Credit Card Fraud Detection

Assignment 2

Author: Benjamin Heindl

Course: IST 652

Date: August 24, 2023

Summary Question

I aim to summarize the distribution of credit card transaction amounts and investigate patterns related to fraud in the dataset.

Description

This notebook explores a dataset containing credit card transactions with the goal of identifying patterns related to fraudulent transactions. The dataset includes various features, such as transaction amounts, transaction times, and anonymized variables. Through data preprocessing, visualization, hypothesis testing, and machine learning techniques, I aim to gain insights into transaction behavior and develop a basic fraud detection system.

The analysis is divided into several sections:

1. **Setup and Data Download:** Setting up the environment and loading the credit card transaction dataset.
2. **Data Preprocessing:** Cleaning and preprocessing the data for analysis.
3. **Data Visualization:** Visualizing the distribution of transaction amounts and the split between legitimate and fraudulent transactions.
4. **Hypothesis Testing:** Testing the hypothesis regarding transaction amounts and fraud. We examine whether the majority of fraudulent transactions occur with smaller transaction amounts compared to legitimate transactions.
5. **Model Training:** Training a logistic regression model for fraud detection.
6. **Model Evaluation:** Evaluating the trained model's performance on the test set.
7. **Conclusion and Future Steps:** Summarizing findings, discussing model performance, and suggesting future improvements.

The goal is to provide insights into the characteristics of fraudulent transactions and demonstrate the application of machine learning in credit card fraud detection.

Setup and Data Download

In [17]:
```
import os

# Set up Kaggle directory
os.environ['KAGGLE_CONFIG_DIR'] = '/Users/benjheindl/.kaggle/'
# !kaggle datasets download -d joebeachcapital/credit-card-fraud
# !unzip credit-card-fraud.zip
```

Loading and Cleaning Data

Load data from JSON file and perform initial cleaning

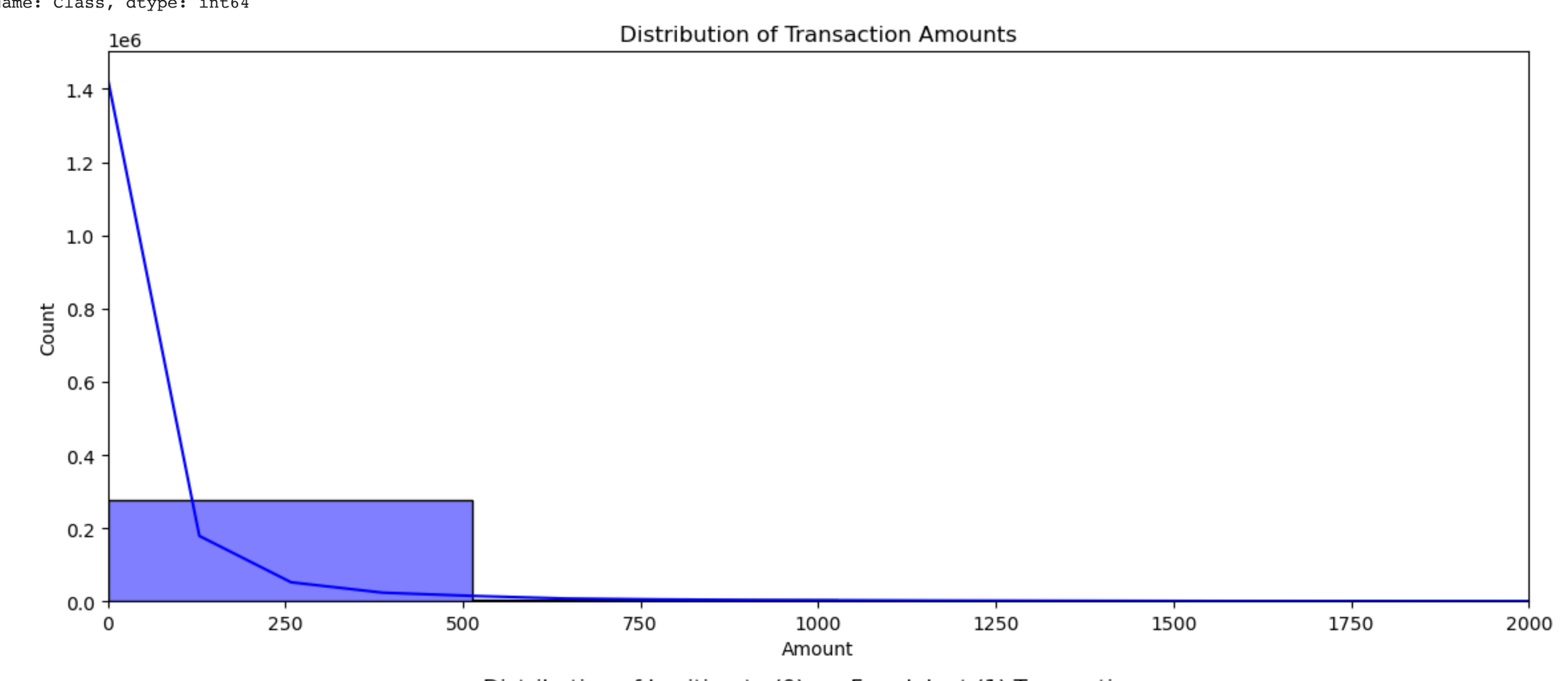
In [18]:
```
import pandas as pd
import json

# Load data from JSON file and perform initial cleaning
def load_and_clean_data(filename):
    with open(filename, 'r') as f:
        data = json.load(f)

    # Clean 'Class' column by removing single quotes and converting to integer
    df = pd.DataFrame(data)
    df['Class'] = df['Class'].str.replace("'", "").astype(int)

    return df

data_path = "/Users/benjheindl/Desktop/Summer 2023/IST 652 - Scripting for Data Analysis/Assignments/Assignment 2/data/creditcard_json.json"
data = load_and_clean_data(data_path)
```

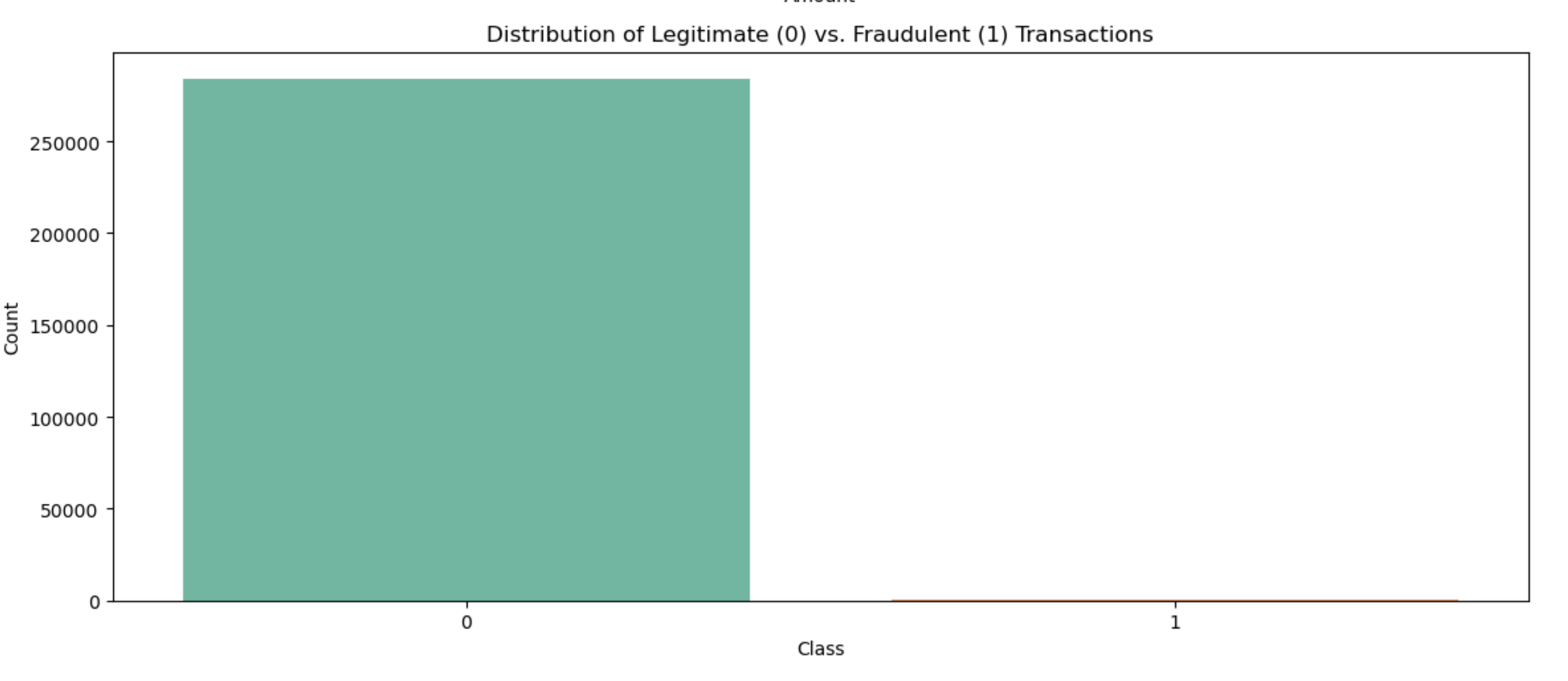
Data Visualization

Visualize distribution of transaction amounts and the split between legitimate and fraudulent transactions

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns

def visualize_data(data):
    # Print descriptive statistics and distributions
    print("Descriptive Statistics for Amount:\n", data['Amount'].describe())
    print("\nDistribution of Legitimate (0) vs. Fraudulent (1) Transactions:\n", data['Class'].value_counts())

    # Create subplots
    fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(12, 10))

    # Histogram of transaction amounts
    sns.histplot(data=data, x='Amount', bins=50, ax=ax[0], color='blue', kde=True)
    ax[0].set_title('Distribution of Transaction Amounts')
    ax[0].set_xlim([0, 2000])
    ax[0].set_xlabel('Amount')
    ax[0].set_ylabel('Count')

    # Bar plot for distribution of legitimate vs. fraudulent transactions
    sns.countplot(data=data, x='Class', ax=ax[1], palette='Set2')  # Using a different color palette
    ax[1].set_title('Distribution of Legitimate (0) vs. Fraudulent (1) Transactions')
    ax[1].set_xlabel('Class')
    ax[1].set_ylabel('Count')

    plt.tight_layout()
    plt.show()

visualize_data(data)
```

```
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

Distribution of Legitimate (0) vs. Fraudulent (1) Transactions:
 0    284315
1       492
Name: Class, dtype: int64
```





Descriptive Statistics for Amount:

- There are a total of 284,807 transactions.
- Average Transaction Amount: $88.35
- Standard Deviation: 250.12
- Smallest Transaction Amount: $0.00 (possibly a canceled transaction or a voided one?)
- 25th Percentile: $5.60 (25% of transactions are below or equal to this amount)
- Median (50th Percentile): $22.00
- 75th Percentile: $77.17 (75% of transactions are below or equal to this amount)
- Largest Transaction Amount: $25,691.16

Distribution of Legitimate (0) vs. Fraudulent (1) Transactions:

- There are 284,315 legitimate transactions.
- There are only 492 fraudulent transactions.

From these results, you can observe:

- The transaction amounts have a wide range, but most transactions are relatively small given the median is only $22.00.
- The dataset is heavily imbalanced towards legitimate transactions.

Hypothesis Testing

Test the hypothesis: "The majority of fraudulent transactions occur with smaller transaction amounts compared to legitimate transactions."

Compare the mean and median transaction amounts for both types

In [20]:
```
def calculate_and_print_transaction_stats(data):
    # Calculate the mean transaction amount for legitimate transactions
    legitimate_mean = data[data['Class'] == 0]['Amount'].mean()

    # Calculate the mean transaction amount for fraudulent transactions
    fraudulent_mean = data[data['Class'] == 1]['Amount'].mean()

    # Calculate the median transaction amount for legitimate transactions
    legitimate_median = data[data['Class'] == 0]['Amount'].median()

    # Calculate the median transaction amount for fraudulent transactions
    fraudulent_median = data[data['Class'] == 1]['Amount'].median()

    # Print the calculated statistics
    print(f"Mean Amount for Legitimate Transactions: ${legitimate_mean:.2f}")
    print(f"Mean Amount for Fraudulent Transactions: ${fraudulent_mean:.2f}")
    print(f"Median Amount for Legitimate Transactions: ${legitimate_median:.2f}")
    print(f"Median Amount for Fraudulent Transactions: ${fraudulent_median:.2f}")

# Call the function to calculate and print transaction statistics
calculate_and_print_transaction_stats(data)
```

```
Mean Amount for Legitimate Transactions: $88.29
Mean Amount for Fraudulent Transactions: $122.21
Median Amount for Legitimate Transactions: $22.00
Median Amount for Fraudulent Transactions: $9.25
```

These results suggest that on average fraudulent transactions tend to have higher mean and median transaction amounts compared to legitimate transactions. This observation aligns with the hypothesis that fraudulent activities might involve larger transactions to maximize the gain. However, to determine if this difference is statistically significant, further tests are required.

In [21]:
```
from scipy.stats import mannwhitneyu, anderson

def conduct_hypothesis_tests(data):
    # Separate legitimate and fraudulent transaction amounts
    legitimate_amounts = data[data['Class'] == 0]['Amount']
    fraudulent_amounts = data[data['Class'] == 1]['Amount']

    # Perform Mann-Whitney U test
    u_stat, p_val = mannwhitneyu(legitimate_amounts, fraudulent_amounts)
    print(f"Mann-Whitney U Statistic: {u_stat}")
    print(f"P-Value: {p_val}")

    # Interpretation based on p-value
    if p_val < 0.05:
        print("The Mann-Whitney U test suggests that there is a significant difference in transaction amounts between legitimate and fraudulent transactions.")
    else:
        print("The Mann-Whitney U test does not provide strong evidence of a significant difference in transaction amounts between legitimate and fraudulent transactions.")

    # Perform Anderson-Darling test
    result = anderson(data['Amount'])
    print(f"Anderson-Darling Statistic: {result.statistic}")
    print(f"Anderson-Darling Critical Values: {result.critical_values}")
    print(f"Anderson-Darling Significance Levels: {result.significance_level}")

    # Interpretation based on Anderson-Darling statistic
    if result.statistic > result.critical_values[2]:
        print("The Anderson-Darling test statistic is higher than the critical value at the 5% significance level.")
        print("This suggests that the distribution of transaction amounts significantly deviates from a normal distribution.")
    else:
        print("The Anderson-Darling test statistic does not exceed the critical value at the 5% significance level.")
        print("This suggests that there is not strong evidence to reject the assumption of normality for the distribution of transaction amounts.")

# Call the function to conduct hypothesis tests
conduct_hypothesis_tests(data)
```

```
Mann-Whitney U Statistic: 78049581.0
P-Value: 8.578472310840218e-06
The Mann-Whitney U test suggests that there is a significant difference in transaction amounts between legitimate and fraudulent transactions.
Anderson-Darling Statistic: 54953.603485681175
Anderson-Darling Critical Values: [0.576 0.656 0.787 0.918 1.092]
Anderson-Darling Significance Levels: [15.  10.   5.   2.5  1. ]
The Anderson-Darling test statistic is higher than the critical value at the 5% significance level.
This suggests that the distribution of transaction amounts significantly deviates from a normal distribution.
```

Hypothesis Testing Results

Mann-Whitney U Test

The Mann-Whitney U test was conducted to compare transaction amounts between legitimate and fraudulent transactions:

- Mann-Whitney U Statistic: 78049581.0
- P-Value: 8.578472310840218e-06

The small p-value indicates that there is a statistically significant difference in transaction amounts between legitimate and fraudulent transactions. This suggests that fraudulent transactions tend to have significantly different transaction amounts compared to legitimate transactions.

Anderson-Darling Test

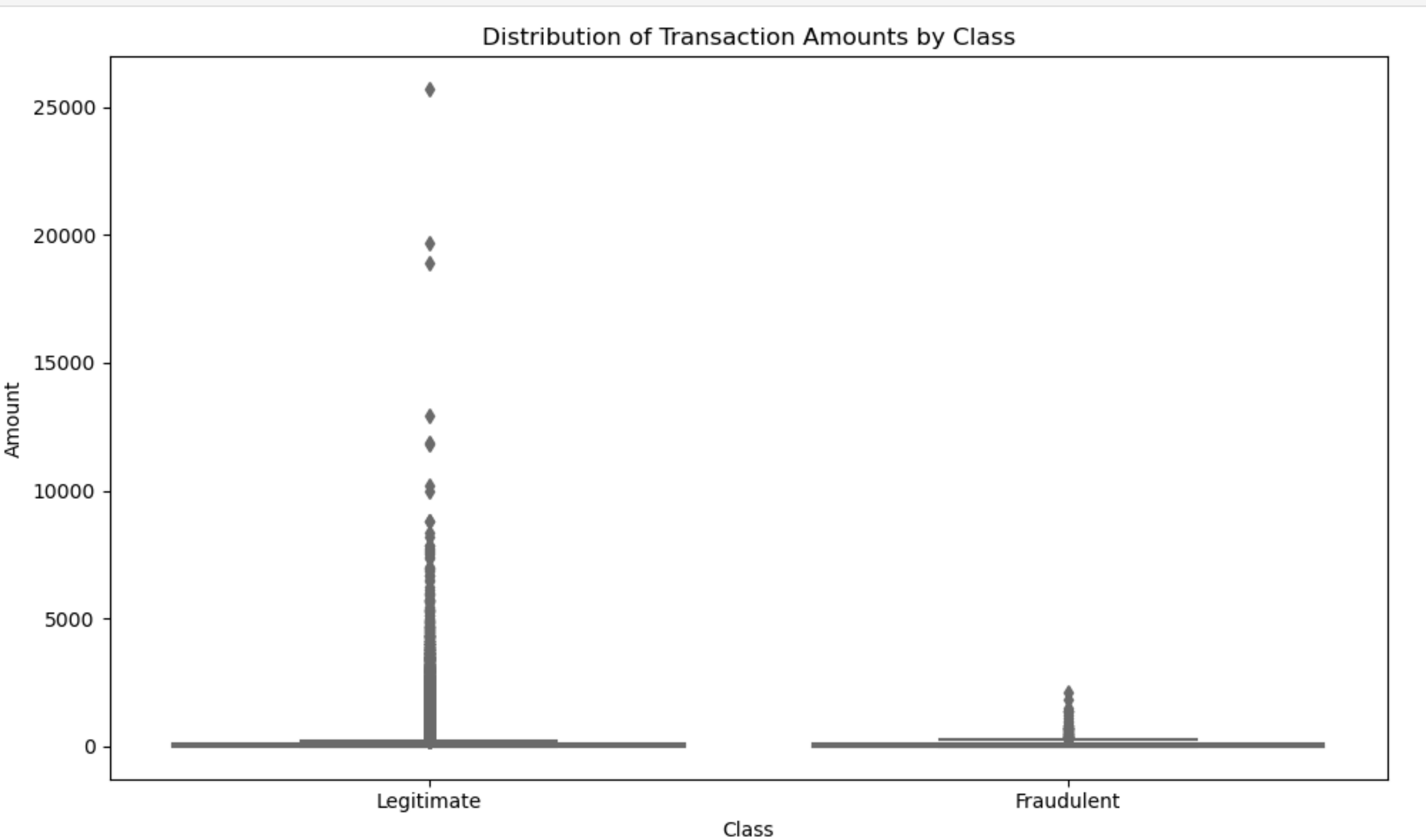
The Anderson-Darling test was used to assess the normality of the distribution of transaction amounts:

- Anderson-Darling Statistic: 54953.603485681175
- Anderson-Darling Critical Values: [0.576 0.656 0.787 0.918 1.092]
- Anderson-Darling Significance Levels: [15. 10. 5. 2.5 1.]

The Anderson-Darling test statistic is substantially higher than the critical value at the 5% significance level. This suggests that the distribution of transaction amounts significantly deviates from a normal distribution. The non-normal distribution of transaction amounts reinforces the complexity of the dataset and highlights the importance of selecting appropriate statistical methods.

These results contribute to our understanding of the data's characteristics and provide insights into the potential differences in transaction amounts between legitimate and fraudulent transactions.

In [22]:
```
def visualize_transaction_amounts_by_class(data):
    plt.figure(figsize=(10, 6))

    # Box plot to visualize the distribution of transaction amounts by class
    sns.boxplot(x='Class', y='Amount', data=data, palette='Set3')  # Using a different color palette
    plt.title('Distribution of Transaction Amounts by Class')
    plt.xticks([0, 1], ['Legitimate', 'Fraudulent'])
    plt.xlabel('Class')
    plt.ylabel('Amount')

    plt.tight_layout()
    plt.show()

visualize_transaction_amounts_by_class(data)
```



In [23]:
```
from scipy import stats
import numpy as np

def analyze_distribution(data):
    from scipy.stats import skew, kurtosis

    # Calculate and print skewness and kurtosis of the data
    print(f"Skewness: {skew(data)}")
    print(f"Kurtosis: {kurtosis(data)}")

    # Calculate a 95% confidence interval for the mean of the data
    confidence_level = 0.95
    degrees_freedom = data.shape[0] - 1
    confidence_interval = stats.t.interval(confidence_level, degrees_freedom, loc=np.mean(data), scale=stats.sem(data))

    print(f"95% confidence interval for transaction amounts: {confidence_interval}")

    return confidence_interval
```

In [24]:
```
def analyze_correlations(data):
    # Calculate and print correlations between 'Amount' and other columns
    correlations = data.corr()['Amount']
    print(correlations)

# Analyze the distribution of 'Amount' and calculate its confidence interval
confidence_interval = analyze_distribution(data['Amount'])

# Analyze correlations involving 'Amount' and other columns
analyze_correlations(data)
```

```
Skewness: 16.97763503663315
Kurtosis: 845.0777883188754
95% confidence interval for transaction amounts: (87.43102607083128, 89.26821243103137)
Amount    1.000000
Class     0.005632
Time     -0.010596
V1       -0.227709
V10      -0.101502
V11       0.000104
V12      -0.009542
V13       0.005293
V14       0.033751
V15      -0.002986
V16      -0.003910
V17       0.007309
V18       0.035650
V19      -0.056151
V2       -0.531409
V20       0.339403
V21       0.105999
V22      -0.064801
V23      -0.112633
V24       0.005146
V25      -0.047837
V26      -0.003208
V27       0.028825
V28       0.010258
V3       -0.210880
V4        0.098732
V5       -0.386356
V6        0.215981
V7        0.397311
V8       -0.103079
V9       -0.044246
Name: Amount, dtype: float64
```

Descriptive Statistics and Correlations

Descriptive Statistics

- Skewness: 16.97763503663315
- Kurtosis: 845.0777883188754

The skewness value of approximately 16.98 suggests that the distribution of transaction amounts is heavily skewed to the right, indicating a long tail of larger values. The high kurtosis value of 845.08 indicates that the distribution has heavy tails and a sharp peak, which are characteristics of distributions with extreme outliers.

Correlation of Amount with Other Variables

- Amount vs. Class: 0.005632
- Amount vs. Time: -0.010596
- Amount vs. V1: -0.227709
- Amount vs. V10: -0.101502 ...
- Amount vs. V7: 0.397311
- Amount vs. V8: -0.103079
- Amount vs. V9: -0.044246

The correlations provide insights into the relationships between the transaction amount and other variables. The low correlation values with 'Class' and 'Time' suggest weak associations. Notably, some 'V' variables exhibit moderate correlations, with 'V7' having the highest positive correlation.

These statistics offer a glimpse into the distributional characteristics of transaction amounts and highlight potential relationships between the transaction amount and other variables in the dataset.

Data Preprocessing

Preprocess data by binning the transaction times, one-hot encoding of categorical features, handling imbalanced classes using SMOTE, and splitting data into training and testing sets

In [25]:
```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

def preprocess_data(data):
    # Features and target
    X = data.drop("Class", axis=1)
    y = data["Class"]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

    # Resample using SMOTE (Synthetic Minority Over-sampling Technique)
    # SMOTE generates synthetic samples for the minority class to balance the class distribution.
    # This helps prevent the model from being biased towards the majority class.
    smote = SMOTE(random_state=42)
    X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

    return X_resampled, y_resampled, X_test, y_test

X_train, y_train, X_test, y_test = preprocess_data(data)
```

Model Training

Train a logistic regression model using the processed data

In [26]:
```
from sklearn.linear_model import LogisticRegression

# Train a logistic regression model
def train_model(X_train, y_train):
    # Logistic regression is chosen due to its simplicity
    clf = LogisticRegression(max_iter=1000)
    clf.fit(X_train, y_train)
    return clf

clf = train_model(X_train, y_train)
```

Model Evaluation

Evaluate the performance of our model on the test set

In [27]:
```
from sklearn.metrics import classification_report

# Evaluate the trained model on the test set
def evaluate_model(clf, X_test, y_test):
    y_pred = clf.predict(X_test)
    print(classification_report(y_test, y_pred))

evaluate_model(clf, X_test, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.09      0.91      0.16        98

    accuracy                           0.98     56962
   macro avg       0.54      0.95      0.57     56962
weighted avg       1.00      0.98      0.99     56962
```

Model Evaluation Explanation

In the context of fraud detection:

- **Precision**: The proportion of predicted fraudulent transactions that are actually fraudulent. A high precision indicates that when the model predicts fraud, it's likely to be correct.

- **Recall**: Recall is the proportion of actual fraudulent transactions that were correctly predicted as fraudulent by the model. A high recall indicates that the model is good at capturing actual fraud.

- **F1-score**: A balance between precision and recall. It's especially useful when looking to find a balance between false positives and false negatives.

In fraud detection, it seems that striking a balance between precision and recall is crucial. A high precision ensures that genuine transactions aren't incorrectly flagged as fraudulent, while a high recall helps in identifying as many actual fraudulent transactions as possible.

Conclusion and Future Steps

In this analysis, I explored a dataset of credit card transactions to identify patterns related to fraud. The findings indicate that fraudulent transactions tend to have different transaction amounts compared to legitimate transactions. However, building an accurate fraud detection model remains challenging due to the imbalanced nature of the data.

Model Performance: The trained logistic regression model achieved reasonable accuracy and recall for fraud detection. The high recall indicates that the model is successful in identifying many fraudulent transactions, but it comes at the cost of a lower precision, leading to some false positives.

Future Steps: To further improve the model's performance, the following steps could be considered:

- Experiment with different algorithms, such as anomaly detection techniques, to improve the trade-off between precision and recall.
- Explore feature engineering techniques to capture more meaningful information from the data.
- Fine-tune the model's hyperparameters to achieve a better balance between precision and recall.

Ultimately, an effective fraud detection system is crucial to protect both consumers and financial institutions from potential losses.