CODE

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression

# Load the dataset
# Replace 'path_to_csv' with the actual file path
data = pd.read_csv('creditcard.csv')
print(data)

# Check for missing values and basic dataset info
print(f"Dataset shape: {data.shape}")
print(f"Missing values:\n{data.isnull().sum()}")

# Separate features and target
X = data.drop(columns=['Class'])  # Replace 'Class' with your target column name if different
y = data['Class']

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

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Handle class imbalance using SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Choose a classification algorithm
# Use Logistic Regression for faster training
# You can switch to RandomForestClassifier if needed
classifier = LogisticRegression(max_iter=500, random_state=42)

# Train the model
print("Training the model...")
classifier.fit(X_train_resampled, y_train_resampled)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model's performance
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Calculate precision, recall, and F1-score
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"\nPrecision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_curve, auc, precision_recall_curve

# Plot 1: Class Distribution5
plt.figure(figsize=(6, 4))
sns.countplot(x=y)
plt.title("Class Distribution")
plt.xlabel("Class (0: Genuine, 1: Fraudulent)")
plt.ylabel("Count")
plt.show()

# Plot 2: Correlation Heatmap
plt.figure(figsize=(12, 10))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()

# Plot 3: Feature Distribution (After Scaling)
plt.figure(figsize=(14, 8))
```

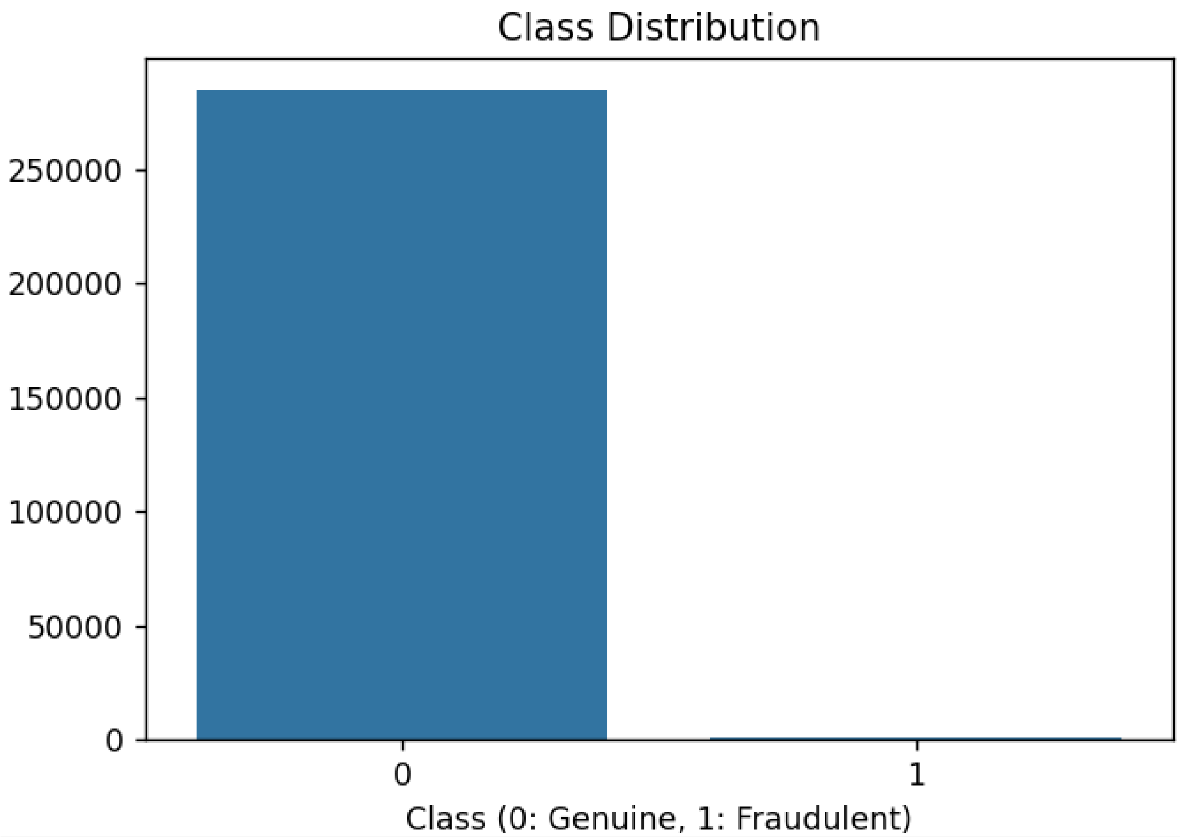
```
for i, col in enumerate(X.columns[:10], 1):  # Adjust for number of features to visualize
    plt.subplot(3, 4, i)
    sns.kdeplot(X_scaled[:, i - 1], color='blue', fill=True)
    plt.title(col)
plt.tight_layout()
plt.show()

# Plot 4: ROC Curve
y_proba = classifier.predict_proba(X_test)[:, 1]  # Probability scores for the positive class
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 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='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.show()

# Plot 5: Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, y_proba)
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', lw=2)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.show()
```

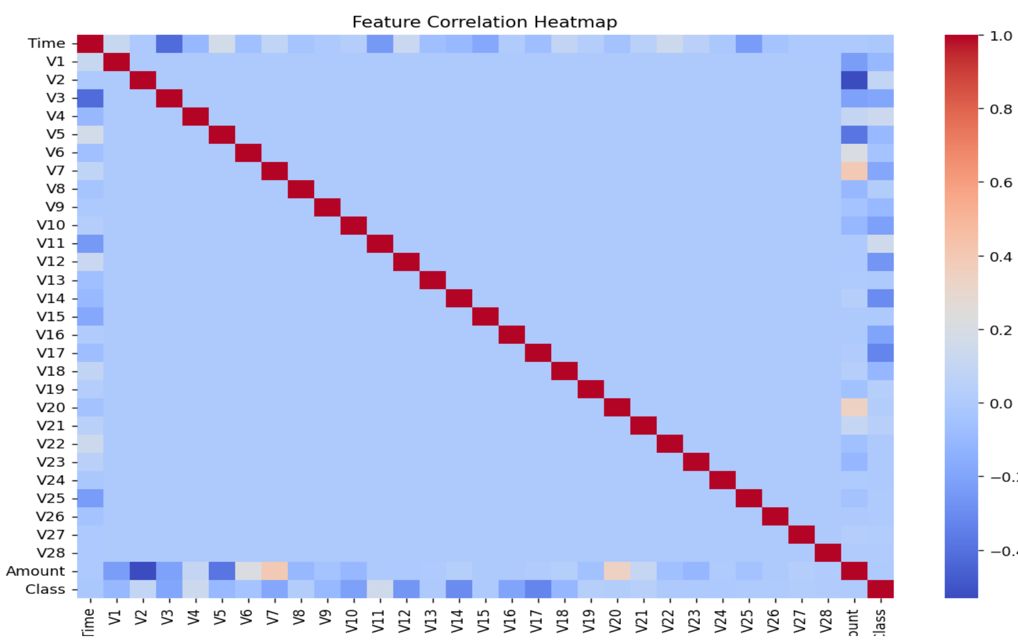
OUTPUT

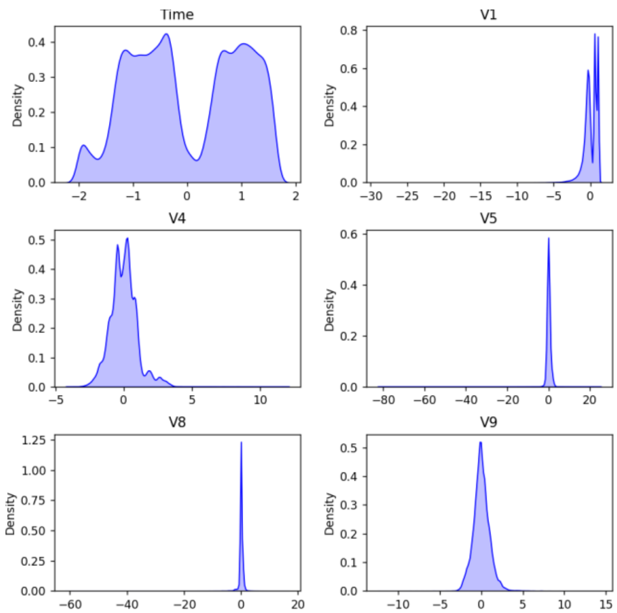
```
   Time        V1        V2        V3        V4        V5        V6        V7  ...       V23       V24       V25       V26       V27
V28  Amount  Class

0          0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  ... -0.110474
0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62      0

1          0.0   1.191857   0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  ...  0.101288 -
0.339846  0.167170  0.125895 -0.008983  0.014724    2.69      0

2          1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  ...  0.909412 -
0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0

3          1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  ... -0.190321 -
1.175575  0.647376 -0.221929  0.062723  0.061458  123.50      0

4          2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941  ... -0.137458
0.141267 -0.206010  0.502292  0.219422  0.215153   69.99      0

...        ...       ...       ...       ...       ...       ...       ...       ...  ...       ...       ...       ...       ...       ...       ...     ...    ...

284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473 -2.606837 -4.918215  ...
1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77      0

284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229  1.058415  0.024330  ...
0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79      0

284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515  3.031260 -0.296827  ... -
0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88      0

284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961  0.623708 -0.686180  ... -
0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00      0

284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546 -0.649617  1.577006  ...
0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00      0


[284807 rows x 31 columns]

Dataset shape: (284807, 31)

Missing values:

Time      0
```

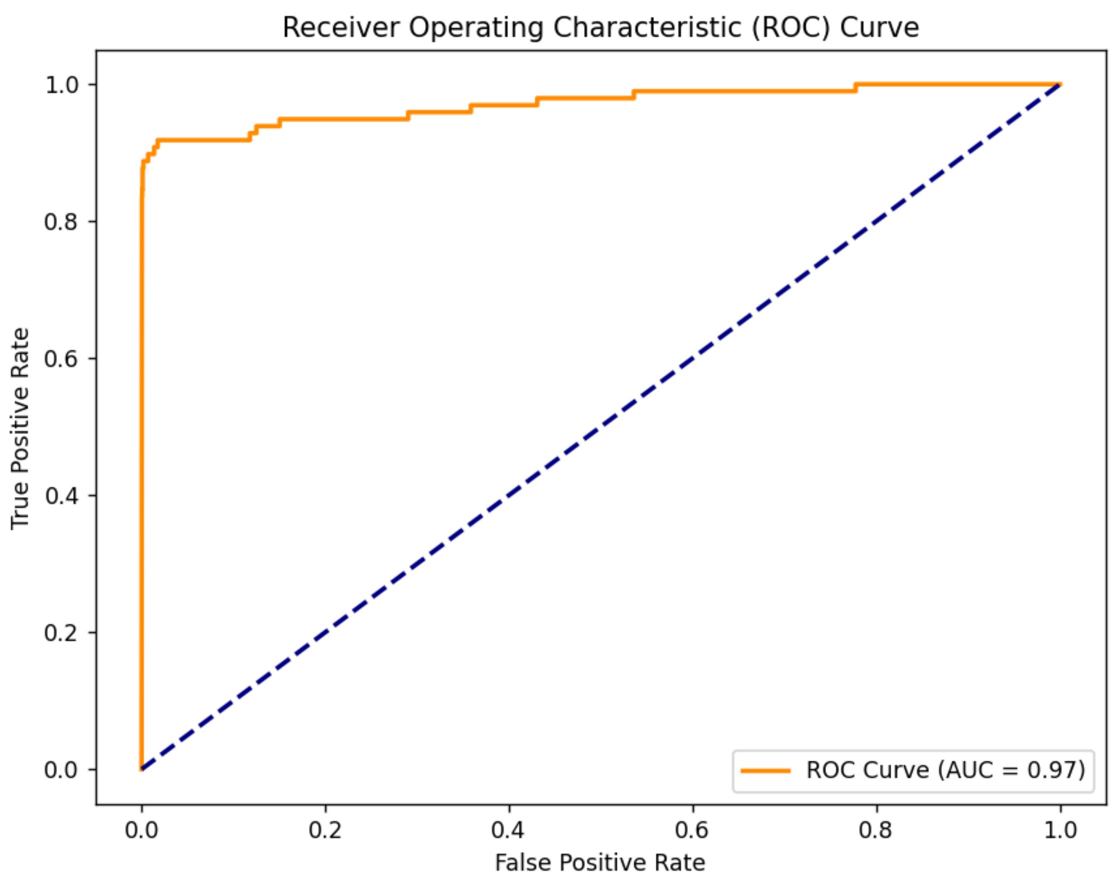
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0
V19	0
V20	0
V21	0
V22	0
V23	0
V24	0
V25	0
V26	0
V27	0
V28	0
Amount	0

```
Class     0
dtype: int64
Training the model...

Confusion Matrix:
[[55399  1465]
 [    8    90]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     56864
           1       0.06      0.92      0.11        98

    accuracy                           0.97     56962
   macro avg       0.53      0.95      0.55     56962
weighted avg       1.00      0.97      0.99     56962


Precision: 0.06
Recall: 0.92
F1-Score: 0.11
```


Class Distribution
250000
200000
150000
100000
50000
0
0
1
Class (0: Genuine, 1: Fraudulent)


Feature Correlation Heatmap
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
Density


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC Curve (AUC = 0.97)