**Program 2**

**Aim:** Perform svd (Singular Value Decomposition) using python.

**Program:**

from numpy import array  
from scipy.linalg import svd

ar = array([[3, 2, 4], [7, 3, 5]])

print(ar)

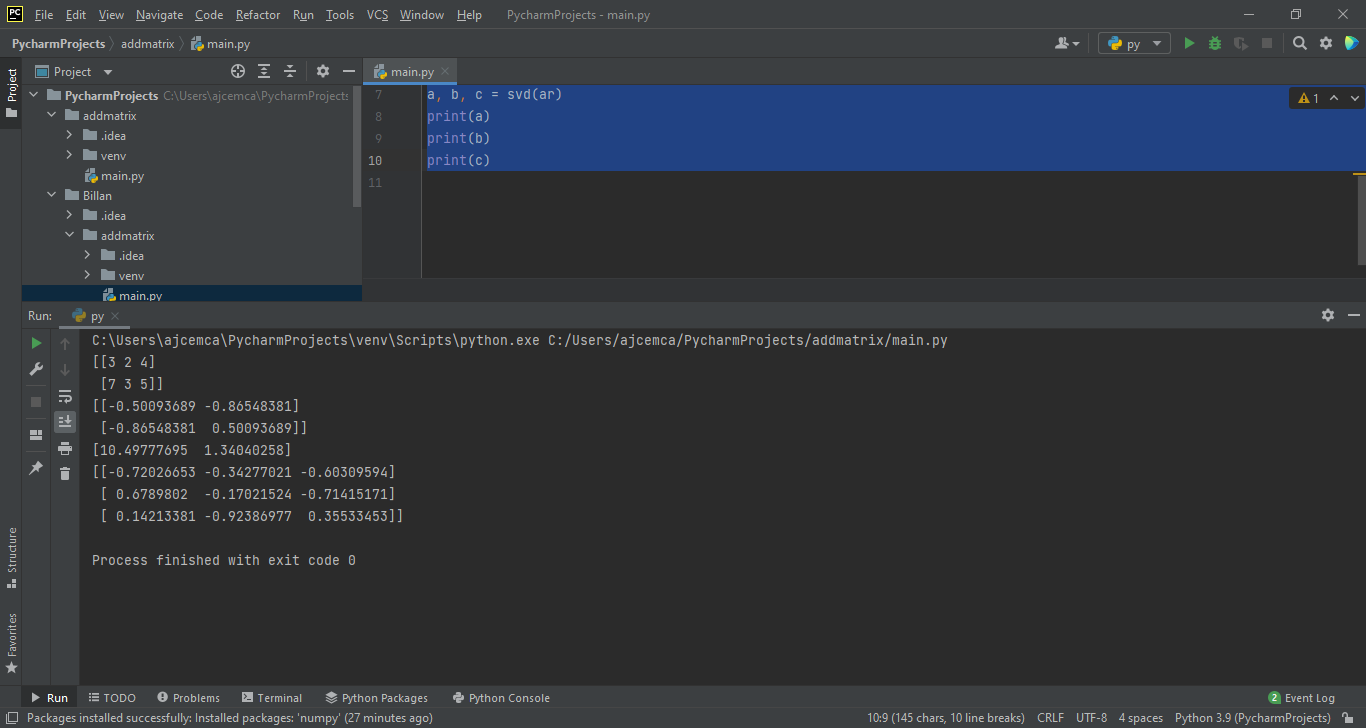
a, b, c = svd(ar)

print(a)

print(b)

print(c)

**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 import metrics

irisData = load\_iris()

x = irisData.data

y = irisData.target

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.35, random\_state=35)

knn = KNeighborsClassifier(n\_neighbors=9)

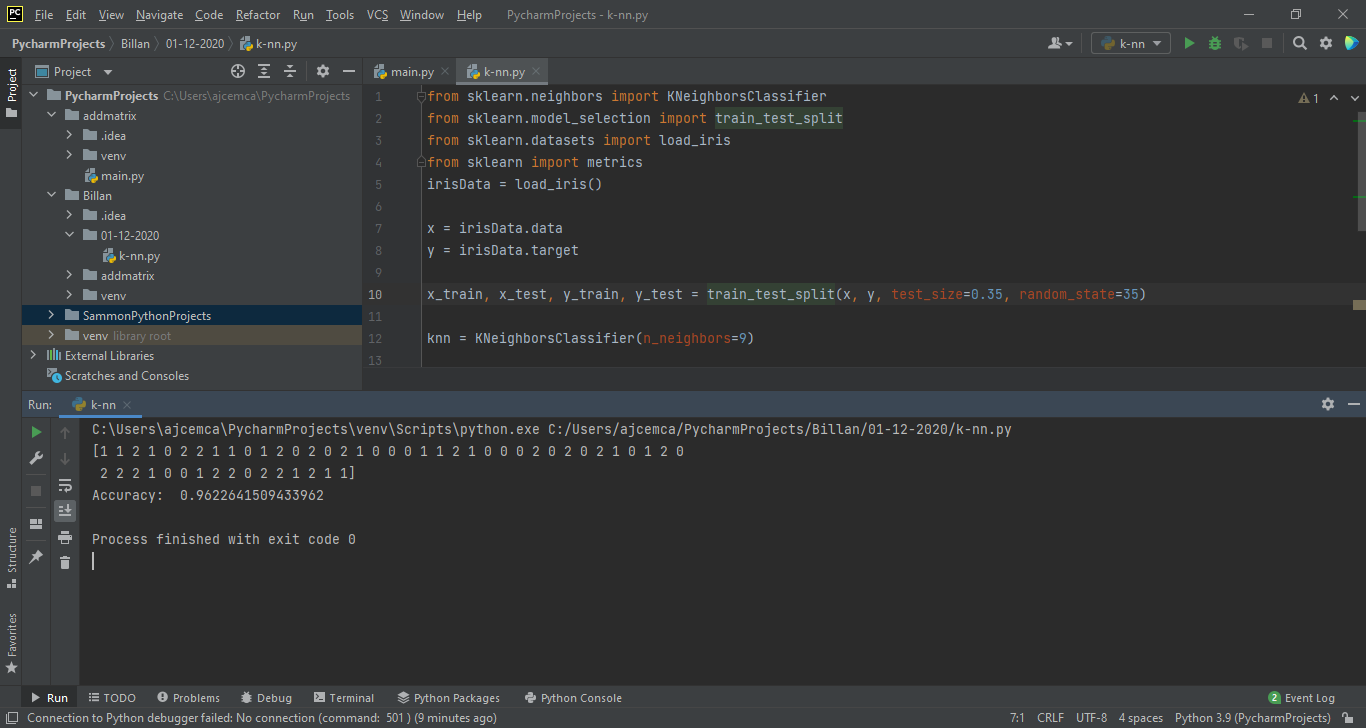
knn.fit(x\_train, y\_train)

print(knn.predict(x\_test))

acc = metrics.accuracy\_score(y\_test, knn.predict(x\_test))

print("Accuracy: ", acc)

**Output:**

****

**Program 3**

**Aim:** Program to implement k-NN algorithm using any random dataset without using inbuilt packages.

**Program:**

import math

from math import sqrt

def euclidean\_dist(row1, row2):

dist = 0.0

for i in range(len(row1)-1):

dist += (row1[i] - row2[i]) \*\* 2

return math.sqrt(dist)

def get\_neighbors(train, test\_row, num\_neighbors):

distances = list()

for train\_row in train:

dist = euclidean\_dist(test\_row, train\_row)

distances.append([train\_row, dist])

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

neighbors = list()

for i in range(num\_neighbors):

neighbors.append(distances[i][0])

return neighbors

def predict\_classif(train, test\_row, num\_neighbors):

neighbors = get\_neighbors(train, test\_row, num\_neighbors)

output\_val = [row[-1] for row in neighbors]

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

return prediction

dataset = [[2.78, 2.55, 0],

[1.26, 2.36, 0],

[3.39, 4.40, 0],

[1.38, 1.85, 0],

[5.74, 3.56, 0],

[3.53, 5.67, 1],

[6.24, 3.67, 1],

[2.45, 1.78, 1]]

prediction = predict\_classif(dataset, dataset[0], 3)

print("Expected ", dataset[0][-1], " got ", prediction)

**Output**

