Introduction

Cluster analysis aims at partitioning the input data into groups of closely related instances that belong to the same cluster and are more similar to each other than to instances that belong to other clusters.

K-means Clustering

The K-means clustering algorithm represents each cluster by its correspoinding cluster centroid. The algorithm partitions the input data into k disjoint clusters by iteratively applying the following steps

- Form K clusters by assigning each instance to its nearest centroid
- Recompute the centroid of each cluster.

In [1]:
```
# Creating the dataset
import pandas as pd
```

In [2]:
```
ratings =[['John',5,5,2,1],['Mary',4,5,3,2],['Bob',4,4,4,3],['Lisa',2,2,4,5],['Lee',1,2,3,4],['Harry',2,1,5,
```

In [3]:
```
ratings
```

Out[3]:
```
[['John', 5, 5, 2, 1],
 ['Mary', 4, 5, 3, 2],
 ['Bob', 4, 4, 4, 3],
 ['Lisa', 2, 2, 4, 5],
 ['Lee', 1, 2, 3, 4],
 ['Harry', 2, 1, 5, 5]]
```

In [4]:
```
titles = ['User','Jaws','Star Wars','Exorcist','Omen']
```

In [5]:
```
titles
```

Out[5]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen']
```

In [6]:
```
movies=pd.DataFrame(ratings, columns=titles)
```

In [7]:
```
movies
```

Out[7]:

	User	Jaws	Star Wars	Exorcist	Omen
0	John	5	5	2	1
1	Mary	4	5	3	2
2	Bob	4	4	4	3
3	Lisa	2	2	4	5
4	Lee	1	2	3	4
5	Harry	2	1	5	5

In [8]:
```
# comments first users- action movies - Jaws, Star Wars
# last 3 users - Horror movies
# Our goal is to use k-means clusting on the users to identify groups of users with similar
# movie preferences k=2
```

In [10]:
```
from sklearn import cluster
```

In [11]:
```
data= movies.drop('User', axis=1)
```

In [12]:
```
data
```

Out[12]:

	Jaws	Star Wars	Exorcist	Omen
0	5	5	2	1
1	4	5	3	2
2	4	4	4	3
3	2	2	4	5
4	1	2	3	4
5	2	1	5	5

In [13]:
```
k_means=cluster.KMeans(n_clusters=2, max_iter=50,random_state=1)
```

In [14]:
```
k_means
```

Out[14]:
```
KMeans(max_iter=50, n_clusters=2, random_state=1)
```

In [15]:
```
k_means.fit(data)
```

Out[15]:
```
KMeans(max_iter=50, n_clusters=2, random_state=1)
```

In [16]:
```
labels=k_means.labels_
```

In [17]:
```
labels
```

Out[17]:
```
array([0, 0, 0, 1, 1, 1])
```

In [20]:
```
pd.DataFrame(labels,index=movies.User, columns=['Cluster ID'])
```

Out[20]:

User	Cluster ID
John	0
Mary	0
Bob	0
Lisa	1
Lee	1
Harry	1

In [22]:
```
# K-means has assigned first users to one cluster and the last 3 users to the second cluster.
# This is consistent with our expectation.
#We can display the centroid for each of the two cluster
```

In [23]:
```
centroid=k_means.cluster_centers_
```

In [24]:
```
centroid
```

Out[24]:
```
array([[4.33333333, 4.66666667, 3.        , 2.        ],
       [1.66666667, 1.66666667, 4.        , 4.66666667]])
```

In [25]:
```
pd.DataFrame(centroid, columns=data.columns)
```

Out[25]:

	Jaws	Star Wars	Exorcist	Omen
0	4.333333	4.666667	3.0	2.000000
1	1.666667	1.666667	4.0	4.666667

In [26]:
```
import numpy as np
```

In [27]:
```
testData=np.array([[4,5,1,2],[3,2,4,4],[2,3,4,1],[3,2,3,3],[5,4,1,4]])
```

In [28]:
```
testData
```

Out[28]:
```
array([[4, 5, 1, 2],
       [3, 2, 4, 4],
       [2, 3, 4, 1],
       [3, 2, 3, 3],
       [5, 4, 1, 4]])
```

In [29]:
```
labels=k_means.predict(testData)
```

In [30]:
```
labels
```

Out[30]:
```
array([0, 1, 0, 1, 0])
```

In [31]:
```
labels=labels.reshape(-1,1)
```

In [32]:
```
labels
```

Out[32]:
```
array([[0],
       [1],
       [0],
       [1],
       [0]])
```

In [33]:
```
usernames=np.array(['Paul','Kim','Liz','Tom','Bill']).reshape(-1,1)
```

In [34]:
```
usernames
```

Out[34]:
```
array([['Paul'],
       ['Kim'],
       ['Liz'],
       ['Tom'],
       ['Bill']], dtype='<U4')
```

In [35]:
```
cols=movies.columns.to_list()
```

In [36]:
```
cols
```

Out[36]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen']
```

In [37]:
```
cols.append('Cluster ID')
```

In [38]:
```
cols
```

Out[38]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen', 'Cluster ID']
```

In [39]:
```
newuser=pd.DataFrame(np.concatenate((usernames,testData,labels), axis=1),columns=cols)
```

In [40]:
```
newuser
```

Out[40]:

	User	Jaws	Star Wars	Exorcist	Omen	Cluster ID
0	Paul	4	5	1	2	0
1	Kim	3	2	4	4	1
2	Liz	2	3	4	1	0
3	Tom	3	2	3	3	1
4	Bill	5	4	1	4	0

To determine the number of clusters, we can vary the clusters 1-6 and then computer the corresponding sum of squared error. The Elbew in the plow of SSE vs the number of clusters can be used to estimate the number of clusters.

In [41]:
```
import matplotlib.pyplot as plt
```

In [42]:
```
%matplotlib inline
```

In [43]:
```
numCluster = [1,2,3,4,5,6]
```

In [44]:
```
SSE=[]
```

In [45]:
```
for k in numCluster:
    k_means=cluster.KMeans(n_clusters=k)
    k_means.fit(data)
    SSE.append(k_means.inertia_)
```

In [46]:
```
SSE
```

Out[46]:
```
[45.66666666666667, 9.333333333333334, 5.5, 2.5, 1.0, 0.0]
```

In [47]:
```
plt.plot(numCluster, SSE)
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
```

Out[47]:
```
Text(0, 0.5, 'SSE')
```

In [49]:
```
from sklearn.datasets import load_digits
```

In [50]:
```
digits=load_digits()
```

In [51]:
```
digits
```

Out[51]:
```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
  'pixel_1_1',
  'pixel_1_2',
  'pixel_1_3',
  'pixel_1_4',
  'pixel_1_5',
  'pixel_1_6',
  'pixel_1_7',
  'pixel_2_0',
  'pixel_2_1',
  'pixel_2_2',
  'pixel_2_3',
  'pixel_2_4',
  'pixel_2_5',
  'pixel_2_6',
  'pixel_2_7',
  'pixel_3_0',
  'pixel_3_1',
  'pixel_3_2',
  'pixel_3_3',
  'pixel_3_4',
  'pixel_3_5',
  'pixel_3_6',
  'pixel_3_7',
  'pixel_4_0',
  'pixel_4_1',
  'pixel_4_2',
  'pixel_4_3',
  'pixel_4_4',
  'pixel_4_5',
  'pixel_4_6',
  'pixel_4_7',
  'pixel_5_0',
  'pixel_5_1',
  'pixel_5_2',
  'pixel_5_3',
  'pixel_5_4',
  'pixel_5_5',
  'pixel_5_6',
  'pixel_5_7',
  'pixel_6_0',
  'pixel_6_1',
  'pixel_6_2',
  'pixel_6_3',
  'pixel_6_4',
  'pixel_6_5',
  'pixel_6_6',
  'pixel_6_7',
  'pixel_7_0',
  'pixel_7_1',
  'pixel_7_2',
  'pixel_7_3',
  'pixel_7_4',
  'pixel_7_5',
  'pixel_7_6',
  'pixel_7_7'],
 'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ..., 15.,  5.,  0.],
         [ 0.,  3., 15., ..., 11.,  8.,  0.],
         ...,
         [ 0.,  4., 11., ..., 12.,  7.,  0.],
         [ 0.,  2., 14., ..., 12.,  0.,  0.],
         [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

        [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
         [ 0.,  0.,  0., ...,  9.,  0.,  0.],
         [ 0.,  0.,  3., ...,  6.,  0.,  0.],
         ...,
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

        [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
         [ 0.,  0.,  3., ..., 14.,  0.,  0.],
         [ 0.,  0.,  8., ..., 16.,  0.,  0.],
         ...,
         [ 0.,  9., 16., ...,  0.,  0.,  0.],
         [ 0.,  3., 13., ..., 11.,  5.,  0.],
         [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

        ...,

        [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ...,  2.,  1.,  0.],
         [ 0.,  0., 16., ..., 16.,  5.,  0.],
         ...,
         [ 0.,  0., 16., ..., 15.,  0.,  0.],
         [ 0.,  0., 15., ..., 16.,  0.,  0.],
         [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

        [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
         [ 0.,  0., 14., ..., 15.,  1.,  0.],
         [ 0.,  4., 16., ..., 16.,  7.,  0.],
         ...,
         [ 0.,  0.,  0., ..., 16.,  2.,  0.],
         [ 0.,  0.,  4., ..., 16.,  2.,  0.],
         [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

        [[ 0.,  0., 10., ...,  1.,  0.,  0.],
         [ 0.,  2., 16., ...,  1.,  0.,  0.],
         [ 0.,  0., 15., ..., 15.,  0.,  0.],
         ...,
         [ 0.,  4., 16., ..., 16.,  6.,  0.],
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
         [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n--------------------------------------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 5620\n    :Number of Attributes: 64\n    :Attribute Information: 8x8 image of integer pixels in the range 0..16.\n    :Missing Attribute Values: None\n    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a copy of the test set of the UCI ML hand-written digits datasets\nhttps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains images of hand-written digits: 10 classes where\neach class refers to a digit.\n\nPreprocessing programs made available by NIST were used to extract\nnormalized bitmaps of handwritten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the training set and different 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the number of on pixels are counted in each block. This generates\nan input matrix of 8x8 where each element is an integer in the range\n0..16. This reduces dimensionality and gives invariance to small\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of\n    Graduate Studies in Science and Engineering, Bogazici University.\n  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityreduction using relevance weighted LDA. School of\n    Electrical and Electronic Engineering Nanyang Technological University.\n    2005.\n  - Claudio Gentile. A New Approximate Maximal Margin Classification\n    Algorithm. NIPS. 2000."}
```

Loading [MathJax]/extensions/Safe.js