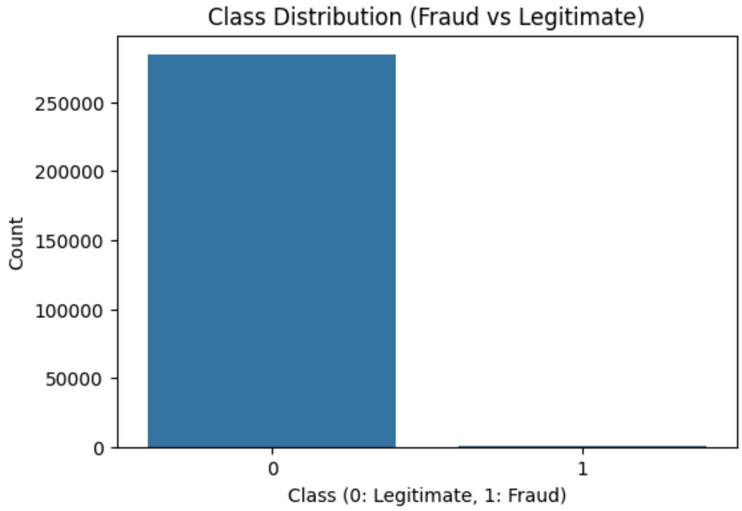
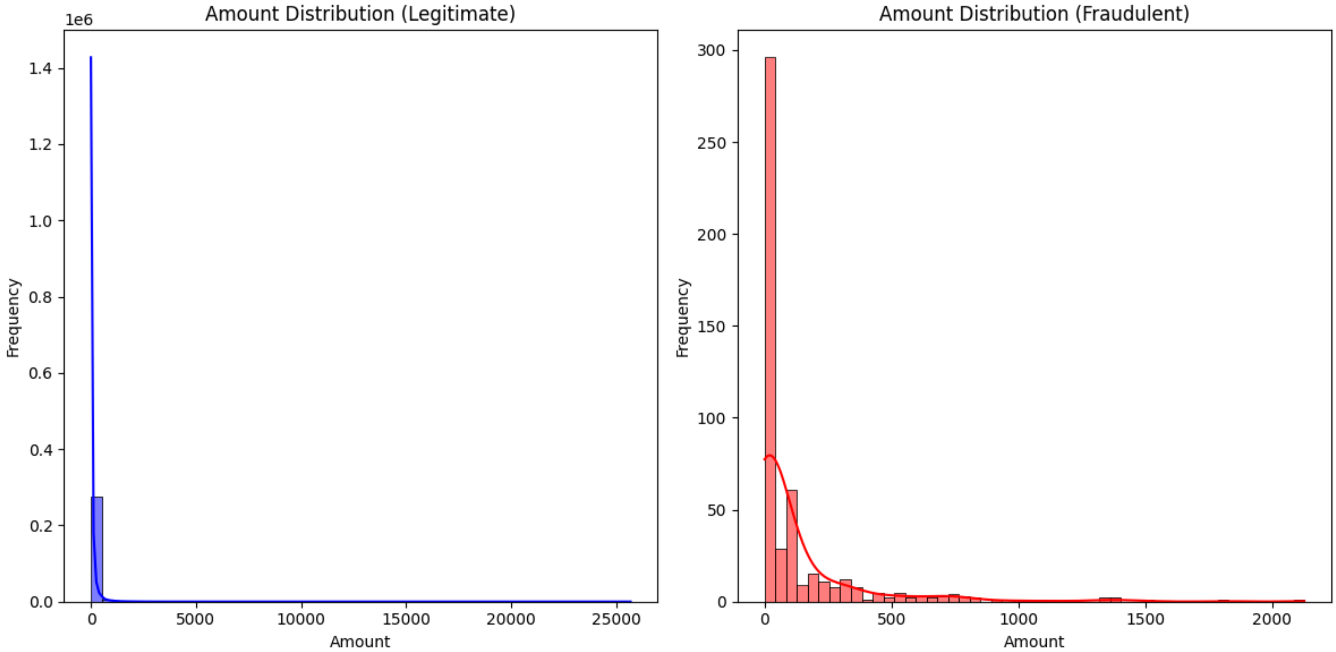
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_curve, auc, confusion_matrix

import kagglehub

# Download the latest version of the dataset using kagglehub
path = kagglehub.dataset_download("mlg-ulb/creditcardfraud")
print("Path to dataset files:", path)

# Load the dataset into a Pandas DataFrame
credit_card_data = pd.read_csv('/content/creditcard.csv')

# Display the first few rows of the dataset to understand its structure
print(credit_card_data.head())

# 1. Visualize Class Distribution (Fraud vs Legitimate)
plt.figure(figsize=(6, 4))
sns.countplot(x='Class', data=credit_card_data)
plt.title('Class Distribution (Fraud vs Legitimate)')
plt.xlabel('Class (0: Legitimate, 1: Fraud)')
plt.ylabel('Count')
plt.show()

# Separate the dataset into two subsets: legitimate and fraudulent transactions
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]

# 2. Visualize Feature Distributions (Amount for Fraud vs Legitimate)
plt.figure(figsize=(12, 6))

# Plot the distribution of transaction amounts for legitimate transactions
plt.subplot(1, 2, 1)
sns.histplot(legit['Amount'], kde=True, color='blue', bins=50)
plt.title('Amount Distribution (Legitimate)')
plt.xlabel('Amount')
plt.ylabel('Frequency')

# Plot the distribution of transaction amounts for fraudulent transactions
plt.subplot(1, 2, 2)
sns.histplot(fraud['Amount'], kde=True, color='red', bins=50)
plt.title('Amount Distribution (Fraudulent)')
plt.xlabel('Amount')
plt.ylabel('Frequency')

plt.tight_layout()  # Adjust the layout to avoid overlap
plt.show()

# 3. Prepare Data for Model Training

# Since the dataset is highly imbalanced, take a random sample of legitimate transactions equal to the number of fraudulent ones
legit_sample = legit.sample(n=492)

# Create a new balanced dataset by combining the legitimate sample and all fraudulent transactions
new_dataset = pd.concat([legit_sample, fraud], axis=0)

# Split the dataset into features (X) and target label (Y)
X = new_dataset.drop(columns='Class', axis=1)  # Features
Y = new_dataset['Class']  # Target label

# Split the data into training and testing sets (80% train, 20% test), ensuring stratified sampling to maintain class balance
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

# 4. Train Logistic Regression Model

# Initialize the logistic regression model
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, Y_train)

# Evaluate model accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy on Training Data:', training_data_accuracy)
```

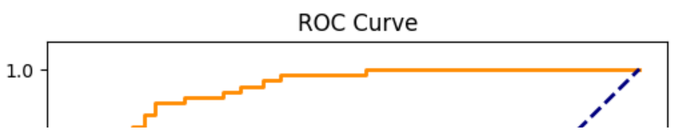
```
# Evaluate model accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy on Test Data:', test_data_accuracy)

# 5. Plot ROC Curve to evaluate model performance

# Predict probabilities for the positive class (fraudulent transactions) on the test set
y_prob = model.predict_proba(X_test)[:, 1]

# Calculate the False Positive Rate (FPR), True Positive Rate (TPR), and thresholds for the ROC curve
fpr, tpr, thresholds = roc_curve(Y_test, y_prob)

# Compute the area under the ROC curve (AUC)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  # Diagonal line for reference
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate (Recall)')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()

# 6. Plot Confusion Matrix to visualize model performance on test data

# Predict class labels on the test set
y_pred = model.predict(X_test)

# Compute the confusion matrix
cm = confusion_matrix(Y_test, y_pred)

# Plot the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Legitimate', 'Fraud'], yticklabels=['Legitimate', 'Fraud'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Print formatted accuracy scores on training and test data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print(f'Accuracy on Training Data: {training_data_accuracy * 100:.2f}%')

X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print(f'Accuracy on Test Data: {test_data_accuracy * 100:.2f}%')
```

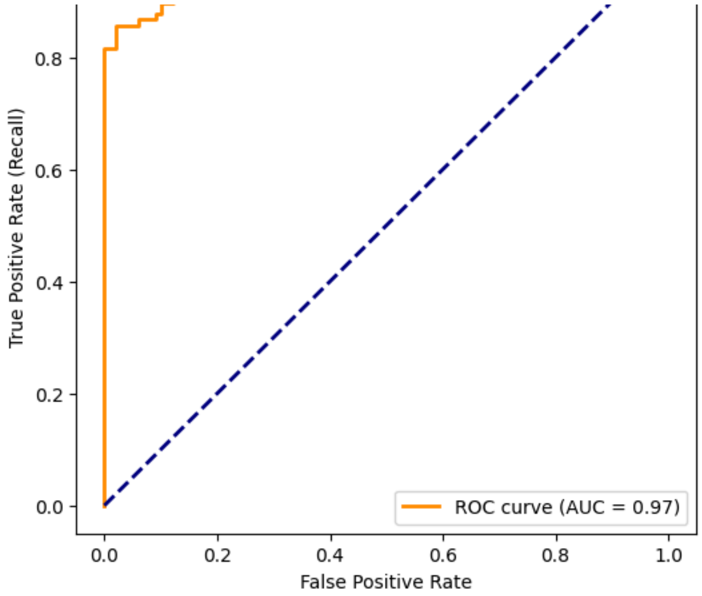
```
Path to dataset files: /root/.cache/kagglehub/datasets/mlg-ulb/creditcardfraud/versions/3
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

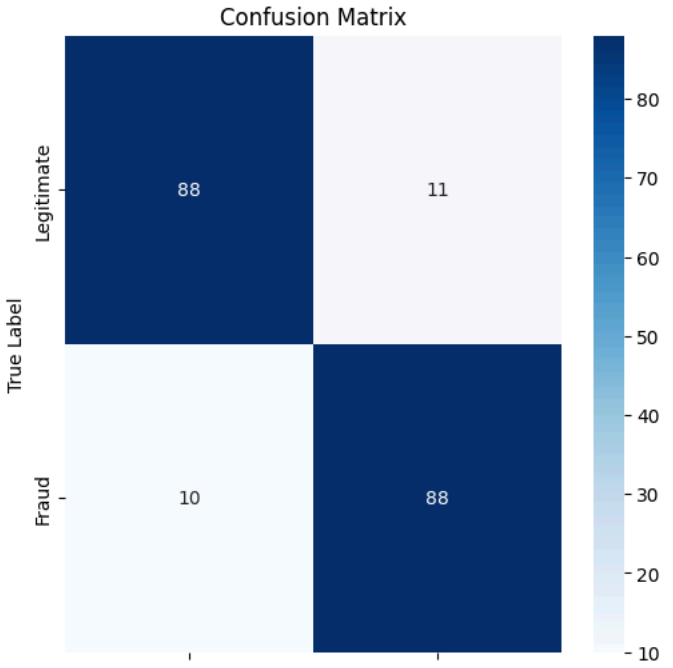





```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy on Training Data: 0.9466327827191868
Accuracy on Test Data: 0.8934010152284264
```




True Positive Rate (Recall)
False Positive Rate
ROC curve (AUC = 0.97)
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix
True Label
Legitimate
Fraud
88
11
10
88