A Report On: -

Clustering Analysis of Wine Dataset Using K-Means

Submitted By: -

Rajan Ghimire
C0924991

Submitted To: -

Victoria Shtern

Contents

Abstract

This project aims to analyze the Wine dataset from the UCI Machine Learning Repository using the K-Means clustering algorithm. The results are visualized in both **2D** and **3D** plots to provide clear insights into the clustering patterns.

Introduction:

Clustering is a fundamental technique in machine learning used to group similar data points together based on their features. This project focuses on the Wine dataset, which contains chemical analysis results of wines grown in the same region in Italy. By applying the K-Means clustering algorithm, we aim to uncover natural groupings within the data.

Objectives:

- The project's goal is to identify the optimal number of clusters and evaluate the clustering performance through visualization.

About the Dataset:

The dataset used in this study is the Wine Quality Dataset from the UCI Machine Learning Repository. It contains 1,143 samples of red "Vinho Verde" wine, each described by 11 physicochemical properties and a quality rating. The features and their descriptions are : 'fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar', 'chlorides, 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density', 'ph', 'sulphates', 'alcohol', 'quality'.

Objectives:

- Find the best clustering algorithm based on their size, shape and density without relying on metrics like ELBOW method.

Data Validation:

Duplicate Values:

```
print("\nNumber of duplicated rows : ", df.drop(columns=['id']).duplicated().sum(),"\n")
```

```
Number of duplicated rows :  125
```

There were **125** duplicate values in the dataset. And the total percentage of duplicate values was **10%** and has been removed in the final dataset.

Valid Data Types:

```
Incorrect df types:
None
Data is correct with following:
fixed_acidity           float64
volatile_acidity        float64
citric_acid             float64
residual_sugar          float64
chlorides               float64
free_sulfur_dioxide     float64
total_sulfur_dioxide    float64
density                 float64
ph                      float64
sulphates               float64
alcohol                 float64
quality                   int64
id                        int64
dtype: object
```

There were no Incorrect data, and all data was correct and was in correct format.

Missing Values:

```
fixed_acidity           0
volatile_acidity        0
citric_acid             0
residual_sugar          0
chlorides               0
free_sulfur_dioxide     0
total_sulfur_dioxide    0
density                 0
ph                      0
sulphates               0
alcohol                 0
quality                 0
id                      0
```

There are no missing values in the dataset.

Overall, besides some duplicate values, data was valid, and we can proceed with the other steps.

Data Preparation:

Feature Selection:

Feature selection involves selecting a subset of relevant features (variables, predictors) for use in model construction. By doing so, we aim to improve the model's performance and reduce computational costs. In our case, I've selected the following features from the dataset:

```
features = ['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar',
       'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
       'ph', 'sulphates', 'alcohol']

X = df[features]
```

These features are numerical attributes of the wine that are relevant for clustering.

StandardScaler

StandardScaler is used to standardize features by removing the mean and scaling to unit variance. This is an important preprocessing step for many machine learning algorithms, especially those that rely on distance metrics like K-Means clustering.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Mathematically:

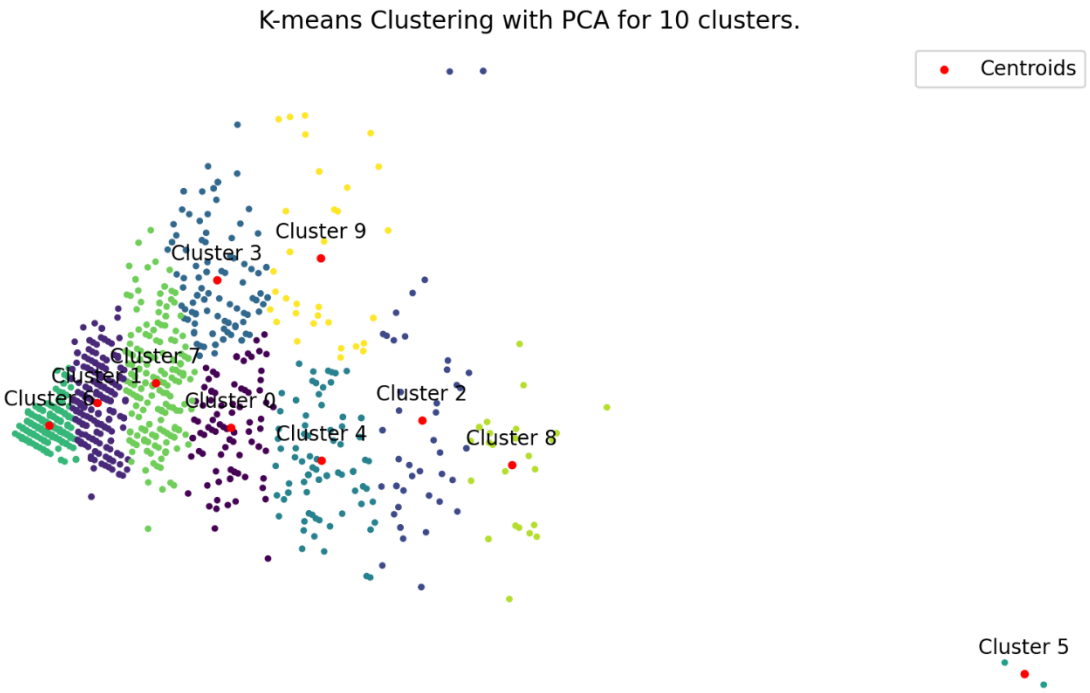
$$Z_i = \frac{X_i - \mu_i}{\sigma_i}$$

where:

- X_i is the original feature.
- μ_i is the mean of the feature X_i.
- σ_i is the standard deviation of the feature X_i.

K-Means Cluster Evaluation:

For 10 Cluster(K=10):



Cluster Analysis

1. **Size**:

The clusters vary in size, with some clusters having more data points than others. Cluster 3 and 9 seem to be the largest cluster and cluster 5 seems to be smallest. There is no uniformity between any of the clusters.

2. **Density**:

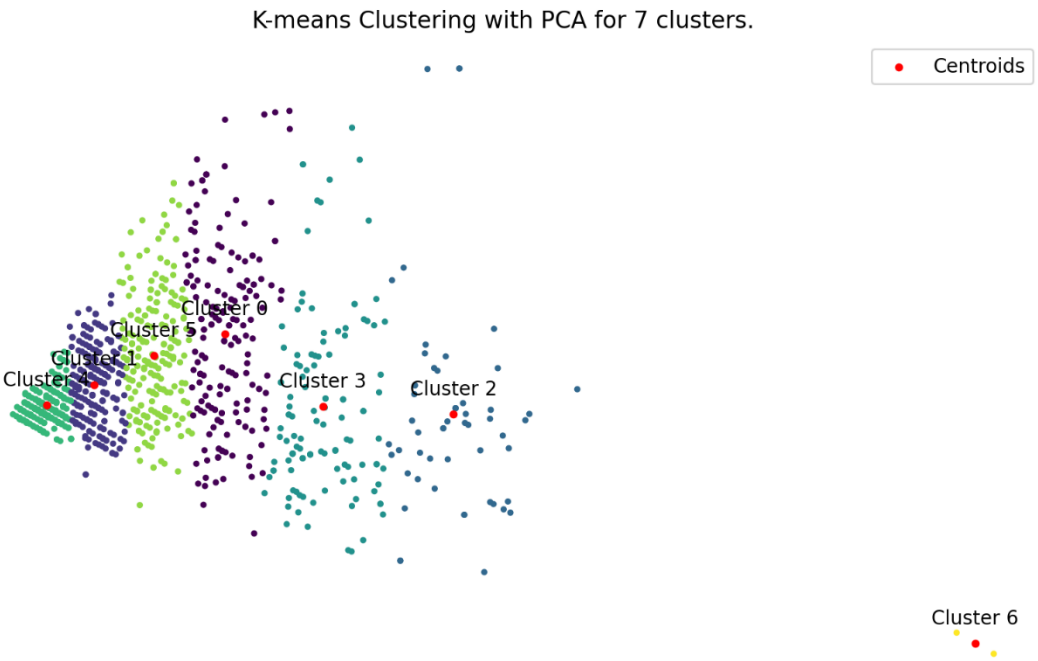
The density of the clusters also varies. Some clusters have points that are closely packed together, while others are more spread out. Cluster 7 seems to be a most dense cluster and cluster 2 seems to have the less density.

3. **Shape**:

The shapes of the clusters are not uniform. Some clusters appear to be more circular, while others are elongated or irregularly shaped. There is no shape pattern between any of the clusters. However, all clusters seem to be elongated.

Clustering with 10 clusters showed different size of cluster, varying density and non-uniform shape also there was some **overlapping** between clusters 0, 3 and 7. This is not ideal clustering.

For 7 Cluster(K=7):



Cluster Analysis

1. **Size**:

Like 10 clusters, clusters vary in size, with some clusters having more data points than others. Cluster 0 and 3 seem to be the largest cluster and cluster 6 seems to be smallest. There is no uniformity between any of the clusters.

2. **Density**:

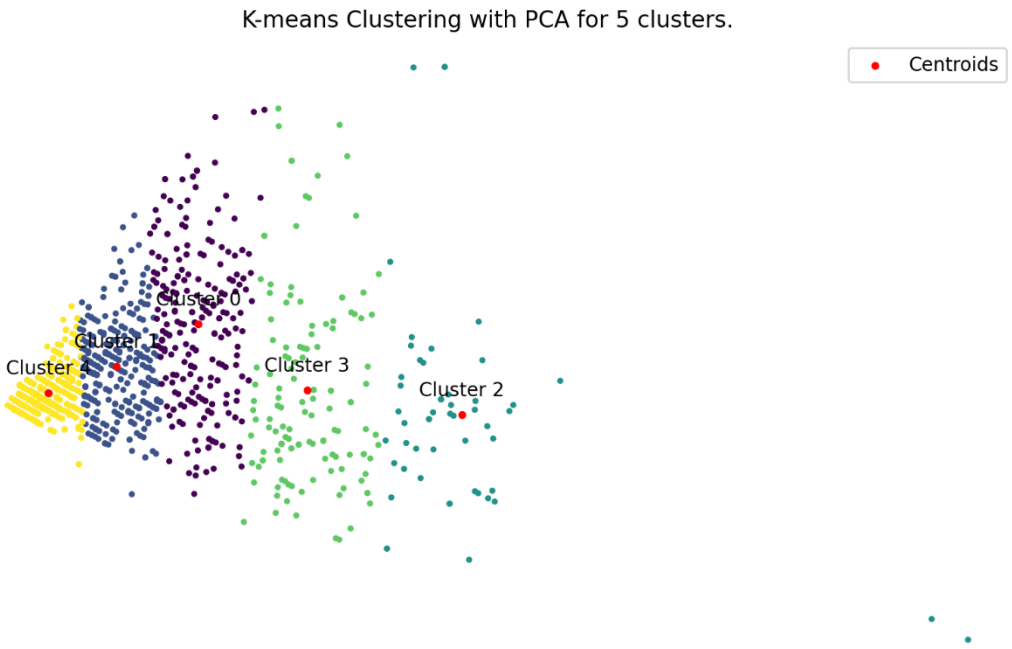
The density of the clusters also varies. Some clusters have points that are closely packed together, while others are more spread out. Cluster 2 appears to be the most densely packed. Cluster 6 seems to be the least dense.

3. **Shape**:

The shapes of the clusters are not uniform. Some clusters appear to be more circular, while others are elongated or irregularly shaped. Cluster 2 is more circular. Cluster 0 is elongated. Cluster 6 has an irregular shape.

In comparison to cluster 10, The shapes of the clusters have changed, with some clusters becoming more circular or elongated. Also, there is some **overlapping** between cluster 3 and cluster 5.

For 5 cluster (K=5)



Cluster Analysis

1. **Size**:

The clusters vary in size, with some clusters having more data points than others.

2. **Density**:

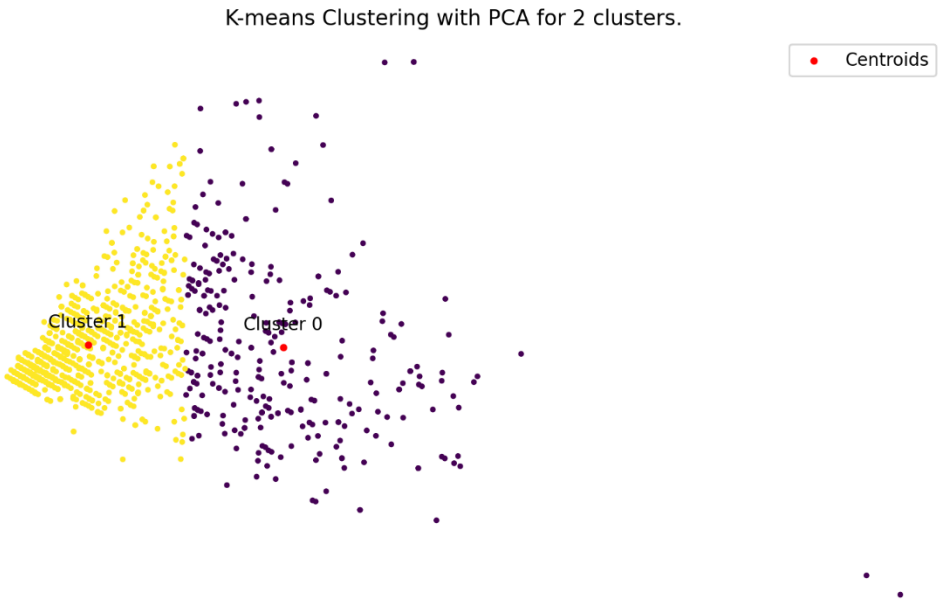
Here though all the clusters have somewhat the same number of data points. Some clusters have points that are closely packed together, while others are more spread out.

3. **Shape**:

The shapes of the clusters are not uniform. Some clusters appear to be more circular, while others are elongated or irregularly shaped. Cluster 2 is more circular.

This clustering is somewhat better than clustering with 7 cluster with minimum **overlapping**. But the shape and density of clusters is not uniform.

For 2 cluster (K=2)



1. **Size**:

The clusters vary in size, with some clusters having more data points than others. **Biggest Cluster**: Cluster 1 appears to be the largest cluster. **Smallest Cluster**: Cluster 0 seems to be the smallest cluster.

2. **Density**:

The density of the clusters also varies. Some clusters have points that are closely packed together, while others are more spread out. **Most Dense Cluster**: Cluster 0 appears to be the most densely packed. **Least Dense Cluster**: Cluster 1 seems to be the least dense.

3. **Shape**:

The shapes of the clusters are not uniform. Cluster 0 appears to be more circular, while cluster 0 is elongated or irregularly shaped.

Why 5 is the optimal number of clusters?

Clustering with (k=5) is considered the best among all the clustering results for several reasons:

1. **Balanced Size**: The clusters in the (k=5) clustering are more balanced in size compared to the other clusterings. This balance ensures that no single cluster dominates the dataset, providing a more even distribution of data points.
2. **Density**: The clusters in the (k=5) clustering have a good balance of density. While some clusters are more densely packed, others are more spread out, capturing the natural variation in the data. This balance in density helps in identifying distinct groups within the data.
3. **Shape**: The shapes of the clusters in the (k=5) clustering are more uniform and well-defined. There are clear distinctions between circular, elongated, and irregular shapes, which helps in better understanding the underlying structure of the data.
4. **Minimal Overlapping**: The (k=5) clustering shows minimal overlapping between clusters. This clear separation ensures that each cluster represents a distinct group of data points, reducing ambiguity and improving the interpretability of the results.
5. **Optimal Number of Clusters**: The choice of (k=5) strikes a balance between having too many clusters (which can lead to overfitting) and too few clusters (which can lead to underfitting). This optimal number of clusters captures the natural groupings in the data without overcomplicating the model.

Conclusion:

In this study, we explored K-means clustering with different values of (k) (10, 7, 5, and 2) to determine the optimal number of clusters for our dataset. Through a detailed analysis of each clustering result, we observed significant variations in cluster size, density, and shape. The (k=5) clustering emerged as the most balanced and interpretable solution, with clusters that were more uniform in shape and density, and minimal overlap between them. This clustering provided a clear and distinct separation of data points, capturing the natural groupings in the data without overcomplicating the model. Overall, the (k=5) clustering was identified as the best choice, demonstrating the importance of selecting an optimal number of clusters to achieve meaningful and interpretable results in K-means clustering.

Appendix:

Code Used for K-Means and 2D plot:

```
CLUSTER_NUMBER = 5

kmeans = KMeans(n_clusters=CLUSTER_NUMBER, random_state=42, n_init="auto")
kmeans.fit(X)

# Get cluster labels
label = kmeans.labels_
# Add the cluster labels to the original dataframe
df['cluster'] = label
centroids = kmeans.cluster_centers_

# 2D Plot
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
centroids_pca = pca.transform(centroids)

# Plot the clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['cluster'], cmap='viridis', s=5)
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title(f'K-means Clustering with PCA for {CLUSTER_NUMBER} clusters.')

# Annotate the clusters
for i, centroid in enumerate(centroids_pca):
    plt.annotate(f'Cluster {i}', (centroid[0], centroid[1]),
                 textcoords="offset points", xytext=(0,10), ha='center')


# Plot the centroids
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], s=10, c='red', label='Centroids')
plt.legend()
plt.show()
```