1. Write Python Code to demonstrate implementation of Decision Trees Using Python. Use IRIS Dataset.
2. Write Python Code to demonstrate Precision, Recall, F1-Score of the decision tree model.

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

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

21.Write Python/R Programming Code to demonstrate implementation of Decision Trees Using Python. Use BREAST CANCER Dataset

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

import pandas as pd

import numpy as np

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

iris = load\_iris() # breast = load\_breast\_cancer()

x = iris.data # breast.data

y = iris.target # breast.target

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20,

random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(X\_train)

xtest = sc\_x.transform(X\_test)

from sklearn.tree import DecisionTreeClassifier

classifier= DecisionTreeClassifier(criterion='entropy', random\_state=0)

classifier.fit(xtrain, y\_train)

y\_pred = classifier.predict(xtest)

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

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

cm

print(classification\_report(y\_test, y\_pred))

print(accuracy\_score(y\_test,y\_pred))

1. Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using IRIS Dataset.
2. Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the KNN Model.
3. Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the IRIS dataset.
4. Write Python/R Programming Code to demonstrate Precision, Recall, F1-Score of the KNN model.
5. Write Python/R Programming Code Print both correct and wrong predictions and Accuracy of the KNN Model.
6. Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using BREAST CANCER Dataset
7. Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the BREAST CANCER dataset.

import pandas as pd

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

iris = load\_iris() # breast = load\_breast\_cancer()

x = iris.data # breast.data

y = iris.target # breast.target

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

# Splitting the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=0)

from sklearn.preprocessing import StandardScaler

# Feature Scaling

sc\_x = StandardScaler()

X\_train = sc\_x.fit\_transform(X\_train)

X\_test = sc\_x.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

# Fitting KNN classifier to the training set

classifier = KNeighborsClassifier(n\_neighbors=5)

classifier.fit(X\_train, y\_train)

# Predicting the test set results

y\_pred = classifier.predict(X\_test)

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

# Creating the Confusion matrix

Print(accuracy\_score(y\_test,y\_pred))

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

print("Confusion Matrix:")

print(cm)

from sklearn.metrics import classification\_report

# Classification report with precision, recall, and F1-score

report = classification\_report(y\_test, y\_pred)

print("Classification Report:")

print(report)

12. Write Python/R Programming Code Print both correct and wrong

predictions and Print Accuracy of the Naive Bayes Classifier Model.

13. Write Python/R Programming Code to implement the implement Naive Bayes Classifier to classify the IRIS dataset.

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

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

25. Write Python/R Programming Code to implement the implement Naive Bayes Classifier to classify the BREAST CANCER dataset.

Import pandas as pd

import numpy as np

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

iris = load\_iris() # breast = load\_breast\_cancer()

x = iris.data # breast.data

y = iris.target # breast.target

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20,

random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(X\_train)

xtest = sc\_x.transform(X\_test)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(xtrain, y\_train)

y\_pred = classifier.predict(xtest)

for i in range(len(y\_test)):

if y\_test[i] == y\_pred[i]:

print("Correct prediction: Expected {}, Predicted {}".format(y\_test[i], y\_pred[i]))

else:

print("Wrong prediction: Expected {}, Predicted {}".format(y\_test[i], y\_pred[i]))

from sklearn.metrics import confusion\_matrix

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

print("Confusion Matrix:")

print(cm)

1. Find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R Square , Adjusted R square of the Linear Regression Model.

import pandas as pd

import numpy as np

dataset = pd.read\_csv('student\_scores.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

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

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

# Calculate Sum of Squared Errors (SSE)

sse = np.sum((y\_test - y\_pred) \*\* 2)

# Calculate Total Sum of Squares (SST)

y\_mean = np.mean(y\_test)

sst = np.sum((y\_test - y\_mean) \*\* 2)

# Calculate R-squared

r\_squared = 1 - (sse / sst)

# Calculate Adjusted R-squared

n = len(y\_test) # Number of data points

p = X.shape[1] # Number of features (excluding the intercept)

adjusted\_r\_squared = 1 - (sse / (n - p - 1)) / (sst / (n - 1))

print('Sum of Squared Errors (SSE):', sse)

print('Total Sum of Squares (SST):', sst)

print('R-squared:', r\_squared)

print('Adjusted R-squared:', adjusted\_r\_squared)

# Write a program to implement Simple Linear Regression Using Python

import pandas as pd

import numpy as np

dataset = pd.read\_csv('student\_scores.csv')

dataset.describe()

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

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=0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

X\_train.shape

X\_test

y\_pred = regressor.predict(X\_test)

y\_pred

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

regressor.score(X\_test,y\_test)

# Write a program to implement Multiple Linear Regression Using Python

import pandas as pd

import numpy as np

dataset = pd.read\_csv('house\_data.csv')

dataset.shape

X = dataset.iloc[:, [5]].values

y = dataset.iloc[:, -1].values

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=0)

y

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

y\_pred

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

regressor.score(X\_test,y\_test)

1. Write a program for multi-variable (two variable) linear regression and find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R Square , Adjusted R square. Input Vector X = ([1, 1], [1, 2], [2, 2], [2, 3], [2, 4], [3, 4], [3, 5]) Output Vector y = ([4, 6, 7, 8, 9, 10, 12])

Write a program for single-variable (two variable) linear regression and find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R Square , Adjusted R square etc. Input Vector X = ([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]]) Output Vector y = ([[1], [3], [2], [5], [7], [8], [8], [9], [10], [12]])

import pandas as pd

import numpy as np

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3], [2, 4], [3, 4], [3, 5]])

y = np.array([4, 6, 7, 8, 9, 10, 12])

# question 19

#X = np.array([[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]])

#y = np.array([[1], [3], [2], [5], [7], [8], [8], [9], [10], [12]])

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X, y)

y\_pred = regressor.predict(X)

# Calculate Sum of Squared Errors (SSE)

sse = np.sum((y - y\_pred) \*\* 2)

# Calculate Total Sum of Squares (SST)

y\_mean = np.mean(y)

sst = np.sum((y - y\_mean) \*\* 2)

# Calculate R-squared

r\_squared = 1 - (sse / sst)

# Calculate Adjusted R-squared

n = len(y) # Number of data points

p = X.shape[1] # Number of features (excluding the intercept)

adjusted\_r\_squared = 1 - (sse / (n - p - 1)) / (sst / (n - 1))

print('Sum of Squared Errors (SSE):', sse)

print('Total Sum of Squares (SST):', sst)

print('R-squared:', r\_squared)

print('Adjusted R-squared:', adjusted\_r\_squared)

26:-

Sky,temp,humidity,water,wind,enjoy\_sport

Sunny,Warm,High,Warm,Weak,Yes

Rainy,Cold,High,Warm,Strong,No

Sunny,Warm,High,Cool,Strong,Yes

Sunny,Cold,Low,Cool,Weak,Yes

Rainy,Warm,Low,Warm,Weak,Yes

Rainy,Cold,Low,Cool,Strong,No

Sunny,Cold,Low,Warm,Strong,No

Rainy,Warm,Hign,Cool,Strong,No

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

#first run this command on cmd (pip install -U scikit-fuzzy)

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

np.random.seed(0)

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

n\_clusters = 3

cntr, u, ue, d, jm, p, fpc = fuzz.cluster.cmeans(data.T, n\_clusters,

2,error=0.005,maxiter=10)

cluster\_membership = np.argmax(u, axis=0)

print('Cluster Membership:', cluster\_membership)

1. Write Python/R Programming Code for Implementing Agglomerative Clustering in Python.

from sklearn import datasets

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

iris = datasets.load\_iris()

X=iris['data']

Y=iris.target

print(X.shape)

X = PCA(n\_components=2).fit\_transform(X)

plt.scatter(X[:,0],X[:,1])

from sklearn.cluster import AgglomerativeClustering

classifier = AgglomerativeClustering(n\_clusters = 3, affinity = 'euclidean',

Linkage=’ward’)

clusters = classifier.fit\_predict(X)

plt.scatter(X[clusters == 0, 0], X[clusters == 0, 1], label = 'Type 1')

plt.scatter(X[clusters == 1, 0], X[clusters == 1, 1], label = 'Type 2')

plt.scatter(X[clusters == 2, 0], X[clusters == 2, 1], label = 'Type 3')

plt.title('Clusters')

plt.show()

plt.scatter(X[Y == 0, 0], X[Y == 0, 1], label = 'Type 1')

plt.scatter(X[Y == 1, 0], X[Y == 1, 1], label = 'Type 2')

plt.scatter(X[Y == 2, 0], X[Y == 2, 1], label = 'Type 3')

27. Write and demonstrate a program to Implement the Candidate- Elimination Inductive Learning algorithm using the above dataset or a dataset of your choice.

import numpy as np

import pandas as pd

data = pd.read\_csv('enjoysport.csv')

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

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

target = np.array(data.iloc[:,-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] ='?'

if target[i] == "No":

print("Instance is Negative ")

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

if h[x]!= 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")

1. Write and demonstrate a program to implement the find-S inductive learning algorithm using the above dataset or a dataset of your choice.

import numpy as np

import pandas as pd

data=pd.read\_csv("Enjoy\_Sport.csv")

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

print("\n Instance are ",concepts)

h=['0','0','0','0','0','0','0']

for row in concepts:

if row[-1]=='Yes':

j=0

for col in row:

if col !=’yes’:

if col !=h[j] and h[j]==’0’:

h[j]=col

elif col !=h[j] and h[j]!=’0’:

h[j]=’?’

j=j+1

print(“maximally specific hypothesis:”,h)

# Write a program to implement Support Vector Machine(SVM)

Using Python

import pandas as pd

import numpy as np

dataset = pd.read\_csv("User\_Data.csv")

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20,

random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(X\_train)

xtest = sc\_x.transform(X\_test)

from sklearn.svm import SVC

classifier = SVC(kernel='linear', random\_state=0)

classifier.fit(xtrain, y\_train)

y\_pred = classifier.predict(xtest)

y\_pred

from sklearn.metrics import confusion\_matrix

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

cm

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))