ASSIGNMENT NO. 3

AIM: Assignment on K-Means Clustering using the CarDekho Dataset.

OBJECTIVES:

- To understand the concept of K-Means clustering and its working principles.
- To implement the K-Means clustering algorithm using Python.
- To analyze the impact of different initialization methods on clustering results.
- To compare K-Means clustering with Hierarchical clustering in terms of scalability and efficiency.

THEORY: K-Means is one of the simplest and most widely used unsupervised learning algorithms. It is primarily used for clustering data into a predefined number of K clusters. The algorithm works by defining K centroids, each representing a cluster. The final clusters depend on the initial centroid configuration, so centroids should be initialized as far apart as possible.

K-Means is iterative and easy to implement, making it a popular choice for clustering tasks.

Algorithm Steps: Given N data points, the algorithm follows these steps:

1. **Initialization:** Choose K initial centroids randomly. (Default K-Means++)
2. **Assignment:** Assign each data point to the nearest centroid, forming K clusters.
3. **Recalculation:** Compute new centroids by averaging the data points in each cluster.
4. **Iteration:** Repeat steps 2 and 3 until centroids no longer change significantly.

The objective function of K-Means aims to minimize the sum of Euclidean distances between data points and their respective cluster centroids.

Dataset: The dataset used for this assignment is **CarDekho.csv**, which contains various features related to used car sales. The dataset includes the following attributes:

- **Vehicle Age:** Age of the car in years.
- **Km Driven:** Distance covered by the car.
- **Selling Price:** The price at which the car was sold (target for clustering).
- **Fuel Type:** Type of fuel used (Petrol/Diesel/CNG/Electric).
- **Transmission Type:** Type of transmission (Manual/Automatic).
- **Engine Capacity:** Engine size in CC.
- **Max Power:** Maximum power output of the car.

Centroid Initialization Methods:

- **Forgy:** Randomly select K points as centroids.
- **Random Partition:** Assign each point to a random cluster before computing centroids.
- **K-Means++:** Optimized initialization for better cluster stability.
- **Canopy Clustering:** Pre-clustering method used for large datasets to improve efficiency.

Determining the Value of K:

- **Elbow Method:** Perform clustering for different K values and calculate the Sum of Squared Errors (SSE). Plot SSE against K and choose the elbow point where SSE stops decreasing significantly.

K-Means vs. Hierarchical Clustering:

1. **Scalability:** K-Means is better suited for large datasets due to its linear time complexity, whereas hierarchical clustering has a quadratic time complexity.
2. **Reproducibility:** Hierarchical clustering provides consistent results, while K-Means depends on initial centroid selection.
3. **Predefined Clusters:** K-Means requires specifying K in advance, whereas hierarchical clustering allows selecting clusters by analyzing a dendrogram.

Key Considerations for Clustering:

- K-Means is sensitive to scale since it relies on Euclidean distance, so data should be normalized if scaling issues exist.
- If the data has inherent symmetries, some clusters may be misclassified.
- Running K-Means multiple times with different initial centroids and choosing the most common result can improve reliability.

Working:

Step 1: Import Necessary Libraries

- Import essential libraries:
    - pandas, numpy: for data handling
    - matplotlib.pyplot, seaborn: for visualizations
    - sklearn.preprocessing.StandardScaler: for normalization
    - sklearn.cluster.KMeans: for clustering

Step 2: Load the CarDekho Dataset

- Load the dataset using pd.read_csv().
- Explore basic structure with .head(), .info().

Step 3: Select Important Numerical Features for Clustering

- Choose relevant numeric columns for clustering such as:
    - vehicle_age, km_driven, mileage, engine, max_power, selling_price
- Avoid non-numeric or ID-based columns (e.g. car_name, brand, model)

Step 4: Handle Missing Values

- Fill missing values:
    - Use median for numeric columns like mileage, engine, max_power.

Step 5: Normalize the Features (Very Important)

- Apply StandardScaler to standardize the numeric features:
    - Mean = 0, Standard Deviation = 1
    - Prevents scale-dominant features (like km_driven) from skewing cluster results

Step 6: Apply K-Means Clustering

- Use KMeans algorithm from sklearn.cluster.
- Select a suitable number of clusters (e.g., k = 3 or 4).
- Fit the model on the normalized data.

Step 7: Add Cluster Labels to Original DataFrame

- Add the resulting cluster labels as a new column in the original DataFrame.
- Enables deeper inspection of cluster characteristics.

Step 8: Visualize the Clusters

- Use 2D scatter plots to visualize clusters:
    - e.g., selling_price vs vehicle_age, color-coded by cluster
- Use PCA or t-SNE if more dimensions need to be visualized in 2D.

Step 9: Final Output

- Provide:
    - Cluster summary: average selling_price, mileage, etc., per cluster
    - Interpretation: What each cluster likely represents (e.g., old low-priced cars, new premium models)
    -

CONCLUSION: K-Means is an efficient and widely used clustering technique that effectively groups cars into distinct clusters based on their attributes. However, its performance depends on proper initialization, scaling, and selecting the optimal K value. Despite its limitations, K-Means remains a fundamental tool in unsupervised learning applications, helping to segment vehicles based on price, age, and other factors.