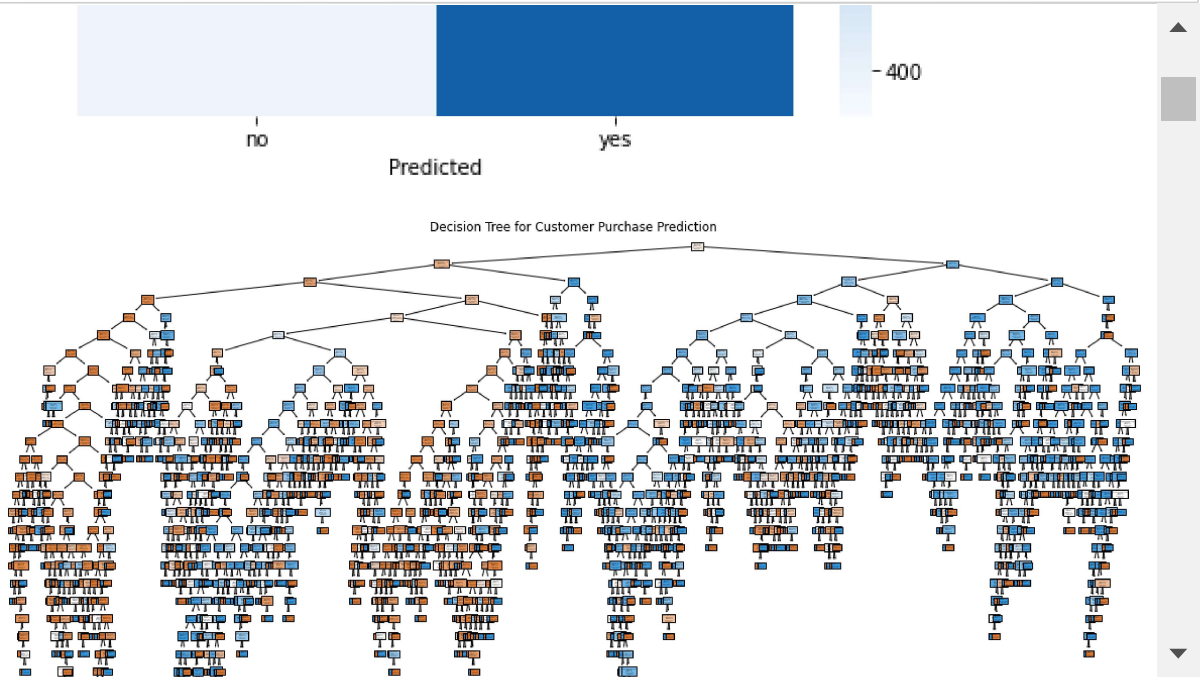
In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree

data = pd.read_csv("C:/Users/91988/Downloads/bank-marketing Dataset.csv")

print(data.head())
print(data.isnull().sum())
data['deposit'] = data['deposit'].astype('category')

# Exploratory Data Analysis (EDA)
sns.countplot(x='deposit', data=data)
plt.title("Count of Deposit Subscriptions")
plt.xlabel("Deposit Subscription (Yes/No)")
plt.ylabel("Count")
plt.show()

# Prepare features and target variable
X = data.drop('deposit', axis=1)
y = data['deposit']

# Convert categorical variables to dummy/indicator variables
X = pd.get_dummies(X, drop_first=True)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc

# Initialize and train the Decision Tree model
model = DecisionTreeClassifier(random_state=123)
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)

# Plot the Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=model.classes_, yticklabels=model.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
# Visualize the Decision Tree
plt.figure(figsize=(20, 10))
tree.plot_tree(model, filled=True, feature_names=X.columns, class_names=model.
plt.title("Decision Tree for Customer Purchase Prediction")
plt.show()

# Export the decision tree rules
tree_rules = export_text(model, feature_names=list(X.columns))
print(tree_rules)

# Feature Importance Visualization
importance = model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': impo
feature_importance_df = feature_importance_df.sort_values(by='Importance', asc

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



In [ ]: