

DATA SCIENCE LAB RECORD

(20MCA241)

ASHA S S3RMCA-A

Roll No:26

Date:24/11/2021

# Program - 1

## Aim:

Perform all matrix operations using python (using numpy).

## Program:

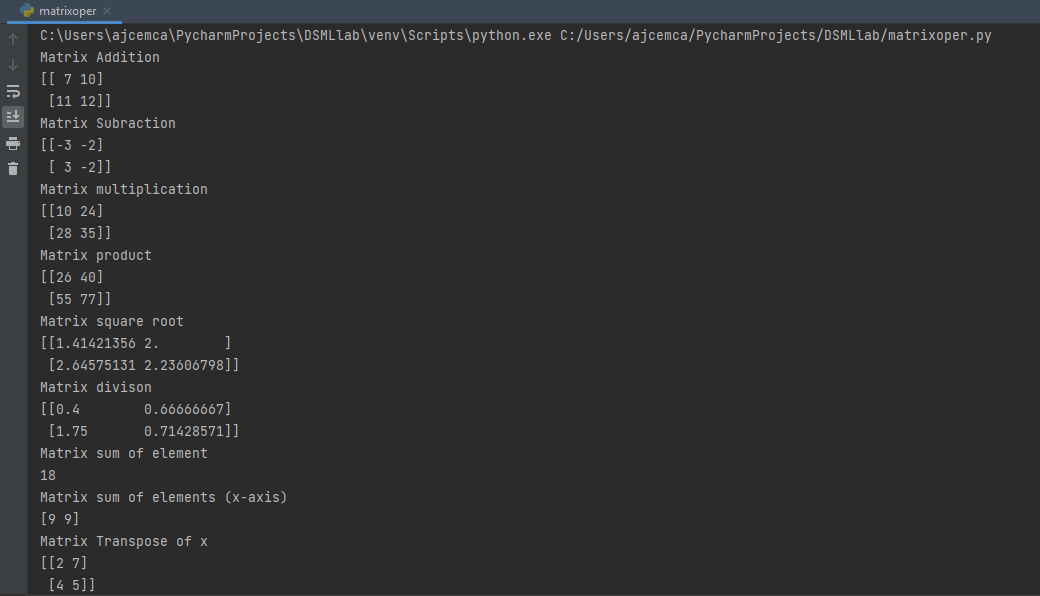
import numpy x=numpy.array([[2,4],[7,5]])

y=numpy.array([[5,6],[4,7]]) print("Matrix Addition") print(numpy.add(x,y)) print("Matrix Subraction") print(numpy.subtract(x,y)) print("Matrix multiplication") print(numpy.multiply(x,y)) print("Matrix product") print(numpy.dot(x,y)) print("Matrix square root") print(numpy.sqrt(x)) print("Matrix divison") print(numpy.divide(x,y)) print("Matrix sum of element") print(numpy.sum(x))

print("Matrix sum of elements (x-axis)") print(numpy.sum(x,axis=0))

print("Matrix Transpose of x") print(x.T)

**OUTPUT**



# Program - 2

Date:01/12/2021



## Aim:

Perform SVD(Singular Value Decomposition)

## Program:

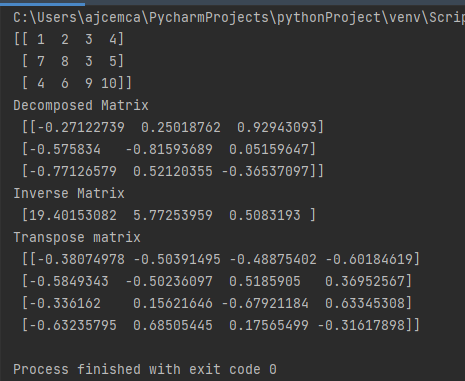
from numpy import array from scipy.linalg import svd

a=array([[1,2,3,4],[7,8,3,5],[4,6,9,10]])

print(a) u,s,vt=svd(a)

print("Decomposed Matrix\n",u) print("Inverse Matrix\n",s) print("Transpose matrix\n",vt)

**OUTPUT**



# Program - 3

## Aim:

Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm

## Program:

from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score

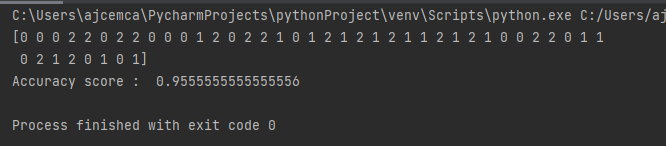
idata=load\_iris() x=idata.data y=idata.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split( x,y,test\_size=0.3,random\_state=55) knn=KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train,y\_train) y\_p=knn.predict(x\_test) print(knn.predict(x\_test))

print(“Accuracy score : ”,accuracy\_score(y\_test,y\_p))

**OUTPUT**



# Program - 4

## Aim:

Program to implement k-NN classification using any random data set without using inbuilt packages.

## Program:

from math import sqrt def e\_dis(r1,r2):

dist=0.0

for i in range(len(r1)-1): dist+=(r1[i]-r2[i])\*\*2

return sqrt(dist)

def get\_ne(train,test\_row,num\_neig): distances=list()

for train\_row in train: dist=e\_dis(test\_row,train\_row) distances.append([test\_row,train\_row])

distances.sort(key=lambda tup:tup[1]) neighbors=list()

for i in range(num\_neig): neighbors.append(distances[i][0])

return neighbors

def predict\_classif(train,test\_row,num\_neig): neighbors = get\_ne(train,test\_row,num\_neig) out\_val=[row[-1] for row in neighbors]

prediction=max(set(out\_val),key=out\_val.count) return prediction

dataset=[[2.734,2.55,0],

[1.45,3.36,0],

[2.334, 2.355, 0],

[1.45, 3.36, 0],

[2.334, 2.55, 0],

[1.45, 3.336, 0],

[3.334, 3.55, 1],

[1.45, 3.36, 1],

[3.734, 4.55, 1],

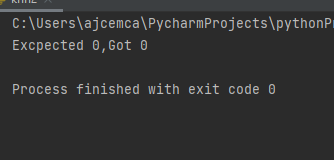
[3.45, 4.36, 1],

[4.734, 5.55, 1],

[3.45, 5.36, 1]]

prediction=predict\_classif(dataset,dataset[0],3) print('Excpected %d,Got %d'%(dataset[0][-1],prediction))

**OUTPUT**



# Program - 5

Date:08/12/2021



## Aim:

Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm

## Program:

import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.naive\_bayes import GaussianNB

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

dataset=pd.read\_csv('Social\_Network\_Ads.csv') x=dataset.iloc[:,[2,3]].values

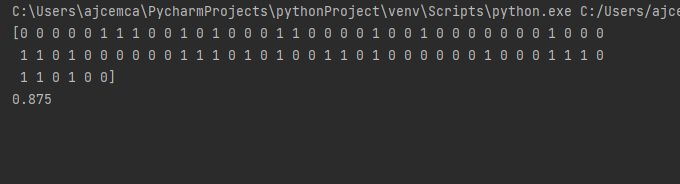
y=dataset.iloc[:,-1].values x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30)

sc=StandardScaler() x\_train=sc.fit\_transform(x\_train) x\_test=sc.transform(x\_test)

classifier=GaussianNB() classifier.fit(x\_train,y\_train)

y\_pred=classifier.predict(x\_test) print(y\_pred)

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

**OUTPUT**

# Program - 6

## Aim:

Program to implement linear regression techniques using any standard dataset available in the public domain and evaluate its performance.

## Program(inbuilt):

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression x=np.array([5,15,25,35,45,55]).reshape((-1,1)) y=np.array([5,20,14,32,22,38])

print(x) print(y)

model=LinearRegression() model.fit(x,y) r\_sq=model.score(x,y)

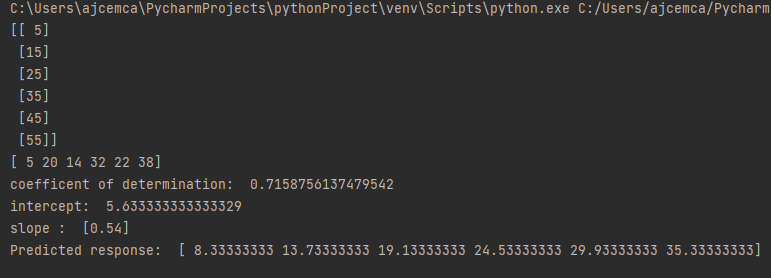
print('coefficent of determination: ',r\_sq) print('intercept: ',model.intercept\_) print('slope : ',model.coef\_) y\_pred=model.predict(x) print('Predicted response: ',y\_pred) plt.scatter(x,y,color="g") plt.plot(x,y\_pred)

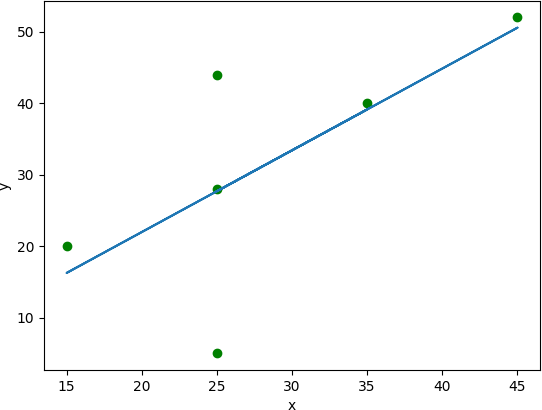
plt.xlabel('x')

plt.ylabel('y')

plt.show()

**OUTPUT**





## Program:7 (Without inbuilt):

### Aim:

Program to implement linear regression techniques using any standard dataset available in the public domain and evaluate its performance.

**Program:**

import numpy as np

import matplotlib.pyplot as plt def estimate\_coef(x,y):

n=np.size(x) m\_x=np.mean(x) m\_y=np.mean(y)

SS\_xy=np.sum(y\*x) - n \*m\_y\* m\_x SS\_xx=np.sum(x\*x) - n \*m\_x\* m\_x b\_1=SS\_xy / SS\_xx

b\_0=m\_y - b\_1\* m\_x return (b\_0,b\_1)

def plot\_regr\_line(x,y,b): plt.scatter(x,y,color="m",marker="o",s=30) y\_pred=b[0]+b[1]\*x plt.plot(x,y\_pred,color="g")

plt.xlabel('x')

plt.ylabel('y') plt.show()

def main():

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])

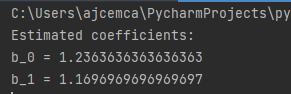
b = estimate\_coef(x, y)

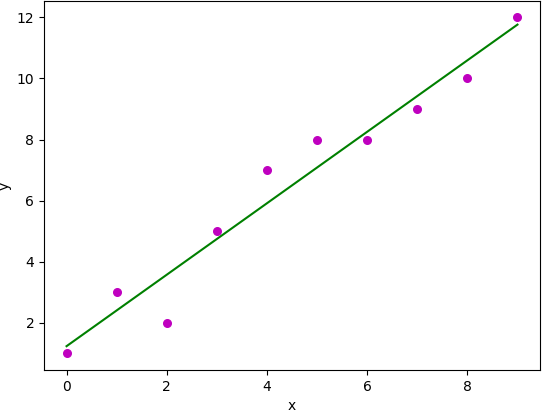
print("Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}".format(b[0], b[1])) plot\_regr\_line(x, y, b)

if name ==" main ": main()

**OUTPUT**





# Program - 8

Date:15-12-2021



## Aim:

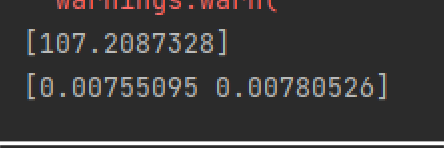
Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

**Program:**

import pandas df=pandas.read\_csv("cars.csv") x=df[['Weight','Volume']] y=df['CO2']

from sklearn import linear\_model regr=linear\_model.LinearRegression() regr.fit(x,y) predictedco2=regr.predict([[2300,1300]]) print(predictedco2)

**OUTPUT**



# Program - 9

Date:15-12-2021



## Aim:

Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

import matplotlib.pyplot as plt

from sklearn import datasets,linear\_model,metrics

boston=datasets.load\_boston()

x=boston.data y=boston.target

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split( x,y,test\_size=0.4,random\_state=1)

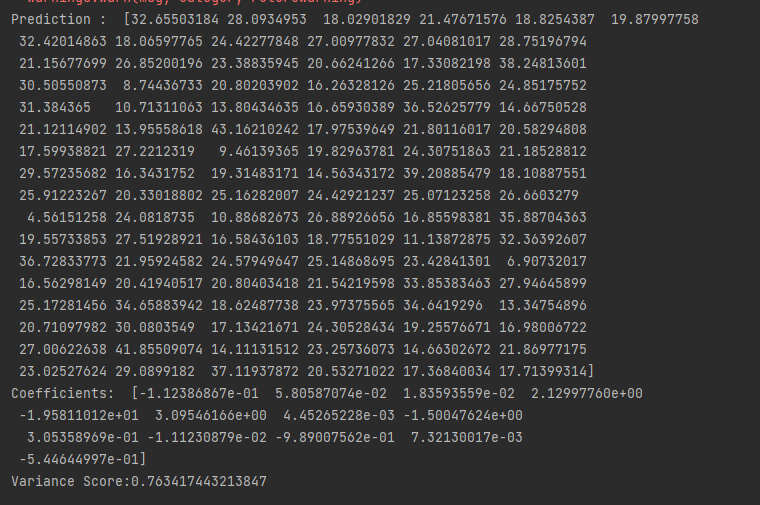
reg=linear\_model.LinearRegression() reg.fit(x\_train,y\_train)

pre=reg.predict(x\_test) print("Prediction : ",pre) print('Coefficients: ',reg.coef\_)

print('Variance Score:{}'.format(reg.score(x\_test,y\_test)))

**OUTPUT**





# Program - 10

Date:22-12-2021



## Aim:

Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm

### Program:

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.tree import plot\_tree df=sns.load\_dataset('iris') print(df.head())

print(df.info())

df.isnull().any() print(df.shape)

sns.pairplot(data=df,hue='species')

plt.savefig("pne.png")

sns.heatmap(df.corr()) plt.savefig("one.png") target=df['species'] df1=df.copy() df1=df1.drop('species',axis=1) print(df1.shape) print(df1.head())

x=df1 print(target)

le=LabelEncoder() target=le.fit\_transform(target) print(target)

y=target

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

print("Training split input",x\_train.shape) print("Testing split input",x\_test.shape)

dtree=DecisionTreeClassifier()

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

print("Decision tree classifier created")

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

print("classsification report \n",classification\_report(y\_test,y\_pred)) cm=confusion\_matrix(y\_test,y\_pred)

plt.figure(figsize=(5,5))

sns.heatmap(data=cm,linewidth=5,annot=True,square=True,cmap='Blues')

plt.ylabel('Actual label') plt.xlabel('Predictd label')

all\_sample\_title='Accuracy Score:{0}'.format(dtree.score(x\_test,y\_test))

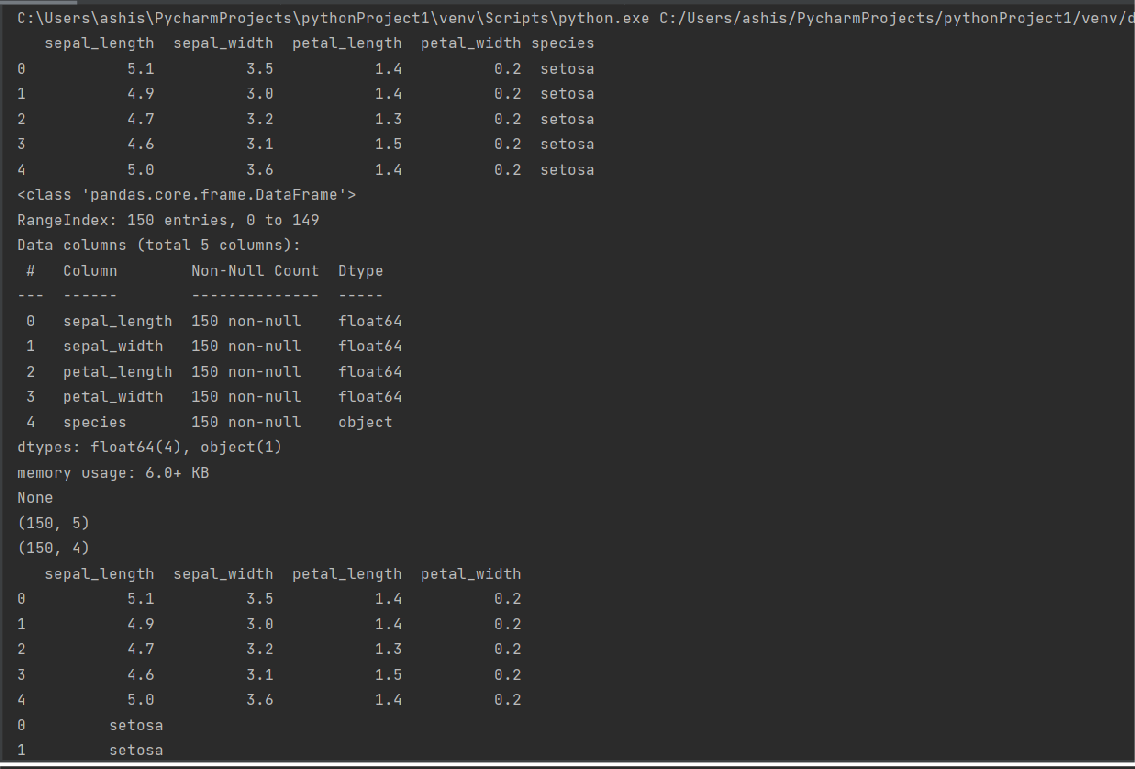
plt.savefig("two.png")

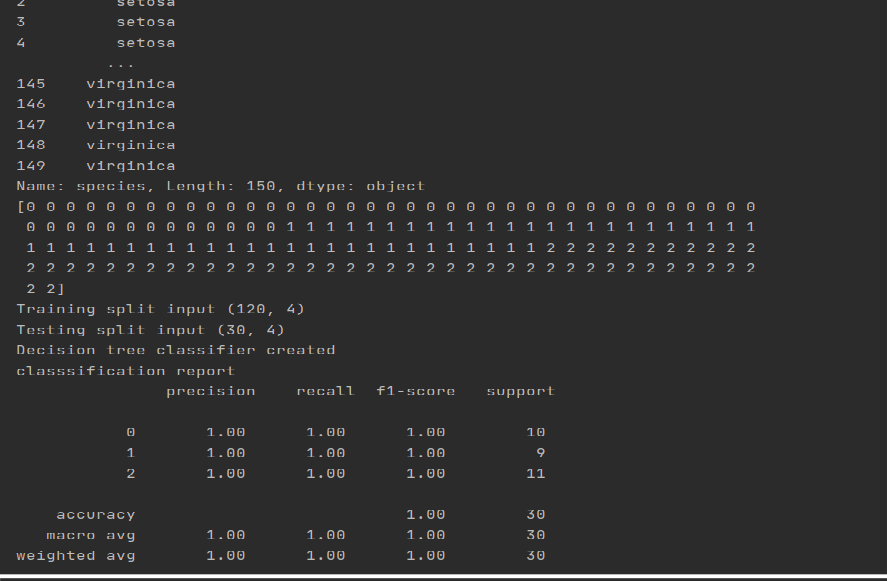
plt.figure(figsize=(20,20)) dec\_tree=plot\_tree(decision\_tree=dtree,feature\_names=df1.columns,

class\_names=["setosa","vercicikor","verginica"],filled=True,precision=4,rounded=True) plt.savefig("three.png")

**OUTPUT**

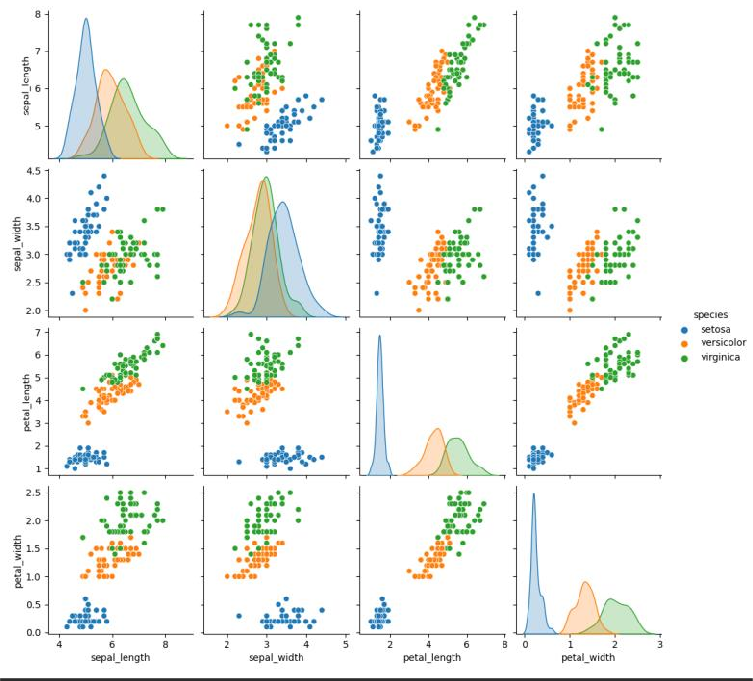




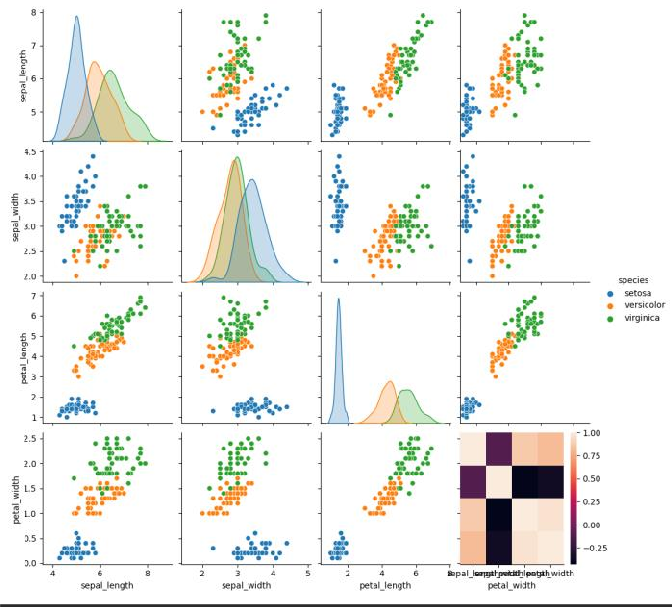


**Pne.png**



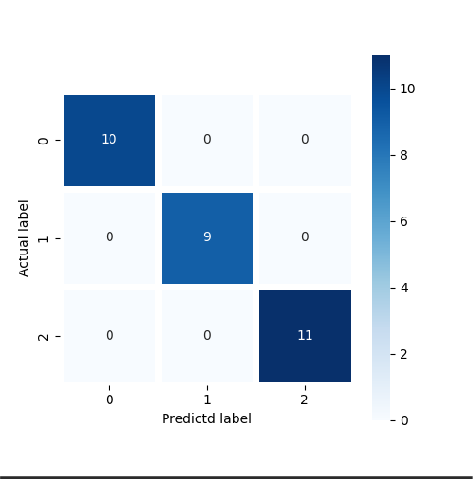


**One.png**

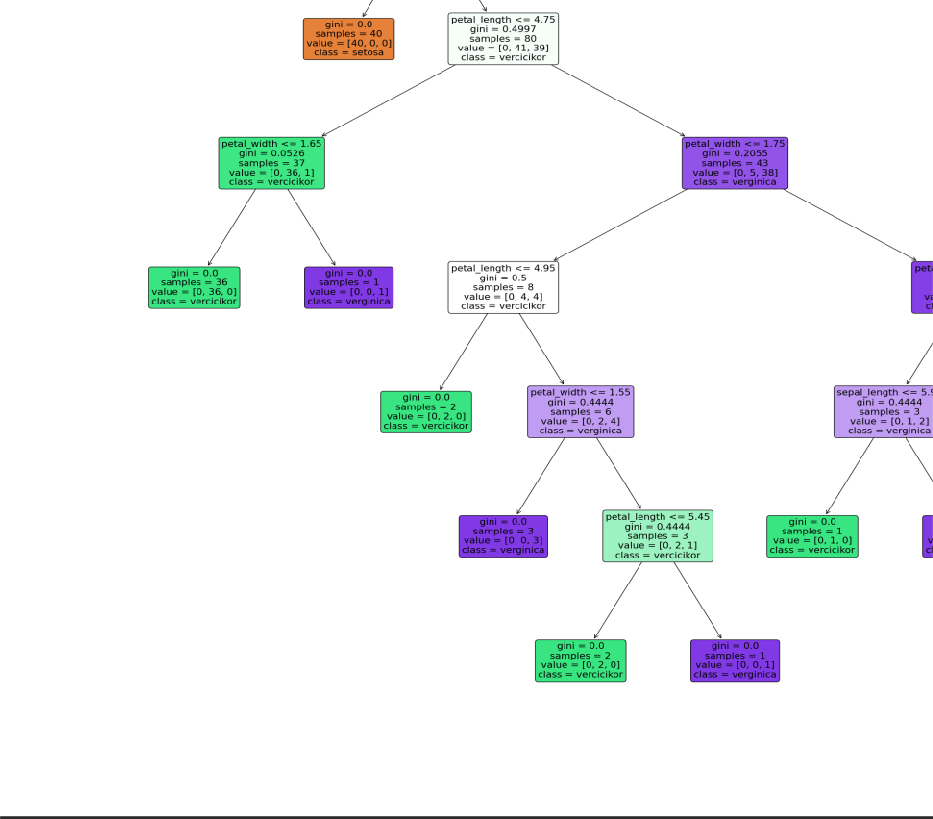


**Two.png**





**Three.png**



# Program - 11

Date:05-01-2022



## Aim:

Program to implement k-means clustering technique using any standard dataset available in the public domain

**Program:**

import numpy as nm

import matplotlib.pyplot as mtp import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv') x=dataset.iloc[:,[3,4]].values

print(x)

from sklearn.cluster import KMeans wcss\_list=[]

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++',random\_state=42) kmeans.fit(x)

wcss\_list.append(kmeans.inertia\_) mtp.plot(range(1,11),wcss\_list) mtp.title('The Elbow Method Graph') mtp.xlabel('Number of clusters(k)') mtp.ylabel('wcss\_list')

mtp.show()

kmeans=KMeans(n\_clusters=5,init='k-means++',random\_state=42) y\_predict=kmeans.fit\_predict(x)

print(y\_predict)

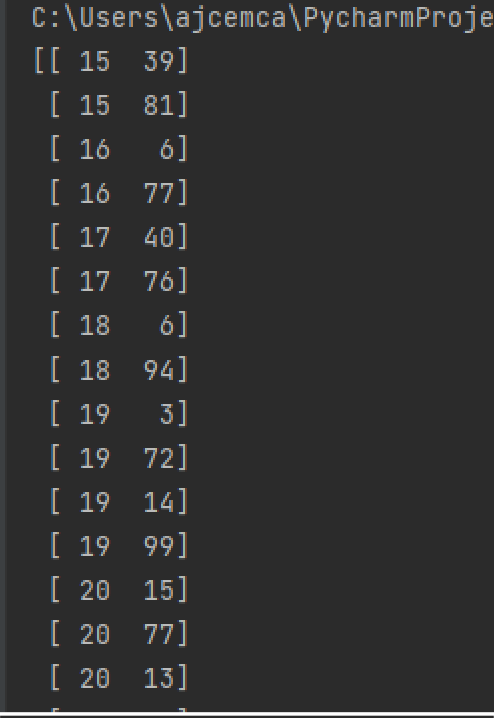
mtp.scatter(x[y\_predict ==0,0],x[y\_predict ==0,1],s=100,c='blue',label='cluster 1') mtp.scatter(x[y\_predict ==1,0],x[y\_predict ==1,1],s=100,c='green',label='cluster 2') mtp.scatter(x[y\_predict ==2,0],x[y\_predict ==2,1],s=100,c='red',label='cluster 3') mtp.scatter(x[y\_predict ==3,0],x[y\_predict ==3,1],s=100,c='cyan',label='cluster 4') mtp.scatter(x[y\_predict ==4,0],x[y\_predict ==4,1],s=100,c='magenta',label='cluster 5') mtp.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=300,c='black',label='cluster') mtp.title('Clusters of customers')

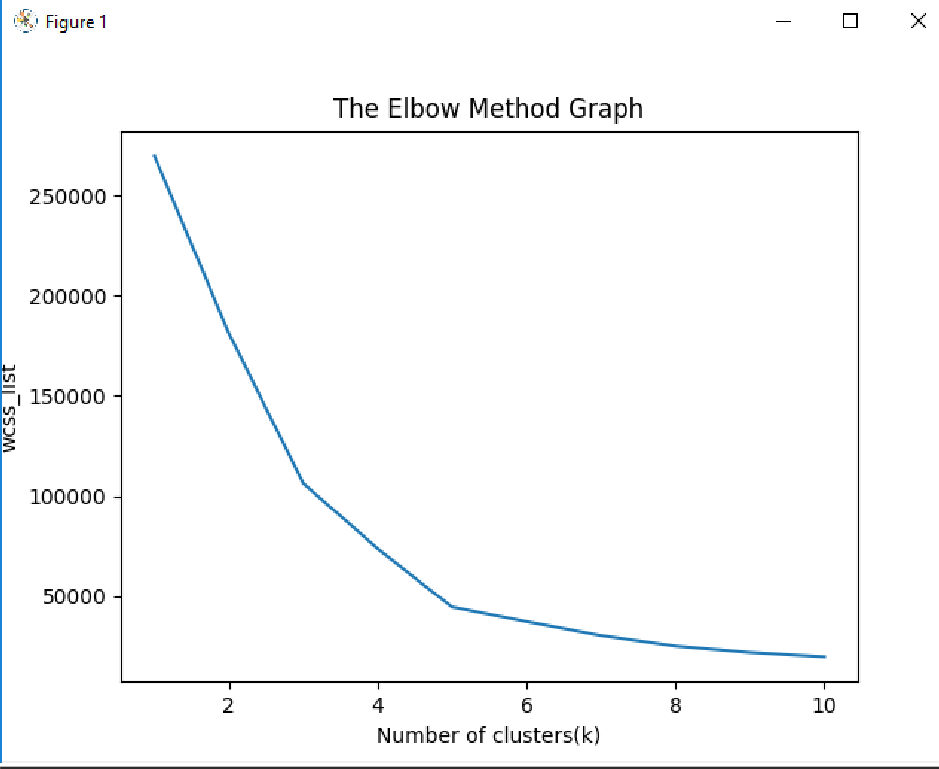
mtp.xlabel('Annual Income (K$)') mtp.ylabel('Spending Score(1-100)') mtp.legend()

mtp.show()

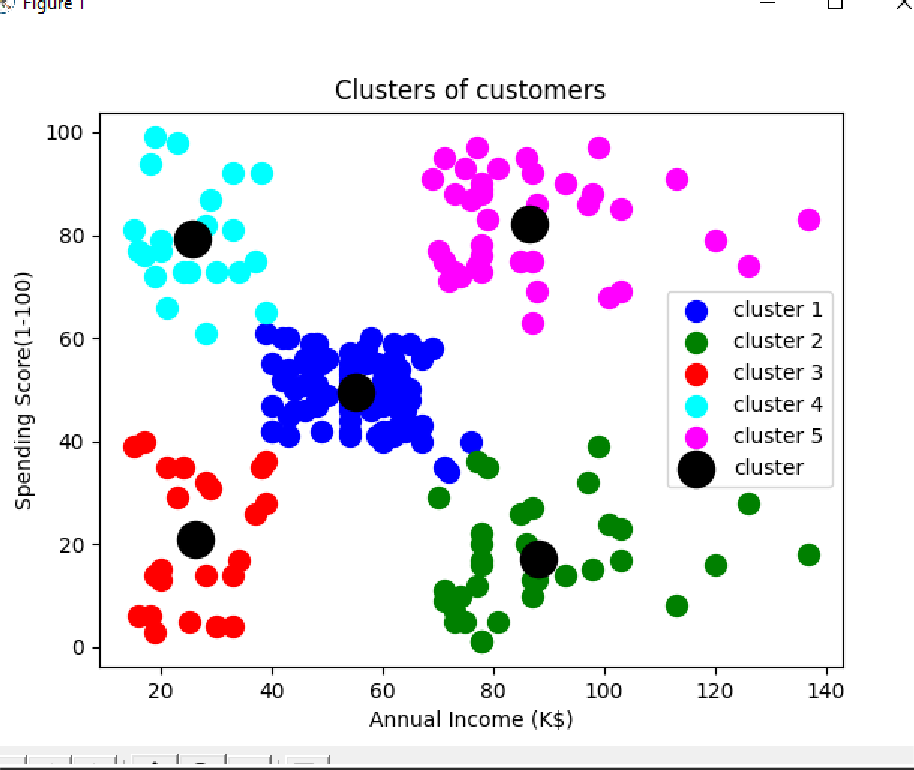
**OUTPUT**











# Program - 12

Date:05-01-2022



## Aim:

Program to implement k-means clustering technique using any standard dataset available in the public domain (Using world\_country\_and\_usa\_states\_latitude\_and\_longitude\_values.csv)

**PROGRAM**

import numpy as nm

import matplotlib.pyplot as mtp import pandas as pd

dataset = pd.read\_csv('world\_country\_and\_usa\_states\_latitude\_and\_longitude\_values.csv') x=dataset.iloc[:,[1,2]].values

print(x)

from sklearn.cluster import KMeans wcss\_list=[]

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++',random\_state=42) kmeans.fit(x)

wcss\_list.append(kmeans.inertia\_) mtp.plot(range(1,11),wcss\_list) mtp.title('The Elbow Method Graph') mtp.xlabel('Number of clusters(k)') mtp.ylabel('wcss\_list')

mtp.show()

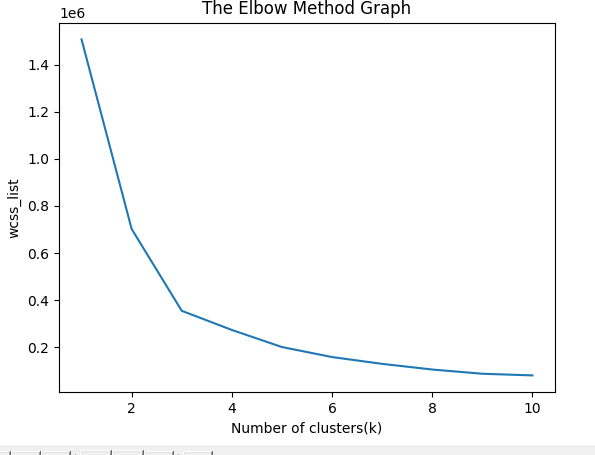
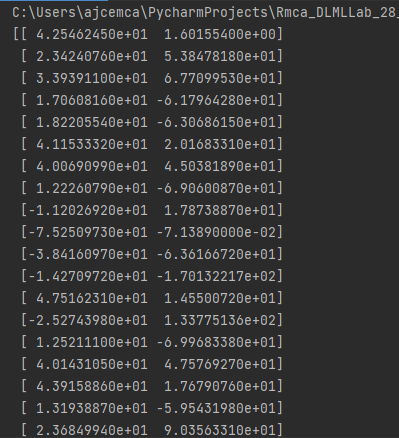
kmeans=KMeans(n\_clusters=3,init='k-means++',random\_state=42) y\_predict=kmeans.fit\_predict(x)

print(y\_predict)

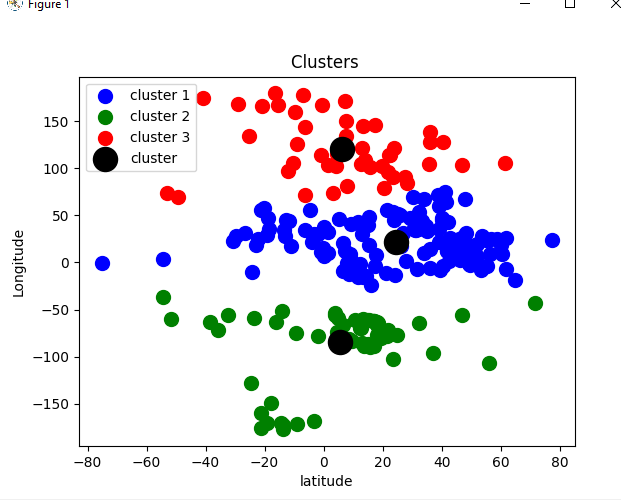
mtp.scatter(x[y\_predict ==0,0],x[y\_predict ==0,1],s=100,c='blue',label='cluster 1') mtp.scatter(x[y\_predict ==1,0],x[y\_predict ==1,1],s=100,c='green',label='cluster 2') mtp.scatter(x[y\_predict ==2,0],x[y\_predict ==2,1],s=100,c='red',label='cluster 3') mtp.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=300,c='black',label='cluster') mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (K$)') mtp.ylabel('Spending Score(1-100)') mtp.legend()

mtp.show()

**OUTPUT**





# Program – 13

# AIM: PROGRAMS ON FEEDFORWARD NETWORK TO CLASSIFY ANY STANDARD DATASET AVAILABLE IN THE PUBLIC DOMAIN

**PROGRAM**

# from tensorflow import keras

# print('Tensorflow/keras : %s'%keras.version)

# from keras.models import Sequential

# from keras import Input

# from keras.layers import Dense

# import pandas as pd

# print('pandas : %s' %pd.version)

# import numpy as np

# print('numpy : %s' %np.version)

# import sklearn

# print('sklearn : %s' %sklearn.version)

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

# from sklearn.metrics import classification\_report

# import plotly

# import plotly.express as px

# import plotly.graph\_objects as go

# print('plotly : %s' %plotly.version)

# pd.options.display.max\_columns=50

# df=pd.read\_csv('weatherAUS.csv', encoding='utf-8')

# df=df[pd.isnull(df['RainTomorrow'])==False]

# #df=df.fillna(df.mean())

# df['RainTodayFlag']=df['RainToday'].apply(lambda x: 1 if x=='Yes' else 0)

# df['RainTomorrowFlag']=df['RainTomorrow'].apply(lambda x: 1 if x=='Yes' else 0)

# print(df)

# X = df[['Humidity3pm']]

# Y = df['RainTomorrowFlag'].values

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

# model = Sequential(name="Model-with-One-Input")

# model.add(Input(shape=(1,), name='Input-Layer'))

# model.add(Dense(2, activation='softplus', name='Hidden-Layer'))

# model.add(Dense(1, activation='sigmoid', name='Output-Layer'))

**OUTPUT**

# 

# 

# 

# Date:02-02-2022

# Program – 14

# AIM: PROGRAMS ON CONVOLUTIONAL NEURAL NETWORK TO CLASSIFY

# IMAGES FROM ANY STANDARD DATASET IN THE PUBLIC DOMAIN.

**PROGRAM**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

np.random.seed(42)

# tf.set.random. seed(42)

fashion\_mnist = keras.datasets.fashion\_mnist

(X\_train, y\_train), (X\_test, y\_test) = fashion\_mnist.load\_data()

print(X\_train.shape, X\_test.shape)

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

plt.imshow(X\_train[1], cmap='binary')

plt.show()

np.unique(y\_test)

class\_names = ['T-Shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', '8ag', 'Ankle Boot']

n\_rows = 5

n\_cols = 10

plt.figure(figsize=(n\_cols \* 1.4, n\_rows \* 1.6))

for row in range(n\_rows):

for col in range(n\_cols):

index = n\_cols \* row + col

plt.subplot(n\_rows, n\_cols, index + 1)

plt.imshow(X\_train[index], cmap='binary', interpolation='nearest')

plt.axis('off')

plt.title(class\_names[y\_train[index]])

plt.show()

model\_CNN = keras.models.Sequential()

model\_CNN.add(keras.layers.Conv2D(filters=32, kernel\_size=7, padding='same', activation='relu', input\_shape=[28, 28, 1]))

model\_CNN.add(keras.layers.MaxPooling2D(pool\_size=2))

model\_CNN.add(keras.layers.Conv2D(filters=64, kernel\_size=3, padding='same', activation='relu'))

model\_CNN.add(keras.layers.MaxPooling2D(pool\_size=2))

model\_CNN.add(keras.layers.Conv2D(filters=32, kernel\_size=3, padding='same', activation='relu'))

model\_CNN.add(keras.layers.MaxPooling2D(pool\_size=2))

model\_CNN.summary()

model\_CNN.add(keras.layers.Flatten())

model\_CNN.add(keras.layers.Dense(units=128, activation='relu'))

model\_CNN.add(keras.layers.Dense(units=64, activation='relu'))

model\_CNN.add(keras.layers.Dense(units=10, activation='softmax'))

model\_CNN.summary()

model\_CNN.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

X\_train = X\_train[..., np.newaxis]

X\_test = X\_test[..., np.newaxis]

history\_CNN = model\_CNN.fit(X\_train, y\_train, epochs=2, validation\_split=0.1)

pd.DataFrame(history\_CNN.history).plot()

plt.grid(True)

plt.xlabel('epochs')

plt.ylabel('loss/accuracy')

plt.title('Training and validation plot')

plt.show()

test\_loss, test\_accuracy = model\_CNN.evaluate(X\_test, y\_test)

print(' Test Loss :{}, Test Accuracy : {}'.format(test\_loss, test\_accuracy))

**output**



