Lab Number: 9

Title
Perform K-means clustering with K = 2

Objective
To apply the K-Means clustering algorithm on a suitable dataset in WEKA, partition the data into 2 clusters, and analyze the resulting cluster.

IDE/Tools Used
Weka 3.8.6

Theory
Clustering: Clustering is an unsupervised machine learning technique that groups similar data points into clusters based on defined similarity measures, such as Euclidean distance. Data points within a cluster are more similar to each other than to those in other clusters, which helps in identifying patterns and structure in unlabeled data. It's a versatile tool with applications in fields like market segmentation, social network analysis, and data compression.

K-Means Algorithm: The k-means clustering algorithm is an unsupervised learning method used to partition data into a pre-defined number (k) of clusters. It works by iteratively assigning each data point to the nearest cluster centroid and then recalculating the position of the centroids based on the new cluster assignments until the centroids no longer change. This process groups similar data points together, minimizing the mean squared distance between each point and its assigned centroid

Steps:
- Initialize K centroids randomly, in this case k=2
- Assign each data point to the nearest centroid.
- Recalculate the centroid of each cluster.
- Repeat steps 2–3 until convergence (centroids no longer change significantly).
- K-Means is widely used due to its simplicity and efficiency but is sensitive to initial centroids and outliers.

Implementation

The following steps are performed to implement k-means clustering in WEKA.

1. Preparing the Dataset

Select or create a dataset suitable for clustering. Save it as .csv, then open it in WEKA =>Tools => ARFF Viewer and save as .arff. Also ensure attributes are numeric, as K-Means operates on continuous data.

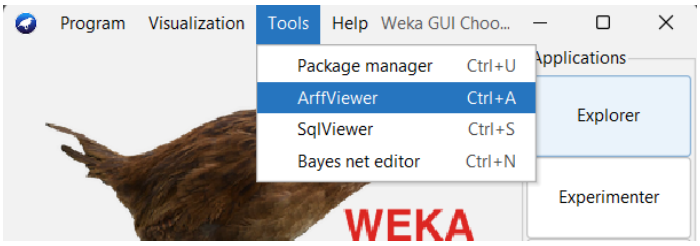


Figure 1: Opening the Arff Viewer

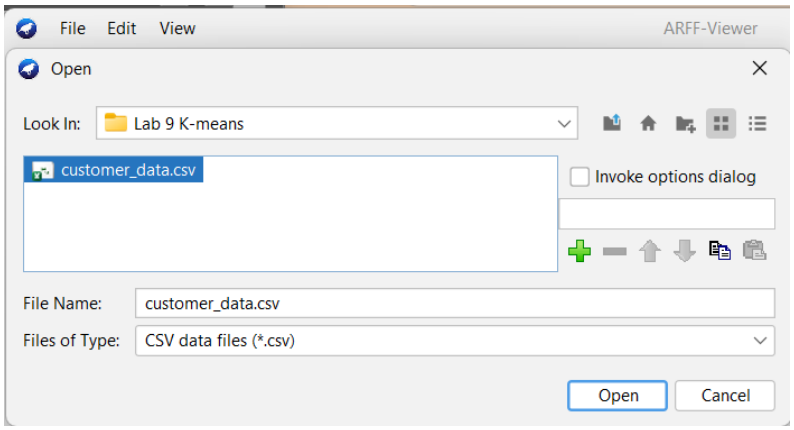
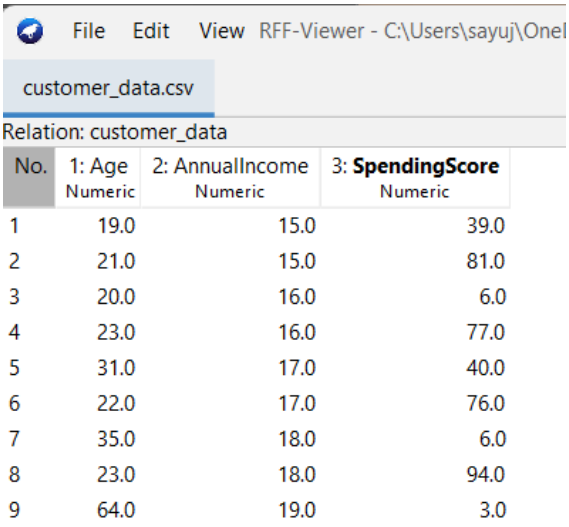


Figure 2: Selecting the csv file



No.	1: Age Numeric	2: AnnualIncome Numeric	3: **SpendingScore** Numeric
1	19.0	15.0	39.0
2	21.0	15.0	81.0
3	20.0	16.0	6.0
4	23.0	16.0	77.0
5	31.0	17.0	40.0
6	22.0	17.0	76.0
7	35.0	18.0	6.0
8	23.0	18.0	94.0
9	64.0	19.0	3.0

Figure 3: Visualization of the dataset

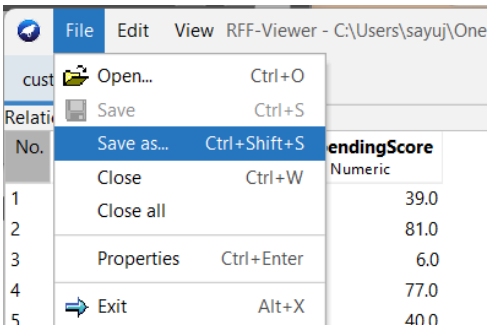


Figure 4: Option to save as arff

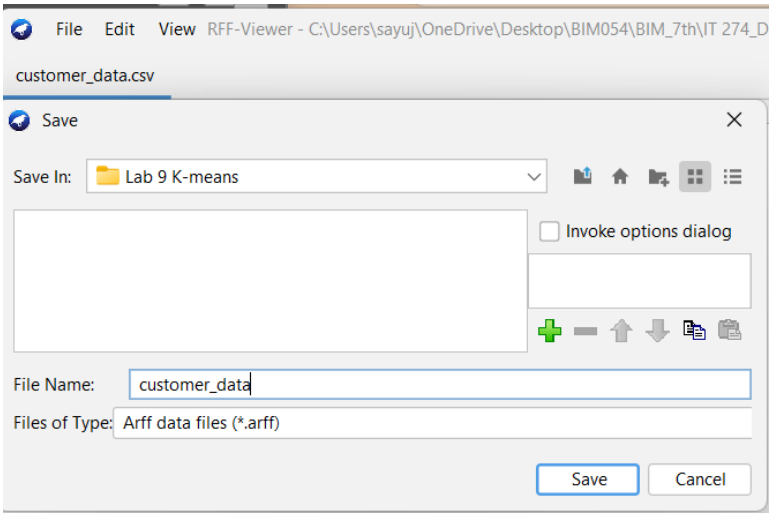


Figure 5: Saving as arff format

2. Loading the Dataset

2.1. Open WEKA Explorer => Preprocess => Open File

2.2. Load the prepared .arff file

2.3. Verify attribute types are numeric and clean any missing values if present

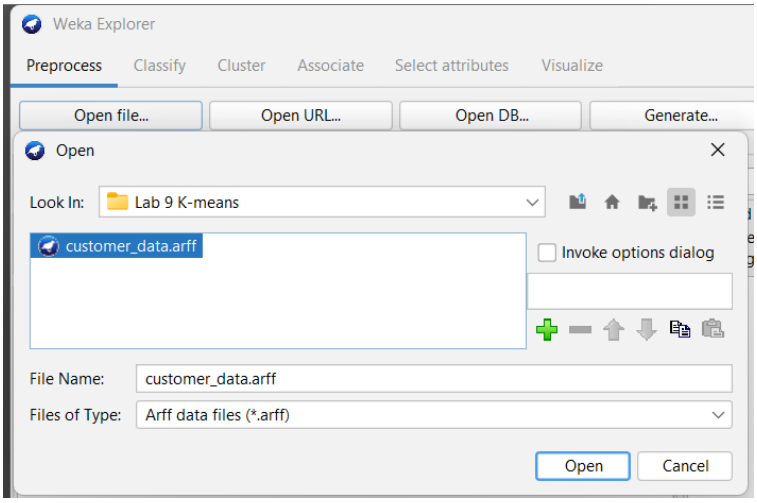
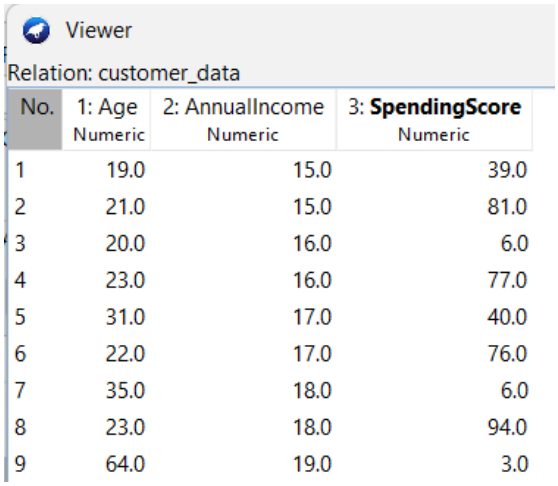


Figure 6: Opening the dataset in WEKA



No.	1: Age Numeric	2: AnnualIncome Numeric	3: **SpendingScore** Numeric
1	19.0	15.0	39.0
2	21.0	15.0	81.0
3	20.0	16.0	6.0
4	23.0	16.0	77.0
5	31.0	17.0	40.0
6	22.0	17.0	76.0
7	35.0	18.0	6.0
8	23.0	18.0	94.0
9	64.0	19.0	3.0

Figure 7: Verifying numeric data type

3. Running K-Means Clustering

3.1. Go to the Cluster tab in WEKA Explorer.

3.2. Select SimpleKMeans as the clustering algorithm.

3.3. Set Number of Clusters (K) = 2.

3.4. Adjust other parameters as needed:

- Distance function (default: Euclidean)
- Seed (for reproducibility)
- Maximum iterations (default 500)

Finally, Click Start to run K-Means clustering. The output will then show in the Clusterer output section in the left.

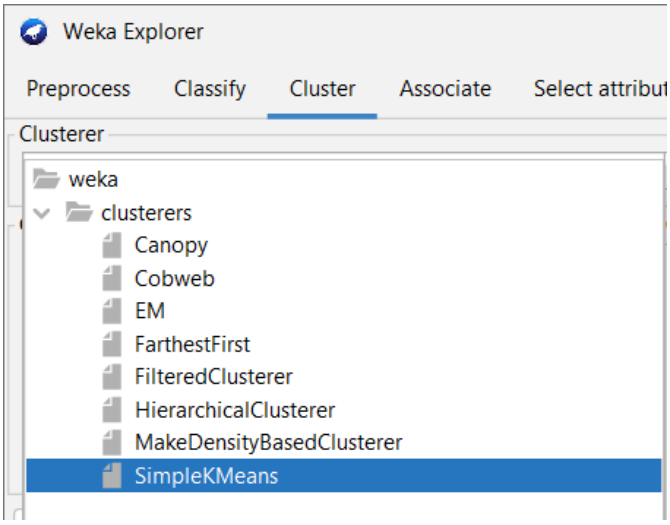


Figure 8: Choosing the SimpleKMeans clusterer

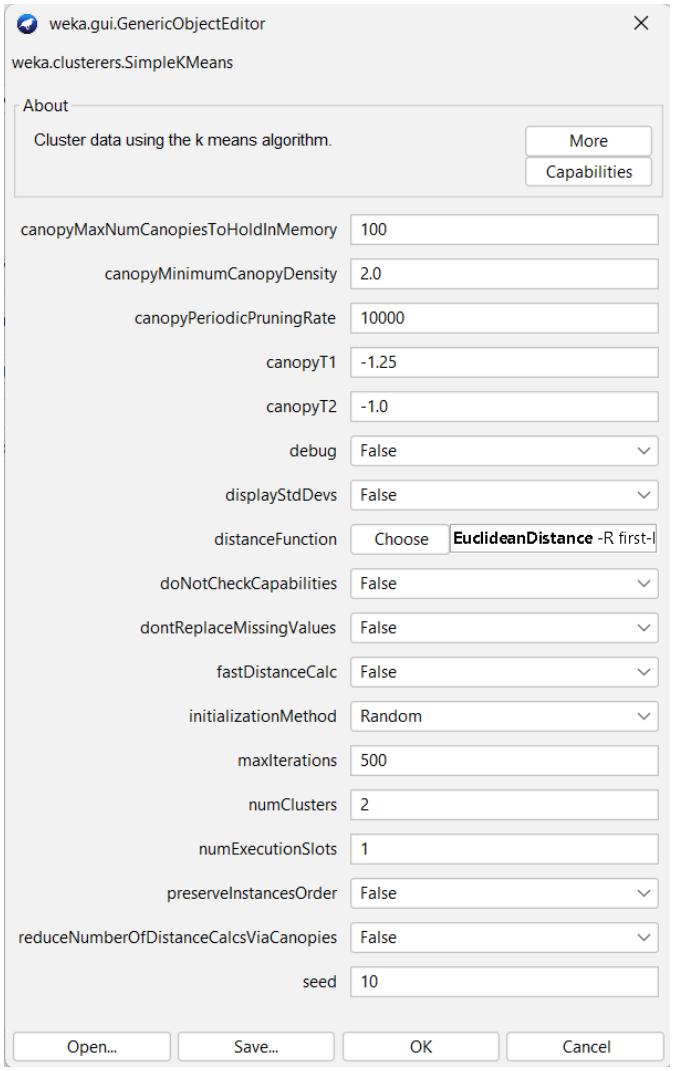


Figure 9: Adjusting the settings for the algorithm, Setting k = 2

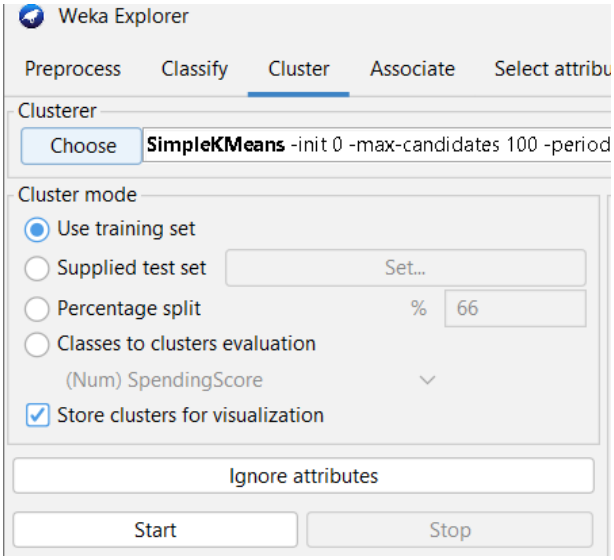


Figure 10: Starting the process

Output

```
=== 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.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation:     customer_data
Instances:    30
Attributes:   3
              Age
              AnnualIncome
              SpendingScore
Test mode:    evaluate on training data


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

kMeans
======
Number of iterations: 6
Within cluster sum of squared errors: 4.595352343661824

Initial starting points (random):

Cluster 0: 23,16,77
Cluster 1: 23,18,94

Missing values globally replaced with mean/mode

Final cluster centroids:
                            Cluster#
Attribute        Full Data          0          1
                    (30.0)     (13.0)     (17.0)
=============================================
Age                34.5333    45.2308    26.3529
AnnualIncome            22    22.9231    21.2941
SpendingScore         50.4    18.3077    74.9412


Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0      13 ( 43%)
1      17 ( 57%)
```

Discussion

The K-Means algorithm was applied to the customer_data dataset with K = 2, partitioning the 30 customers into two distinct clusters. The algorithm converged after 6 iterations, producing a within-cluster sum of squared errors (SSE) of 4.595, indicating that the data points are reasonably close to their respective cluster centroids.

The final cluster centroids reveal meaningful patterns:

- Cluster 0 (13 instances, 43%):
  - Average Age: 45.23 years
  - Average Annual Income: 22.92 k$
  - Average Spending Score: 18.31
- Cluster 1 (17 instances, 57%):
  - Average Age: 26.35 years
  - Average Annual Income: 21.29 k$
  - Average Spending Score: 74.94

From these centroids, we can interpret the clusters as two customer segments:

1. **Cluster 0:** Older, lower-spending customers: Customers in this cluster are older on average and exhibit lower spending scores, despite having slightly higher income than Cluster 1.
2. **Cluster 1:** Younger, higher-spending customers: Customers in this cluster are younger, with moderately lower income, but they have significantly higher spending scores, indicating more active engagement or purchasing behavior.

This segmentation illustrates K-Means' ability to uncover natural groupings in the data based on demographic and behavioral attributes. The results could be used to guide targeted marketing strategies, such as designing promotions for younger high-spending customers or loyalty programs for older, lower-spending customers.

Conclusion

Hence, in this lab, the K-Means algorithm was successfully implemented and it separated the dataset into two meaningful clusters, distinguishing older low-spending customers from younger high-spending customers. This demonstrates K-Means' effectiveness for customer segmentation and pattern discovery in numerical datasets.