Name: - Abhijit Saha *Batch: - B1*
SAP ID: - 60004200112 *Subject: - Data Mining & Warehousing*

Experiment – 5

Aim: - Implementation of Linear Regression for Single Variate and Multi-variate

Theory: -

Clustering: - Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group." It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns. It is an unsupervised learning method; hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

Types of Clustering: -
The clustering methods are broadly divided into Hard clustering (data point belongs to only one group) and Soft Clustering (data points can belong to another group also). But there are also other various approachestof Clustering exist. Below are the main clustering methods used in Machine learning:

1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

K-Means algorithm: - The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of O(n).

Implementation: -

```
# Connecting to drive
from google.colab import drive
drive.mount('/content/drive')

# Importing python packages or libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

# Pre-processing
df = pd.read_csv('/content/drive/MyDrive/cars.csv')
```

df.head()

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Car     36 non-null     object
 1   Model   36 non-null     object
 2   Volume  36 non-null     int64
 3   Weight  36 non-null     int64
 4   CO2     36 non-null     int64
dtypes: int64(3), object(2)
memory usage: 1.5+ KB
```

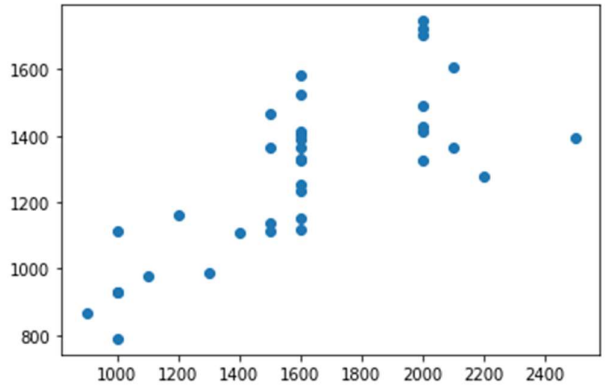
```
for i in df.columns:
  print(i)
```

```
Car
Model
Volume
Weight
CO2
```

Part A: Simple KMeans Clustering

```
# Using two features for clustering
X = pd.DataFrame(df, columns = ['Volume', 'Weight'])

# Plotting a scatter plot
plt.scatter(df['Volume'], df['Weight'])
plt.show()
```




```
# Forming clusters using KMeans
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
```

```
KMeans(n_clusters=3)
```

```
clusters = kmeans.fit_predict(X)
clusters
```

```
array([1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 2, 0, 2, 0, 2,
       2, 0, 2, 0, 2, 2, 0, 2, 2, 2, 0, 0, 0, 2], dtype=int32)
```

```
kmeans.labels_
```

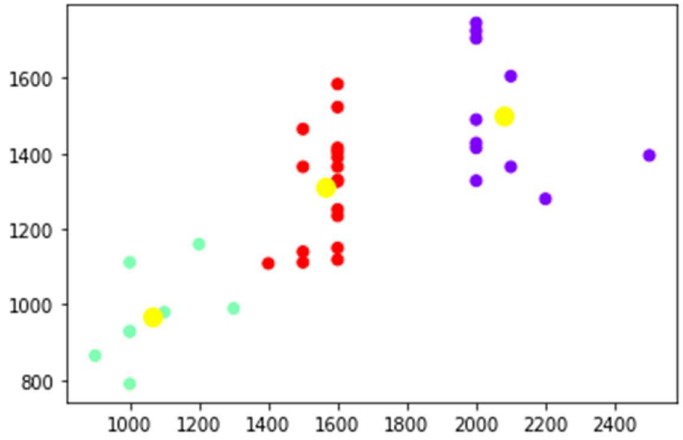
```
array([1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 2, 0, 2, 0, 2,
       2, 0, 2, 0, 2, 2, 0, 2, 2, 2, 0, 0, 0, 2], dtype=int32)
```

kmeans.cluster_centers_

```
array([[1564.70588235, 1310.88235294],
       [1062.5       ,  969.375     ],
       [2081.81818182, 1498.36363636]])
```

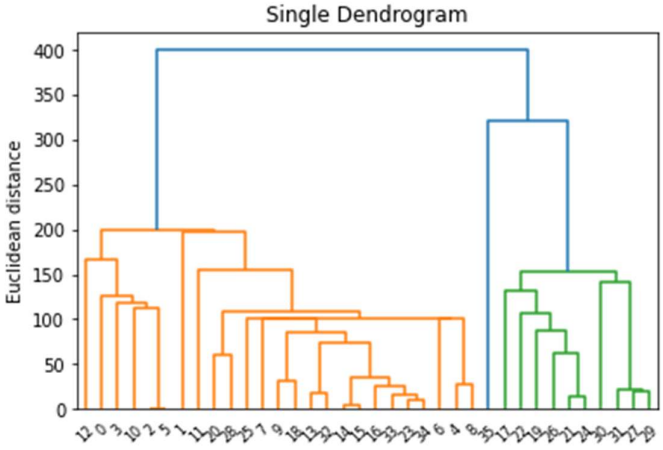
 # Plotting scatter plot after clustering

```
data_with_clusters = df.copy()
data_with_clusters['Clusters'] = clusters
plt.scatter(data_with_clusters['Volume'], data_with_clusters['Weight'],
c=data_with_clusters['Clusters'], cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, color
= 'yellow')
```

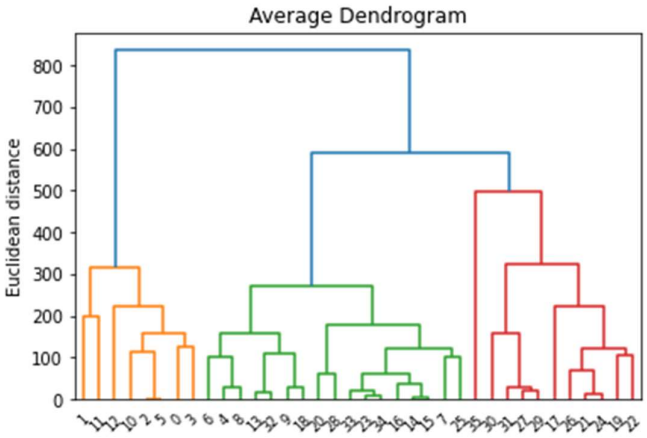


Part B: Hierarchical Clustering

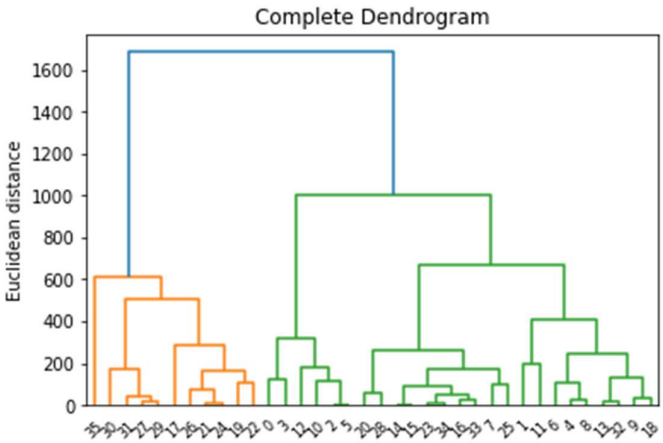
# *Simple Dendrogram*
Z = linkage(X, method = 'single')
dendro = dendrogram(Z)
plt.title('Single Dendrogram')
plt.ylabel('Euclidean distance')
plt.show()



# *Average Dendrogram*
Z = linkage(X, method = 'average')
dendro = dendrogram(Z)
plt.title('Average Dendrogram')
plt.ylabel('Euclidean distance')
plt.show()



```
# Complete Dendrogram
Z = linkage(X, method = 'complete')
dendro = dendrogram(Z)
plt.title('Complete Dendrogram')
plt.ylabel('Euclidean distance')
plt.show()
```

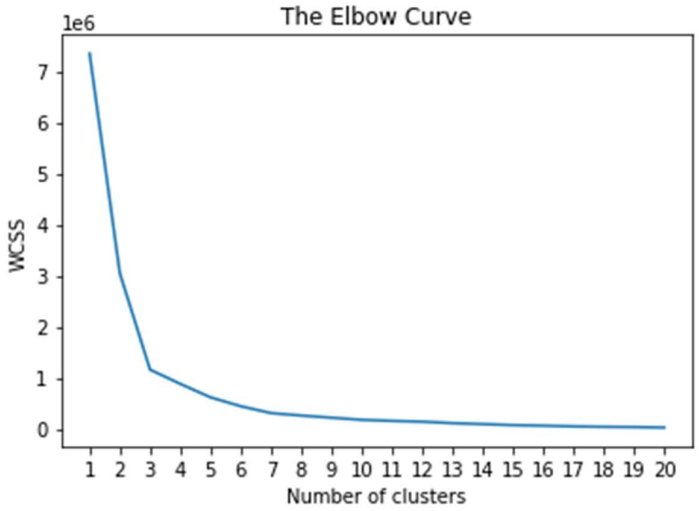


Part C: Elbow method

```
# Finding the optimum no. of clusters using Elbow method
wcss = []
for i in range(1, 21):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10,
random_state = 1274)
  kmeans.fit(X)
  wcss.append(kmeans.inertia_)
  print(f'Cluster {i} Inertia {kmeans.inertia_}')

plt.plot(range(1, 21), wcss)
plt.title('The Elbow Curve')
plt.xlabel('Number of clusters')
plt.xticks(list(range(1, 21)))
plt.ylabel('WCSS')
plt.show()
```

```
Cluster 1 Inertia 7347394.777777779
Cluster 2 Inertia 3045987.4690909097
Cluster 3 Inertia 1158577.3509358289
Cluster 4 Inertia 880446.2282467533
Cluster 5 Inertia 616493.0821428571
Cluster 6 Inertia 442824.12976190477
Cluster 7 Inertia 308300.51666666666
Cluster 8 Inertia 260583.2916666667
Cluster 9 Inertia 218225.3357142857
Cluster 10 Inertia 176592.00238095236
Cluster 11 Inertia 155727.98571428572
Cluster 12 Inertia 140200.38333333333
Cluster 13 Inertia 111245.25238095238
Cluster 14 Inertia 92605.38333333333
Cluster 15 Inertia 72837.75
Cluster 16 Inertia 62364.3
Cluster 17 Inertia 47878.66666666667
Cluster 18 Inertia 40066.16666666667
Cluster 19 Inertia 35223.5
Cluster 20 Inertia 25712.0
```



Conclusion: -

Implemented KMeans Clustering Algorithm for clustering of variables in a dataset and found the three dendrograms for the hierarchical clustering as well as found the optimum clusters for the dataset through the elbow method.