**# 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)

**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)

***# Write a program to implement Logistic Regression 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.linear\_model **import** LogisticRegression

classifier **=** LogisticRegression(random\_state **=** 0)

classifier.fit(xtrain, y\_train)

y\_pred **=** classifier.predict(xtest)

y\_pred

**from** sklearn.metrics **import** accuracy\_score

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

***# Write a program to implement Decision Tress 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.tree **import** DecisionTreeClassifier

classifier**=** DecisionTreeClassifier(criterion**=**'entropy', 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** confusion\_matrix

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

cm

***# 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))

***# Write a program to implement KNN 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.neighbors **import** KNeighborsClassifier

classifier **=** KNeighborsClassifier(n\_neighbors**=**5)

classifier.fit(xtrain, y\_train)

y\_pred **=** classifier.predict(xtest)

y\_pred

**from** sklearn.metrics **import** classification\_report,confusion\_matrix

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

cm

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

***# Write a program to implement Naive Bayes Classifier 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.naive\_bayes **import** GaussianNB

classifier **=** GaussianNB()

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))

***#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

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')

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

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** sklearn.datasets **import** load\_breast\_cancer

**from** sklearn.tree **import** DecisionTreeClassifier

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

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn **import** tree

data **=** load\_breast\_cancer()

data.feature\_names

data.target\_names

data.data

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(data.data[:,0:9],data.target,shuffle**=True**,

test\_size**=**0.3, random\_state**=**42)

clf\_dt **=** DecisionTreeClassifier(criterion**=**"entropy")

clf\_dt.fit(X\_train,y\_train)

y\_test\_pred **=** clf\_dt.predict(X\_test);

a\_dt\_test **=** accuracy\_score(y\_test, y\_test\_pred);

print("Training data accuracy is : " )

a\_dt\_test

fig **=** plt.figure(figsize**=**(30,25))

\_ **=** tree.plot\_tree(clf\_dt,feature\_names**=**data.feature\_names,

class\_names**=**data.target\_names,

filled**=True**)

***#Implement the Candidate-Elimination Inductive Learning algorithm.***

**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")

***#Implement the Candidate-Elimination Inductive Learning algorithm.***

**import** numpy **as** np

**import** pandas **as** pd

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

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

print("\nInstances are:\n",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 program for linear regression and find parameters like sum of squared error SSE,SSR,SST LINEAR REGRESSION**

**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)

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

print(sse)

ssr = np.sum((y\_test – y.mean())\*\*2)

print(ssr)

sst = ssr + sse

print(sst)

**#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)

#Backpropagation

\*import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

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

import matplotlib.pyplot as plt

\*data= load\_iris()

X=data.data

y=data.target

\*X\_train ,X\_test, y\_train, y\_test= train\_test\_split(X,y, test\_size=20, random\_state=0)

\*learning\_rate =0.1

iterations=5000

N=y\_train.size

input\_size=4

hidden\_size=2

output\_size=3

\*np.random.seed(10)

W1= np.random.normal(scale=0.5,size=(input\_size,hidden\_size))

W2= np.random.normal(scale=0.5,size=(hidden\_size, output\_size))

\*def sigmoid(x):

return 1/(1+np.exp(-x))

def mean\_squared\_error(y\_pred, y\_true):

y\_true\_one\_hot=np.eye(output\_size)[y\_true]

y\_true\_reshaped= y\_true\_one\_hot.reshape(y\_pred.shape)

error= ((y\_pred - y\_true\_reshaped)\*\*2).sum()/(2\*y\_pred.size)

return error

def accuracy(y\_pred,y\_true):

acc=y\_pred.argmax(axis=1) == y\_true.argmax(axis=1)

return acc.mean()

results = pd.DataFrame(columns=["mse","accuracy"])

\*for itr in range(iterations):

Z1 = np.dot(X\_train, W1)

A1 = sigmoid(Z1)

Z2 = np.dot(A1, W2)

A2 = sigmoid(Z2)

mse = mean\_squared\_error(A2, y\_train)

acc = accuracy (np.eye(output\_size) [y\_train], A2)

new\_row = pd.DataFrame({"mse": [mse], "accuracy": [acc]})

results = pd.concat([results, new\_row], ignore\_index=True)

E1 = A2 - np.eye (output\_size) [y\_train]

dW1 = E1 \* A2 \* (1 - A2)

E2= np.dot(dW1, W2.T)

dW2 = E2 \* A1 \* (1 - A1)

W2\_update = np.dot (A1.T, dW1) / N

W1\_update = np.dot (X\_train. T, dW2) / N

W2 = W2 - learning\_rate\* W2\_update

W1 = W1 - learning\_rate\* W1\_update

\*Z1 = np.dot (X\_test, W1)

A1 = sigmoid(Z1)

Z2 = np.dot (A1, W2)

A2 = sigmoid (Z2)

test\_acc = accuracy (np.eye(output\_size) [y\_test], A2)

print("Test accuracy: {}".format(test\_acc))

**1)simple linear Regression usin python**

**2)find sse,ssr,sst using linear Regression**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

data=load\_iris()

x=data.data

y=data.target

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

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

from sklearn.linear\_model import LinearRegression

sc=LinearRegression()

sc.fit(x\_train,y\_train)

ypred=sc.predict(x\_test)

print("score",sc.score(x\_test,y\_test))

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

ssr=np.sum((ypred-y.mean())\*\*2)

sst=sse+ssr

print("sse =",sse)

print("ssr= ",ssr)

print("sst= ",sst)

**3)implement Decision tree using iris dataset**

**4) implement precision,recall,f1-score of decision tree**

**5) and 22)implement confusion matrix,accuracy score using decision tree**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

data=load\_iris()

x=data.data

y=data.target

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

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

from sklearn.tree import DecisionTreeClassifier

sc=DecisionTreeClassifier(criterion='entropy')

sc.fit(x\_train,y\_train)

y\_pred=sc.predict(x\_test)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**7) knn using iris dataset**

**8) find accuracy score and confusion matrix of knn**

**9) knn using iris dataset**

**10) demonstrate precision,recall,f1-score of knn**

**11) correct and wrong prediction and accuracy of knn**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

data=load\_iris()

x=data.data

y=data.target

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,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.neighbors import KNeighborsClassifier

sc=KNeighborsClassifier(n\_neighbors=5)

sc.fit(xtrain,y\_train)

y\_pred=sc.predict(x\_test)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**12) print correct and wrong prediction and print accuracy of naïve bayes classifier model**

**13) implement naïve bayes classification using iris dataset**

**14) Print Precision,recall,F1-score of naïve bayes classification model**

**15) Demonstrate Accuracy and confusion matrix of naïve bayes classification**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

data=load\_iris()

x=data.data

y=data.target

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,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

sc= GaussianNB()

sc.fit(xtrain,y\_train)

y\_pred=sc.predict(x\_test)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**20) and 21) implement decision tree using breast cancer dataset**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_breast\_cancer

data=load\_breast\_cancer()

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)

from sklearn.tree import DecisionTreeClassifier

sc=DecisionTreeClassifier(criterion='entropy')

sc.fit(x\_train,y\_train)

y\_pred=sc.predict(x\_test)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**24) Implement knn to classify breast cancer dataset**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_breast\_cancer

data=load\_breast\_cancer()

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

xtrain=sc.fit\_transform(x\_train)

xtest=sc.transform(x\_test)

from sklearn.neighbors import KNeighborsClassifier

sc\_x=KNeighborsClassifier(n\_neighbors=5)

sc\_x.fit(x\_train,y\_train)

y\_pred=sc.predict(x\_test)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**25) implement naïve bayes classifier to classify breast cancer dataset**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_breast\_cancer

data=load\_breast\_cancer()

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

xtrain=sc.fit\_transform(x\_train)

xtest=sc.transform(x\_test)

from sklearn.naive\_bayes import GaussianNB

sc\_x=GaussianNB()

sc\_x.fit(x\_train,y\_train)

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

print(confusion\_matrix(y\_test,y\_pred))

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

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

**19)write a program to implement multivariable linear regression and find sse,ssr,sst**

import pandas as pd

import numpy as np

data=pd.read\_csv("abc.csv")

x=data.iloc[:,:-1].values

y=data.iloc[:,2].values

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=0)

from sklearn.linear\_model import LinearRegression

sc=LinearRegression()

sc.fit(x\_train,y\_train)

ypred=sc.predict(x\_test)

print("score",sc.score(x\_test,y\_test))

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

ssr=np.sum((ypred-y.mean())\*\*2)

sst=sse+ssr

print("sse =",sse)

print("ssr= ",ssr)

print("sst= ",sst)

**18) Implement Fuzzy c mean clustering technique**

import pandas as pd

import numpy as np

import skfuzzy as f

np.random.seed(0)

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

nc=3

cnt,u,u0,d,jm,p,fpc=f.cluster.cmeans(data.T,nc,2,error=0.005,maxiter=10)

clm=np.argmax(u,axis=0)

print("cluster center : ",cnt)

print("cluster membership:",clm)