Practical 7

Logistic Regression and Decision Tree

* Build a logistic regression model to predict a binary outcome.
* Evaluate the model's performance using classification metrics (e.g., accuracy, precision, recall).
* Construct a decision tree model and interpret the decision rules for classification

Code:

# Import Required Libraries 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 StandardScaler from sklearn.linear\_model import LogisticRegression from sklearn.metrics import (

accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report, roc\_curve, auc

)

# Step 1: Create Sample Dataset (Binary Classification) np.random.seed(42)

data = {

'Age': np.random.randint(20, 60, 100),

'Salary': np.random.randint(30000, 100000, 100),

'Purchased': np.random.choice([0, 1], size=100) # 0 = No Purchase, 1 = Purchased

}

df = pd.DataFrame(data) print("\nDataset Sample:\n", df.head())

# Step 2: Split Data into Training & Testing Sets X = df[['Age', 'Salary']]

y = df['Purchased']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 3: Standardize Features (for better performance) scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)

# Step 4: Train Logistic Regression Model log\_model = LogisticRegression() log\_model.fit(X\_train\_scaled, y\_train)

# Step 5: Make Predictions

y\_pred = log\_model.predict(X\_test\_scaled)

# Step 6: Compute Classification Metrics accuracy = accuracy\_score(y\_test, y\_pred) precision = precision\_score(y\_test, y\_pred) recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print("\n🔹 Classification Metrics:") print(f"Accuracy: {accuracy:.4f}") print(f"Precision: {precision:.4f}") print(f"Recall: {recall:.4f}")

print(f"F1-Score: {f1:.4f}")

# Step 7: Confusion Matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(6, 4))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap="Blues", xticklabels=['No Purchase', 'Purchase'], yticklabels=['No Purchase', 'Purchase'])

plt.xlabel("Predicted") plt.ylabel("Actual") plt.title("Confusion Matrix") plt.show()

# Step 8: ROC Curve & AUC Score

y\_prob = log\_model.predict\_proba(X\_test\_scaled)[:, 1] fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

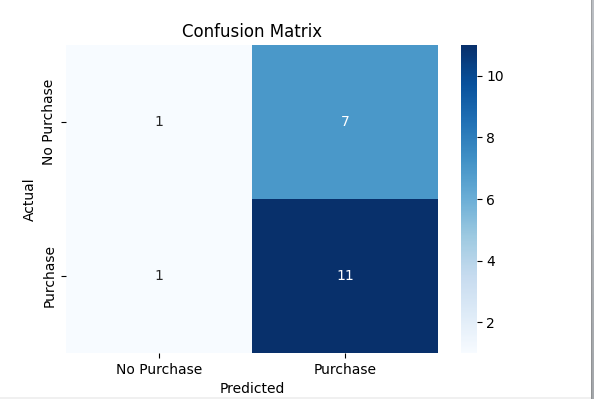
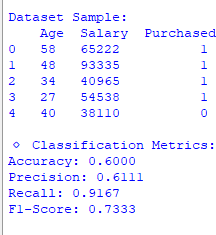
roc\_auc = auc(fpr, tpr)

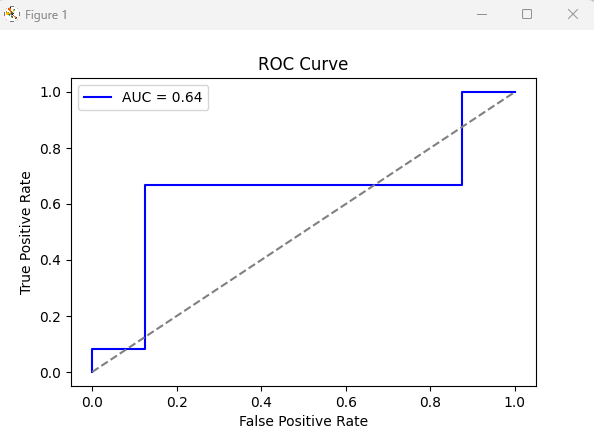
plt.figure(figsize=(6, 4))

plt.plot(fpr, tpr, label=f"AUC = {roc\_auc:.2f}", color='blue') plt.plot([0, 1], [0, 1], linestyle='--', color='grey') # Diagonal Line plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate") plt.title("ROC Curve")

plt.legend() plt.show()





**Construct a decision tree model and interpret the decision rules for classification Code:**

# Import Required Libraries 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 StandardScaler

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Step 1: Create Sample Dataset (Binary Classification) np.random.seed(42)

data = {

'Age': np.random.randint(20, 60, 100),

'Salary': np.random.randint(30000, 100000, 100),

'Purchased': np.random.choice([0, 1], size=100) # 0 = No Purchase, 1 = Purchased

}

df = pd.DataFrame(data) print("\nDataset Sample:\n", df.head())

# Step 2: Split Data into Training & Testing Sets X = df[['Age', 'Salary']]

y = df['Purchased']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 3: Train Decision Tree Classifier

dt\_model = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=42) dt\_model.fit(X\_train, y\_train)

# Step 4: Make Predictions

y\_pred = dt\_model.predict(X\_test)

# Step 5: Compute Model Accuracy accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\n🔹 Decision Tree Accuracy: {accuracy:.4f}")

# Step 6: Display Confusion Matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(6, 4))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap="Blues", xticklabels=['No Purchase', 'Purchase'], yticklabels=['No Purchase', 'Purchase'])

plt.xlabel("Predicted") plt.ylabel("Actual") plt.title("Confusion Matrix") plt.show()

# Step 7: Display Classification Report print("\n Classification Report:") print(classification\_report(y\_test, y\_pred))

# Step 8: Visualize the Decision Tree

plt.figure(figsize=(12, 6))

plot\_tree(dt\_model, feature\_names=['Age', 'Salary'], class\_names=['No Purchase', 'Purchase'], filled=True, rounded=True)

plt.title("Decision Tree Visualization") plt.show()

# Step 9: Interpret Decision Rules from sklearn.tree import export\_text

tree\_rules = export\_text(dt\_model, feature\_names=['Age', 'Salary']) print("\n🔹 Decision Rules:\n", tree\_rules)

