**CSA4724 – DEEP LEARNING FOR NUTRITION ANALYSIS**

**HABEEB MOHAMED KASIM M**

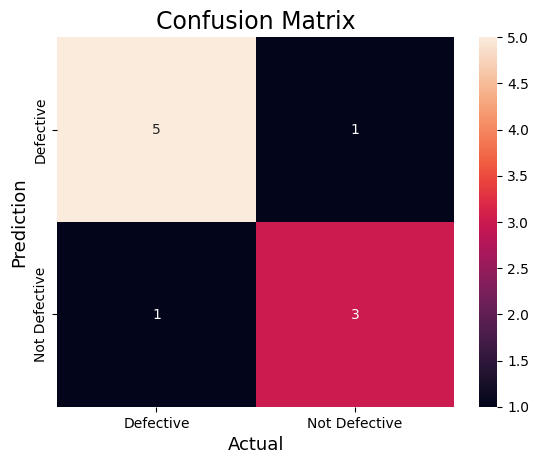
**REG NO : 192224093**

**DAY 01 – 21.06.2024**

01.

import numpy as npfrom sklearn.metrics import confusion\_matriximport seaborn as snsimport matplotlib.pyplot as pltactual = np.array( ['Defective','Defective','Defective','Not Defective','Defective','Not Defective','Defective','Defective','Not Defective','Not Defective'])predicted = np.array( ['Defective','Not Defective','Defective','Not Defective','Defective','Defective','Defective','Defective','Not Defective','Not Defective'])cm = confusion\_matrix(actual,predicted)sns.heatmap(cm, annot=True, fmt='g', xticklabels=['Defective','Not Defective'], yticklabels=['Defective','Not Defective'])plt.ylabel('Prediction',fontsize=13)plt.xlabel('Actual',fontsize=13)plt.title('Confusion Matrix',fontsize=17)plt.show()

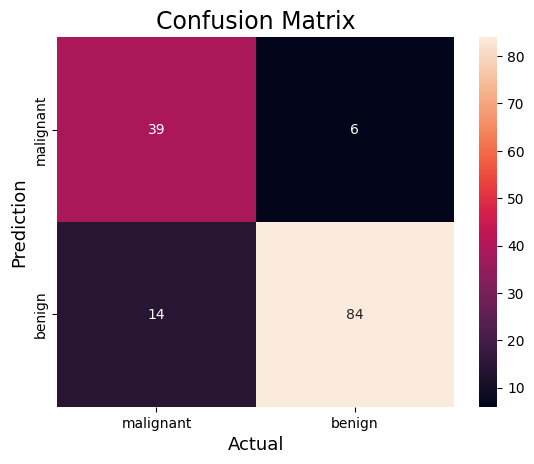
OUTPUT



02.

#Import the necessary librariesfrom sklearn.datasets import load\_breast\_cancerfrom sklearn.model\_selection import train\_test\_splitfrom sklearn.tree import DecisionTreeClassifierfrom sklearn.metrics import confusion\_matriximport seaborn as snsimport matplotlib.pyplot as pltfrom sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_scoreX, y= load\_breast\_cancer(return\_X\_y=True)X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.25)tree = DecisionTreeClassifier(random\_state=23)tree.fit(X\_train, y\_train)y\_pred = tree.predict(X\_test)cm = confusion\_matrix(y\_test,y\_pred)sns.heatmap(cm, annot=True, fmt='g', xticklabels=['malignant', 'benign'], yticklabels=['malignant', 'benign'])plt.ylabel('Prediction',fontsize=13)plt.xlabel('Actual',fontsize=13)plt.title('Confusion Matrix',fontsize=17)plt.show()accuracy = accuracy\_score(y\_test, y\_pred)print("Accuracy :", accuracy)precision = precision\_score(y\_test, y\_pred)print("Precision :", precision)recall = recall\_score(y\_test, y\_pred)print("Recall :", recall)F1\_score = f1\_score(y\_test, y\_pred)print("F1-score :", F1\_score)

OUTPUT



Accuracy : 0.8601398601398601Precision : 0.9333333333333333Recall : 0.8571428571428571F1-score : 0.8936170212765957

03.

from sklearn.datasets import load\_digitsfrom sklearn.model\_selection import train\_test\_splitfrom sklearn.ensemble import RandomForestClassifierfrom sklearn.metrics import confusion\_matriximport seaborn as snsimport matplotlib.pyplot as pltfrom sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score X, y= load\_digits(return\_X\_y=True)X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.25) clf = RandomForestClassifier(random\_state=23)clf.fit(X\_train, y\_train)y\_pred = clf.predict(X\_test)cm = confusion\_matrix(y\_test,y\_pred)sns.heatmap(cm, annot=True, fmt='g')plt.ylabel('Prediction',fontsize=13)plt.xlabel('Actual',fontsize=13)plt.title('Confusion Matrix',fontsize=17)plt.show()accuracy = accuracy\_score(y\_test, y\_pred)print("Accuracy :", accuracy)

OUTPUT:



04.

import numpy as npimport matplotlib.pyplot as pltfrom sklearn.pipeline import Pipelinefrom sklearn.preprocessing import PolynomialFeaturesfrom sklearn.linear\_model import LinearRegressionfrom sklearn.model\_selection import cross\_val\_scoredef true\_fun(X): return np.cos(1.5 \* np.pi \* X)np.random.seed(0)n\_samples = 30degrees = [1, 4, 15]X = np.sort(np.random.rand(n\_samples))y = true\_fun(X) + np.random.randn(n\_samples) \* 0.1plt.figure(figsize=(14, 5))for i in range(len(degrees)): ax = plt.subplot(1, len(degrees), i + 1) plt.setp(ax, xticks=(), yticks=()) polynomial\_features = PolynomialFeatures(degree=degrees[i], include\_bias=False) linear\_regression = LinearRegression() pipeline = Pipeline( [ ("polynomial\_features", polynomial\_features), ("linear\_regression", linear\_regression), ] ) pipeline.fit(X[:, np.newaxis], y) scores = cross\_val\_score( pipeline, X[:, np.newaxis], y, scoring="neg\_mean\_squared\_error", cv=10 ) X\_test = np.linspace(0, 1, 100) plt.plot(X\_test, pipeline.predict(X\_test[:, np.newaxis]), label="Model") plt.plot(X\_test, true\_fun(X\_test), label="True function") plt.scatter(X, y, edgecolor="b", s=20, label="Samples") plt.xlabel("x") plt.ylabel("y") plt.xlim((0, 1)) plt.ylim((-2, 2)) plt.legend(loc="best") plt.title( "Degree {}\nMSE = {:.2e}(+/- {:.2e})".format( degrees[i], -scores.mean(), scores.std() ) )plt.show()

OUTPUT:

