Experiment 6

Aim:

A. To perform a simple k-mean clustering algorithm on any data set(at least 1500 instant and 6 classes).
B. To Implement k mean clustering algorithm in python.

Theory: K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition 'n' observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modelling. They both use cluster centres to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbour classifier, a popular machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbour classifier to the cluster centres obtained by k-means classifies new data into the existing clusters. To process the learning data, the K-means algorithm in data mining starts with the first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids It halts creating and optimizing clusters when either:

- The centroids have stabilized—there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

Algorithm:

```
A. Initialize k means with random values
B. For a given number of iterations:
    a. Iterate through items:
        i.   Find the mean closest to the item
        ii.  Assign item to mean
        iii. Update mean
```

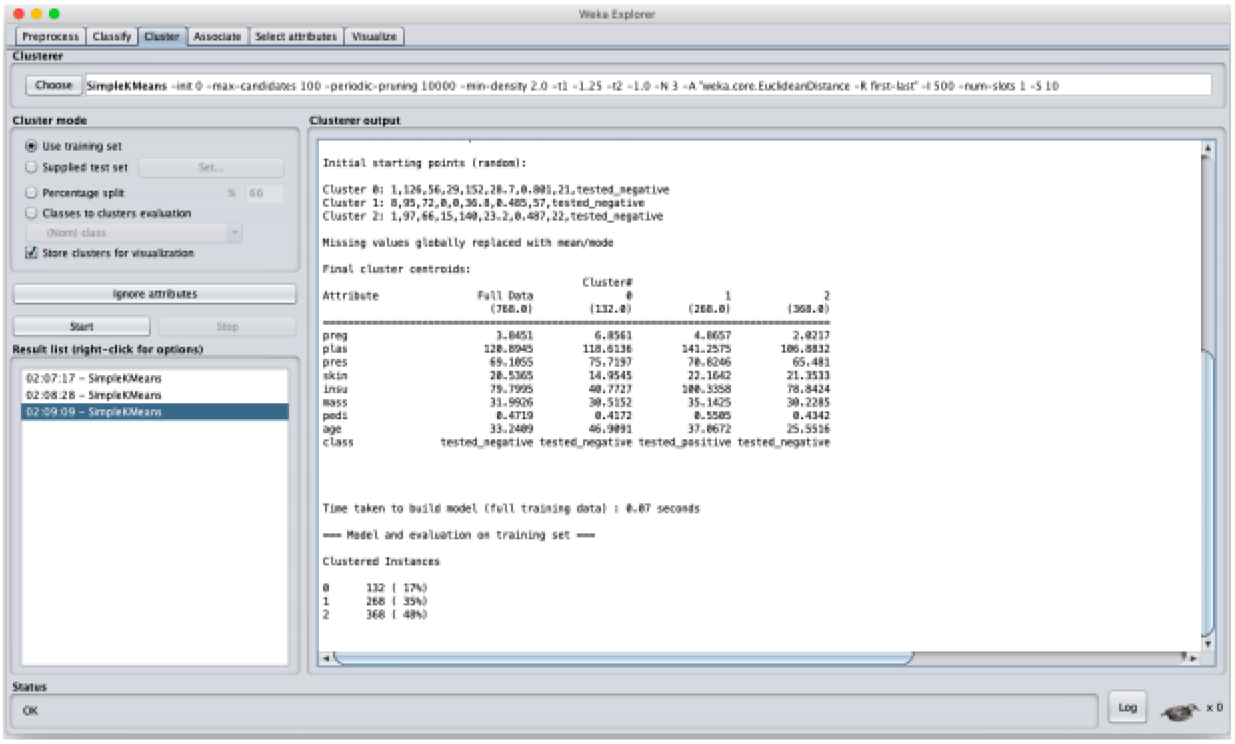
Part A

To perform a simple k-mean clustering algorithm on any data set.

Procedure:

1. Go to Weka Explorer.
2. Choose dataset in weka/data
3. Go to cluster tab
4. Choose cluster algorithm, in this case k-means
5. Click start.
6. Visualize the results

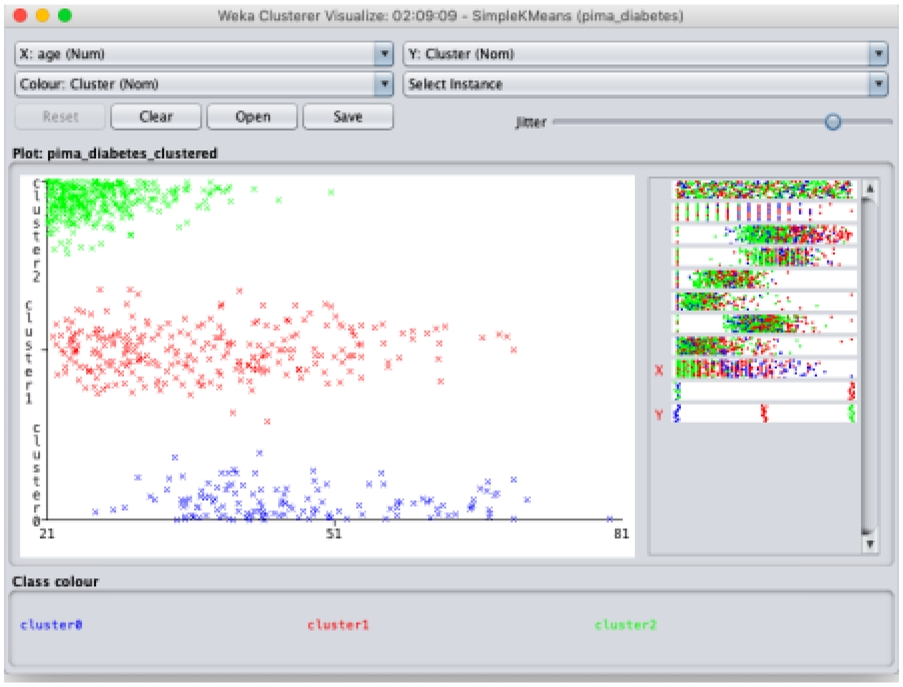
Output:



Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode: [✓ Use training set]; [✓ Store clusters for visualization]

```
Initial starting points (random):

Cluster 0: 1,126,56,29,152,28.7,0.801,21,tested_negative
Cluster 1: 8,95,72,0,0,36.8,0.485,57,tested_negative
Cluster 2: 1,97,66,15,140,23.2,0.487,22,tested_negative

Missing values globally replaced with mean/mode

Final cluster centroids:
                                         Cluster#
Attribute                 Full Data             0             1             2
                            (768.0)       (132.0)       (268.0)       (368.0)
=============================================================================
preg                         3.8451        6.8561        4.8657        2.0217
plas                       120.8945      118.6136      141.2575      106.0832
pres                        69.1055       75.7197       70.8246        65.481
skin                        20.5365       14.9545       22.1642       21.3533
insu                        79.7995       40.7727      100.3358       78.8424
mass                        31.9926       30.5152       35.1425       30.2285
pedi                         0.4719        0.4172        0.5505        0.4342
age                         33.2409       46.9091       37.0672       25.5516
class               tested_negative tested_negative tested_positive tested_negative




Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       132 ( 17%)
1       268 ( 35%)
2       368 ( 48%)
```



Findings and Learnings :

1. K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem.
2. Weka software makes the implementation and visualization of K-means algorithms very simple and efficient.
3. It makes K-means easily available to non-programmer users as well.

Part B

To Implement k mean clustering algorithm in python.

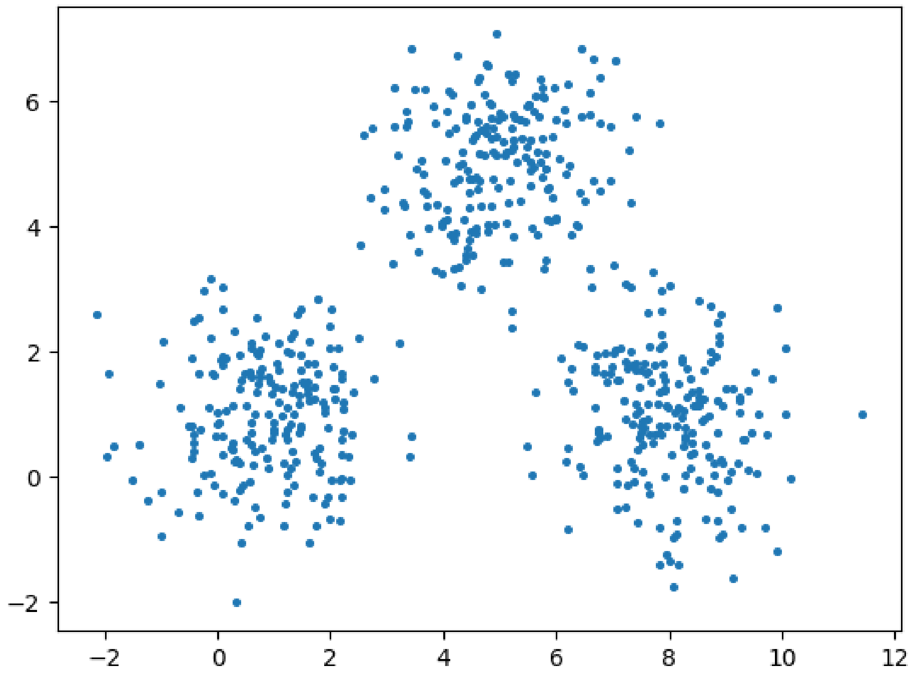
Source Code:

```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

center_1 = np.array([1,1])
center_2 = np.array([5,5])
center_3 = np.array([8,1])
# Generate random data and center it to the three centers
data_1 = np.random.randn(200, 2) + center_1
data_2 = np.random.randn(200,2) + center_2
data_3 = np.random.randn(200,2) + center_3
data = np.concatenate((data_1, data_2, data_3), axis = 0)
plt.scatter(data[:,0], data[:,1], s=7)
plt.show()
# Number of clusters
k = 3
# Number of training data
n = data.shape[0]
# Number of features in the data
c = data.shape[1]
# Generate random centers, here we use sigma and mean to ensure it represent the whole data
mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c)*std + mean
# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
plt.scatter(centers[:,0], centers[:,1], marker='*', c='g', s=150)
plt.show()
centers_old = np.zeros(centers.shape) # to store old centers
centers_new = deepcopy(centers) # Store new centers
print(data.shape)
clusters = np.zeros(n)
distances = np.zeros((n, k))
error = np.linalg.norm(centers_new - centers_old)
# When, after an update, the estimate of that center stays the same, exit loop
```

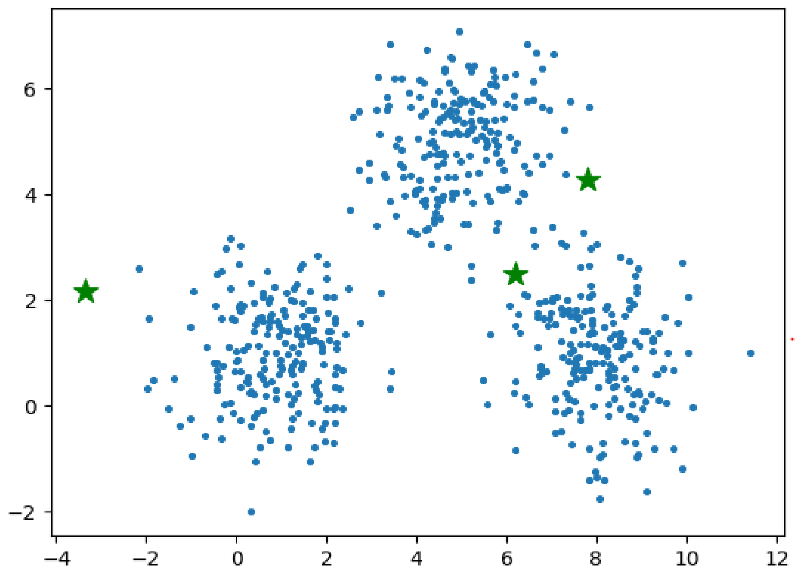
```
while error != 0:
    for i in range(k):
        distances[:, i] = np.linalg.norm(data - centers[i], axis=1)
     # Assign all training data to closest center
    clusters = np.argmin(distances, axis=1)
    centers_old = deepcopy(centers_new)
     # Calculate mean for every cluster and update the center
    for i in range(k):
        centers_new[i] = np.mean(data[clusters == i], axis=0)
    error = np.linalg.norm(centers_new - centers_old)
print(centers_new)
# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
plt.scatter(centers_new[:,0], centers_new[:,1], marker='*', c='g',s=150)
plt.show()
```

Output:

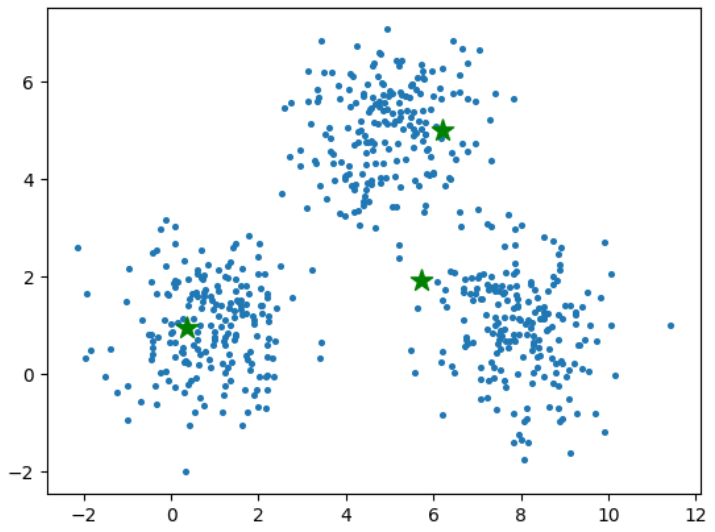
Sample data



Random centroid initialization



Final cluster assignment



Findings and Learnings :

1. K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem.
2. We have successfully implemented k-means clustering in Python