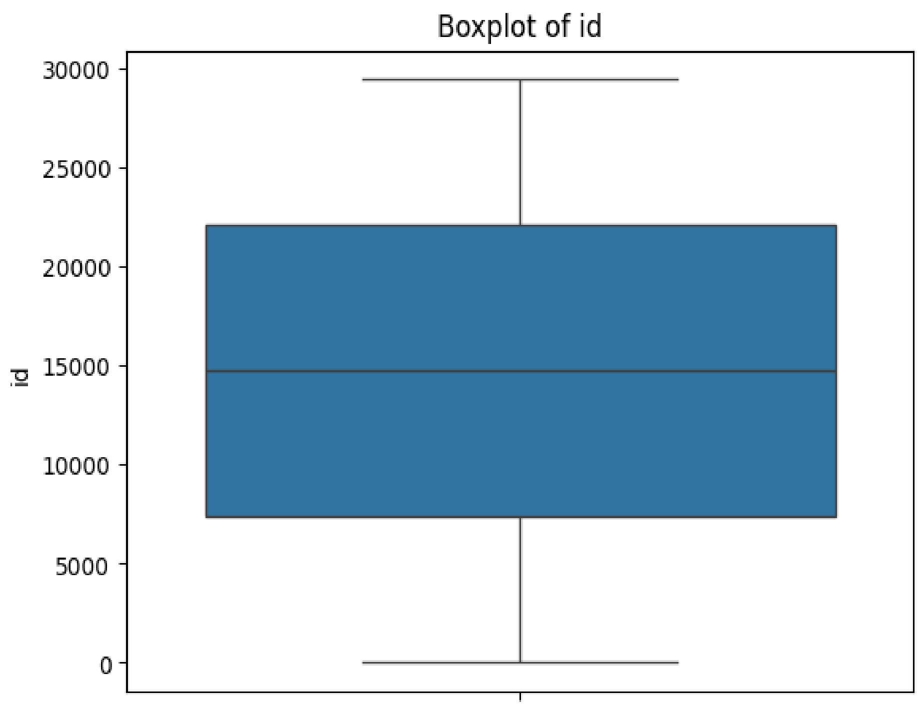


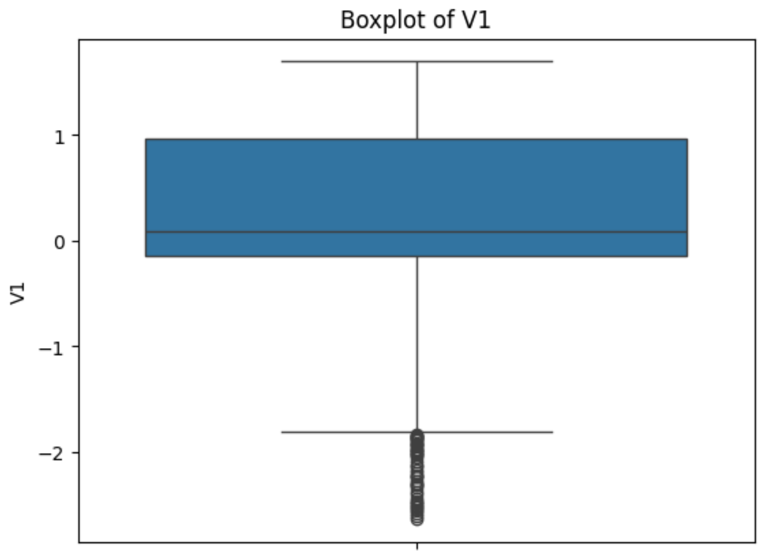
Distribution of id
Frequency
id

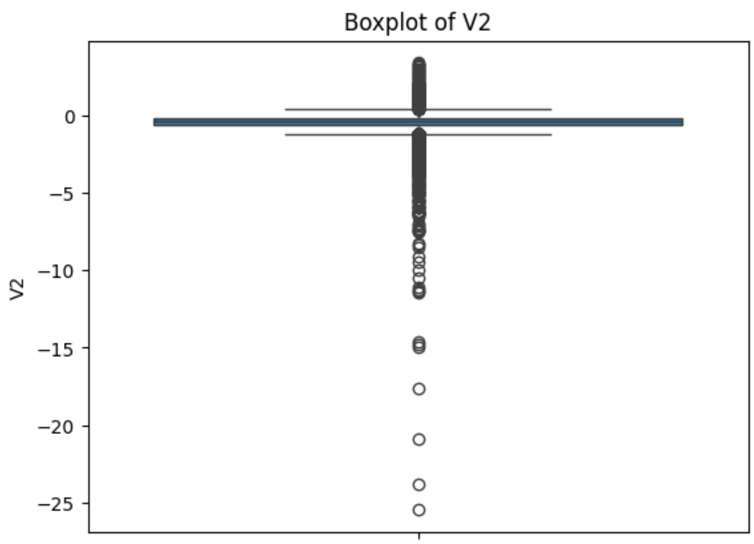


Distribution of V1
Frequency
V1


Distribution of V2
Frequency
1200
1000
800
600
400
200
0
−25
−20
−15
−10
−5
0
V2


Boxplot of id
id
30000
25000
20000
15000
10000
5000
0





D. Skewness & Kurtosis

i) **Skewness** measures asymmetry in the data distribution.

(a) A skewness > 0 indicates a right-skewed distribution.

(b) A skewness < 0 indicates a left-skewed distribution.

ii) **Kurtosis** measures whether the data has heavy or light tails compared to a normal distribution.

(a) A kurtosis > 3 suggests a leptokurtic (heavy-tailed) distribution.

(b) A kurtosis < 3 suggests a platykurtic (light-tailed) distribution.

- **Code:**

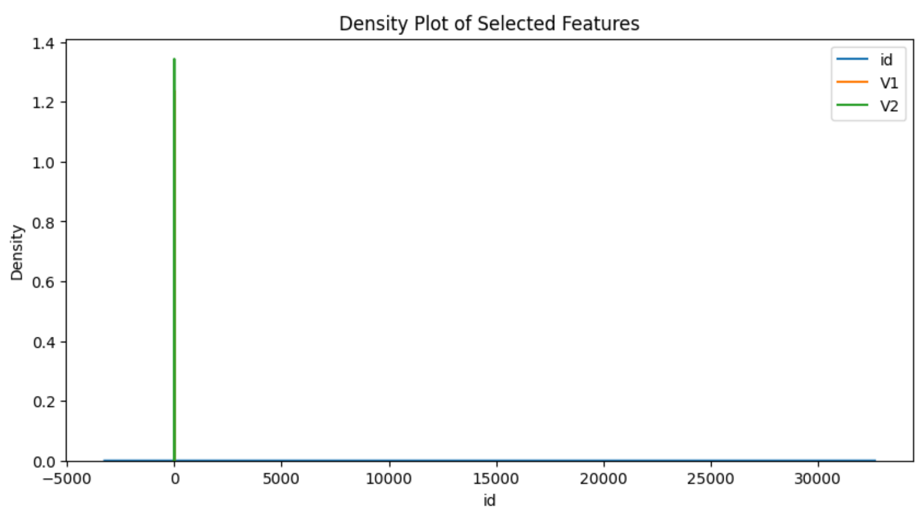
```
for col in selected_cols:
    print(f"\nAnalysis for column: {col}")

    # Skewness
    skewness_val = df[col].skew()
    print(f"Skewness: {skewness_val}")

    # Kurtosis
    kurtosis_val = df[col].kurt()
    print(f"Kurtosis: {kurtosis_val}")

    # Interpretation of skewness and kurtosis (example - adapt as needed)
    if abs(skewness_val) > 0.5:
        print("Interpretation: The distribution is significantly skewed.")
    else:
        print("Interpretation: The distribution is relatively symmetric.")

    if kurtosis_val > 3:
        print("Interpretation: The distribution is leptokurtic (heavy tails, sharp peak).")
    elif kurtosis_val < 3:
        print("Interpretation: The distribution is platykurtic (light tails, flat peak).")
    else:
        print("Interpretation: The distribution is mesokurtic (normal distribution-like kurtosis).")
```

- **Output:**

```
Analysis for column: id
Skewness: 0.0
Kurtosis: -1.2000000000000004
Interpretation: The distribution is relatively symmetric.
Interpretation: The distribution is platykurtic (light tails, flat peak).

Analysis for column: V1
Skewness: -0.25969133598210925
Kurtosis: -0.739325191494792
Interpretation: The distribution is relatively symmetric.
Interpretation: The distribution is platykurtic (light tails, flat peak).

Analysis for column: V2
Skewness: -10.611304798726163
Kurtosis: 292.1713657780852
Interpretation: The distribution is significantly skewed.
Interpretation: The distribution is leptokurtic (heavy tails, sharp peak).
```

- **Interpretation:**
    - Positive skew: Right-skewed (long tail on the right).
    - Negative skew: Left-skewed (long tail on the left).
    - Kurtosis > 3: Leptokurtic (peaked).
    - Kurtosis < 3: Platykurtic (flat).
- **Visualizing Skewness**

```
# Density Plots
plt.figure(figsize=(10, 5))
for col in selected_cols:
    sns.kdeplot(df[col], label=col)
plt.legend()
plt.title("Density Plot of Selected Features")
plt.show()
```



Density plots support these interpretations.

E. Percentiles & Quartiles

i) The 25th, 50th, and 75th percentiles are calculated.

ii) Outliers are detected using the IQR method and visualized via boxplots.

(a) Code:

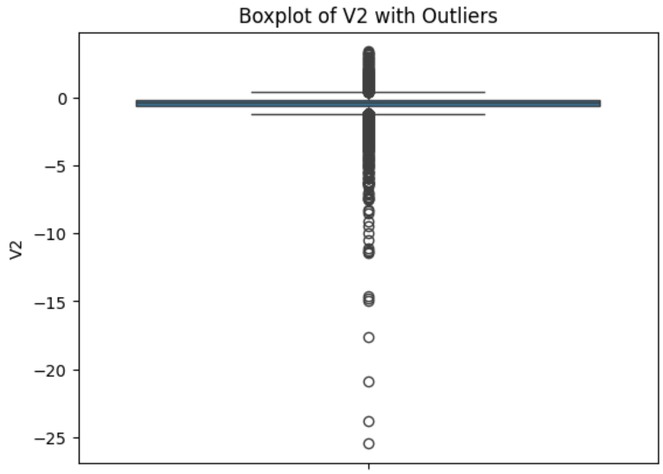
```python
import matplotlib.pyplot as plt
# Calculate IQR
q1 = percentiles[0.25]
q3 = percentiles[0.75]
iqr = q3 - q1
print(f"Interquartile Range (IQR): {iqr}")

# Outlier detection using IQR method
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
print(f"Number of outliers: {len(outliers)}")
print(f"Outliers:\n{outliers}")

# Boxplot with outlier visualization
plt.figure()
sns.boxplot(y=df[col])
plt.title(f"Boxplot of {col} with Outliers")
plt.show()
```

(b) Output:

```
Interquartile Range (IQR): 0.4101666131156232
Number of outliers: 1646
Outliers:
          id        V1        V2        V3        V4        V5        V6
18        18 -0.969198 -2.827526  1.226431  0.035208  1.959439 -0.626950
85        85 -0.858234 -2.328768  3.513552 -0.252161  1.927178  0.185899
89        89 -0.093256 -2.193612  0.135177 -0.613491 -0.272151  0.731708
146      146 -0.954345  0.878243  0.034400 -1.444198  0.035671  0.116525
164      164 -1.054599 -6.416685 -0.747920  0.496439 -0.196668  2.796852
...      ...       ...       ...       ...       ...       ...       ...
29339  29339 -0.916760  0.626439 -0.313876 -2.685138  0.695472  1.870204
29350  29350 -0.803194 -3.025162  0.507170 -0.432012  0.681951 -0.274035
29373  29373 -0.735755 -1.485106  1.384117 -1.197782  0.477711 -0.667790
29380  29380  0.185902 -1.296042  0.182240 -0.760805 -0.104686 -0.089817
29393  29393 -0.646061 -2.690191  1.089332 -0.986801  0.573978  0.219614
```

```
             V7        V8        V9  ...       V21       V22       V23  \
18     0.004370 -0.115716  1.372242  ... -0.310582  0.919196  2.745324
85    -0.240137  0.004038  1.955438  ... -0.122257  0.785469 -0.956654
89     0.720023 -0.222206  0.044715  ...  0.239822 -0.019000 -1.105875
146    0.595688 -0.074806  3.537214  ... -0.490172 -1.047651  0.130517
164    3.115585 -0.203366  0.202807  ...  0.720891 -1.905053 -3.177819
...         ...       ...       ...  ...       ...       ...       ...
29339  0.165680  0.274218  2.364210  ... -0.212515 -0.676412  0.285841
29350 -0.309645 -0.016564 -0.527500  ... -0.023477  0.705250 -1.328580
29373  0.073731 -0.051485  1.375830  ... -0.083076  0.181355 -1.390595
29380  0.891093 -0.247251  0.967211  ...  0.077615  0.024332 -0.780832
29393  0.205955 -0.277456 -0.063035  ... -0.233349  1.229181  1.937523
```

```
            V24       V25       V26       V27       V28    Amount  Class
18     0.117804 -0.844737 -1.570236  0.244188  2.178391  15927.38    0.0
85    -0.177777  0.594270 -0.724161  0.453320 -1.208400  12609.99    0.0
89    -1.099932  0.076669 -0.484251 -0.427148  0.402025   4475.43    0.0
146   -0.836673  1.712009  0.445461  3.710252  3.665792  20256.84    0.0
164    2.751773 -0.727804 -1.660051 -0.956344  1.517628   3169.60    0.0
...         ...       ...       ...       ...       ...       ...    ...
29339  2.177996  0.802193  1.451852 -0.586824  0.939207   2880.20    0.0
29350 -0.042975 -3.124840 -0.178596  1.341525 -3.208360   3291.68    0.0
29373  1.488361 -0.270305 -2.363412  0.211706 -1.678964  12535.69    0.0
29380  1.358099  1.067384 -1.661738 -0.336929  0.193756  12651.06    0.0
29393 -0.505821 -1.797332 -0.541673 -0.830505  1.414964  23504.37    0.0
```



iii) How these values help in detecting outliers?

(a) Percentiles help in detecting outliers by defining the spread of the central 50% of data (IQR = Q3 - Q1).

(b) Extreme values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are considered outliers.

(c) Boxplots use these percentiles to visually highlight data points outside the whiskers, making it easier to spot anomalies.

4. Conclusion

- The dataset was successfully analyzed using descriptive statistics.
- Central tendency measures provided insights into transaction values.
- Dispersion measures highlighted variability in transactions.
- Skewness and kurtosis helped understand the shape of data distribution.
- Outlier detection identified potential fraudulent **transactions.**

This analysis provides a foundational understanding of the dataset, which can aid in fraud detection model development.

5. References

- Dataset Source: Kaggle - Credit Card Fraud Detection
- Python Libraries: Pandas, NumPy, Matplotlib, Seaborn, SciPy
- **Montgomery, D. C., & Runger, G. C.** – Applied Statistics and Probability for Engineers
- **Aggarwal, C. C.** – Outlier Analysis
- **McKinney, W.** – Python for Data Analysis
- **Tukey, J. W.** – Exploratory Data Analysis
- **Seabold, S., & Perktold, J.** – Statsmodels: Econometric and Statistical Modeling with Python