**Assignment No. 5**

**Problem Statement:** Implement the K-Means clustering algorithm using Python to analyze and visualize a given dataset.

**Objective:**

1. Understand and implement the K-Means clustering algorithm.
2. Apply K-Means to a dataset and analyze the results.
3. Use the Elbow Method to determine the optimal number of clusters.
4. Visualize clusters and interpret results.

**Prerequisite :**

1. A Python environment with essential libraries like pandas, numpy, matplotlib, seaborn, and scikit-learn.
2. Basic knowledge of Python, statistics, and machine learning principles.
3. Statistics: Understanding of mean, variance, and standard deviation.
4. Machine Learning: Basics of unsupervised learning, clustering, and K-Means.

**Theory :**

**K-Means Clustering is** an **unsupervised machine learning algorithm** usedfor **partitioning a dataset** into **K distinct, non-overlapping clusters.** Itaimsto **group similar data points together** while ensuring that different clusters are as distinct as possible.

### ****Key Properties of K-Means:****

1. **Unsupervised learning**: No labeled data is required.
2. **Centroid-based**: Each cluster is represented by its centroid (mean of the points).
3. **Iterative algorithm**: Repeatedly assigns data points and updates centroids.
4. **Works well with large datasets**: Faster than hierarchical clustering.

# **Working of K-Means Algorithm**

### ****Step 1: Choose K (Number of Clusters)****

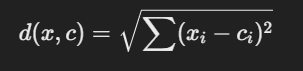
* Decide the number of clusters KKK manually or using methods like the **Elbow Method** or **Silhouette Score**.

### ****Step 2: Initialize Centroids****

* Randomly select **K data points** as the initial **cluster centroids**.

### ****Step 3: Assign Each Data Point to the Nearest Centroid****

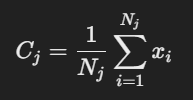
* Compute the **Euclidean distance** between each data point and the centroids:



* + Assign each data point to the **closest centroid**.
  + This step **forms K clusters**.

### ****Step 4: Compute New Centroids****

* For each cluster, calculate the **new centroid** by taking the **mean of all points** assigned to it:



where:

* + Cj​ = New centroid of cluster j
  + Nj​ = Number of points in cluster j
  + xi​ = Data points in cluster j

### ****Step 5: Repeat Until Convergence****

* **Reassign points** to the new centroids.
* **Recalculate centroids**.
* Repeat **until centroids no longer change** (or changes are minimal).

# **3. Understanding WCSS (Within-Cluster Sum of Squares)**

The **WCSS (Within-Cluster Sum of Squares)** measures **how well data points fit within a cluster**. It is used in the **Elbow Method** to find the optimal number of clusters.

### ****Formula for WCSS:****

### 

where:

* K= Number of clusters,
* Nj​ = Number of points in cluster j
* xi​ = Data points in cluster j
* Cj​ = Centroid of cluster

### ****Interpreting WCSS****

* **Lower WCSS** = Clusters are well-defined and compact.
* **Higher WCSS** = Clusters are too spread out (not well-defined).

# **4. The Elbow Method to Find Optimal K**

The **Elbow Method** helps determine the **best value of K** by plotting **WCSS vs. K**.

### ****How to Use the Elbow Method?****

1. Compute **WCSS for different values of K**.
2. Plot **WCSS vs. K**.
3. Find the **"elbow point"** where WCSS stops decreasing sharply.
4. The corresponding **K is optimal**.

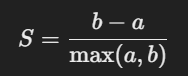
### ****Why Does the Elbow Method Work?****

* Adding more clusters **always decreases WCSS**.
* But after a certain **K**, **adding more clusters has little effect**.
* The **"elbow"** is where **WCSS reduction slows down significantly**.

# **5. Evaluation Metrics for K-Means**

### ****Silhouette Score****

Measures how well a point fits within its cluster:



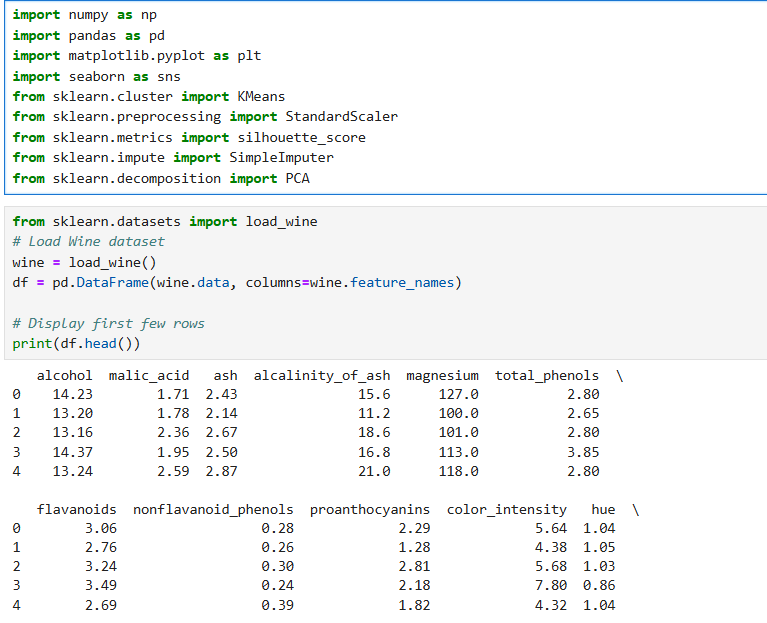
where:

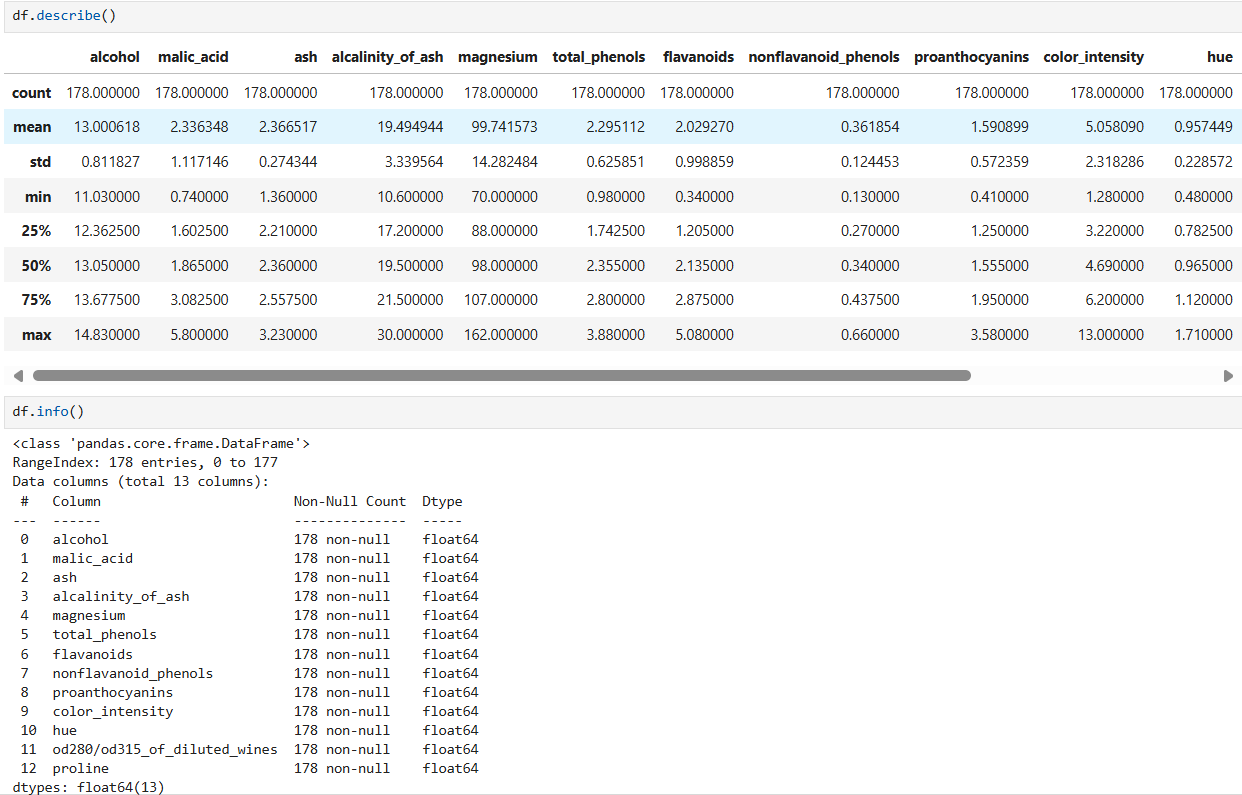
* a = Average distance to other points in the same cluster.
* b = Average distance to points in the nearest cluster.

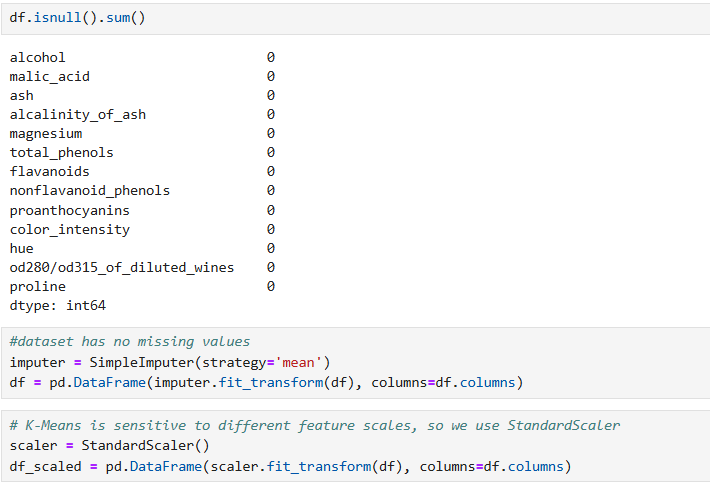
**Interpretation:**

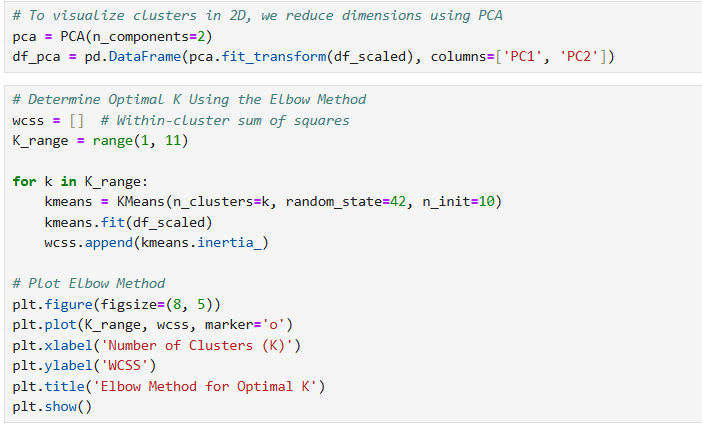
* **S ≈ 1:** Well-clustered
* **S ≈ 0:** Overlapping clusters
* **S ≈ -1:** Wrong clustering

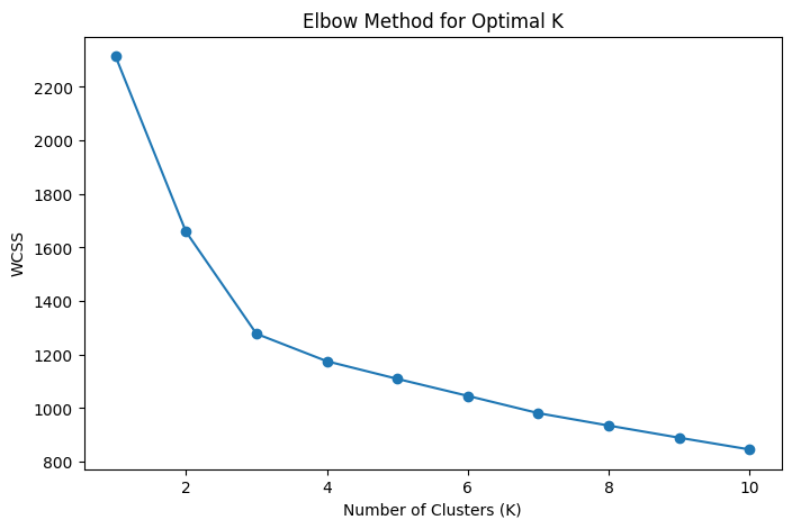
**Code & Output**

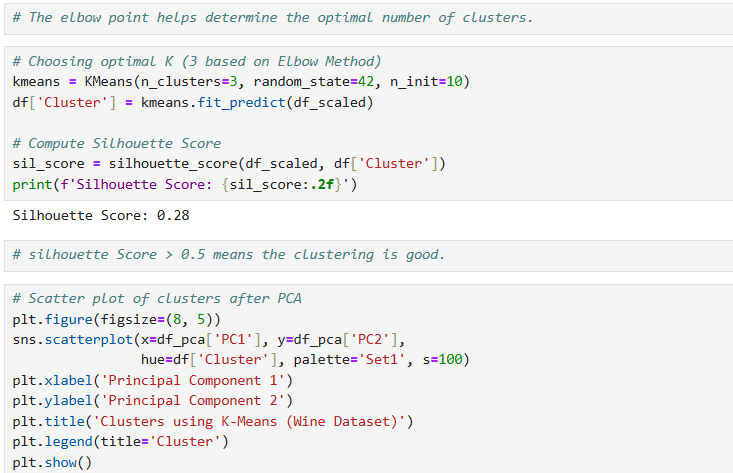


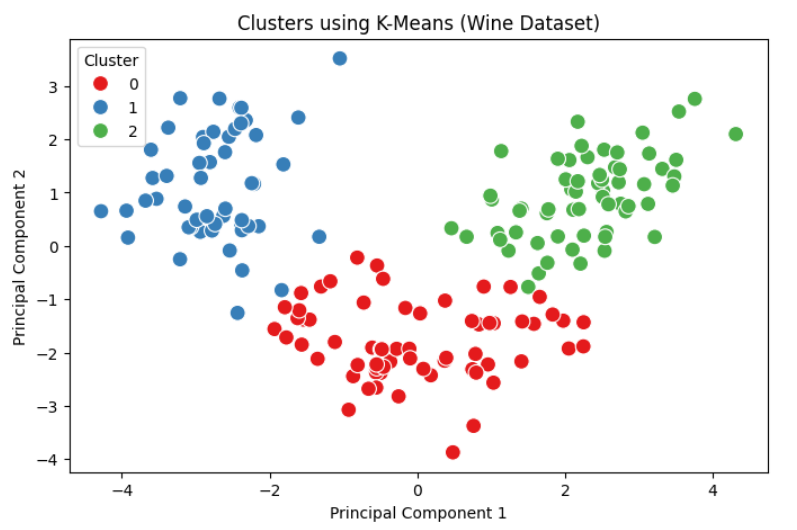




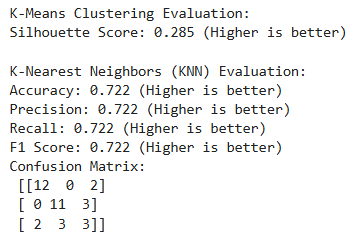












## Github: <https://github.com/dnyaneshwardhere/ML>

## **Conclusion:**

In this assignment, we implemented **K-Means clustering** and **KNN classification** on the **Wine dataset**. The **K-Means model** resulted in a **Silhouette Score of 0.285**, indicating weak cluster separation. The **KNN classifier achieved 72.2% accuracy**, with balanced precision, recall, and F1-score. However, the confusion matrix showed misclassifications, particularly in class 2.