Date:01/12/2021

**Program - 2**

**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)

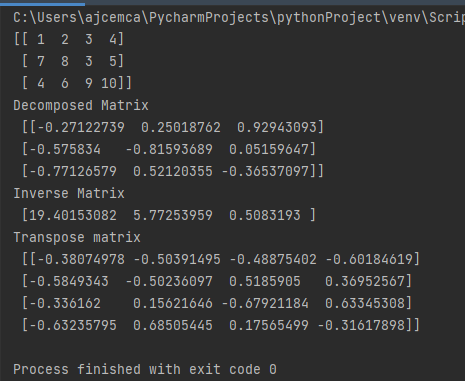
u,s,vt=svd(a)

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)

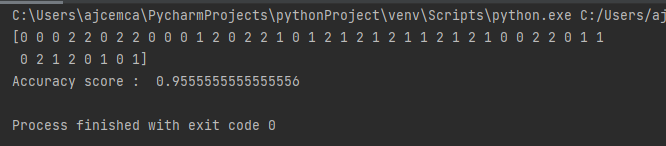
knn.fit(x\_train,y\_train)

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.

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**

