#liner regression SSE,SST,R2,adjusted R2

import numpy as np  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.metrics import r2\_score  
  
*# Input data*x = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])  
y = np.array([3, 4, 5, 6])  
  
model = LinearRegression() *# Create a linear regression model*model.fit(x, y) *# Fit the model to the data*y\_pred = model.predict(x) *# Predict the output*m\_se = mean\_squared\_error(y,y\_pred)*#mean square error*sse = np.sum((y\_pred - y) \*\* 2) *# Calculate SSE (Sum of Squared Errors)*sst = np.sum((y - np.mean(y)) \*\* 2) *# Calculate SST (Total Sum of Squares)*r2 = r2\_score(y, y\_pred) *# Calculate R2 score  
  
# Calculate adjusted R2*n = x.shape[0] *# Number of samples*p = x.shape[1] *# Number of predictors*adjusted\_r2 = 1 - (1 - r2) \* (n - 1) / (n - p - 1)  
  
*# Print the results*print("mean square error",m\_se)  
print("Sum of Squared Errors(SSE):- ", sse)  
print("Total Sum of Squares(SST):- ", sst)  
print("R Square(R2):- ", r2)  
print("Adjusted Square(R2):- ", adjusted\_r2 )

#liner regertion with graph

import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
df = pd.read\_csv('placement.csv')  
X = df.iloc[:,0:1]  
y = df.iloc[:,-1]  
from sklearn.model\_selection import train\_test\_split  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=2)  
from sklearn.linear\_model import LinearRegression  
lr = LinearRegression()  
lr.fit(X\_train,y\_train)  
plt.scatter(df['cgpa'],df['package'])  
plt.plot(X\_train,lr.predict(X\_train),color='red')  
plt.xlabel('CGPA')  
plt.ylabel('Package(in lpa)')  
plt.show()

**WRITE A PROGRAM TO IMPLEMENT DECISION TREE USING PYTHON/R/PROGRAMMING LANGUAGE OF YOUR CHOICE (**load\_iris()**)**

import matplotlib.pyplot as plt  
import pandas as pd  
import sklearn.datasets  
data\_b = sklearn.datasets.load\_iris()  
df=pd.DataFrame(data\_b.data,columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#df['target']  
print(df)  
#print(data\_b)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.tree import DecisionTreeClassifier  
from sklearn import metrics  
from sklearn import tree  
from sklearn.model\_selection import train\_test\_split  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'])  
print(x\_train)  
print(x\_test)  
print(y\_train)  
print(y\_test)  
clf = DecisionTreeClassifier(max\_depth = 5,random\_state=1, criterion='gini') #'gini'  
clf = clf.fit(x\_train, y\_train)  
y\_pred = clf.predict(x\_test)  
print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
fn=['sepal length (cm)','sepal width (cm)', 'petal length (cm)', 'petal width (cm)']  
cn=['setosa', 'versicolor', 'virginica']  
  
fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (4, 4), dpi = 300)  
tree.plot\_tree(clf, feature\_names = fn, class\_names = cn,filled = True); fig.savefig('dstimq.png')

**Write Python Code to demonstrate implementation of Decision Trees Using Python.Use BREAST CANCER Dataset**

*# Import necessary libraries*import numpy as np  
import pandas as pd  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.tree import DecisionTreeClassifier, plot\_tree  
import matplotlib.pyplot as plt  
  
*# Load the Breast Cancer dataset*data = load\_breast\_cancer()  
X = pd.DataFrame(data.data, columns=data.feature\_names)  
y = pd.Series(data.target)  
  
*# Split the dataset 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 and train the Decision Tree classifier*clf = DecisionTreeClassifier(random\_state=42)  
clf.fit(X\_train, y\_train)  
  
*# Make predictions on the test set*y\_pred = clf.predict(X\_test)  
  
*# Visualize the Decision Tree*plt.figure(figsize=(15, 10))  
plot\_tree(clf, filled=True, feature\_names=data.feature\_names, class\_names=data.target\_names, rounded=True)  
plt.show()

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the decision tree model.**

from pandas import DataFrame  
from sklearn.datasets import load\_iris  
data\_b = load\_iris()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
*#print(df)  
#print(data\_b.DESCR)*print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) *# model is trained*y\_pred=clf.predict(X\_test)  
*#print(y\_test, y\_pred)*print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using BREAST CANCER Dataset**

from pandas import DataFrame  
*# from sklearn.datasets import load\_iris*from sklearn.datasets import load\_breast\_cancer  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
data\_b = load\_breast\_cancer()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
*#print(df)  
#print(data\_b.DESCR)*print("Dataset Labels=",data\_b.target\_names)  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) *# model is trained*y\_pred=clf.predict(X\_test)  
*#print(y\_test, y\_pred)*print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)

**find-S**

import pandas as pd  
import numpy as np  
data = pd.read\_csv('FIND-S2.CSV')  
concept = np.array(data)[:,:-1]  
target = np.array(data)[:,-1]  
def train(con,tar):  
 for i,val in enumerate(tar):  
 if val =='yes':  
 sp\_h=con[i].copy()  
 break  
 for i,val in enumerate(con):  
 if tar[i] =='yes':  
 for x in range(len(sp\_h)):  
 if val[x] != sp\_h[x]:  
 sp\_h[x] = '?'  
 else:  
 pass  
 return sp\_h  
print(train(concept,target))

**Candidate-Elimination**

import numpy as np

import pandas as pd

data = pd.read\_csv('C:/Users/sarvadnya/Desktop/Sheet01.csv')

concepts = np.array(data)[:,:-1]

print("\nInstances are:\n",concepts)

target = np.array(data)[:,-1]

print("\nTarget Values are: ",target)

def learn(concepts, target):

specific\_h = concepts[0].copy()

print("\nInitialization of specific\_h and genearal\_h")

print("\nSpecific Boundary: ", specific\_h)

general\_h = [["?" for i in range(len(specific\_h))] for i in range(len(specific\_h))]

print("\nGeneric Boundary: ",general\_h)

for i, h in enumerate(concepts):

print("\nInstance", i+1 , "is ", h)

if target[i] == "Yes":

print("Instance is Positive ")

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

specific\_h[x] ='?'

general\_h[x][x] ='?'

else:

print("Instance is Negative ")

for x in range(len(specific\_h)):

if h[x] != specific\_h[x] and specific\_h[x] != '?':

general\_h[x][x] = specific\_h[x]

else:

general\_h[x][x] = '?'

print("Specific Bundary after ", i+1, "Instance is ", specific\_h)

print("Generic Boundary after ", i+1, "Instance is ", general\_h)

print("\n")

indices = [i for i, val in enumerate(general\_h) if val == ['?', '?', '?', '?', '?', '?']]

for i in indices:

general\_h.remove(['?', '?', '?', '?', '?', '?'])

return specific\_h, general\_h

s\_final, g\_final = learn(concepts, target)

print("Final Specific\_h: ", s\_final, sep="\n")

print("Final General\_h: ", g\_final, sep="\n")

**Write Python Code to demonstrate Precision, Recall, F1-Score of the decision tree model.**

from sklearn.datasets import load\_iris, load\_breast\_cancer

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score

# Load the Irish dataset

iris = load\_iris()

X\_iris = iris.data

y\_iris = iris.target

# Split the Irish dataset into training and testing sets

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

# Train the KNN classifier on the Irish d3ataset

knn\_iris = KNeighborsClassifier()

knn\_iris.fit(X\_train\_iris, y\_train\_iris)

# Make predictions on the Irish testing set

y\_pred\_iris = knn\_iris.predict(X\_test\_iris)

# Calculate the confusion matrix for Irish dataset

cm\_iris = confusion\_matrix(y\_test\_iris, y\_pred\_iris)

print("Confusion Matrix (Irish Dataset):")

print(cm\_iris)

# Calculate precision, recall, and F-measure for Irish dataset

precision\_iris = precision\_score(y\_test\_iris, y\_pred\_iris, average='macro')

recall\_iris = recall\_score(y\_test\_iris, y\_pred\_iris, average='macro')

f1\_iris = f1\_score(y\_test\_iris, y\_pred\_iris, average='macro')

print("Precision (Irish Dataset):", precision\_iris)

print("Recall (Irish Dataset):", recall\_iris)

print("F-measure (Irish Dataset):", f1\_iris)

**Write Python/R Programming Code to demonstrate calculate popular attribute selection measures (ASM) like Information Gain, Gain Ratio, and Gini Index etc.**

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using IRIS Dataset**

from pandas import DataFrame  
from sklearn.datasets import load\_iris  
data\_b = load\_iris()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)  
cor=0  
for i in range(len(data\_b.target\_names)):  
 cor=cor+cm[i,i]  
wrg=len(y\_test)-cor  
print("number of correct prediction:",cor)  
print("number of worng prediction:",wrg)

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using breast cancer Dataset**

from pandas import DataFrame  
from sklearn.datasets import load\_breast\_cancer  
data\_b = load\_breast\_cancer()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)  
cor=0  
for i in range(len(data\_b.target\_names)):  
 cor=cor+cm[i,i]  
wrg=len(y\_test)-cor  
print("number of correct prediction:",cor)  
print("number of worng prediction:",wrg)

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the KNN Model**

**using IRIS Dataset**

from pandas import DataFrame  
from sklearn.datasets import load\_iris  
data\_b = load\_iris()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the KNN Model**

**using brest Dataset**

from pandas import DataFrame  
from sklearn.datasets import load\_breast\_cancer  
data\_b = load\_breast\_cancer()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)

**Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the IRIS dataset.**

from pandas import DataFrame  
from sklearn.datasets import load\_iris  
data\_b = load\_iris()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)  
cor=0  
for i in range(len(data\_b.target\_names)):  
 cor=cor+cm[i,i]  
wrg=len(y\_test)-cor  
print("number of correct prediction:",cor)  
print("number of worng prediction:",wrg)

**Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the Brest cancer dataset.**

from pandas import DataFrame  
from sklearn.datasets import load\_breast\_cancer  
data\_b = load\_breast\_cancer()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)  
cor=0  
for i in range(len(data\_b.target\_names)):  
 cor=cor+cm[i,i]  
wrg=len(y\_test)-cor  
print("number of correct prediction:",cor)  
print("number of worng prediction:",wrg)

**Write Python/R Programming Code to demonstrate Precision, Recall, F1- Score of the KNN model.**

from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score

# Load the Irish dataset

iris = load\_iris()

X\_iris = iris.data

y\_iris = iris.target

# Split the Irish dataset into training and testing sets

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

# Train the KNN classifier on the Irish d3ataset

knn\_iris = KNeighborsClassifier()

knn\_iris.fit(X\_train\_iris, y\_train\_iris)

# Make predictions on the Irish testing set

y\_pred\_iris = knn\_iris.predict(X\_test\_iris)

# Calculate the confusion matrix for Irish dataset

cm\_iris = confusion\_matrix(y\_test\_iris, y\_pred\_iris)

print("Confusion Matrix (Irish Dataset):")

print(cm\_iris)

# Calculate precision, recall, and F-measure for Irish dataset

precision\_iris = precision\_score(y\_test\_iris, y\_pred\_iris, average='macro')

recall\_iris = recall\_score(y\_test\_iris, y\_pred\_iris, average='macro')

f1\_iris = f1\_score(y\_test\_iris, y\_pred\_iris, average='macro')

print("Precision (Irish Dataset):", precision\_iris)

print("Recall (Irish Dataset):", recall\_iris)

print("F-measure (Irish Dataset):", f1\_iris)

**Write Python/R Programming Code Print both correct and wrong predictions and Accuracy of the KNN Model**

from pandas import DataFrame  
from sklearn.datasets import load\_breast\_cancer  
data\_b = load\_breast\_cancer()  
df= DataFrame(data\_b.data, columns=data\_b.feature\_names)  
df['target'] = data\_b.target  
#print(df)  
#print(data\_b.DESCR)  
print("Dataset Labels=",data\_b.target\_names)  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, y\_test = train\_test\_split(df[data\_b.feature\_names], df['target'], random\_state=1)  
print(X\_train.head(6))  
print(Y\_train.head(6))  
print(X\_test.head())  
clf = KNeighborsClassifier(n\_neighbors=6)  
clf.fit(X\_train, Y\_train) # model is trained  
y\_pred=clf.predict(X\_test)  
#print(y\_test, y\_pred)  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
cm = confusion\_matrix(y\_test, y\_pred)  
print("Confusion Matrix:")  
print(cm)  
cor=0  
for i in range(len(data\_b.target\_names)):  
 cor=cor+cm[i,i]  
wrg=len(y\_test)-cor  
print("number of correct prediction:",cor)  
print("number of worng prediction:",wrg)

**Write Python/R Programming Code Print both correct and wrong predictions and Print Accuracy of the Naive Bayes Classifier Model**

*#naive Basesian Classfier  
# for dataset*import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import datasets  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import confusion\_matrix  
iris=datasets.load\_iris()  
x=iris.data  
y=iris.target  
print("Features:",iris['feature\_names'])  
  
*#Accuracy Confusion Matrix*x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)  
NB=GaussianNB()  
NB.fit(x\_train,y\_train)  
y\_pred=NB.predict(x\_test)  
cm=confusion\_matrix(y\_test,y\_pred)  
print("Confusion Matrix")  
print(cm)

**Write Python/R Programming Code to implement the implement Naïve Bayes Classifier to classify the IRIS dataset**

#naive Basesian Classfier  
# for dataset  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import datasets  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import confusion\_matrix  
iris=datasets.load\_iris()  
x=iris.data  
y=iris.target  
print("Features:",iris['feature\_names'])  
  
#Accuracy Confusion Matrix  
  
x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)  
NB=GaussianNB()  
NB.fit(x\_train,y\_train)  
y\_pred=NB.predict(x\_test)  
cm=confusion\_matrix(y\_test,y\_pred)  
print("Confusion Matrix")  
print(cm)

**Write Python/R Programming Code Print Precision, Recall, F1-Score of the Naive Bayes Classifier Model.**

*#naive Basesian Classfier  
# for dataset*import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import datasets  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score  
iris=datasets.load\_iris()  
x=iris.data  
y=iris.target  
print("Features:",iris['feature\_names'])  
  
*#Accuracy Confusion Matrix*x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)  
NB=GaussianNB()  
NB.fit(x\_train,y\_train)  
y\_pred=NB.predict(x\_test)  
cm=confusion\_matrix(y\_test,y\_pred)  
print("Confusion Matrix")  
print(cm)  
  
*# Calculate precision, recall, and F-measure for Irish dataset*precision\_iris = precision\_score(y\_test, y\_pred, average='macro')  
recall\_iris = recall\_score(y\_test, y\_pred, average='macro')  
f1\_iris = f1\_score(y\_test, y\_pred, average='macro')  
  
print("Precision (Irish Dataset):", precision\_iris)  
print("Recall (Irish Dataset):", recall\_iris)  
print("F-measure (Irish Dataset):", f1\_iris)

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the Naive Bayes Classifier Model.**

*#naive Basesian Classfier  
# for dataset*import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import datasets, \_\_all\_\_  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import confusion\_matrix,accuracy\_score  
from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score  
iris=datasets.load\_iris()  
x=iris.data  
y=iris.target  
print("Features:",iris['feature\_names'])  
  
*#Accuracy Confusion Matrix*x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)  
NB=GaussianNB()  
NB.fit(x\_train,y\_train)  
y\_pred=NB.predict(x\_test)  
cm=confusion\_matrix(y\_test,y\_pred)  
print("Confusion Matrix")  
print(cm)  
print("Accuracy:",accuracy\_score(y\_test, y\_pred))

**Write Python/R Programming Code for Implementing Agglomerative Clustering in Python**

from sklearn.cluster import AgglomerativeClustering  
from sklearn.datasets import make\_blobs  
import matplotlib.pyplot as plt  
  
*# Generate sample data*X, y = make\_blobs(n\_samples=200, centers=4, random\_state=0)  
  
*# Create an instance of AgglomerativeClustering*clustering = AgglomerativeClustering(n\_clusters=4)  
  
*# Perform clustering*clustering.fit(X)  
  
*# Retrieve the cluster labels*labels = clustering.labels\_  
  
*# Plot the data points with their corresponding cluster labels*plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')  
plt.xlabel("Feature 1")  
plt.ylabel("Feature 2")  
plt.title("Agglomerative Clustering")  
plt.show()

**Write a Program for Fuzzy c-means clustering in Python.**

import numpy as np  
import skfuzzy as fuzz  
from skfuzzy import control as ctrl  
  
*# Generate some example data*np.random.seed(0)  
data = np.random.rand(100, 2)  
  
*# Define the number of clusters*n\_clusters = 3  
  
*# Apply fuzzy c-means clustering*cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(  
data.T, n\_clusters, 2, error=0.005, maxiter=1000, init=None  
  
)