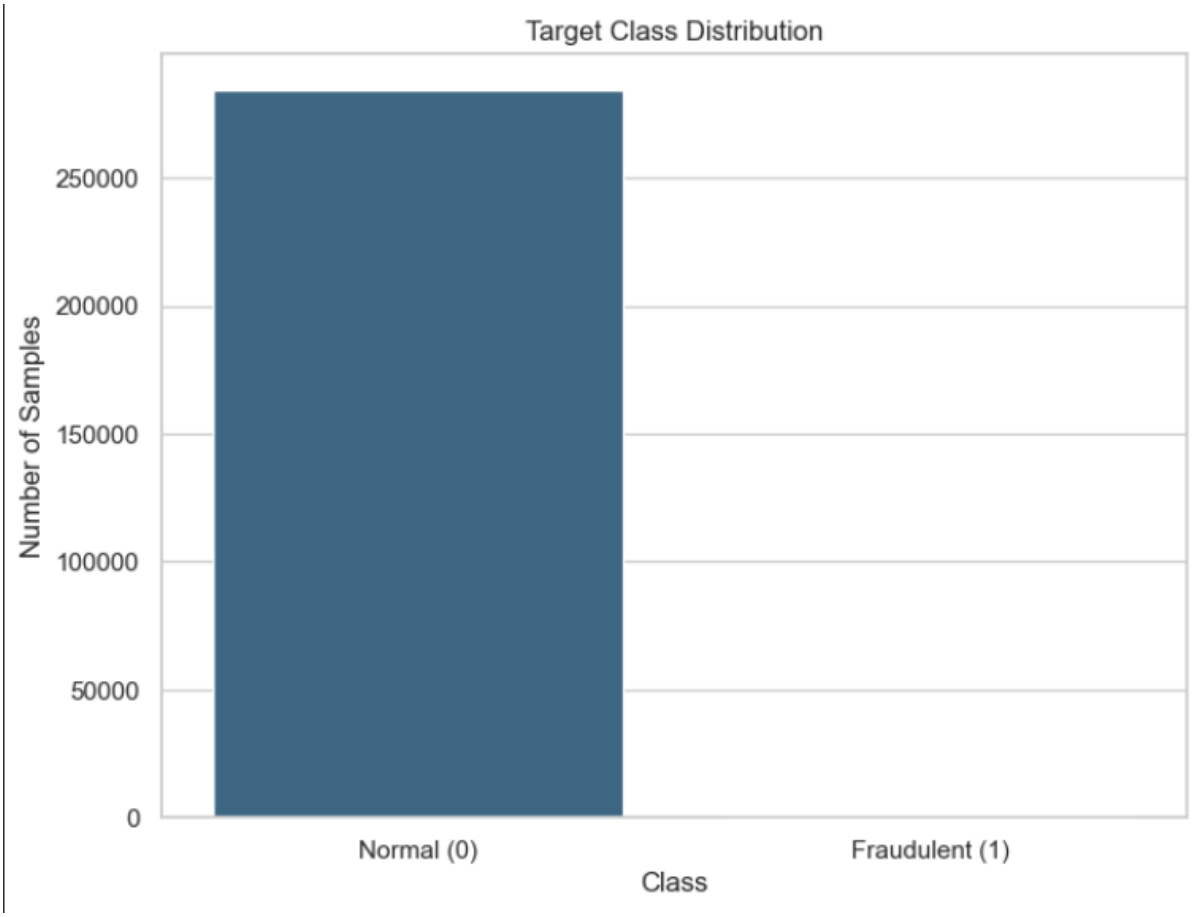
در ابتدا دیتاست را دانلود کرده و آن را لود می کنیم.

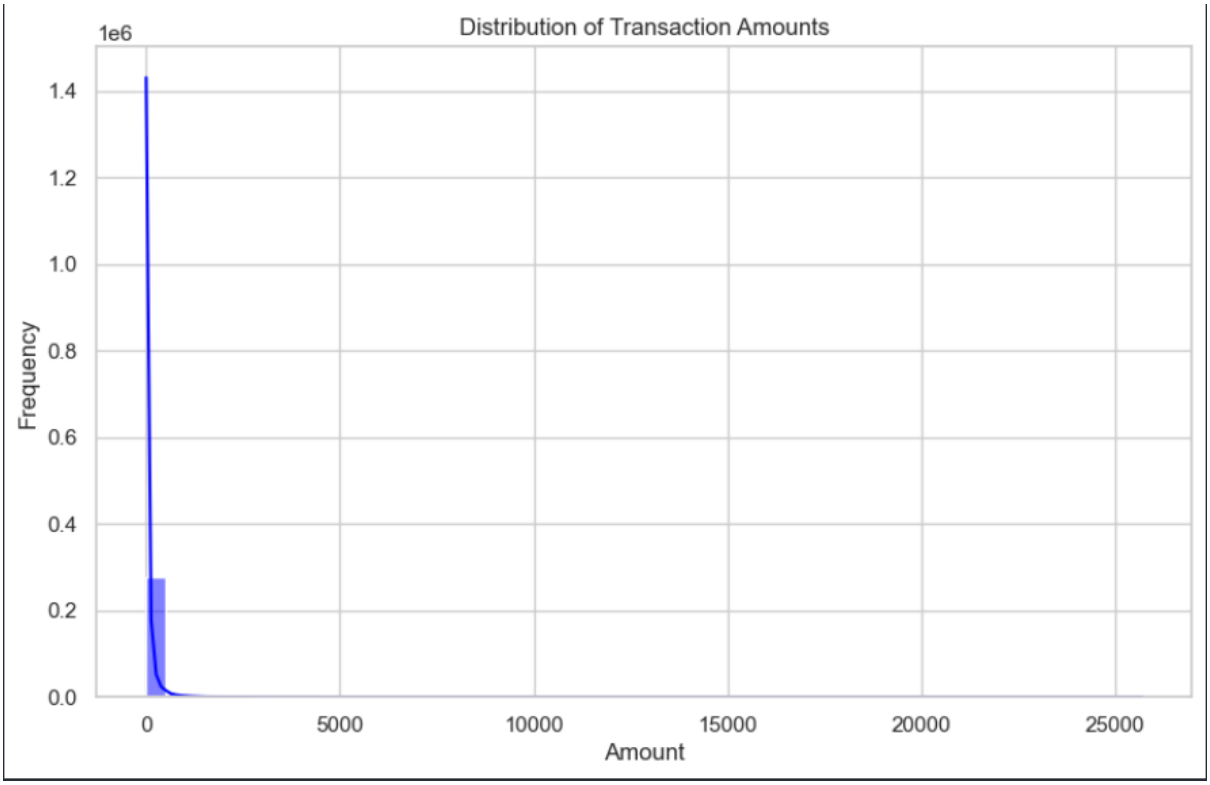
```
data = pd.read_csv("creditcard.csv")
```

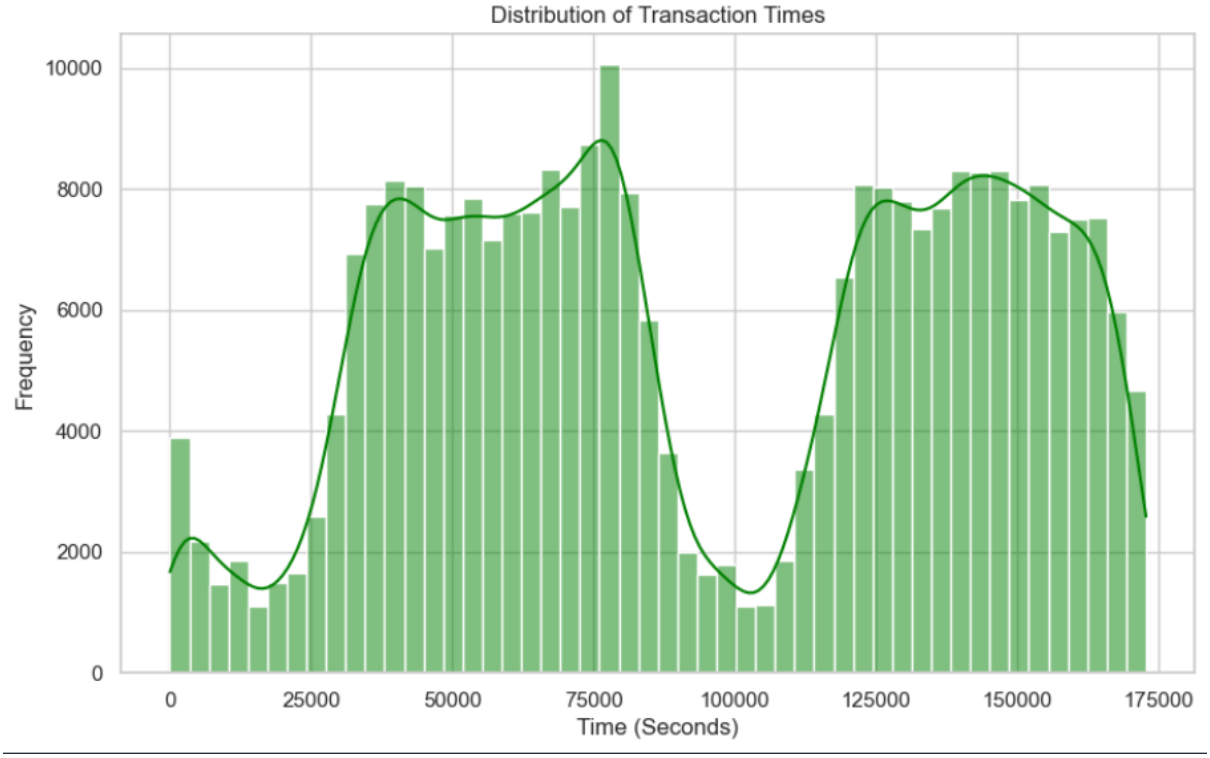
سپس ساختار آن را بررسی می کنیم.

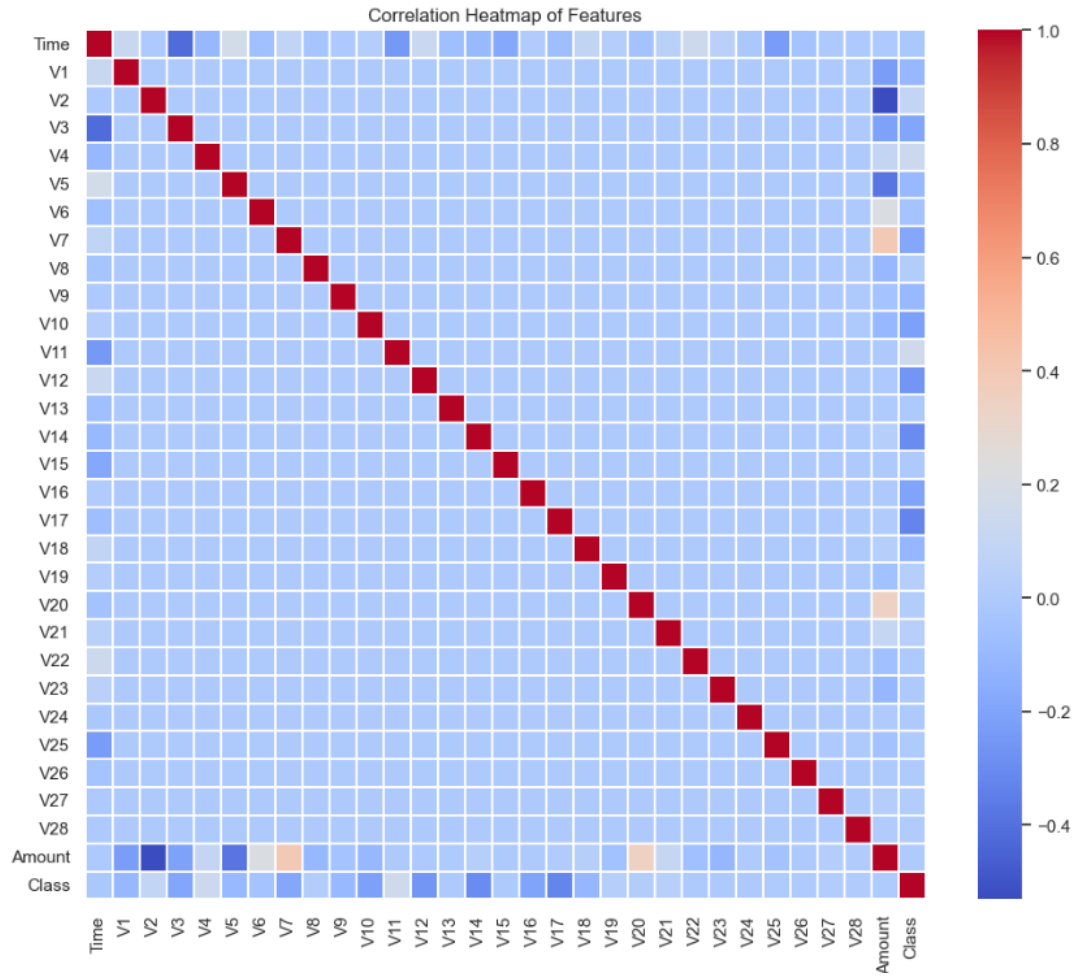
```
data.info()
```

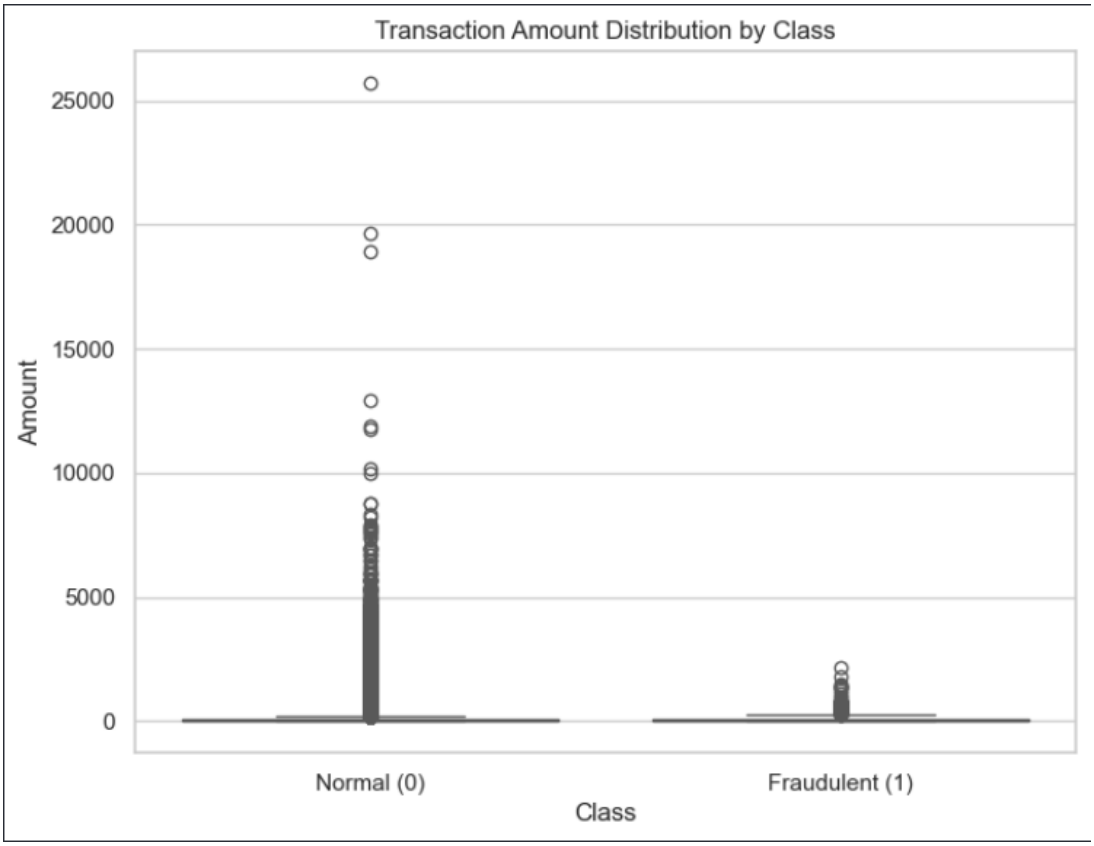
سپس توزیع کلاس های هدف را بررسی کرده و نمودارهای مناسبی برای نمایش توزیع کلاس ها رسم می کنیم.




Distribution of Transaction Amounts
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
0
5000
10000
15000
20000
25000
Amount


Distribution of Transaction Times
10000
8000
6000
4000
2000
0
Frequency
0
25000
50000
75000
100000
125000
150000
175000
Time (Seconds)


Correlation Heatmap of Features
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


Transaction Amount Distribution by Class
Amount
25000
20000
15000
10000
5000
0
Normal (0)
Fraudulent (1)
Class

مجموعه داده ما دارای یک ویژگی به نام Amount است که نشان دهنده مبلغ تراکنش است و درمقایسه با سایر ویژگی ها متفاوت است. ویژگی Time نیز مقیاس خاصی دارد. بنابراین این دو مقدار را scale می کنیم.

```
scaler = StandardScaler()
data[['Time', 'Amount']] = scaler.fit_transform(data[['Time', 'Amount']])
```

در الگوریتم‌های درخت تصمیم (Decision Trees)، مقیاس‌بندی یا استانداردسازی ویژگی‌ها معمولاً تاثیر چندانی بر عملکرد مدل ندارد. درخت تصمیم بر اساس مقادیر ویژگی‌ها، داده‌ها را به صورت متوالی به دو بخش تقسیم می‌کند و از معیارهایی مانند ناخالصی جینی (Gini Impurity) یا آنتروپی برای انتخاب بهترین نقطه تقسیم استفاده می‌کند. در واقع، این الگوریتم‌ها به مقدار مطلق ویژگی‌ها توجه دارند و به مقیاس ویژگی‌ها حساس نیستند.

در مرحله بعد برای تقسیم داده‌ها به سه مجموعه آموزش، اعتبارسنجی و تست با رعایت توزیع کلاس‌ها (با توجه به نامتوازن بودن داده‌ها)، از روش تقسیم تصادفی طبقاتی (Stratified Sampling) استفاده می‌کنیم. این روش تضمین می‌کند که نسبت نمونه‌های هر کلاس در هر مجموعه مشابه نسبت آن‌ها در کل مجموعه داده باشد.

```
X = data.drop('Class', axis=1)
y = data['Class']

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=0.25,
stratify=y_train_val, random_state=42)
```

```
Training set shape: (170883, 30)
Validation set shape: (56962, 30)
Test set shape: (56962, 30)

Class distribution in Training set:
 Class
0    0.998274
1    0.001726
Name: proportion, dtype: float64

Class distribution in Validation set:
 Class
0    0.998262
1    0.001738
Name: proportion, dtype: float64

Class distribution in Test set:
 Class
0    0.99828
1    0.00172
Name: proportion, dtype: float64
```

با استفاده از DecisionTreeClassifier مدل درخت تصمیم را ایجاد می‌کنیم و پارامتر random_state را تنظیم می‌کنیم تا نتایج قابل تکرار باشند. سپس مدل را با داده‌های آموزشی X_train و y_train آموزش می‌دهیم. مدل را برای داده‌های تست (X_test) به کار می‌بریم و پیش‌بینی‌های آن را ذخیره می‌کنیم.

```
Accuracy: 0.9991222218320986

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.74      0.76      0.75        98

    accuracy                           1.00     56962
   macro avg       0.87      0.88      0.87     56962
weighted avg       1.00      1.00      1.00     56962


Confusion Matrix:
 [[56838    26]
 [   24    74]]
```

در مرحله بعد برای بررسی تاثیر معیارهای مختلف ناخالصی بر ساختار درخت و عملکرد مدل، می‌توانیم از سه معیار gini index و Entropy و Misclassification Error استفاده کنیم.

در این کد تابع با معیار ناخالصی مشخص شده یک مدل درخت تصمیم ایجاد کرده و آن را روی داده‌های آموزش آموزش می‌دهد. سپس، پیش‌بینی‌ها را برای داده‌های تست انجام داده و معیارهای مختلف را محاسبه و نمایش می‌دهد. برای هر معیار، مدل درخت تصمیم را آموزش می‌دهیم و سپس دقت، دقت مثبت (Precision)، بازیابی (Recall)، امتیاز F1 و ماتریس درهم‌ریختگی را محاسبه می‌کنیم. ماتریس درهم‌ریختگی به صورت Heatmap نمایش داده می‌شود تا تفاوت‌های عملکرد مدل با هر معیار مشخص شود.

```
Results for criterion: gini
Accuracy: 0.9991222218320986
Precision: 0.74
Recall: 0.7551020408163265
F1 Score: 0.7474747474747474

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.74      0.76      0.75        98

    accuracy                           1.00     56962
   macro avg       0.87      0.88      0.87     56962
weighted avg       1.00      1.00      1.00     56962
```

```
Results for criterion: entropy
Accuracy: 0.9992451107756047
Precision: 0.8021978021978022
Recall: 0.7448979591836735
F1 Score: 0.7724867724867726

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.80      0.74      0.77        98

    accuracy                           1.00     56962
   macro avg       0.90      0.87      0.89     56962
weighted avg       1.00      1.00      1.00     56962
```

```
Results for criterion: log_loss
Accuracy: 0.9992451107756047
Precision: 0.8021978021978022
Recall: 0.7448979591836735
F1 Score: 0.7724867724867726

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.80      0.74      0.77        98

    accuracy                           1.00     56962
   macro avg       0.90      0.87      0.89     56962
weighted avg       1.00      1.00      1.00     56962
```

نتایج به دست آمده برای Gini index به مقدار بسیار بسیار کمتری از دو مورد دیگر ضعیف تر است.

برای جلوگیری از بیش‌برازش (Overfitting) در مدل‌های درخت تصمیم، استفاده از پارامترهایی مانند max_depth، min_samples_split و min_samples_leaf کمک می‌کند که رشد درخت محدود شود و مدل بتواند بهتر بر روی داده‌های جدید تعمیم یابد.

برای یافتن بهترین مقدار برای این پارامترها، از GridSearchCV در Scikit-Learn استفاده می‌کنیم.

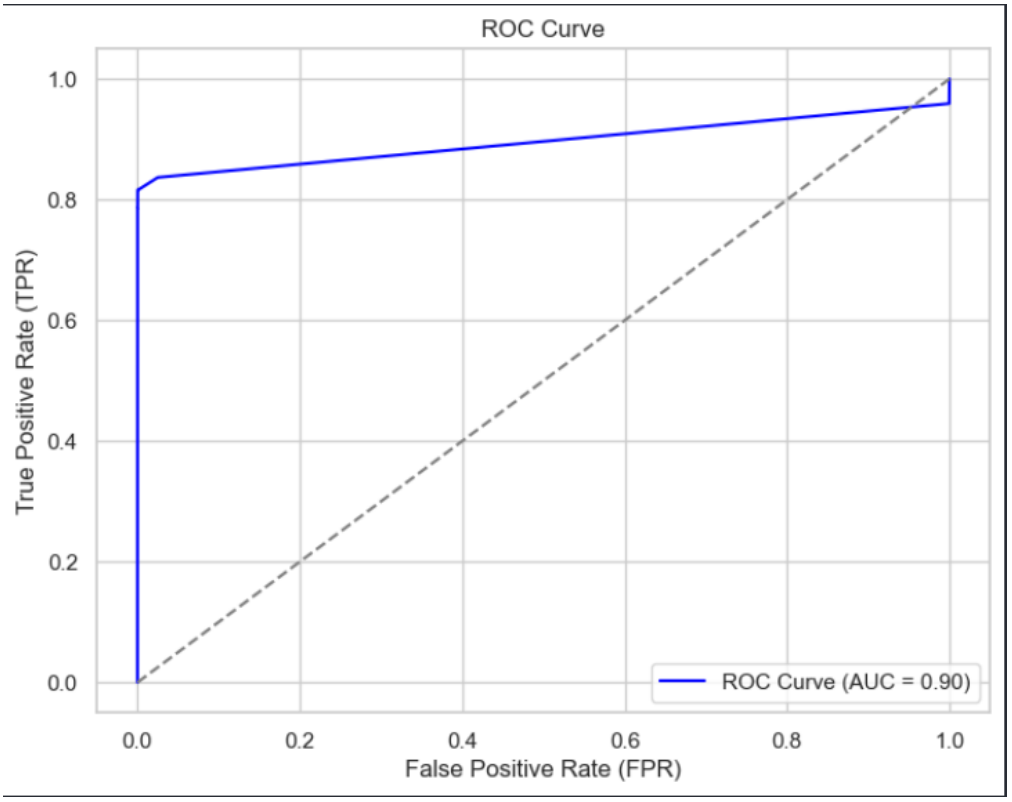
```
Best parameters found: {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 10}
Best F1 Score achieved: 0.8159780070842018

Test Accuracy: 0.9994382219725431
Test Precision: 0.875
Test Recall: 0.7857142857142857
Test F1 Score: 0.8279569892473119

Confusion Matrix:
 [[56853    11]
 [   21    77]]
```

در این مرحله بهترین مدل و دقت به دست آمد.

در مرحله آخر نمودار ROC را رسم می کنیم.



AUC نزدیک به 1: نشان‌دهنده عملکرد خوب مدل در تفکیک کلاس‌ها است.

AUC نزدیک به 0.5: مدل عملکردی تصادفی دارد.