NAME :Ashutosh Rode ROLL NO : 72 DIV : B

Q1] Write a program to Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.

import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt # Load the Iris dataset

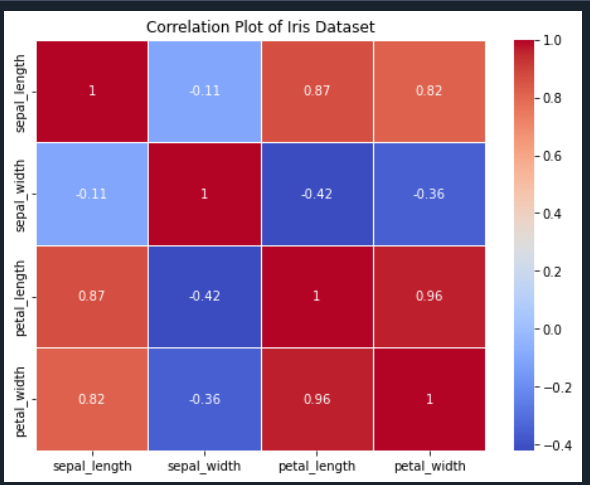
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data" column\_names = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width", "class"] iris\_data = pd.read\_csv(url, names=column\_names)

# Calculate the correlation matrix correlation\_matrix = iris\_data.corr()

# Create a heatmap to visualize the correlations plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", linewidths=0.5) plt.title("Correlation Plot of Iris Dataset")

plt.show()



NAME :Ashutosh Rode ROLL NO : 72 DIV : B

Q2] Write a program to implement linear regression algorithm to create and evaluate a model on a given dataset.

import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score # Sample dataset (replace this with your own dataset)

# X represents the independent variable, and y represents the dependent variable. X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)

y = np.array([2, 4, 5, 4, 5, 6, 8, 8, 10, 10])

# Split the data 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) # Create a linear regression model

model = LinearRegression()

# Train the model on the training data model.fit(X\_train, y\_train)

# Make predictions on the test data y\_pred = model.predict(X\_test)

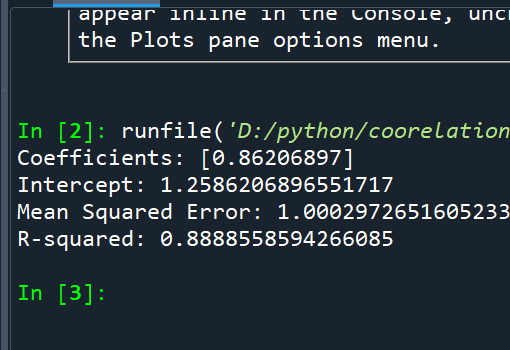
# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred) r2 = r2\_score(y\_test, y\_pred)

# Print the model's coefficients and evaluation metrics print("Coefficients:", model.coef\_)

print("Intercept:", model.intercept\_) print("Mean Squared Error:", mse) print("R-squared:", r2)

NAME :Ashutosh Rode ROLL NO : 72 DIV : B



Q3] Write a program to classify the given dataset using logistic regression and evaluate the model.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report # Generating a synthetic dataset for binary classification

np.random.seed(0)

X = np.random.rand(100, 2)

y = (X[:, 0] + X[:, 1] > 1).astype(int)

# Split the data 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) # Create a logistic regression model

model = LogisticRegression()

# Train the model on the training data model.fit(X\_train, y\_train)

# Make predictions on the test data y\_pred = model.predict(X\_test)

NAME :Ashutosh Rode ROLL NO : 72 DIV : B

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) conf\_matrix = confusion\_matrix(y\_test, y\_pred) class\_report = classification\_report(y\_test, y\_pred) # Print the evaluation metrics

print("Accuracy:", accuracy) print("Confusion Matrix:\n", conf\_matrix) print("Classification Report:\n", class\_report)

# Visualize the decision boundary (works for 2D data only) if X.shape[1] == 2:

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Spectral) h = .02 # step size in the mesh

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

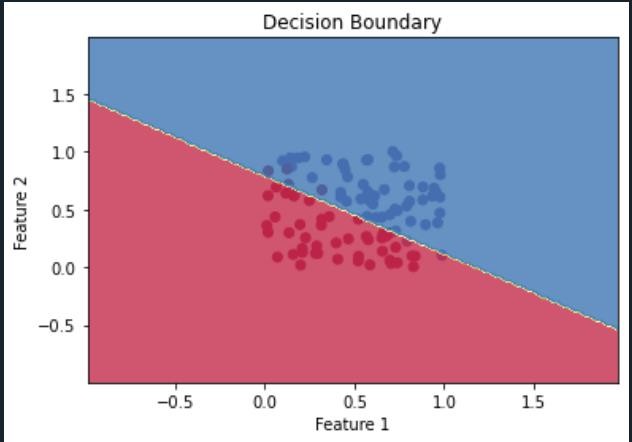
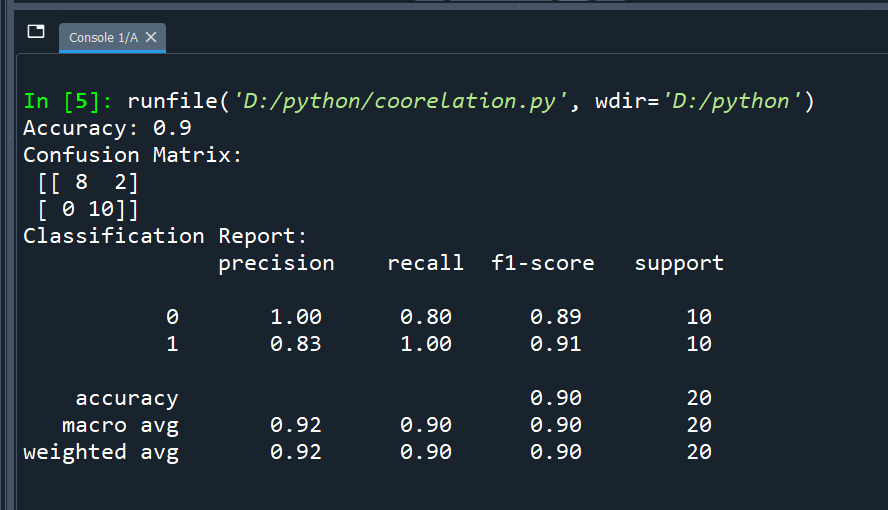
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h)) Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral, alpha=0.8) plt.xlabel("Feature 1")

plt.ylabel("Feature 2") plt.title("Decision Boundary") plt.show()

NAME :Ashutosh Rode ROLL NO : 72 DIV : B



Q4] Write a program to implement support vector machine algorithm import numpy as np

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report # Load a sample dataset (you can replace this with your own data)

# We'll use the built-in Iris dataset for this example. iris = datasets.load\_iris()

X = iris.data

NAME :Ashutosh Rode ROLL NO : 72 DIV : B

y = iris.target

# For binary classification, let's consider only two classes (0 and 1). X = X[y != 2]

y = y[y != 2]

# Split the data 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) # Create an SVM classifier

model = SVC(kernel='linear')

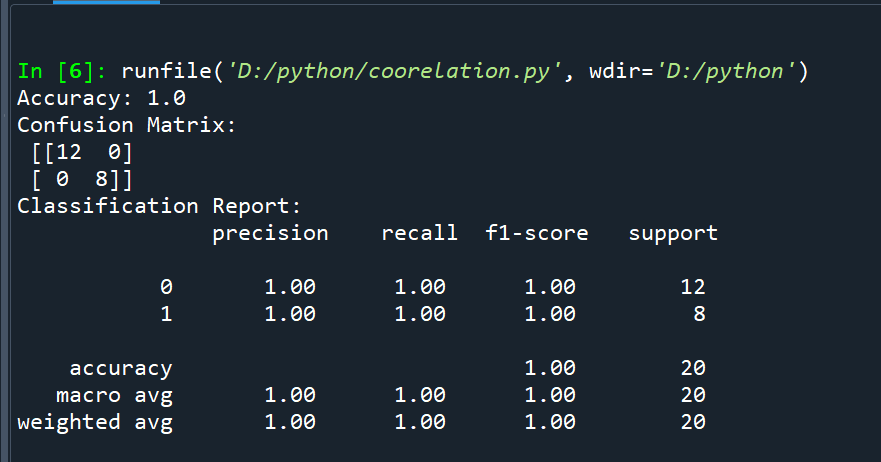
# Train the SVM model on the training data model.fit(X\_train, y\_train)

# Make predictions on the test data y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) conf\_matrix = confusion\_matrix(y\_test, y\_pred) class\_report = classification\_report(y\_test, y\_pred) # Print the evaluation metrics

print("Accuracy:", accuracy) print("Confusion Matrix:\n", conf\_matrix) print("Classification Report:\n", class\_report)



NAME :Ashutosh Rode ROLL NO : 72 DIV : B

Q5] Write a program to implement Decision Tree model on the given dataset import numpy as np

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report # Load a sample dataset (you can replace this with your own data)

# We'll use the built-in Iris dataset for this example. iris = datasets.load\_iris()

X = iris.data y = iris.target

# Split the data 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) # Create a Decision Tree classifier

model = DecisionTreeClassifier()

# Train the Decision Tree model on the training data model.fit(X\_train, y\_train)

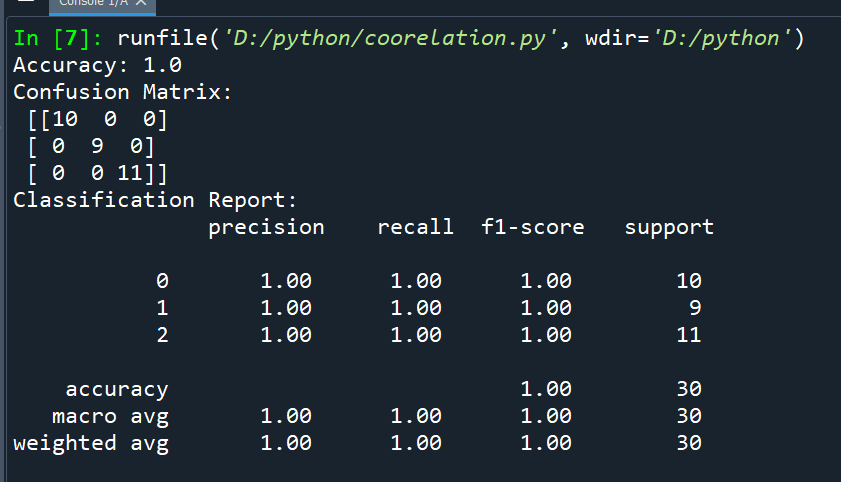
# Make predictions on the test data y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) conf\_matrix = confusion\_matrix(y\_test, y\_pred) class\_report = classification\_report(y\_test, y\_pred) # Print the evaluation metrics

print("Accuracy:", accuracy) print("Confusion Matrix:\n", conf\_matrix) print("Classification Report:\n", class\_report)

NAME :Ashutosh Rode ROLL NO : 72 DIV : B



Q6] Write a program to implement Bayesian classification on given dataset. import numpy as np

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report # Load a sample dataset (you can replace this with your own data)

# We'll use the built-in Iris dataset for this example. iris = datasets.load\_iris()

X = iris.data y = iris.target

# Split the data 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) # Create a Naive Bayes classifier (Gaussian Naive Bayes)

model = GaussianNB()

# Train the Naive Bayes model on the training data model.fit(X\_train, y\_train)

# Make predictions on the test data y\_pred = model.predict(X\_test)

NAME :Ashutosh Rode ROLL NO : 72 DIV : B

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) conf\_matrix = confusion\_matrix(y\_test, y\_pred) class\_report = classification\_report(y\_test, y\_pred) # Print the evaluation metrics

print("Accuracy:", accuracy)

print("Confusion Matrix:\n", conf\_matrix)

print("Classification Report:\n", class\_report)

