Aim: To implement k-means Algorithm on large dataset using Open source tool WEKA.

Objective: To make students well versed with open source tool like WEKA to implement k-means algorithm.

Theory:

- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.
- A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression.
- Cluster analysis is an important human activity. Cluster analysis has been widely used in numerous applications including market research, pattern recognition, data analysis and image processing.
- Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity.
- Clustering can also be used in outlier detection where outliers may be more interesting than common case.

WEKA contains "clusterers" for finding groups of similar instances in a dataset. The clustering schemes available in WEKA are k-means, Cobwebs, DBSCAN, OPTICS. Clusters can be visualized and compared to true clusters. Evaluation is based on log likelihood if clustering scheme produces a probability distribution.In 'preprocess' window click on 'open file...' button to select data file. Choosing Clustering scheme: In the 'clusterer' box click on 'choose' button. In pull-down menu select WEKA Clusteres, and select the cluster scheme 'simple K means'. Some implementations of K -means only allow numerical values for attributes ; therefore we do not need to use a filter.

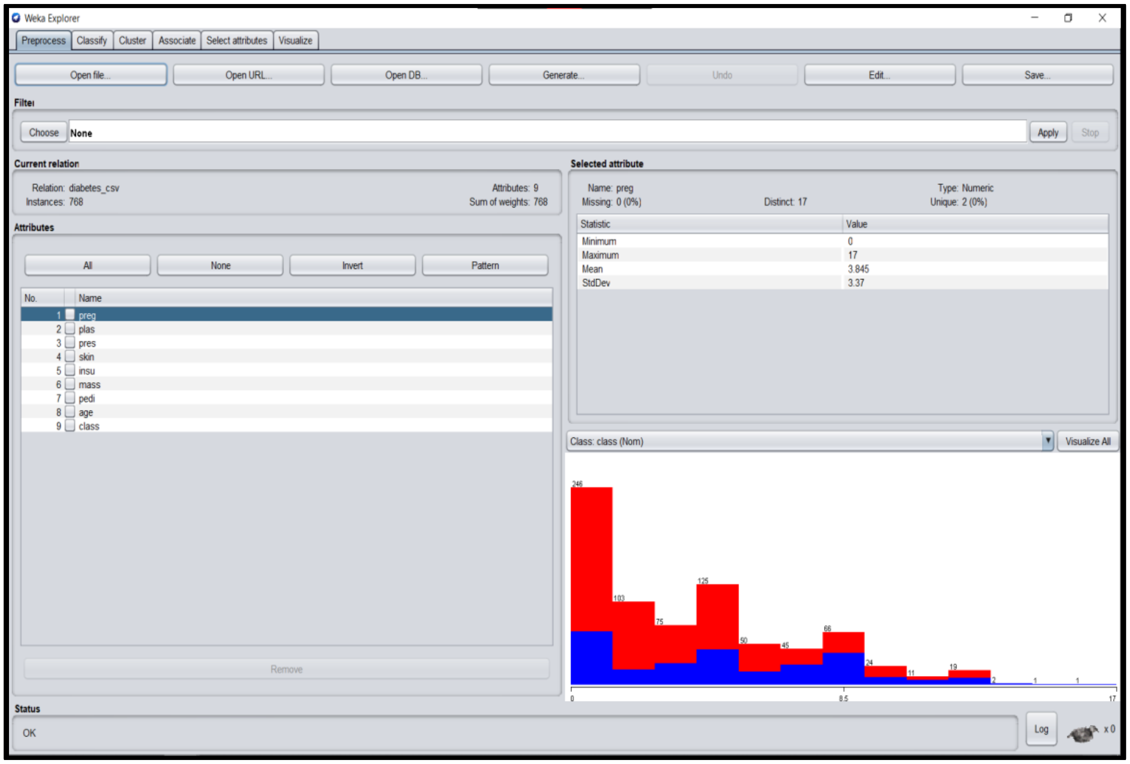
Once the clustering algorithm is chosen, right click on algorithm, 'weak.gui.GenericObjectEditor' comes up to the screen. Set the value in 'numclusters' box to number of clusters required. The seed value is used in generating a random number, which is used for making the initial assignments of instances to clusters. Before we run the clustering algorithm, we need to select 'cluster mode'. Click on 'Classes to cluster evaluation' radio-button in 'Cluster mode' box. Click the start button to run the program. When training set is complete, the 'Cluster' output area on the right panel of 'Cluster' window is filled with text describing the results of training and testing. A new entry appears in the 'Result list' box on the left of the result. Run information gives the information about : the clustering scheme used, the relation name, the number of instances, number of attributes. The clustering model shows the centroid of each cluster and statistics on the number and percentage of instances assigned to different clusters.

Cluster centroid is the mean vector of each cluster so each dimension value and centroid represents mean value for that dimension in the cluster. Thus centroids can be used to characterize the cluster.

Another way of representation of results of clustering is through visualization. Right click on the entry in the 'Result list' and select ' Visualize cluster assignments' in the pull-down window. This brings up Weka clusterer visualize window. This window displays clusters in different colors for better visibility.

Output:

Weka Tool



Filter: None

Current relation

Relation: diabetes_csv
Instances: 768
Attributes: 9
Sum of weights: 768

Attributes

No.		Name
1	□	preg
2	□	plas
3	□	pres
4	□	skin
5	□	insu
6	□	mass
7	□	pedi
8	□	age
9	□	class

Selected attribute

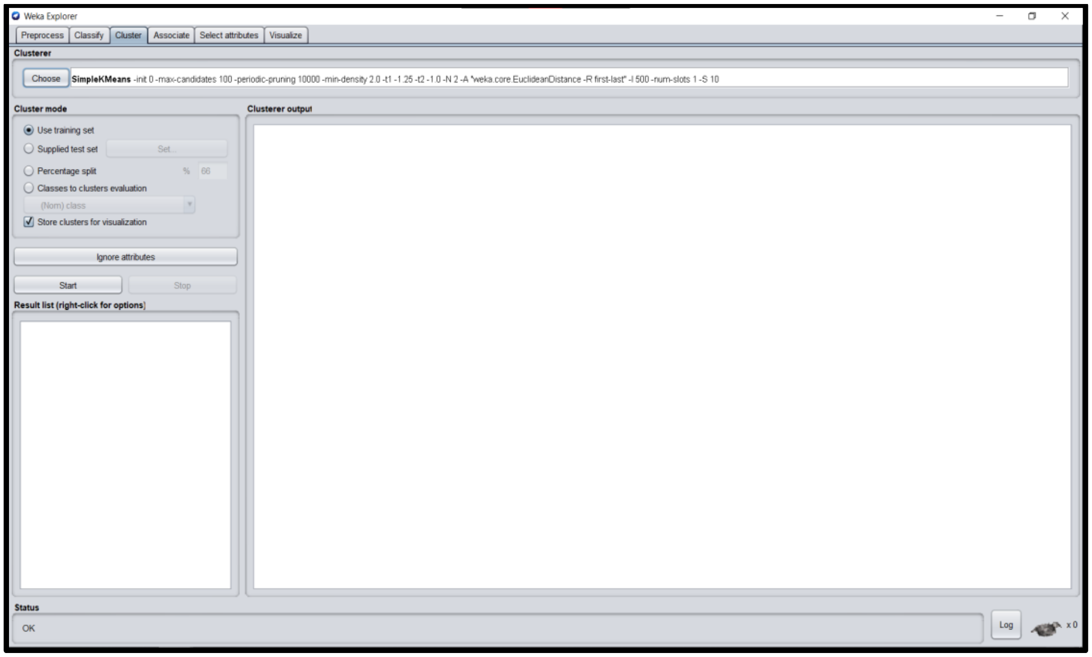
Name: preg
Missing: 0 (0%)
Distinct: 17
Type: Numeric
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	17
Mean	3.845
StdDev	3.37

Class: class (Nom)

Status

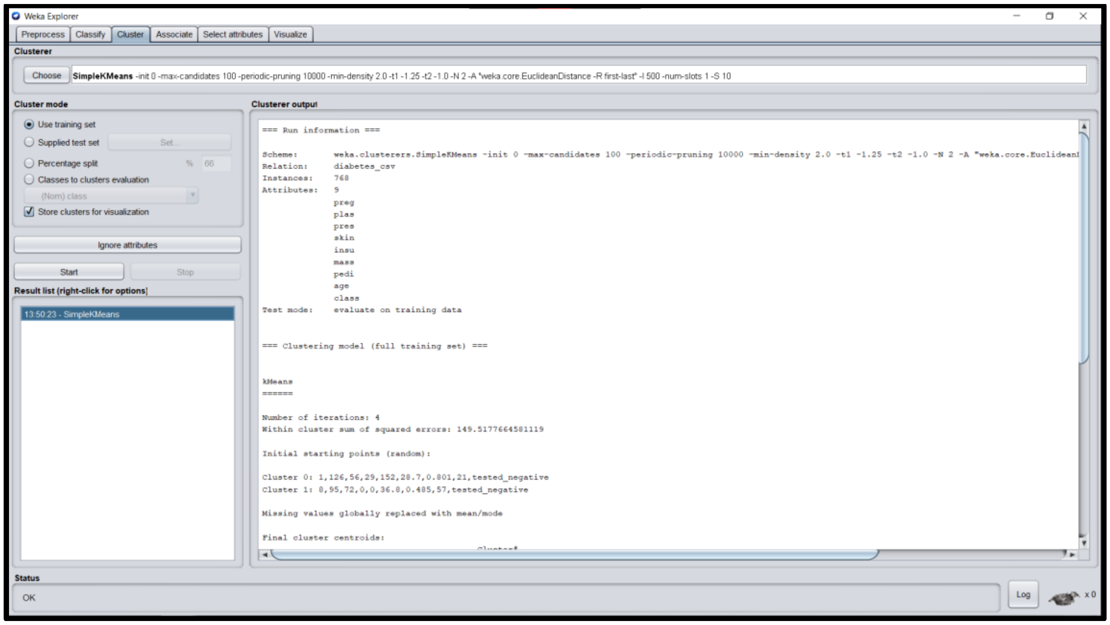
OK



SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

- ● Use training set
- ○ Supplied test set
- ○ Percentage split % 66
- ○ Classes to clusters evaluation (Nom) class
- ☑ Store clusters for visualization

Status: OK

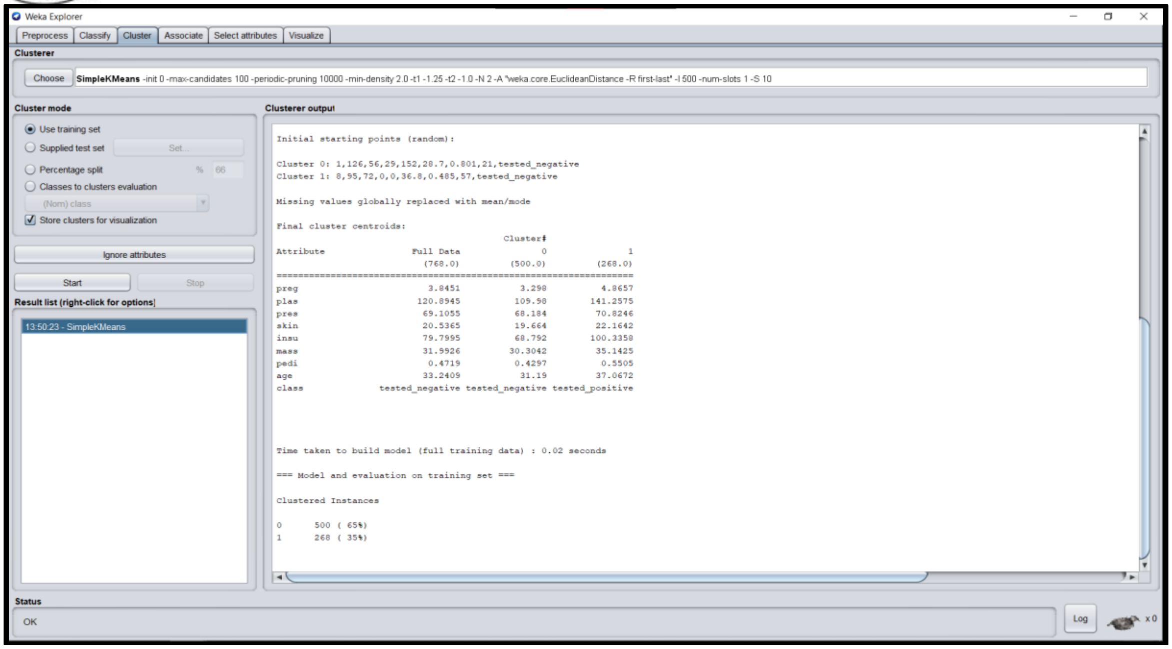


SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

- ● Use training set
- ○ Supplied test set
- ○ Percentage split % 66
- ○ Classes to clusters evaluation (Nom) class
- ☑ Store clusters for visualization

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanI
Relation:     diabetes_csv
Instances:    768
Attributes:   9
              preg
              plas
              pres
              skin
              insu
              mass
              pedi
              age
              class
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 149.5177664581119

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

Missing values globally replaced with mean/mode

Final cluster centroids:
                                   Cluster#
```

Status: OK



Clusterer

SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode

- ● Use training set
- ○ Supplied test set
- ○ Percentage split % 66
- ○ Classes to clusters evaluation (Nom) class
- ☑ Store clusters for visualization

Result list (right-click for options)

13:50:23 - SimpleKMeans

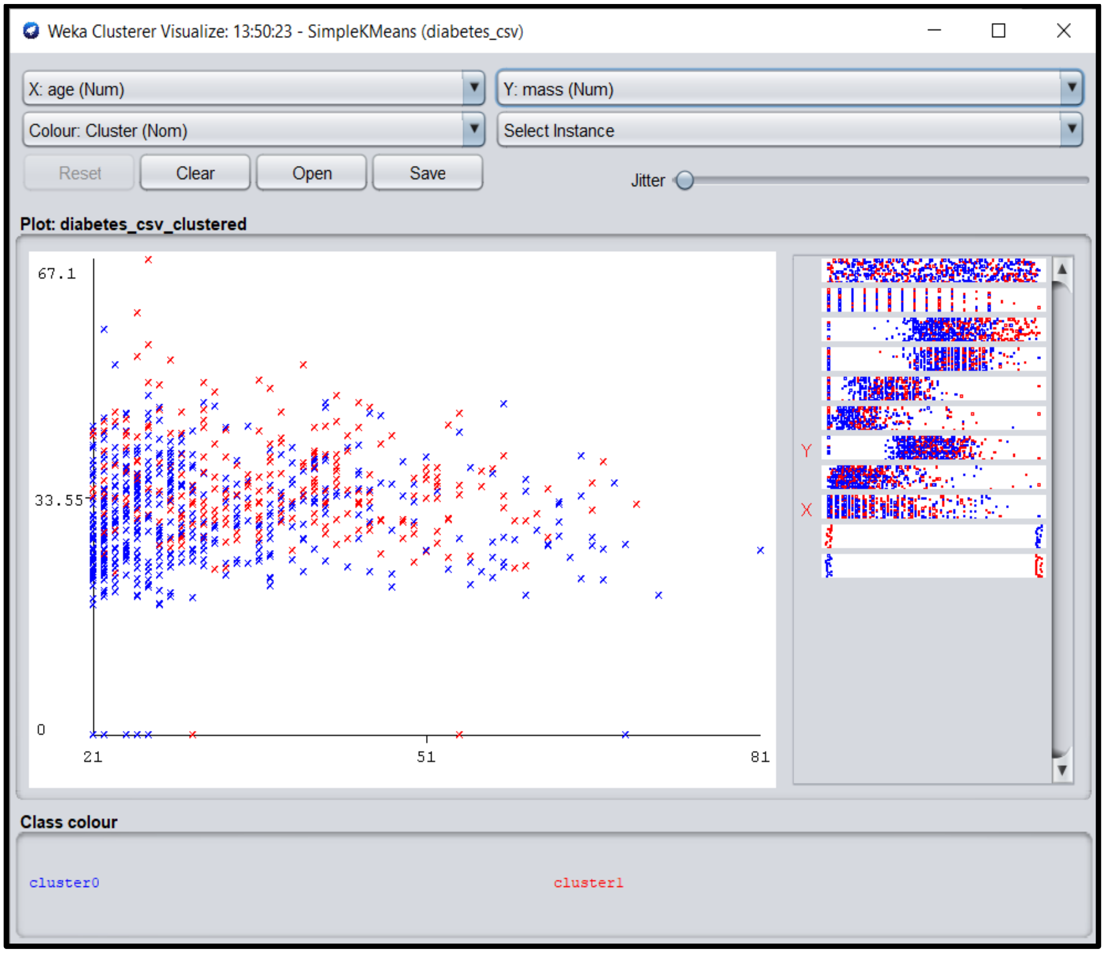
Clusterer output

```
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

Missing values globally replaced with mean/mode

Final cluster centroids:
                                    Cluster#
Attribute              Full Data           0                1
                         (768.0)     (500.0)          (268.0)
================================================================
preg                      3.8451       3.298           4.8657
plas                    120.8945      109.98         141.2575
pres                     69.1055      68.184          70.8246
skin                     20.5365      19.664          22.1642
insu                     79.7995      68.792         100.3358
mass                     31.9926     30.3042          35.1425
pedi                      0.4719      0.4297           0.5505
age                      33.2409       31.19          37.0672
class             tested_negative tested_negative tested_positive




Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0       500 ( 65%)
1       268 ( 35%)
```

Status

OK

Weka Output



Conclusion:

K -means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. A cluster refers to a collection of data points aggregated together because of certain similarities. The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

Hence we've successfully implemented K-means clustering through Python as well as Weka Tool