1) **Perform SVD(Singular Value Composition)using python.**

Program

```
from numpy import array
from scipy.linalg import svd
# define a matrix
A= array([[2,2], [3,2], [5,3], [8,5]])
print(A)
#svd
a,b,c = svd(A)
print(a)
print(b)
print(c)
```

Output

```
[[2 2]
 [3 2]
 [5 3]
 [8 5]]
[[-0.23054399  0.94657383  0.21844711  0.05593136]
 [-0.30083898  0.10578668 -0.58495139 -0.74574591]
 [-0.48640597 -0.26171355  0.72573128 -0.41015773]
 [-0.78724495 -0.15592687 -0.28883706  0.52202046]]
[11.98286716  0.64101067]
[[-0.8423355  -0.53895353]
 [-0.53895353  0.8423355 ]]
```

2) Program to implement KNN classification using any standard available in the public domain and find the accuracy of the algorithm.

Program

```
# Import necessary modules
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


# Loading data
irisData = load_iris()
```

```
# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size = 0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=2)

knn.fit(X_train, y_train)

# Predict on dataset which model has not seen before
print(knn.predict(X_test))
a = accuracy_score(y_test, p)
print(a)
```

Output

```
C:\Users\ajcemca\PycharmProjects\file\venv\Scripts\python.exe
C:/Users/ajcemca/PycharmProjects/file/knn.py
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
1.0
```

3) Program to implement KNN classification using random data set without using inbuilt packages.

Program

```
# Example of making predictions
from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
  distance = 0.0
  for i in range(len(row1)-1):
     distance += (row1[i] - row2[i])**2
  return sqrt(distance)

# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
  distances = list()
  for train_row in train:
    dist = euclidean_distance(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

# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
  neighbors = get_neighbors(train, test_row, num_neighbors)
  output_values = [row[-1] for row in neighbors]
  prediction = max(set(output_values), key=output_values.count)
  return prediction

# Test distance function
dataset = [[2.7810836,2.550537003,0],
  [1.465489372,2.362125076,0],
  [3.396561688,4.400293529,0],
  [1.38807019,1.850220317,0],
  [3.06407232,3.005305973,0],
  [7.627531214,2.759262235,1],
  [5.332441248,2.088626775,1],
  [6.922596716,1.77106367,1],
  [8.675418651,-0.242068655,1],
  [7.673756466,3.508563011,1]]
prediction = predict_classification(dataset, dataset[0], 3)
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

Output

```
C:\Users\ajcemca\PycharmProjects\file\venv\Scripts\python.exe
C:/Users/ajcemca/PycharmProjects/file/math.py
Expected 0, Got 0.
```