```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, roc_curve, auc
```

```
# Load the dataset
file_path = "netflix_users_expanded.csv"
df = pd.read_csv('netflix_users_expanded.csv')
```

```
# Preprocessing: Handling missing values
df.dropna(inplace=True)
```

```
# Encoding categorical variables if any
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

```
# Splitting data into features and target (Assuming last column is target)
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

```
# Standardizing features
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
# Splitting dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Model training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

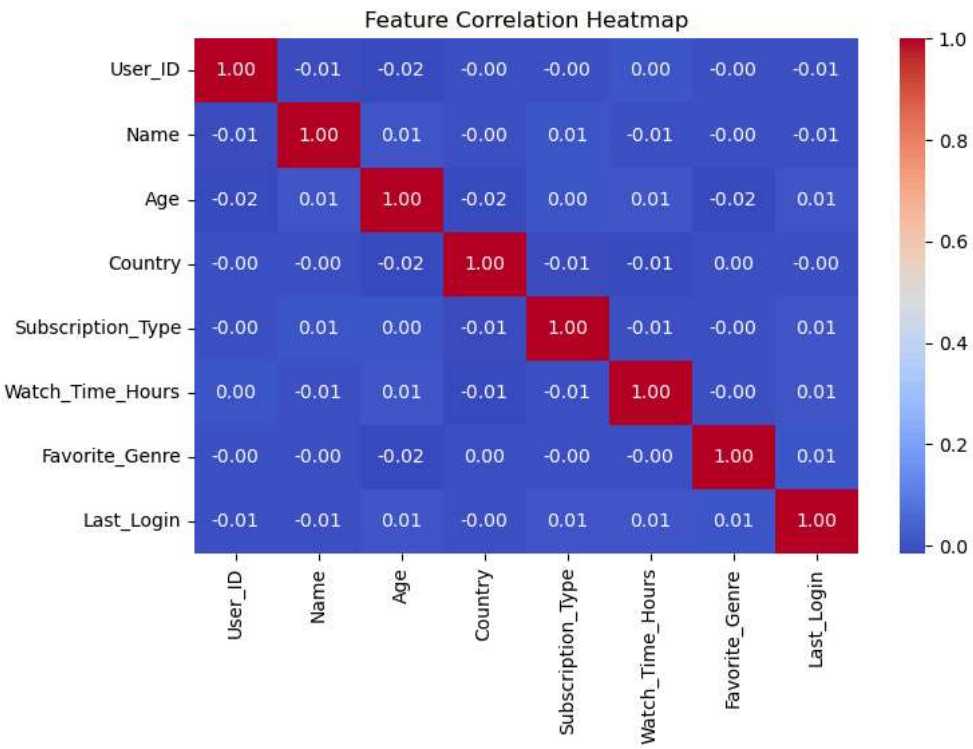
```
# Model evaluation
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9906
Classification Report:
                 precision    recall  f1-score   support

              0       1.00      1.00      1.00        54
              1       1.00      1.00      1.00        45
              2       1.00      1.00      1.00        49
              3       1.00      1.00      1.00        52
              4       1.00      1.00      1.00        40
              5       1.00      1.00      1.00        53
              6       1.00      1.00      1.00        69
              7       1.00      1.00      1.00        64
              8       1.00      1.00      1.00        42
              9       1.00      0.93      0.96        54
             10       1.00      1.00      1.00        52
             11       1.00      1.00      1.00        58
             12       1.00      1.00      1.00        52
             13       1.00      0.91      0.95        45
             14       1.00      1.00      1.00        50
             15       1.00      1.00      1.00        65
             16       0.92      1.00      0.96        46
             17       1.00      0.93      0.96        54
             18       1.00      1.00      1.00        47
             19       1.00      0.92      0.96        52
             20       1.00      1.00      1.00        50
```

```
21       1.00      1.00      1.00        47
22       1.00      0.94      0.97        66
23       1.00      1.00      1.00        48
24       1.00      0.92      0.96        51
25       1.00      1.00      1.00        66
26       1.00      1.00      1.00        67
27       1.00      0.93      0.96        55
28       1.00      1.00      1.00        53
29       1.00      1.00      1.00        40
30       1.00      0.92      0.96        51
31       0.93      0.93      0.93        55
32       0.93      1.00      0.96        55
33       1.00      1.00      1.00        83
34       1.00      1.00      1.00        73
35       1.00      1.00      1.00        75
36       1.00      1.00      1.00        57
37       1.00      0.92      0.96        49
38       1.00      1.00      1.00        49
39       1.00      1.00      1.00        49
40       1.00      1.00      1.00        66
41       1.00      1.00      1.00        69
42       1.00      1.00      1.00        61
43       1.00      1.00      1.00        73
44       0.93      1.00      0.96        54
45       1.00      1.00      1.00        53
46       1.00      1.00      1.00        62
47       1.00      1.00      1.00        52
48       0.92      1.00      0.96        49
49       1.00      1.00      1.00        58
50       1.00      1.00      1.00        56
51       1.00      1.00      1.00        50
52       0.94      1.00      0.97        58
```

```
plt.figure(figsize=(8, 5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Heatmap")
plt.show()
```



Start coding or generate with AI.