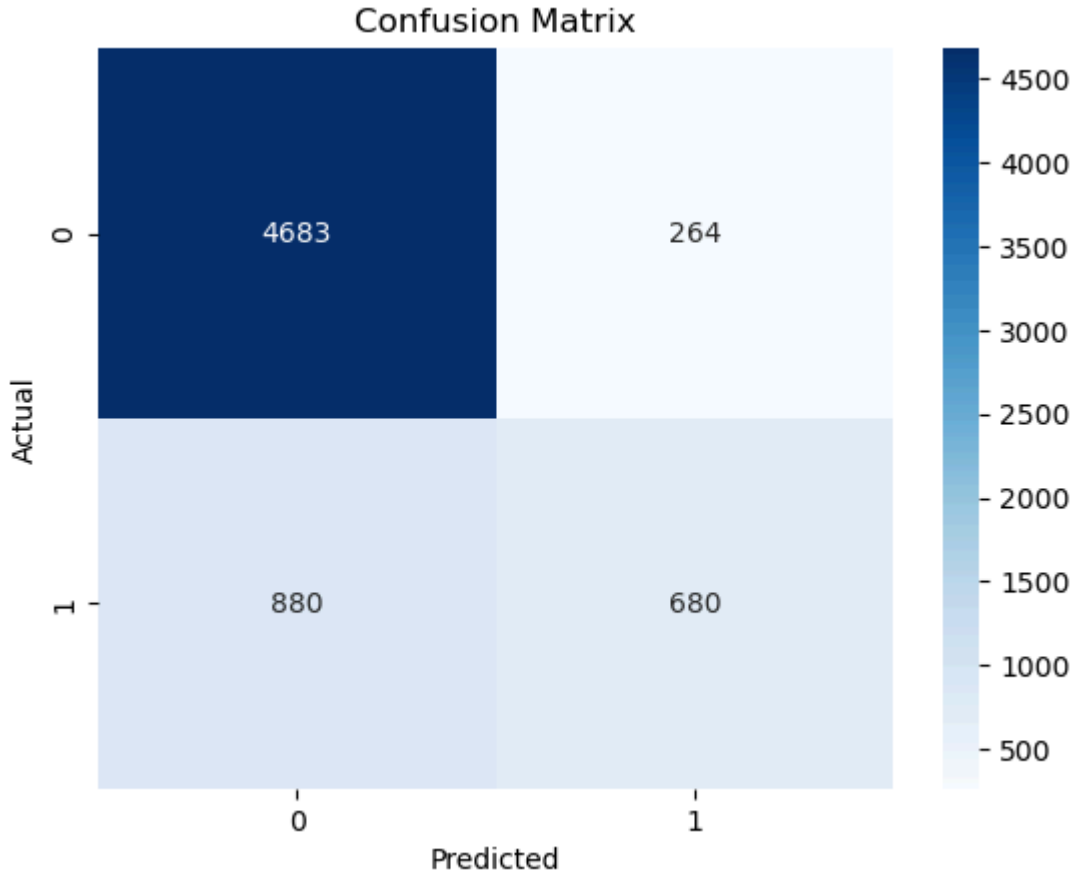
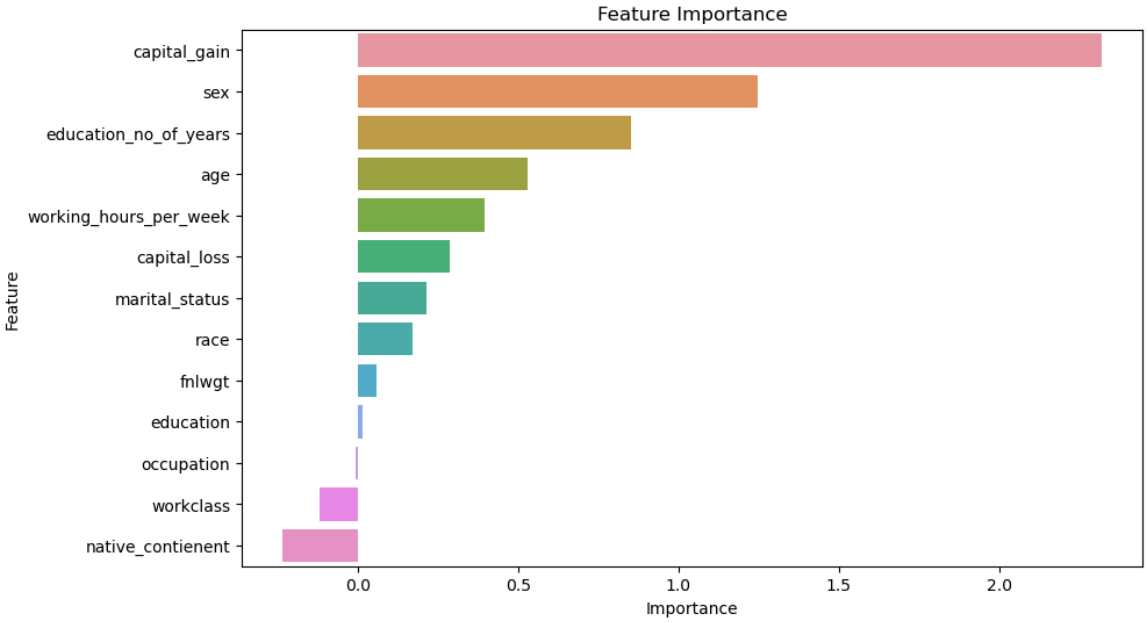
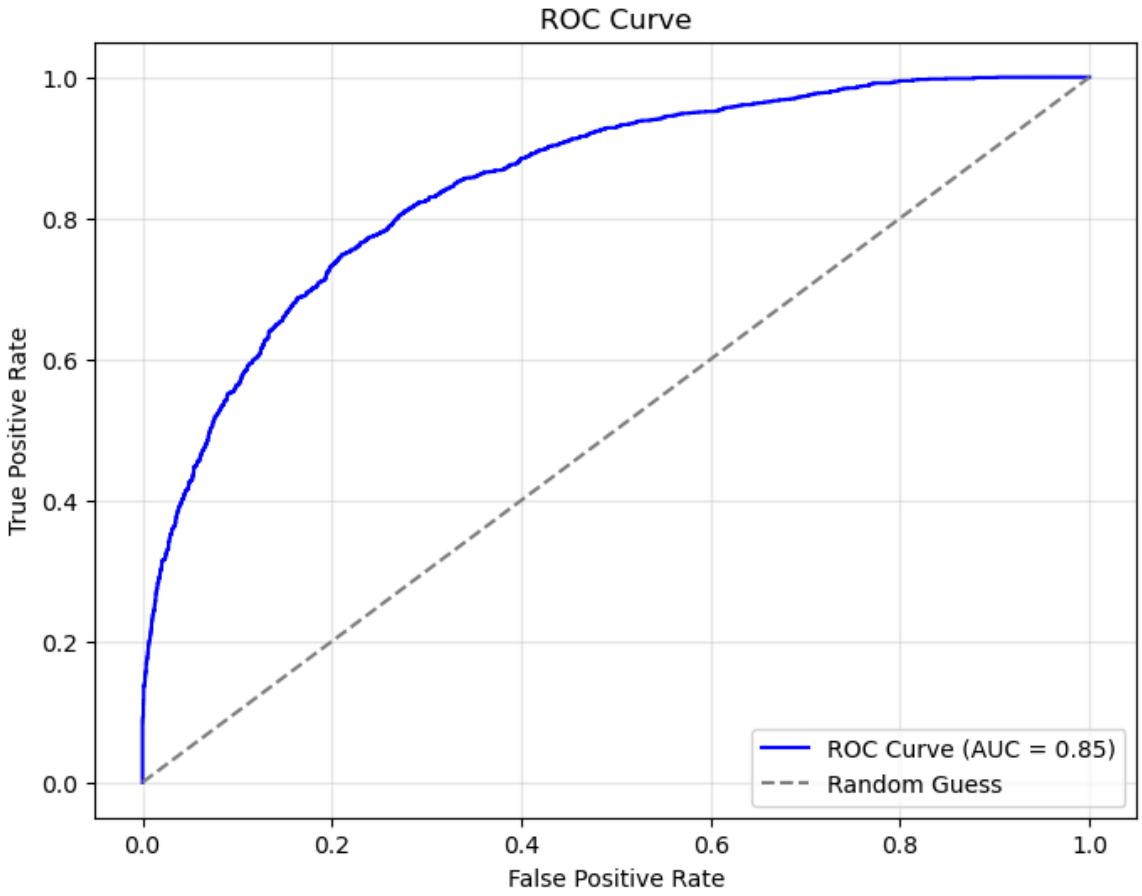
In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, roc
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
data = pd.read_csv('Downloads/WHO.csv')

# Step 1: Data Cleaning
# Convert categorical 'salary' into binary
data['salary'] = data['salary'].apply(lambda x: 1 if x.strip() == '>50

# Encode categorical variables
categorical_features = ['workclass', 'education', 'marital_status', 'o
label_encoders = {}
for col in categorical_features:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Step 2: Scale numerical features
scaler = StandardScaler()
numerical_features = ['age', 'fnlwgt', 'education_no_of_years', 'capit
data[numerical_features] = scaler.fit_transform(data[numerical_feature

# Step 3: Prepare features and target
X = data.drop(columns=['salary'])  # Features
y = data['salary']                 # Target

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Step 4: Train a Logistic Regression Model
log_reg_model = LogisticRegression(random_state=42, max_iter=1000)
log_reg_model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = log_reg_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pr

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Step 6: Feature Importance
# Logistic Regression coefficients as feature importance
feature_importances = log_reg_model.coef_[0]
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feat

# Visualize feature importance
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importance')
plt.show()

# Step 7: ROC Curve and AUC
y_probs = log_reg_model.predict_proba(X_test)[:, 1]  # Predicted proba
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f
plt.plot([0, 1], [0, 1], color='gray', linestyle='--', label='Random C
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(alpha=0.3)
plt.show()

# Display AUC
print(f'ROC AUC: {roc_auc:.4f}')
```

```
Accuracy: 0.8241893345627785

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4947
           1       0.72      0.44      0.54      1560

    accuracy                           0.82      6507
   macro avg       0.78      0.69      0.72      6507
weighted avg       0.81      0.82      0.81      6507
```


Confusion Matrix
4683
264
880
680
Actual
0
1
Predicted
0
1
4500
4000
3500
3000
2500
2000
1500
1000
500


Feature Importance
capital_gain
sex
education_no_of_years
age
working_hours_per_week
capital_loss
marital_status
race
fnlwgt
education
occupation
workclass
native_contienent
Feature
0.0
0.5
1.0
1.5
2.0
Importance




```
ROC AUC: 0.8496
```

In [ ]:

In [ ]: