In [12]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve

from sklearn.model_selection import train_test_split
```

In [3]:
```python
# ccdf = Credit_card_Data_frame
ccdf = pd.read_csv('creditcard_2023.csv')
ccdf.head(10)
```

Out[3]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
0	0	-0.260648	-0.469648	2.496266	-0.083724	0.129681	0.732898	0.519014	-0.130006	0.727159	...	-0.110552	0.217606	-0.134794
1	1	0.985100	-0.356045	0.558056	-0.429654	0.277140	0.428605	0.406466	-0.133118	0.347452	...	-0.194936	-0.605761	0.079469
2	2	-0.260272	-0.949385	1.728538	-0.457986	0.074062	1.419481	0.743511	-0.095576	-0.261297	...	-0.005020	0.702906	0.945045
3	3	-0.152152	-0.508959	1.746840	-1.090178	0.249486	1.143312	0.518269	-0.065130	-0.205698	...	-0.146927	-0.038212	-0.214048
4	4	-0.206820	-0.165280	1.527053	-0.448293	0.106125	0.530549	0.658849	-0.212660	1.049921	...	-0.106984	0.729727	-0.161666
5	5	0.025302	-0.140514	1.191138	-0.707979	0.430490	0.458973	0.611050	-0.092629	0.180811	...	-0.187739	-0.538518	-0.050465
6	6	1.016482	-0.397181	0.497868	-0.144463	0.331022	0.629243	0.431262	-0.134007	0.796159	...	-0.171137	-0.287017	-0.178197
7	7	-0.051306	-0.007194	1.139941	-0.877880	0.684668	0.714326	0.892615	-0.908409	0.901938	...	0.620676	-0.920426	0.034660
8	8	-0.130680	-0.349547	0.425786	-0.760444	1.702777	2.324816	0.568968	0.049100	0.273118	...	-0.132787	-0.284700	-0.227779
9	9	0.058419	-0.093507	1.117270	-0.735172	0.466111	0.332371	0.683425	-0.136674	0.096409	...	-0.203634	-0.601581	-0.145082

10 rows × 31 columns

In [4]:
```
ccdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568630 entries, 0 to 568629
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   id      568630 non-null  int64  
 1   V1      568630 non-null  float64
 2   V2      568630 non-null  float64
 3   V3      568630 non-null  float64
 4   V4      568630 non-null  float64
 5   V5      568630 non-null  float64
 6   V6      568630 non-null  float64
 7   V7      568630 non-null  float64
 8   V8      568630 non-null  float64
 9   V9      568630 non-null  float64
 10  V10     568630 non-null  float64
 11  V11     568630 non-null  float64
 12  V12     568630 non-null  float64
 13  V13     568630 non-null  float64
 14  V14     568630 non-null  float64
 15  V15     568630 non-null  float64
 16  V16     568630 non-null  float64
 17  V17     568630 non-null  float64
 18  V18     568630 non-null  float64
 19  V19     568630 non-null  float64
 20  V20     568630 non-null  float64
 21  V21     568630 non-null  float64
 22  V22     568630 non-null  float64
 23  V23     568630 non-null  float64
 24  V24     568630 non-null  float64
 25  V25     568630 non-null  float64
 26  V26     568630 non-null  float64
 27  V27     568630 non-null  float64
 28  V28     568630 non-null  float64
 29  Amount  568630 non-null  float64
 30  Class   568630 non-null  int64  
dtypes: float64(29), int64(2)
memory usage: 134.5 MB
```

In [5]:
```
ccdf.isnull().sum()
```

Out[5]:
```
id        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
```

In [6]:
```
sns.countplot(ccdf['Class'])
plt.title("Class Distribution")
```

Out[6]:
```
Text(0.5, 1.0, 'Class Distribution')
```

In [7]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
ccdf['Amount'] = scaler.fit_transform(ccdf['Amount'].values.reshape(-1, 1))

ccdf.head(5)
```

Out[7]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
0	0	-0.260648	-0.469648	2.496266	-0.083724	0.129681	0.732898	0.519014	-0.130006	0.727159	...	-0.110552	0.217606	-0.134794
1	1	0.985100	-0.356045	0.558056	-0.429654	0.277140	0.428605	0.406466	-0.133118	0.347452	...	-0.194936	-0.605761	0.079469
2	2	-0.260272	-0.949385	1.728538	-0.457986	0.074062	1.419481	0.743511	-0.095576	-0.261297	...	-0.005020	0.702906	0.945045
3	3	-0.152152	-0.508959	1.746840	-1.090178	0.249486	1.143312	0.518269	-0.065130	-0.205698	...	-0.146927	-0.038212	-0.214048
4	4	-0.206820	-0.165280	1.527053	-0.448293	0.106125	0.530549	0.658849	-0.212660	1.049921	...	-0.106984	0.729727	-0.161666

5 rows × 31 columns

In [8]:
```
X = ccdf.drop(['Class','id'], axis=1)
y = ccdf['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

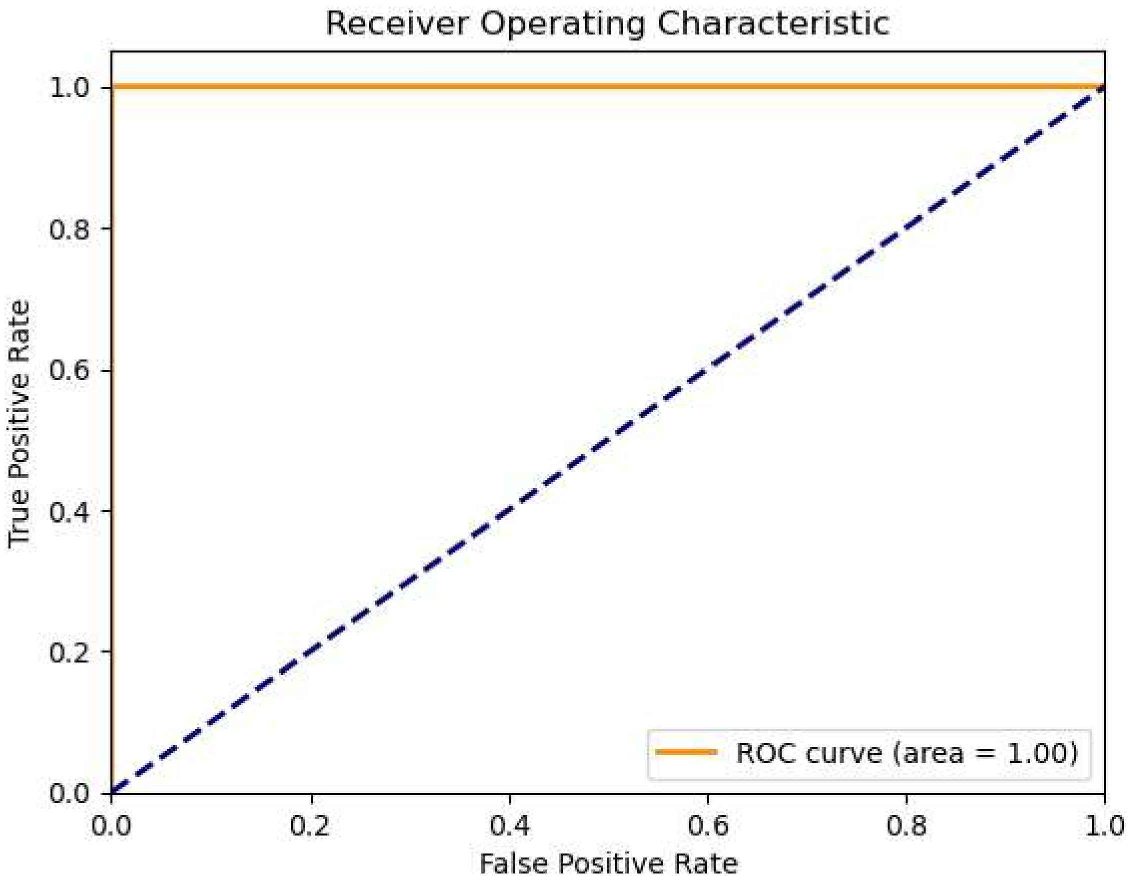
In [9]:
```
random_forest_model = RandomForestClassifier()

random_forest_model.fit(X_train, y_train)
```

Out[9]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [11]:

```
# Make predictions
y_pred = random_forest_model.predict(X_test)

# Print classification report
print(classification_report(y_test, y_pred))

# Compute ROC curve and AUC
y_prob = random_forest_model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = roc_auc_score(y_test, y_prob)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85149
           1       1.00      1.00      1.00     85440

    accuracy                           1.00    170589
   macro avg       1.00      1.00      1.00    170589
weighted avg       1.00      1.00      1.00    170589
```


Receiver Operating Characteristic
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve (area = 1.00)

In [*]: