In [26]:
```
# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier  # For building the frau

from sklearn.model_selection import train_test_split  # To split data into t

from sklearn.metrics import (

    accuracy_score, precision_score, recall_score, f1_score, confusion_matri

    roc_curve, auc  # For evaluating the model

)

import matplotlib.pyplot as plt  # For visualization
```

In [28]:
```
# The dataset contains transaction details with labeled fraud cases (isFraud
file_path = r"/Users/kattykya/Documents/python quiz/python quiz 5/frauddatas
df = pd.read_csv(file_path)  # Load the dataset into a DataFrame
```

In [29]:
```
print(df)
```

```
         step      type      amount     nameOrig  oldbalanceOrg  \
0           1   PAYMENT     9839.64  C1231006815      170136.00
1           1   PAYMENT     1864.28  C1666544295       21249.00
2           1  TRANSFER      181.00  C1305486145         181.00
3           1  CASH_OUT      181.00   C840083671         181.00
4           1   PAYMENT    11668.14  C2048537720       41554.00
...       ...       ...         ...          ...            ...
6362615   743  CASH_OUT   339682.13   C786484425      339682.13
6362616   743  TRANSFER  6311409.28  C1529008245     6311409.28
6362617   743  CASH_OUT  6311409.28  C1162922333     6311409.28
6362618   743  TRANSFER   850002.52  C1685995037      850002.52
6362619   743  CASH_OUT   850002.52  C1280323807      850002.52

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFrau
d  \
0             160296.36  M1979787155            0.00            0.00
0
1              19384.72  M2044282225            0.00            0.00
0
2                  0.00   C553264065            0.00            0.00
1
3                  0.00    C38997010        21182.00            0.00
1
4              29885.86  M1230701703            0.00            0.00
0
...                 ...          ...             ...             ...      ..
.
6362615            0.00   C776919290            0.00       339682.13
1
6362616            0.00  C1881841831            0.00            0.00
1
6362617            0.00  C1365125890        68488.84      6379898.11
1
6362618            0.00  C2080388513            0.00            0.00
1
6362619            0.00   C873221189      6510099.11      7360101.63
1

         isFlaggedFraud
0                     0
1                     0
2                     0
3                     0
4                     0
...                 ...
6362615               0
6362616               0
6362617               0
6362618               0
6362619               0

[6362620 rows x 11 columns]
```

In [32]:

```
# Step 2: Explore and preprocess the data

# ------------------------------------------------------------
```

```
# Let's preprocess the data to prepare it for training the model.


# Encode the 'type' column into numerical values (e.g., CASH-IN -> 0, PAYMEN

df['type'] = df['type'].astype('category').cat.codes


# Fill any missing values with 0 (common for balance columns)

df.fillna(0, inplace=True)
```

In [34]:

```
# Step 3: Define features (X) and the target (y)

# ------------------------------------------------------------

# X: Features we use to predict fraud (independent variables)

# y: The target column ('isFraud'), indicating if a transaction is fraudulen

X = df[['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbal

y = df['isFraud']
```

In [36]:

```
# Step 4: Split the data into training and testing sets

# ------------------------------------------------------------

# Splitting the dataset into 70% training and 30% testing.

# This ensures the model is evaluated on unseen data for reliability.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran
```

In [38]:

```
# Step 5: Train the model

# ------------------------------------------------------------

# Using a Random Forest Classifier because:

# - It works well with imbalanced datasets

# - It handles both categorical and numerical features well

model = RandomForestClassifier(random_state=42)

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

Out[38]:

```
▾       RandomForestClassifier        i ?

RandomForestClassifier(random_state=42)
```

In [40]:
```
# Step 6: Make predictions

# ------------------------------------------------------------

# Predict the outcomes for the test data

y_pred = model.predict(X_test)  # Predicted class labels (0 or 1)

y_prob = model.predict_proba(X_test)[:, 1]  # Predicted probabilities for th
```

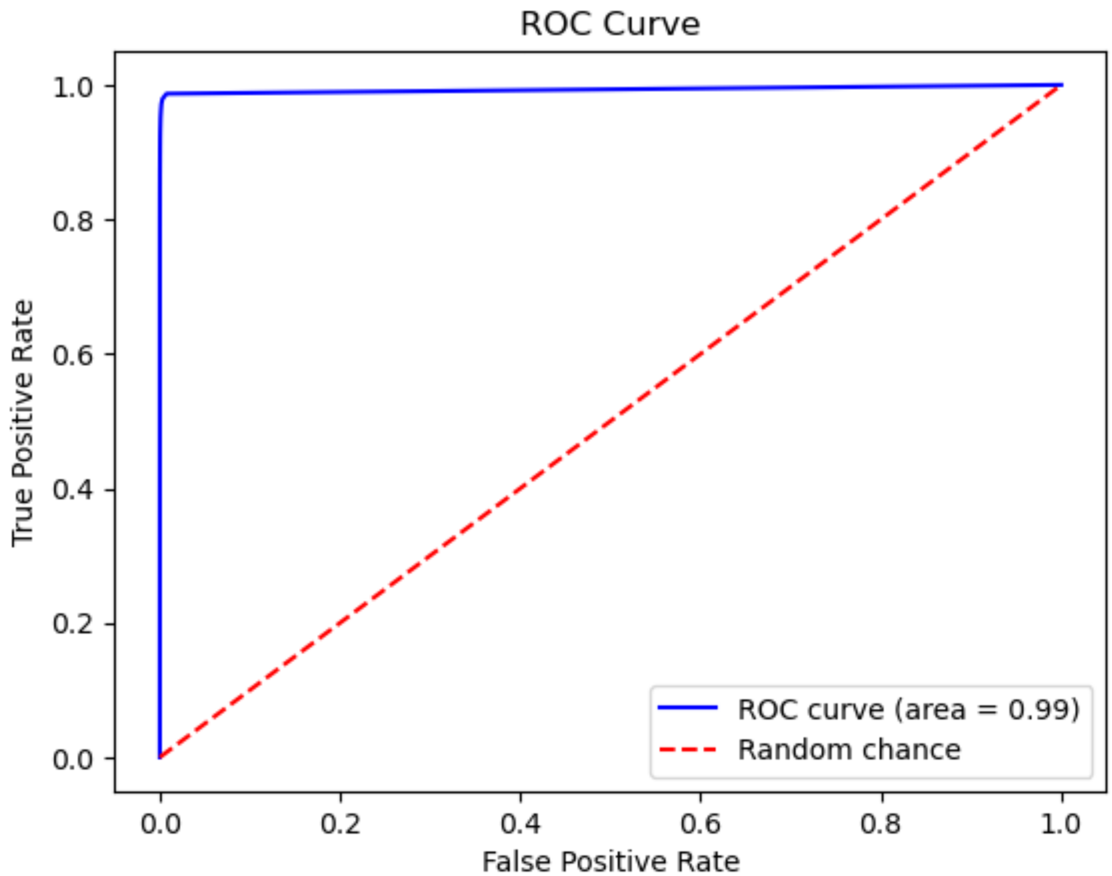
In [42]:
```
# Step 7: Evaluate the model

# ------------------------------------------------------------

# Metrics are used to assess the model's performance

print("=== Evaluation Metrics ===")

accuracy = accuracy_score(y_test, y_pred)  # Overall correctness

precision = precision_score(y_test, y_pred)  # Fraction of correct fraud pre

recall = recall_score(y_test, y_pred)  # Fraction of fraud cases detected

f1 = f1_score(y_test, y_pred)  # Harmonic mean of precision and recall

conf_matrix = confusion_matrix(y_test, y_pred)  # Breakdown of predictions (



# Print the metrics

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")

print(f"Confusion Matrix:\n{conf_matrix}")
```

```
=== Evaluation Metrics ===
Accuracy: 0.999705572023265
Precision: 0.9785385794583547
Recall: 0.7864476386036962
F1 Score: 0.8720400728597449
Confusion Matrix:
[[1906309      42]
 [    520    1915]]
```

In [44]:
```
# Step 8: Plot the ROC Curve

# ------------------------------------------------------------

# The ROC Curve shows the trade-off between the true positive rate (TPR) and
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

roc_auc = auc(fpr, tpr)  # Area under the curve (AUC)



# Plot the ROC Curve

plt.figure()

plt.plot(fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})', color='blue')

plt.plot([0, 1], [0, 1], 'k--', label='Random chance', color='red')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc='lower right')

plt.show()
```

```
/var/folders/1m/rzrgll2n2tbd654y_lgs97k40000gn/T/ipykernel_53275/295897862
4.py:19: UserWarning: color is redundantly defined by the 'color' keyword ar
gument and the fmt string "k--" (-> color='k'). The keyword argument will ta
ke precedence.
  plt.plot([0, 1], [0, 1], 'k--', label='Random chance', color='red')
```



In [46]:
```
# Step 9: Analyze Feature Importance

# ------------------------------------------------------------

# Feature importance helps us understand which features contribute most to f

importances = model.feature_importances_

feature_importance = pd.DataFrame({'Feature': X.columns, 'Importance': impor



print("\n=== Feature Importance ===")

print(feature_importance)
```

```
=== Feature Importance ===
          Feature  Importance
6  newbalanceDest    0.329721
3   oldbalanceOrg    0.268478
2          amount    0.165631
0            step    0.092602
5  oldbalanceDest    0.067778
1            type    0.047391
4  newbalanceOrig    0.028399
```

In [48]:
```
# Step 10: Save the cleaned data or model for future use

# Save the cleaned data to a CSV file
```

```
df.to_csv("cleaned_fraud_data.csv", index=False)  # Saves without the index

print("Cleaned data saved as 'cleaned_fraud_data.csv'")



# ------------------------------------------------------------

# Save the trained model for later use (e.g., deployment)

import joblib

joblib.dump(model, "fraud_detection_model.pkl")  # Save the model as a .pkl
```

```
Cleaned data saved as 'cleaned_fraud_data.csv'
```

Out[48]:

```
['fraud_detection_model.pkl']
```

In [ ]: