```
# RFE Implementation

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score

# loading dataset from UCI Repository
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
columns = ["age", "workclass", "fnlwgt", "education", "education-num", "marital-status",
           "occupation", "relationship", "race", "gender", "capital-gain", "capital-loss",
           "hours-per-week", "native-country", "income"]

df = pd.read_csv(url, names=columns, na_values="?", skipinitialspace=True)

# drop missing values
df.dropna(inplace=True)

# converting categorical variables to numerical using Label Encoding
for column in df.select_dtypes(include=['object']).columns:
    df[column] = LabelEncoder().fit_transform(df[column])

# spliting features and target variable
X = df.drop("income", axis=1)
y = df["income"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# applying RFE with Random Forest (Selecting Top 10 Features)
rfe = RFE(estimator=RandomForestClassifier(random_state=42), n_features_to_select=10)
rfe.fit(X_train, y_train)

# getting selected features
selected_features = X_train.columns[rfe.support_]
print(f"Selected Features: {list(selected_features)}")

# reducing dataset to selected features
X_train_rfe = X_train[selected_features]
X_test_rfe = X_test[selected_features]

# training a model with selected features
model_rfe = RandomForestClassifier(random_state=42)
model_rfe.fit(X_train_rfe, y_train)

# evaluating model
y_pred_rfe = model_rfe.predict(X_test_rfe)
accuracy_rfe = accuracy_score(y_test, y_pred_rfe)

print(f"\nAccuracy after RFE: {accuracy_rfe:.4f}")

# visualizing feature importances
importances = model_rfe.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': selected_features, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10,5))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df, palette="coolwarm")
plt.title('Feature Importance After RFE')
plt.show()
```

```
Selected Features: ['age', 'workclass', 'fnlwgt', 'education-num', 'marital-status', 'occupation', 'relationship', 'capital-

Accuracy after RFE: 0.8551
<ipython-input-2-4e26a7e36ce3>:63: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and

  sns.barplot(x='Importance', y='Feature', data=feature_importance_df, palette="coolwarm")
```