**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** 3](#_Toc192360813)

[2. **Methodology** 3](#_Toc192360814)

[A. Data Preprocessing: 3](#_Toc192360815)

[B. Descriptive Statistics: 3](#_Toc192360816)

[C. Outlier Detection: 4](#_Toc192360817)

[D. Tools & Libraries: 4](#_Toc192360818)

[3. **Results & Analysis** 4](#_Toc192360819)

[A. Data Preprocessing 4](#_Toc192360820)

[B. Measures of Central Tendency 7](#_Toc192360821)

[C. Measures of Dispersion 8](#_Toc192360822)

[D. Skewness & Kurtosis 13](#_Toc192360823)

[E. Percentiles & Quartiles 15](#_Toc192360824)

[4. **Conclusion** 18](#_Toc192360825)

[5. **References** 18](#_Toc192360826)

# 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.

# Methodology

## Data Preprocessing:

* + 1. Load the dataset (284,807 transactions) and inspect its structure.
    2. Check for and handle missing values (if present) to ensure data integrity.
    3. Select three numerical columns for analysis: id, V1 and V2.

## Descriptive Statistics:

* + 1. **Central Tendency:** Compute mean, median, and mode to identify central values.
    2. **Dispersion:** Calculate range, variance, standard deviation, and interquartile range (IQR) to assess spread.
    3. **Skewness & Kurtosis:** Analyze symmetry (skewness) and tail behavior (kurtosis) of data distributions.

## Outlier Detection:

* + 1. Compute percentiles (25th, 75th) and use the IQR method to detect outliers.
    2. Visualize outliers and distributions using boxplots, histograms, and density plots.

## Tools & Libraries:

* + 1. **Python libraries:** Pandas (data handling), NumPy (statistical computations), Matplotlib/Seaborn (visualization).

# Results & Analysis

## Data Preprocessing

* + 1. The dataset is loaded and the first 5 rows are displayed.
       1. **Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

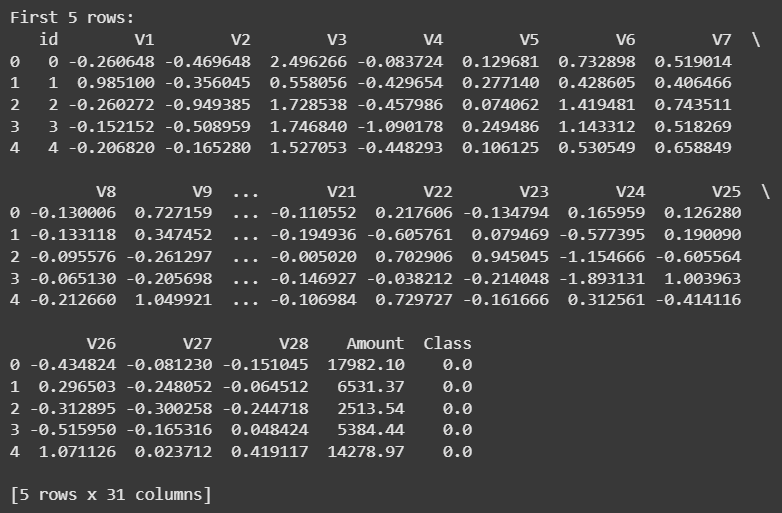
df = pd.read\_csv('/content/creditcard\_2023.csv')  # Replace with actual dataset path

# Display first five rows

print("First 5 rows:")

print(df.head())

* + - 1. **Output:**

****

* + 1. Missing values are checked and handled.
       1. **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

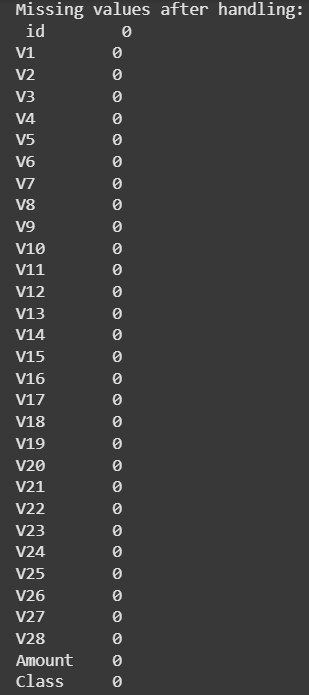
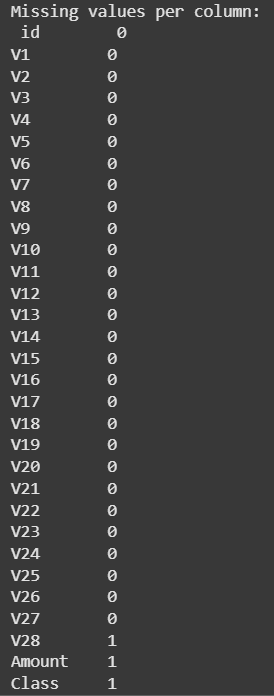
        df[column].fillna(df[column].mode()[0], inplace=True)

    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())

* + - 1. **Output:**

****

* + 1. Three numerical columns, e.g., **id**, **V1 and V2** are selected for further analysis.
       1. **Code:**

import matplotlib.pyplot as plt

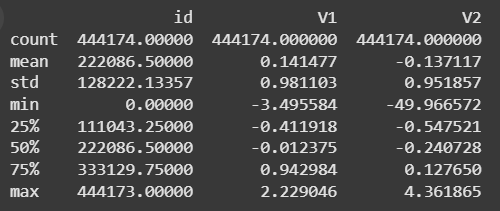
import numpy as np

numerical\_cols = df.select\_dtypes(include=np.number).columns

selected\_cols = numerical\_cols[:3]

print(df[selected\_cols].describe())

* + - 1. **Output:**

****

## Measures of Central Tendency

For each selected column, we calculate:

* + 1. **Mean**: The average value.
    2. **Median**: The middle value when sorted.
    3. **Mode**: The most frequently occurring value.

These measures help understand the central location of data points.

1. **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.")

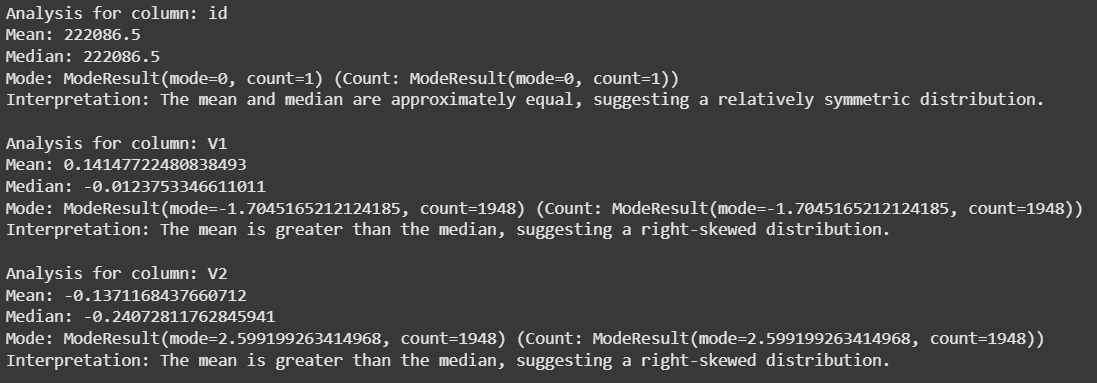
  elif mean\_val < median\_val:

    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.")

1. **Output:**



1. **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.

## Measures of Dispersion

For each selected column, we calculate:

* + 1. **Range**: The difference between the maximum and minimum values.
    2. **Variance**: The spread of data around the mean.
    3. **Standard Deviation**: The average deviation from the mean.
    4. **Interquartile Range (IQR):** The range between the 25th and 75th percentiles.
       1. **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)

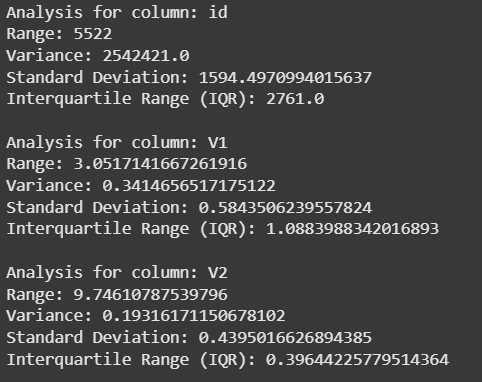
    q1 = df[col].quantile(0.25)

    q3 = df[col].quantile(0.75)

    iqr\_val = q3 - q1

    print(f"Interquartile Range (IQR): {iqr\_val}")

* + - 1. **Output:**

****

* + - 1. **Interpretation:**
         * **High variance & standard deviation** indicate high variability.
         * **IQR** helps detect outliers by showing data spread between Q1 and Q3.
    1. **Visualization**
       1. **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:

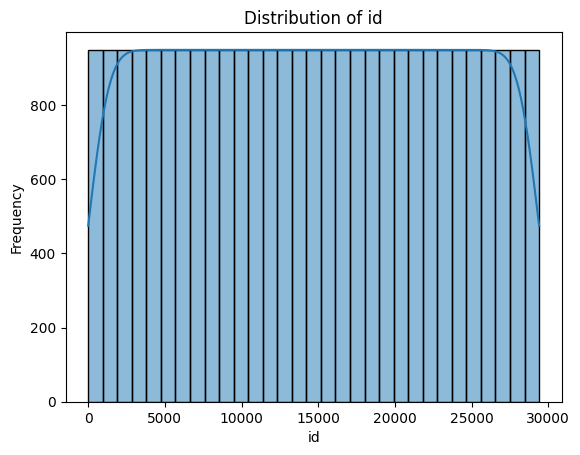
    plt.figure()

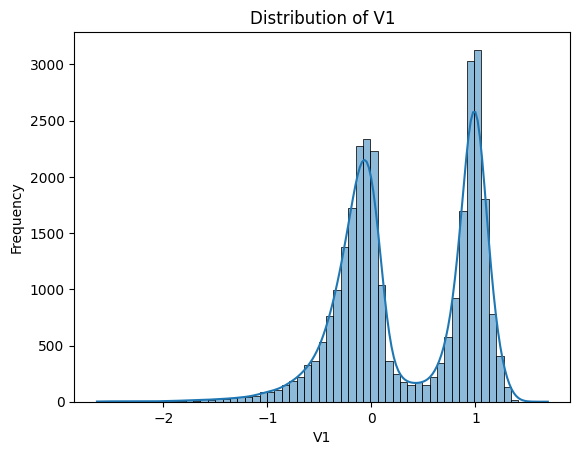
    sns.boxplot(y=df[col])

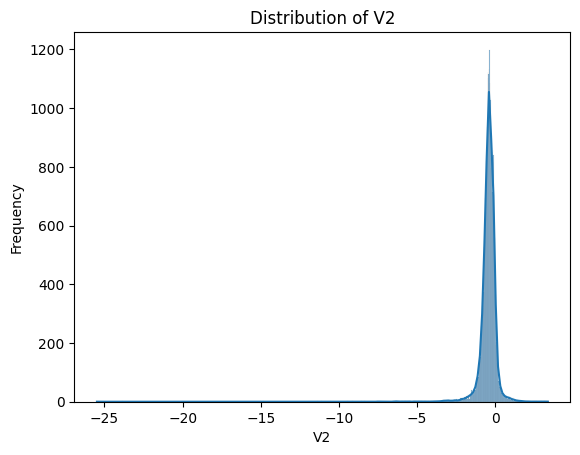
    plt.title(f"Boxplot of {col}")

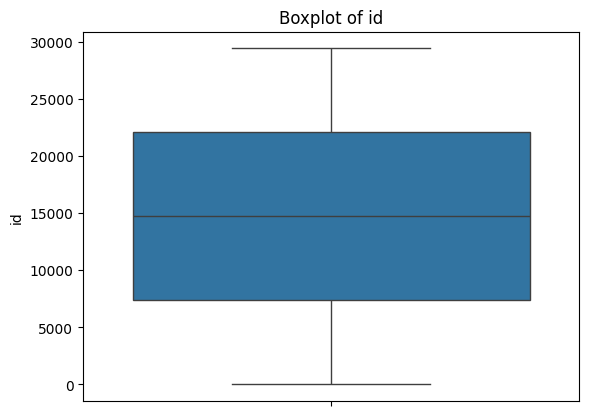
    plt.show()

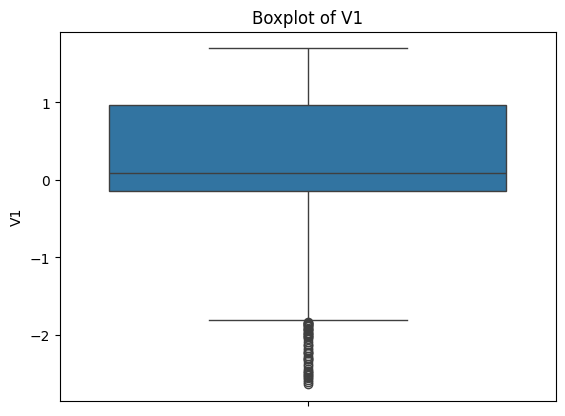
* + - * + **Output:**

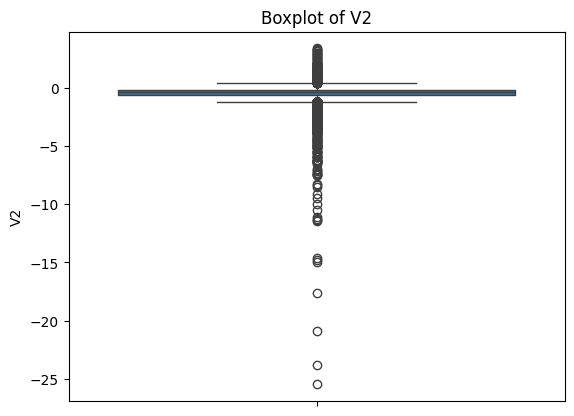












## Skewness & Kurtosis

* + 1. **Skewness** measures asymmetry in the data distribution.
       1. A skewness > 0 indicates a right-skewed distribution.
       2. A skewness < 0 indicates a left-skewed distribution.
    2. **Kurtosis** measures whether the data has heavy or light tails compared to a normal distribution.
       1. A kurtosis > 3 suggests a leptokurtic (heavy-tailed) distribution.
       2. 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).")

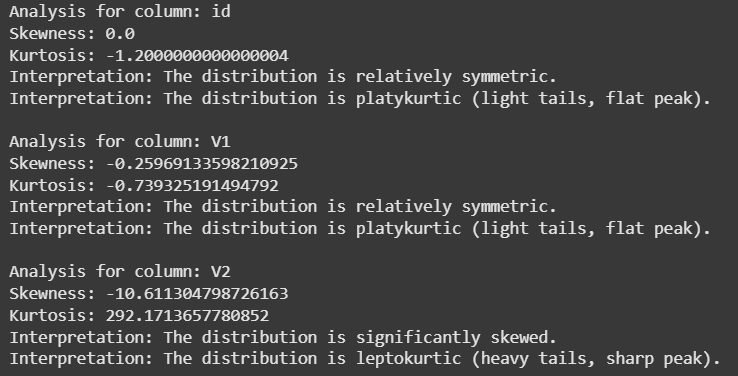
    elif kurtosis\_val < 3:

        print("Interpretation: The distribution is platykurtic (light tails, flat peak).")

    else:

        print("Interpretation: The distribution is mesokurtic (normal distribution-like kurtosis).")

* + - * + **Output:**



* **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))

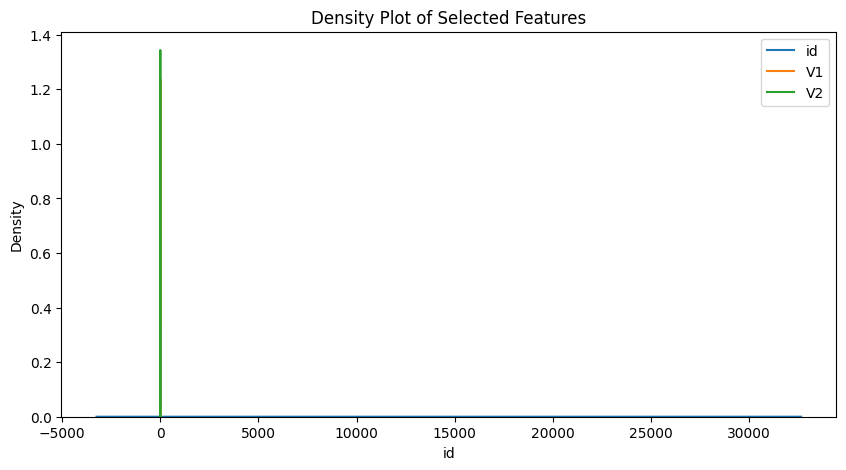
for col in selected\_cols:

    sns.kdeplot(df[col], label=col)

plt.legend()

plt.title("Density Plot of Selected Features")

plt.show()



Density plots support these interpretations.

## Percentiles & Quartiles

* + 1. The 25th, 50th, and 75th percentiles are calculated.
    2. Outliers are detected using the IQR method and visualized via boxplots.
       1. **Code:**

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}")

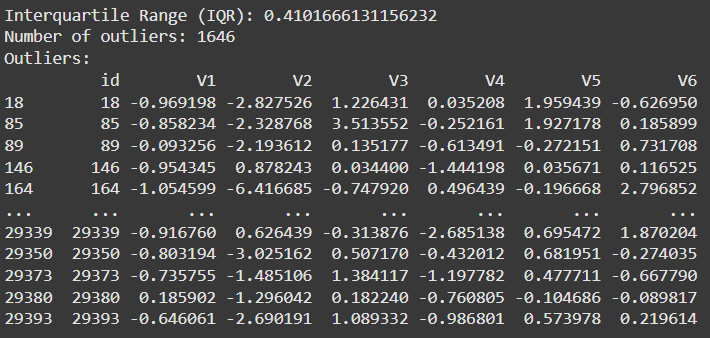
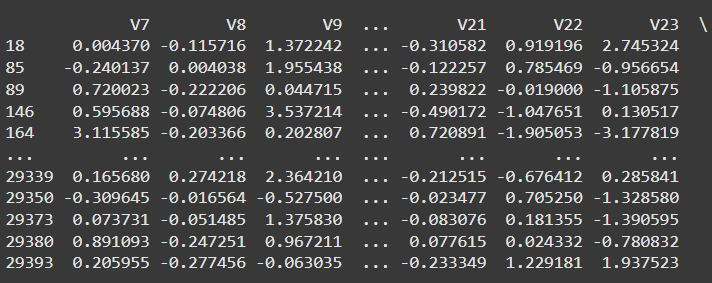
# Boxplot with outlier visualization

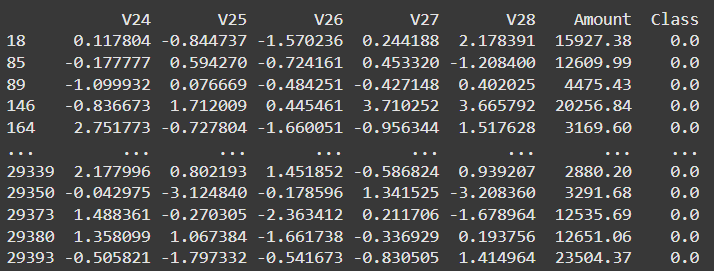
plt.figure()

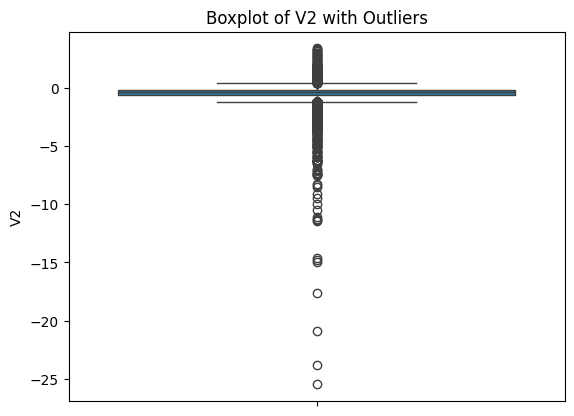
sns.boxplot(y=df[col])

plt.title(f"Boxplot of {col} with Outliers")

plt.show()

* + - 1. **Output:**

****



* + 1. **How these values help in detecting outliers?**
       1. Percentiles help in detecting outliers by defining the spread of the central 50% of data (IQR = Q3 - Q1).
       2. Extreme values that fall below Q1 - 1.5 \* IQR or above Q3 + 1.5 \* IQR are considered outliers.
       3. Boxplots use these percentiles to visually highlight data points outside the whiskers, making it easier to spot anomalies**.**

# 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.

# References

* Dataset Source: [Kaggle - Credit Card Fraud Detection](https://www.kaggle.com/code/samanfatima7/credit-card-fraud-detection-achieving-99-acc/input)
* 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