**1.Finding Factorial of a number**

n = int(input("Enter a number:"))

result = 1

for i in range(n,0,-1):

result = result\*i

print("The factorial of",n,"is: ",result)

**OUTPUT:**

Enter a number: 6

The factorial of 6 is: 720

Enter a number: 5

The factorial of 5 is: 120

**2.To check Prime Number**

num = int(input("Enter a number: "))

if num > 1:

for i in range(2,num):

if(num%i)==0:

print(num,"is not a prime number")

break

else:

print(num,"is a prime number")

**OUTPUT:**

Enter a number: 15

15 is not a prime number

Enter a number: 13

13 is a prime number

**3.Write a python program to get the length of the first two lists of a given list.**

list1=[[1,2,3,4,5],"Python Programming","HTML"]

print("Original list:")

print(list1)

print("Length of the first two lists in the given list:")

len1 = len(list1[0])

len2 = len(list1[1])

print(len1)

print(len2)

**OUTPUT:**

Original list:

[[1, 2, 3, 4, 5], 'Python Programming', 'HTML']

Length of the first two lists in the given list:

5

18

**1.write a python program to create a list containing string, numbers, vectors and a logical values**

list\_data = list(("Python","PHP",(5,7,9,11),True, 125.17, 75.83))

print("Data of the list:")

print(list\_data)

**2.write a python program to merge two given lists into one list :**

n1 = (1,2,3)

c1 = ("Red", "Green", "Black")

print("Original lists:")

print(n1)

print(c1)

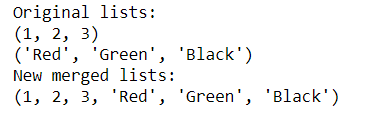
mlist = n1 + c1

print("New merged lists:")

print(mlist)

**OUTPUT :**





**KNN ALGORITHM**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

data.head()

data.isnull().any()

x=data.iloc[:,1:4].values

y=data.iloc[:,4:5].values

x[:5]

y[:5]

from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

x[:,0] **=** lb.fit\_transform(x[:,0])

x[:5]

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

x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()

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

x\_test = sc.transform(x\_test)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier ( n\_neighbors = 5 , p = 2 )

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

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

y\_pred

y\_test

from sklearn.metrics import accuracy\_score

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

from sklearn.metrics import confusion\_matrix

import sklearn.metrics as metrics

fpr, tpr ,threshold **=** metrics.roc\_curve(y\_test,y\_pred)

roc\_auc **=** metrics.auc(fpr,tpr)

roc\_auc

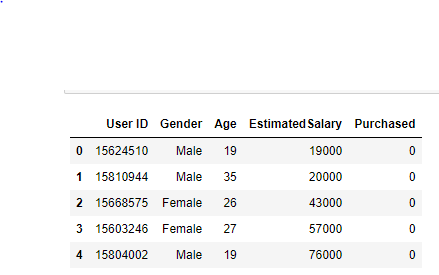
plt.plot(fpr,tpr)

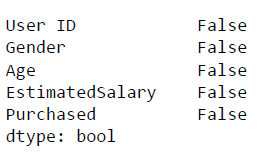
plt.xlim([0,1])

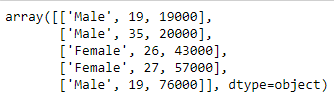
plt.ylim([0,1])

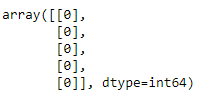
plt.style.use("fivethirtyeight")

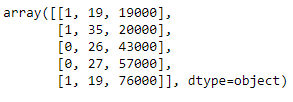
**OUTPUT:**

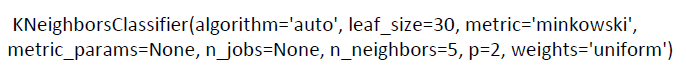
****

****

****

****

****

****

****

array([[0],

[0],

[0],

[0],

[0],

[0],

[0],

[1],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[1],

[0],

[0],

[1],

[0],

[1],

[0],

[1],

[0],

[0],

[0],

[0],

[0],

[1],

[1],

[0],

[0],

[0],

[0],

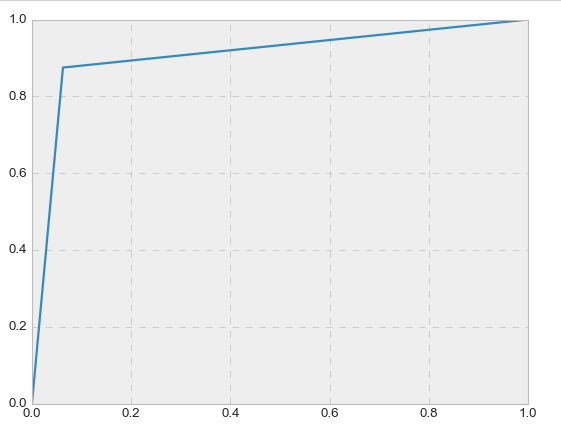
[0],

[0],

[1]], dtype=int64)

0.925 19

0.90625

****

**Decision Tree**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

data.head()

data.isnull().any()

x = data.iloc[:,1:4].values

y = data.iloc[:,4:5].values

x

y[:10]

from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

x[:,0] = lb.fit\_transform(x[:,0])

x

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

x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

x\_train

x\_test[:10]

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy')

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

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

y\_pred

from sklearn.metrics import accuracy\_score

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

from sklearn.metrics import confusion\_matrix

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

import sklearn.metrics as metrics

fpr, tpr ,threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

roc\_auc

plt.plot(fpr,tpr)

plt.xlim([0,1])

plt.ylim([0,1])

plt.style.use("dark\_background")

from sklearn.tree import export\_graphviz

export\_graphviz(dt, out\_file ='tree.dot',

feature\_names = ["Gender","Age","Salary"], class\_names = ['0','1'],

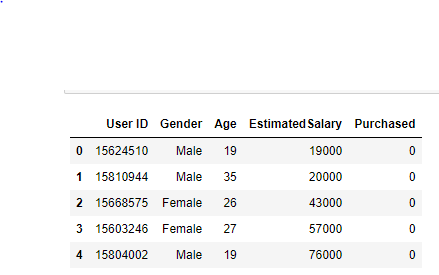
rounded = True, proportion = False, precision = 2, filled = True)

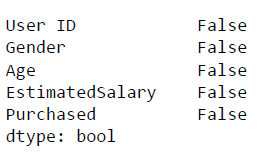
!pip install pydotplus

!conda install graphviz

!dot tree.dot -Tpng -o image.png

**OUTPUT:**

****

****

array([[0],

[0],

[0],

[0], 24

[0],

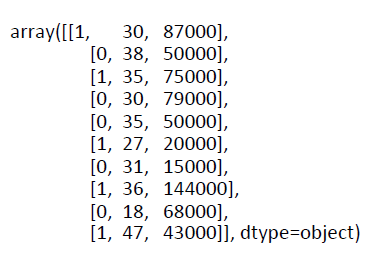
[0],

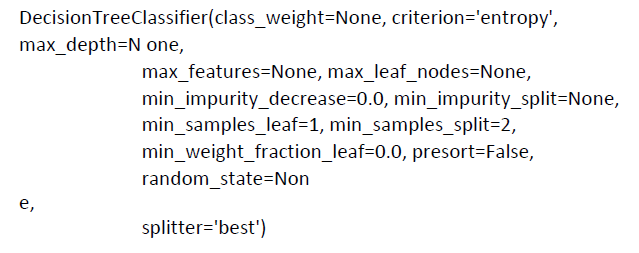
[0],

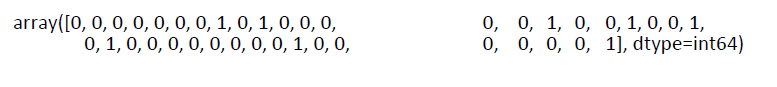
[1],

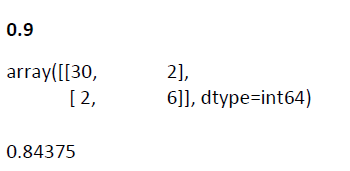
[0],

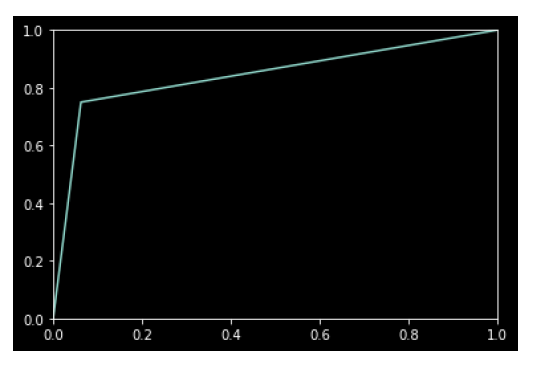
[0]], dtype=int64)







****

****

**Random Forest Classification**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv("ads.csv")

dataset.head()

dataset.isnull().any()

x = dataset.iloc[:, 1:4].values y = dataset.iloc[:, 4].values

x[:5]

y[:5]

from sklearn.preprocessing import LabelEncoder lb=LabelEncoder() x[:,0]=lb.fit\_transform(x[:,0])

x[:5]

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.1, random\_state = 0

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n\_estimators=10000,criterion='entropy')

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

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

from sklearn.metrics import accuracy\_score

print("Accuracy Score: ",accuracy\_score(y\_test,y\_pred)\*100,"%")

from sklearn.metrics import confusion\_matrix

pd.DataFrame(confusion\_matrix(y\_test,y\_pred),columns=["Prediction -0","Prediction -1"],inde

import sklearn.metrics as metrics

fpr, tpr, threshold = metrics.roc\_curve(y\_test, y\_pred) roc\_auc = metrics.auc(fpr, tpr)

print("AUC:",roc\_auc)

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label **=** 'AUC = %0.2f' **%** roc\_auc)

plt.legend(loc **=** 'lower right')

plt.xlim([0, 1])

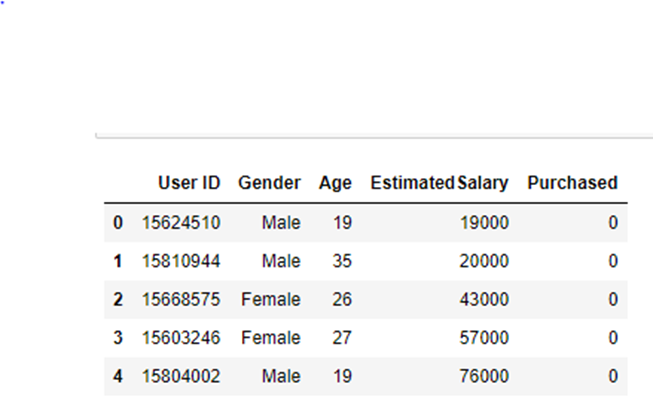
plt.ylim([0, 1])

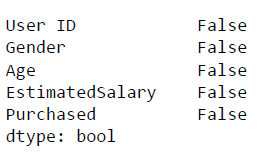
plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

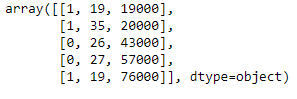
plt.show()

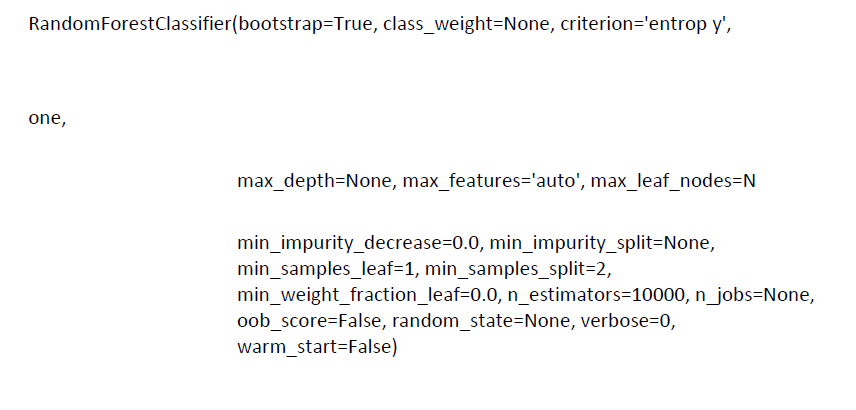
**OUTPUT:**

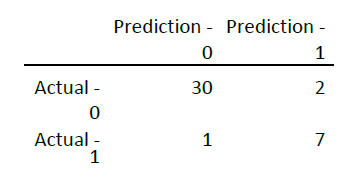
****

****

****

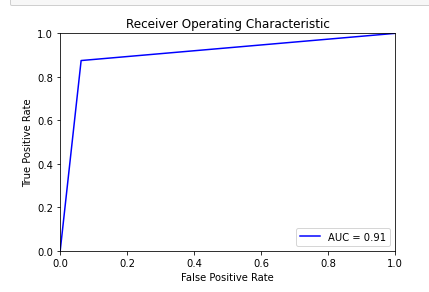
****

****

****

****

****

****

**Decision Tree**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

data.head()

data.isnull().any()

x = data.iloc[:,1:4].values y = data.iloc[:,4:5].values

x

y[:10]

from sklearn.preprocessing import LabelEncoder lb = LabelEncoder()

x[:,0] = lb.fit\_transform(x[:,0])

x

from sklearn.model\_selection import train\_test\_split as tts x\_train,x\_test,y\_train,y\_test = tts(x, y, test\_size = 0.1,random\_state=0)

x\_train

x\_test[:10]

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy')

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

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

y\_pred

from sklearn.metrics import accuracy\_score

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

from sklearn.metrics import confusion\_matrix

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

import sklearn.metrics as metrics

fpr, tpr ,threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

roc\_auc

plt.plot(fpr,tpr)

plt.xlim([0,1])

plt.ylim([0,1])

plt.style.use("ggplot")

from sklearn.tree import export\_graphviz

export\_graphviz(dt, out\_file ='tree.dot',

feature\_names = ["Gender","Age","Salary"], class\_names = ['0','1'],

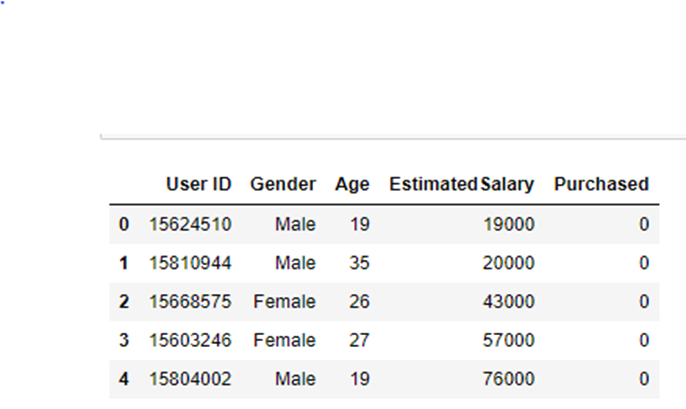
rounded = True, proportion = False, precision = 2, filled = True)

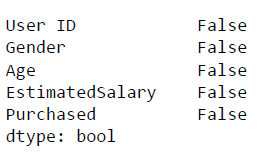
!pip install pydotplus

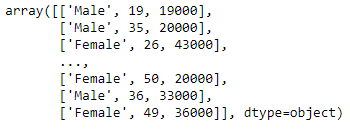
!conda install graphviz

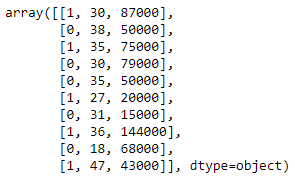
!dot tree.dot -Tpng -o image.png

**OUTPUT:**

****

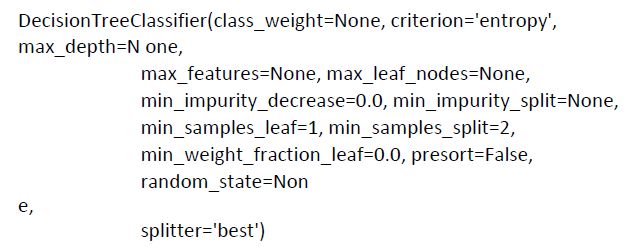
****

****

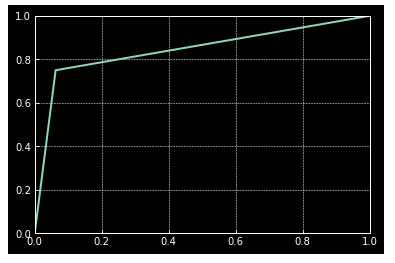
****

**C:\Users\Admin\OneDrive\Pictures\ML lab 6th ex\5.PNG**

**C:\Users\Admin\OneDrive\Pictures\ML lab 6th ex\6.PNG**

****

**C:\Users\Admin\OneDrive\Pictures\ML lab 6th ex\8 (2).PNG**

****

**Artificial Neural Network**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Churn\_Modelling.csv')

dataset.head()

dataset.isnull().any()

x = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values

x[:5]

y[:5]

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder\_X\_1 = LabelEncoder()

x[:, 1] = labelencoder\_X\_1.fit\_transform(x[:, 1])

labelencoder\_X\_2 = LabelEncoder()

x[:, 2] = labelencoder\_X\_2.fit\_transform(x[:, 2])

onehotencoder = OneHotEncoder(categorical\_features = [1])

x = onehotencoder.fit\_transform(x).toarray()

x = x[:, 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 = 0)

from sklearn.preprocessing import MinMaxScaler

sc=MinMaxScaler()

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

x\_test=sc.transform(x\_test)

from keras.models import Sequential

from keras.layers import Dense

model=Sequential() model.add(Dense(input\_dim=11,init="random\_uniform",activation="relu",output\_dim=20)) model.add(Dense(init="random\_uniform",activation="relu",output\_dim=15)) model.add(Dense(init="random\_uniform",activation="sigmoid",output\_dim=1)) model.compile(optimizer="adam",loss="binary\_crossentropy",metrics=["accuracy"]) model.fit(x\_train,y\_train,batch\_size=32,epochs=100)

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

y\_pred

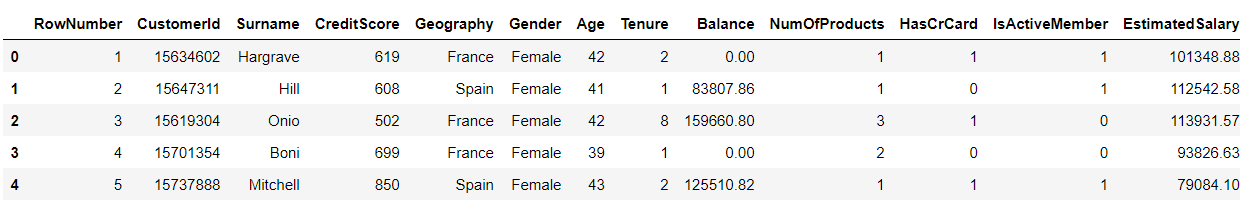
y\_pred=y\_pred>0.5

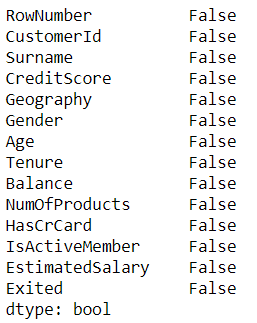
y\_pred

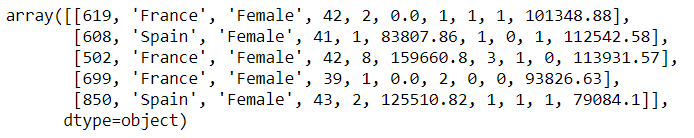
from sklearn.metrics import accuracy\_score

print("Accuracy score",accuracy\_score(y\_test,y\_pred)\*100,"%")

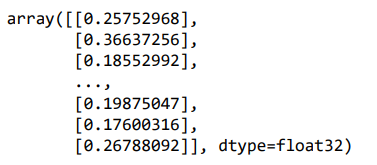
**OUTPUT:**

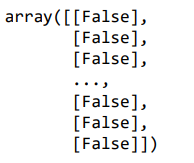














**K-Means Algorithm**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

dataset.head()

dataset.isnull().any()

x = dataset.iloc[:,[3,4]].values

x[:5]

from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):

kmeans = KMeans(n\_clusters = i,init = 'k-means++' , max\_iter = 300, n\_init = 10)

kmeans.fit(x)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters=5,init = 'k-means++' , max\_iter = 300, n\_init=10,random\_state=0)

y\_kmeans = kmeans.fit\_predict(x)

y\_kmeans[:5]

plt.scatter(x[y\_kmeans == 0,0],x[y\_kmeans == 0,1],s=100,c='red',label='cluster 1')

plt.scatter(x[y\_kmeans == 1,0],x[y\_kmeans == 1,1],s=100,c='blue',label='cluster 2')

plt.scatter(x[y\_kmeans == 2,0],x[y\_kmeans == 2,1],s=100,c='green',label='cluster 3')

plt.scatter(x[y\_kmeans == 3,0],x[y\_kmeans == 3,1],s=100,c='yellow',label='cluster 4')

plt.scatter(x[y\_kmeans == 4,0],x[y\_kmeans == 4,1],s=100,c='brown',label='cluster 5')

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=300,c='black',label='centroids')

plt.title('Clusters of clients')

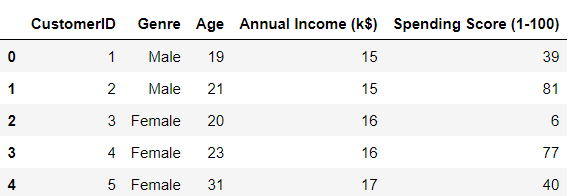
plt.xlabel("Annual Income in 1000 $")

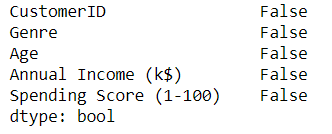
plt.ylabel("Spending Score (1-1000")

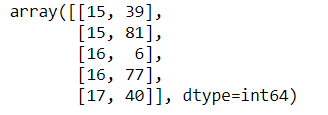
plt.legend()

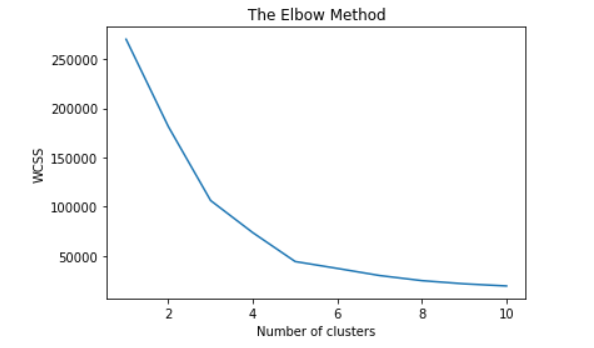
plt.show()

**OUTPUT:**

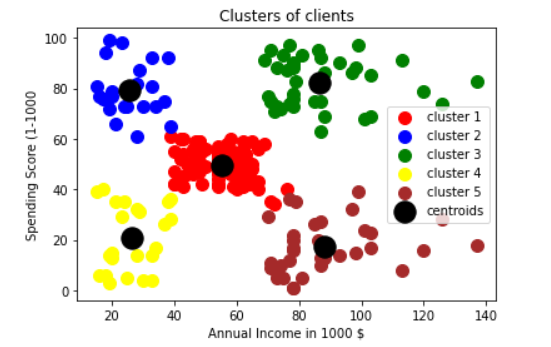












**Hierarchical Clustering**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

x = dataset.iloc[:, [3, 4]].values

x[:5]

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(x)

y\_hc[:5]

plt.scatter(x[y\_hc == 0, 0], x[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(x[y\_hc == 1, 0], x[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(x[y\_hc == 2, 0], x[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(x[y\_hc == 3, 0], x[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(x[y\_hc == 4, 0], x[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

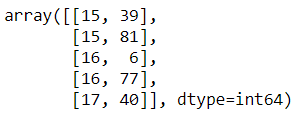
plt.xlabel('Annual Income (k$)')

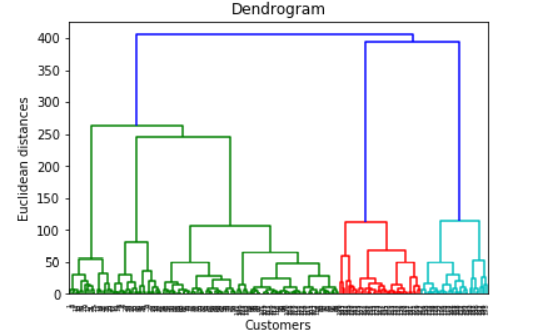
plt.ylabel('Spending Score (1-100)')

plt.legend()

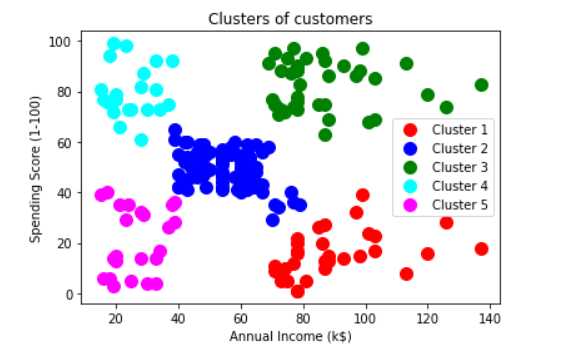
plt.show()

**OUTPUT:**









**Apriori Algorithm**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Market\_Basket\_Optimisation.csv',header = None)

dataset.head()

transactions = []

for i in range(0, 7501):

transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

from apyori import apriori

rules = apriori(transactions, min\_support = 0.003, min\_confidence = 0.2, min\_lift = 3, min\_right = 3)

results = list(rules)

results

**OUTPUT:**

