Machine Learning Experiment 8

Guneet Singh Sethi

02913202717

CSE-1

EXPERIMENT-8

Problem Statement

Apply the k-means algorithm and apply it to the selected data. Evaluate the process by measuring the sum of euclidean distance of each example from its class center. Test the performance of the algorithm as a function of the parameter k.

Algorithm

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

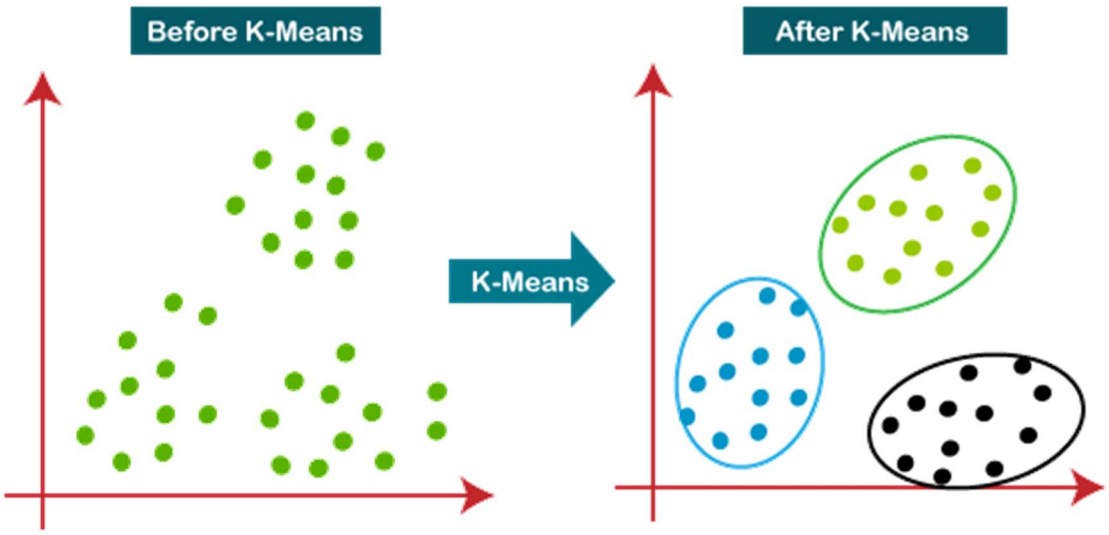
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Program Screenshots

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [2]:
```
df1 = pd.read_csv("Country-data.csv")
df2 = pd.read_csv("creditcard.csv")
```

In [3]:
```
df1.head()
```

Out[3]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

In [4]:
```
df2.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601

5 rows × 31 columns

In [5]:
```
np.unique(df2["Class"].values)
```

Out[5]:
```
array([0, 1], dtype=int64)
```

In [6]:
```
df11 = df2[df2.columns[1:]]
df21 = df2[df2.columns[:-1]]
```

In [7]:
```
X1 = df11.values
X2 = df21.values
```

In [8]:
```
X1.shape, X2.shape
```

Out[8]:
```
((284807, 30), (284807, 30))
```

Country Data for k = 2

In [9]:
```
model = KMeans(n_clusters=2, n_init=4, verbose=1, max_iter=3)
```

In [10]:
```
model.fit(X1)
```

```
Initialization complete
Iteration 0, inertia 15355477175.982346
Iteration 1, inertia 11029741483.103212
Iteration 2, inertia 10392052620.156
Initialization complete
Iteration 0, inertia 11114980702.64728
Iteration 1, inertia 9264719549.548733
Iteration 2, inertia 9102721435.752607
Initialization complete
Iteration 0, inertia 12335933988.245417
Iteration 1, inertia 10054582314.42063
Iteration 2, inertia 9625872693.42902
Initialization complete
Iteration 0, inertia 12731282274.103073
Iteration 1, inertia 10247849537.812023
Iteration 2, inertia 9758780502.942104
```

Out[10]:
```
KMeans(max_iter=3, n_clusters=2, n_init=4, verbose=1)
```

In [11]:
```
model.cluster_centers_
```

Out[11]:
```
array([[ 3.26057855e-02,  5.76572977e-02,  2.51826835e-02,
        -1.17976378e-02,  3.44565898e-02, -1.85262634e-02,
        -3.69604248e-02,  9.59436742e-03,  3.28556924e-03,
         7.94181926e-03, -7.87280491e-04,  8.33871426e-04,
        -5.53236011e-04, -3.21306631e-03, -6.21150418e-04,
        -1.23735478e-03, -3.89169934e-04, -1.20998738e-03,
         3.51151916e-03, -2.11103215e-02, -6.01010005e-03,
         4.99521799e-03,  6.32069988e-03, -7.33291634e-04,
         2.19754330e-03, -1.24054131e-04, -3.76137105e-04,
        -9.19022240e-04,  7.06226133e+01,  1.71319122e-03],
       [-3.19518565e+00, -5.65009452e+00, -2.46776293e+00,
         1.15610289e+00, -3.37655418e+00,  1.81547079e+00,
         3.62191609e+00, -9.40194650e-01, -3.21967514e-01,
        -7.78254120e-01,  7.71491098e-02, -8.17147623e-02,
         5.42140524e-02,  3.14862629e-01,  6.08692865e-02,
         1.21253887e-01,  3.81364892e-02,  1.18572034e-01,
        -3.44109348e-01,  2.06869410e+00,  5.88956381e-01,
        -4.89503583e-01, -6.19393437e-01,  7.18585020e-02,
        -2.15347022e-01,  1.21566150e-02,  3.68593445e-02,
         9.00590685e-02,  1.82549762e+03,  3.12825860e-03]])
```

In [12]:
```
model.inertia_
```

Out[12]:
```
8993108022.120855
```

Credit Card Data for k = 2

In [13]:
```
model = KMeans(n_clusters=2, n_init=4, verbose=1, max_iter=3)
model.fit(X2)
```

```
Initialization complete
Iteration 0, inertia 133197499804769.97
Iteration 1, inertia 118387122772311.6
Iteration 2, inertia 118320729057132.44
Converged at iteration 2: center shift 4116.304743573539 within tolerance 7517.262262351321.
Initialization complete
Iteration 0, inertia 217397134618389.03
Iteration 1, inertia 119969342982985.11
Iteration 2, inertia 118332620375948.28
Initialization complete
Iteration 0, inertia 186568546424018.4
Iteration 1, inertia 119451025740449.81
Iteration 2, inertia 118329474501541.7
Initialization complete
Iteration 0, inertia 179793823891819.7
Iteration 1, inertia 118406789111879.89
Iteration 2, inertia 118323452868119.0
```

Out[13]:
```
KMeans(max_iter=3, n_clusters=2, n_init=4, verbose=1)
```

In [14]:
```
model.cluster_centers_
```

Out[14]:
```
array([[ 1.41083313e+05,  2.73767049e-01, -5.15444169e-02,
        -7.06911008e-01, -1.55103617e-01,  2.79380502e-01,
        -8.09820247e-02,  1.21060177e-01, -6.12691562e-02,
         4.03453498e-02,  3.45519063e-02, -2.43862400e-01,
         3.77698228e-02, -3.37779537e-02, -6.54987759e-02,
        -2.12086853e-01,  1.37932481e-02, -5.10405524e-02,
         8.24898455e-02,  2.46551972e-02, -4.59766665e-02,
         4.32965774e-02,  1.27668887e-01,  3.32715680e-02,
        -1.36283144e-02, -1.35737230e-01, -2.67064777e-02,
        -2.18643134e-03, -3.32139465e-03,  8.80042894e+01],
       [ 5.50524450e+04, -2.35260292e-01,  4.42944270e-02,
         6.07480303e-01,  1.33287488e-01, -2.40084184e-01,
         6.95914823e-02, -1.04032434e-01,  5.26513312e-02,
        -3.46705668e-02, -2.96920012e-02,  2.09561887e-01,
        -3.24573010e-02,  2.90269091e-02,  5.62860329e-02,
         1.82255736e-01, -1.18531561e-02,  4.38614336e-02,
        -7.08872202e-02, -2.11873157e-02,  3.95098096e-02,
        -3.72066889e-02, -1.09711595e-01, -2.85917491e-02,
         1.17114212e-02,  1.16645083e-01,  2.29500729e-02,
         1.87889841e-03,  2.85422324e-03,  8.86463767e+01]])
```

In [15]:
```
model.inertia_
```

Out[15]:
```
118320095020223.72
```

Country Data for k = 5

In [16]:
```
model = KMeans(n_clusters=5, n_init=4, verbose=1, max_iter=3)
model.fit(X1)
```

```
Initialization complete
Iteration 0, inertia 2889028093.155498
Iteration 1, inertia 2717269558.1921086
Iteration 2, inertia 2704110391.975314
Initialization complete
Iteration 0, inertia 3295333163.6207714
Iteration 1, inertia 2869522403.4372406
Iteration 2, inertia 2832973997.1933174
Initialization complete
Iteration 0, inertia 3025270588.6172094
Iteration 1, inertia 2909840001.0567784
Iteration 2, inertia 2874583597.3048234
Initialization complete
Iteration 0, inertia 3040410093.5436487
Iteration 1, inertia 2717410920.4940886
Iteration 2, inertia 2693101368.7125025
```

Out[16]:
```
KMeans(max_iter=3, n_clusters=5, n_init=4, verbose=1)
```

```
In [17]: model.cluster_centers_

Out[17]: array([[ 9.83651344e-02,  2.54103108e-01,  6.04003529e-02,
                 -1.70041712e-02,  1.27166399e-01, -6.45736185e-02,
                 -8.38463653e-02,  2.53674301e-02,  1.18147742e-02,
                  2.45559224e-02,  5.11664845e-03, -5.22905530e-03,
                  6.41475476e-03, -4.40675480e-03,  1.05158842e-02,
                  2.29428429e-02, -4.59008062e-03, -1.39598504e-02,
                  1.08309624e-02, -5.92953649e-02, -1.97182136e-02,
                  1.03818799e-03,  6.74124616e-03,  2.97953032e-03,
                  1.75356983e-03, -1.64555584e-03,  2.97617850e-03,
                 -1.46491495e-03,  3.16655816e+01,  1.59640849e-03],
                [-3.70843022e+00, -6.38774863e+00, -2.80870489e+00,
                  1.32446444e+00, -3.81013619e+00,  2.09938947e+00,
                  4.12031962e+00, -1.12419876e+00, -2.94638143e-01,
                 -8.68205805e-01,  4.89664347e-02, -9.68806826e-02,
                  9.78763645e-02,  3.40398805e-01,  9.45513707e-02,
                  1.77688719e-01,  4.07348875e-02,  1.55725960e-01,
                 -4.07262315e-01,  2.37250672e+00,  7.02706113e-01,
                 -5.78397900e-01, -7.62235684e-01,  9.56489667e-02,
                 -2.85532479e-01, -1.01650919e-02,  5.41269191e-02,
                  9.12386135e-02,  2.07124164e+03,  2.23048327e-03],
                [-3.10957233e-01, -1.11101288e+00, -1.20167167e-01,
                 -2.16317855e-02, -5.01885500e-01,  2.48264810e-01,
                  1.54256143e-01, -7.00612325e-02, -4.37929135e-02,
                 -7.79515240e-02, -4.54305461e-02,  4.89125795e-02,
                 -4.97971230e-02, -1.13612576e-02, -6.95372340e-02,
                 -1.79293164e-01,  2.27674197e-02,  1.01438459e-01,
                 -4.33594737e-02,  1.51677950e-01,  6.53152144e-02,
                  5.41138015e-02,  1.52359310e-02, -2.67435074e-02,
                  1.26740819e-02,  7.45773661e-03, -2.61530043e-02,
                  2.83325223e-03,  2.78443409e+02,  2.26011234e-03],
                [-1.00340340e+01, -1.52117704e+01, -7.44986489e+00,
                  3.91149207e+00, -1.17452096e+01,  6.79241210e+00,
                  1.31767681e+01, -2.84844360e+00, -6.84045575e-01,
                 -2.79192665e+00,  3.53040705e-01, -5.03014438e-01,
                  6.00109117e-01,  5.92747029e-01,  7.29773808e-01,
                  1.28400942e+00, -1.76304914e-01,  4.77845733e-01,
                 -1.16044822e+00,  5.33346697e+00,  1.35002813e+00,
                 -1.60514740e+00, -1.86909508e+00,  2.38337193e-01,
                 -7.61303979e-01, -2.71437588e-01,  5.89841604e-01,
                  1.10187178e-01,  5.56343753e+03,  2.16840434e-19],
                [-1.31802813e+00, -2.87535853e+00, -1.00018534e+00,
                  3.88334705e-01, -1.47011877e+00,  7.24064284e-01,
                  1.33647871e+00, -3.46292734e-01, -1.71545086e-01,
                 -3.33045022e-01, -2.07692763e-03,  3.52741020e-03,
                 -4.73464181e-02,  1.37751503e-01, -1.14694799e-01,
                 -1.08085396e-01,  6.51827229e-02,  1.77667089e-02,
```

Github link-

https://github.com/SethiGuneet/ML-Lab-Work/blob/main/Machine%20Learning%20Experiment%208.ipynb