K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent College of Somaiya Vidyavihar University)

Batch:- H2-1	**Roll No:-** 16010122151
Experiment No. 7	

Title: To implement clustering using K-means algorithm

AIM: To understand the Clustering algorithm.

Expected Outcome of Experiment:

CO4: Understand the basic concept and techniques of Machine Learning clustering.

Books/ Journals/ Websites referred:

1. https://uc-r.github.io/kmeans_clustering
2. https://en.wikipedia.org/wiki/K-means_clustering
3.
4.

Pre Lab/ Prior Concepts:

K-means Algorithm

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled 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, andfor K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discoverthe categories of groups in the unlabelled 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 pointand their corresponding clusters.

The algorithm takes the unlabelled dataset as input, divides the dataset into k-number ofclusters, 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 centre 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.

Description of the dataset used in implementation:

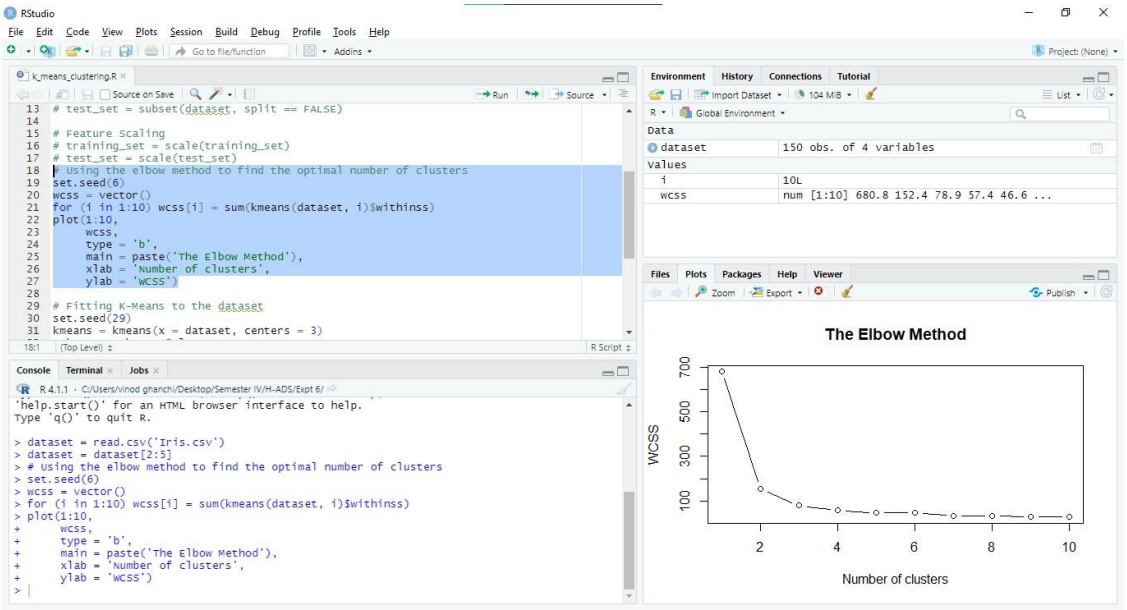
Iris dataset

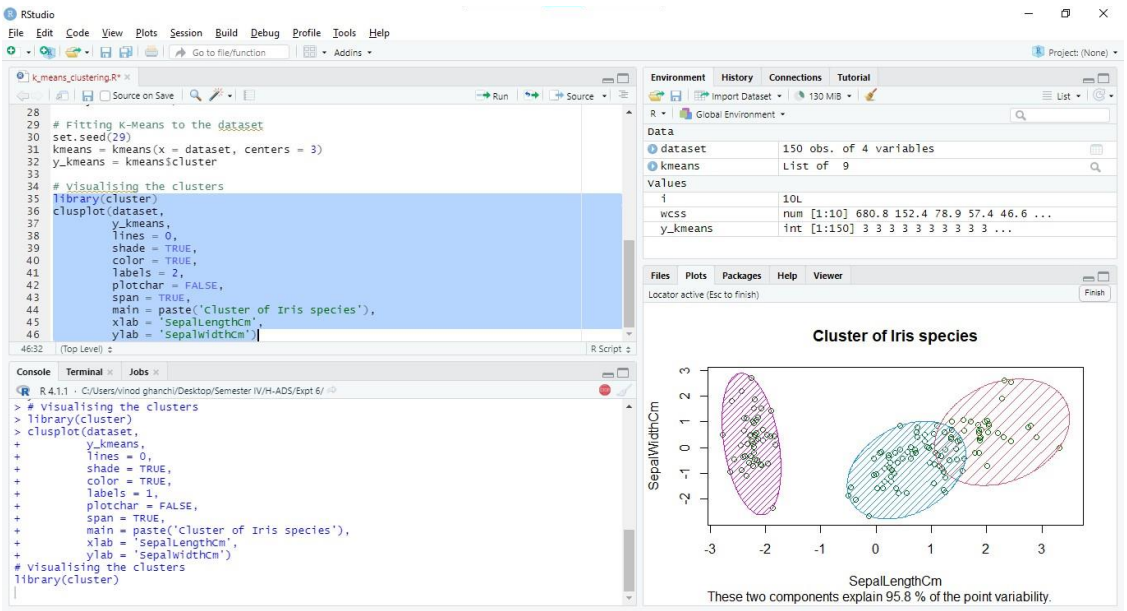
Code (R code):

```
# K-Means Clustering

# Importing the dataset
dataset = read.csv('Iris.csv')
dataset = dataset[2:5]

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
# library(caTools)
# set.seed(123)
# split = sample.split(dataset$DependentVariable, SplitRatio = 0.8)
# training_set = subset(dataset, split == TRUE)
# test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
 # test_set = scale(test_set)
# Using the elbow method to find the optimal number of clusters
 set.seed(6)
wcss = vector()
for (i in 1:10) wcss[i] = sum(kmeans(dataset, i)$withinss)
plot(1:10,
wcss,
 type = 'b',
main = paste('The Elbow Method'),
xlab = 'Number of clusters',
ylab = 'WCSS')


# Fitting K-Means to the dataset set.seed(29)
kmeans = kmeans(x = dataset, centers = 3)
y_kmeans = kmeans$cluster

# Visualising the clusters
library(cluster)
clusplot(dataset,
```

```
y_kmeans,
 lines = 0,
shade = TRUE,
color = TRUE,
 labels = 2,
plotchar = FALSE,
span = TRUE,
main = paste('Cluster of Iris species'),
 xlab = 'SepalLengthCm',
ylab = 'SepalWidthCm')
```

Output:





Conclusion: In the above experiment we have implemented k-means clustering. . To find the optimal number of clusters (k value) we have used the Elbow method.

Post Lab Questions

Q] Explain Apriori algorithm with suitable Numerical based example.

ANSWER:
Apriori Algorithm:
The Apriori algorithm is an association rule mining algorithm used to find frequent item sets and association rules. It operates in two phases:

- Generating frequent item sets: This phase involves finding item sets that have a support value (frequency of occurrence) greater than or equal to a specified threshold.

- Generating association rules: In this phase, association rules are generated from the frequent item sets found in the previous phase. An association rule typically has the form {A} -> {B}, meaning if A occurs, B is likely to occur.

EXAMPLE:

Transaction ID	Items
1	{milk, bread, eggs}
2	{bread, butter, eggs}
3	{milk, bread}
4	{bread, eggs}
5	{milk, eggs}

Transaction ID	Items
1	{1, 2, 3}
2	{1, 2, 4}
3	{1, 3, 4}
4	{2, 3, 4}
5	{1, 2}

6 {1, 3}

7 {2, 4}

8 {2, 3}

9 {3, 4}

Q] Write a program to implement the Apriori algorithm.
Program:

```
from itertools import combinations

def generate_candidates(prev_candidates, length):
    candidates = set()
    for candidate1 in prev_candidates:
        for candidate2 in prev_candidates:
            union_candidate = candidate1.union(candidate2)
            if len(union_candidate) == length:
                candidates.add(union_candidate)
    return candidates

def apriori(transactions, min_support):
    item_counts = {}
    for transaction in transactions:
        for item in transaction:
            if item in item_counts:
                item_counts[item] += 1
            else:
                item_counts[item] = 1

    frequent_item_sets = []
    n_transactions = len(transactions)

    for item, count in item_counts.items():
        if count >= min_support:
            frequent_item_sets.append({item})

    length = 2
    while True:
        candidates = generate_candidates(frequent_item_sets, length)
        frequent_candidates = []
        for candidate in candidates:
            count = 0
```

```
            for transaction in transactions:
                if candidate.issubset(transaction):
                    count += 1
            if count >= min_support:
                frequent_candidates.append(candidate)
        if not frequent_candidates:
            break
        frequent_item_sets.extend(frequent_candidates)
        length += 1

    return frequent_item_sets

transactions = [
    {1, 2, 3},
    {1, 2, 4},
    {1, 3, 4},
    {2, 3, 4},
    {1, 2},
    {1, 3},
    {2, 4},
    {2, 3},
    {3, 4}
]

min_support = 2

frequent_item_sets = apriori(transactions, min_support)
print("Frequent Item Sets:")
for item_set in frequent_item_sets:
    print(item_set)
```