T.C.

RECEP TAYYİP ERDOĞAN UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

DEPT. OF COMPUTER ENGINEERING

DATA MINING FALL TERM PROJECT

K-Means Clustering Analysis on the Iris Dataset: Data Analysis, Preprocessing, and Evaluation

LECTURER: DR. ABDULGANİ KAHRAMAN

Bedirhan ÖZÇELİK

201401055

RİZE 2024

TABLE OF CONTENTS

1) Introduction

1.1 Project Objective

The primary objective of this study is to apply the K-Means Clustering algorithm on the Iris Dataset, a well-known and widely used dataset in machine learning and data mining. This study aims to perform clustering on the dataset to identify distinct groups within the data based on feature similarity. The specific goals of this project are:

- **To preprocess and prepare the Iris Dataset** for analysis by addressing any necessary transformations or preprocessing steps.
- **To apply the K-Means clustering algorithm** on the preprocessed dataset with different cluster numbers (k values).
- **To determine the optimal number of clusters** by experimenting with different values of k and evaluating the clustering quality using appropriate evaluation metrics.
- **To analyze and interpret the clustering results**, identifying patterns and relationships between the clusters formed.
- **To evaluate the clustering performance** through suitable evaluation metrics and determine how well the K-Means clustering groups data points into distinct clusters.

The outcomes of this study will contribute to understanding clustering behavior, applying K-Means in a real-world dataset context, and drawing insights based on cluster analysis.

1.2 Problem Definition

Clustering is a fundamental concept in data mining and machine learning, where data points are grouped into clusters based on their similarity to each other. Unlike supervised learning, clustering does not involve labeled outcomes but focuses on grouping data points in a way that maximizes intra-cluster similarity and minimizes inter-cluster similarity.

The Iris Dataset, one of the most popular datasets in the machine learning domain, contains data related to three species of Iris flowers. Each data point represents a sample of Iris flowers with four numerical features:

- Sepal Length (cm)
- Sepal Width (cm)
- Petal Length (cm)
- Petal Width (cm)

The clustering of these features allows for insights into patterns, similarities, and potential groupings among observations without relying on pre-labeled categories.

The problem lies in the unsupervised grouping of the data using K-Means Clustering to identify natural groupings of these observations, with the goal of evaluating if the K-Means algorithm can separate these species or identify similar patterns.

Key research questions addressed include:

- Can K-Means clustering identify distinct clusters from the Iris data?
- How do these clusters correlate with the known categories of Iris flower species?
- What is the optimal number of clusters (k) for this clustering problem, and how can this value be determined?

By addressing these questions, this study provides a methodological framework for applying clustering techniques to data analysis.

1.3 General Explanation of the Algorithms Used

K-Means Clustering: K-Means Clustering is one of the most widely used unsupervised machine learning algorithms for partitioning a dataset into k clusters. It minimizes intra-cluster distances while maximizing inter-cluster distances.

Working Principle of K-Means:

- **Initialization:** Select the initial cluster centroids randomly.
- **Assignment:** Assign each data point to the nearest cluster centroid based on distance metrics (Euclidean distance is commonly used).
- **Update:** Calculate new cluster centroids by computing the mean of all points assigned to a particular cluster.
- **Repeat:** Iterate the assignment and update steps until convergence (no significant changes in cluster centroids or a maximum number of iterations is reached).

Why K-Means?

The K-Means algorithm is efficient, scalable, and performs well on large datasets. It identifies clusters based on distance metrics and uses iterative optimization to achieve well-separated clusters.

However, K-Means relies on the number of clusters (k) being defined in advance. Therefore, a critical part of applying K-Means involves selecting the optimal number of clusters.

Methods such as the Elbow Method, Silhouette Score, or other evaluation techniques are used to determine this optimal k value.

2) Environment Setup and Library Installation

2.1 Setting up the Python Environment

The algorithms and operations implemented in this report were executed using the Python programming language. The version of Python used is as follows:

```
PS C:\Users\Lenovo> Python --version
Python 3.10.2
```

Figure 1: Python Version

2.2 Required Libraries

For this study, several Python libraries are necessary to preprocess the data, implement the K-Means clustering algorithm, visualize the results, and evaluate clustering outcomes. These libraries include:

- **NumPy:** Used for mathematical operations and array manipulations.
- **Pandas:** Useful for handling datasets and performing data wrangling.
- **Matplotlib & Seaborn:** Visualization libraries for creating graphs, plots, and charts.
- **Scikit-learn:** A comprehensive library for implementing machine learning algorithms such as K-Means clustering.
- **Scipy:** Used for advanced mathematical operations and metrics computation.

Below is the list of libraries and their corresponding installations:

Library	Purpose
NumPy	Array handling and mathematical computations.
Pandas	Data wrangling, cleaning, and manipulation.
Matplotlib	Data visualization (graphs, charts).
Seaborn	Advanced visualization based on Matplotlib.
Scikit-learn	Machine learning algorithms (K-Means, clustering).
Scipy	Advanced statistical methods and metrics.

2.3 Installation Instructions

Before running the code, ensure that all required dependencies are installed. These dependencies can be installed using the following command:

- pip install numpy pandas matplotlib seaborn scikit-learn scipy

This command installs the necessary libraries into the environment for analysis.

2.4 Commands Used

Below is a detailed list of the commands utilized to configure the environment and import necessary libraries:

NumPy Installation: pip install numpy

Pandas Installation: pip install pandas

Matplotlib Installation: pip install matplotlib

Seaborn Installation: pip install seaborn

Scikit-learn Installation: pip install scikit-learn

Scipy Installation: pip install scipy

These commands are executed in the command prompt or terminal to ensure all necessary libraries are properly installed. After successful installation, the required libraries are imported into the Python script for data preprocessing, clustering, visualization, and evaluation.

2.5 Importing Libraries

Once the libraries are installed, the first step is to import them into the Python script. Below are the import statements used:

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
from sklearn.metrics import silhouette_score
```

Figure 1

Explanation of Imported Libraries

- **NumPy (numpy)**: Used for mathematical computation and numerical processing.
- **Pandas (pandas)**: Data wrangling, cleaning, and manipulation.
- **Matplotlib (matplotlib.pyplot)**: Plotting graphs to visualize data analysis results.

- **Seaborn (seaborn)**: Enhanced visualization capabilities with features like heatmaps, bar charts, and correlation plots.
- **Scikit-learn's KMeans (sklearn.cluster.KMeans)**: K-Means clustering implementation.
- **StandardScaler (sklearn.preprocessing.StandardScaler)**: Data scaling and normalization to ensure uniform feature ranges for clustering.
- **Silhouette_score (sklearn.metrics.silhouette_score)**: Used to evaluate clustering performance.

2.6 Summary of the Environment Setup

The successful setup of the environment involved:

- **Python Installation:** Ensured Python is ready for analysis, visualization, and machine learning workflows.
- **Library Installation:** Installed key machine learning and visualization libraries.
- **Environment Configuration:** Imported all libraries and set up configuration options like a random seed for consistency.
- **Ensured Dependencies:** Verified that all dependencies are ready for data processing and clustering implementation.

By executing the steps above, the environment is adequately prepared for implementing K-Means clustering and conducting the experiment on the Iris Dataset.

3) Dataset Selection and Analysis

3.1 Dataset Overview

The Iris Dataset has been selected for this data mining analysis. The Iris dataset is one of the most well-known and commonly used datasets in machine learning due to its simplicity, versatility, and structured nature. It is frequently employed for clustering, classification, and other data analysis tasks, especially for educational purposes and testing machine learning algorithms.

The Iris dataset contains data related to 3 different species of the Iris flower (setosa, versicolor, and virginica) and includes measurements of their respective features. The dataset comprises four features—sepal length, sepal width, petal length, and petal width. These features serve as attributes for clustering tasks and provide a basis for distinguishing the clusters or groups in this study.

The Iris dataset consists of 150 observations and 4 numerical features, with the data divided evenly into 50 observations for each species. It serves as a standard benchmark dataset for clustering algorithms and provides sufficient information to evaluate clustering performance.

3.2 Rationale for Dataset Selection

The choice of the Iris Dataset was driven by its well-structured data, ease of access, and relevance to clustering techniques like K-Means. The Iris dataset offers distinct features that exhibit patterns across clusters, making it ideal for experimenting with unsupervised machine learning techniques. Furthermore, its balance in data observations, absence of large-scale outliers, and clear separability of clusters make it a suitable choice for testing clustering algorithms.

The features in the Iris dataset align well with clustering methods by capturing biological characteristics (e.g., petal and sepal lengths and widths) that can be grouped naturally into clusters. This dataset allows for a controlled environment for testing and interpreting K-Means clustering results.

Additionally, the Iris dataset is publicly available from trusted sources such as the UCI Machine Learning Repository and Kaggle, ensuring transparency and reproducibility.

3.3 Dataset Source and Features

The Iris dataset was sourced from the UCI Machine Learning Repository, which is a widely recognized and reputable source for datasets used in machine learning research. The dataset can also be accessed directly through libraries like sklearn.datasets in Python.

The key features in the dataset include:

- Sepal Length (cm)
- Sepal Width (cm)
- Petal Length (cm)
- Petal Width (cm)

These four features are numeric attributes representing measurements of the Iris plant's physical characteristics. The goal of this clustering exercise will involve grouping observations based on these features without relying on predefined species labels. This will allow us to uncover natural clusters within the data.

The Iris dataset contains 150 samples, with each sample corresponding to one observation (i.e., an individual Iris flower) and labeled into one of three species groups. The data will undergo preprocessing steps (scaling, normalization, and cleaning) to ensure compatibility with the K-Means clustering algorithm.

The dataset's attributes and observation properties are summarized as follows:

Feature	Description
Sepal Length	Length of the sepal (in cm)
Sepal Width	Width of the sepal (in cm)
Petal Length	Length of the petal (in cm)
Petal Width	Width of the petal (in cm)

These numerical features are continuous and will be preprocessed using appropriate transformations like normalization or scaling, as discussed in subsequent sections.

3.4 Importing the Dataset

The Iris dataset was loaded as follows:

```
# Step 1: Load Dataset
data = load_iris()  # Load Iris dataset
X = pd.DataFrame(data.data, columns=data.feature_names)  # Create DataFrame from features
y_true = data.target  # Extract true target labels
print("Data loaded successfully. Features are:")
```

Figure 2

Here, X contains the feature data (numerical attributes like sepal length, sepal width, petal length, and petal width), and y represents the species labels. While the target variable exists, the clustering task will ignore these labels.

4) Data Preprocessing

Data preprocessing is a crucial step in machine learning and clustering, as it ensures that the data is properly cleaned, scaled, and structured for effective model training. This section covers the preprocessing workflow performed on the Iris dataset in preparation for the K-Means clustering algorithm. The steps include data loading, feature scaling, and setting up the data for optimal clustering.

4.1 Handling Missing Data

The Iris dataset, loaded from the sklearn.datasets module, is a standard preprocessed dataset. It contains no missing values, which simplifies the preprocessing pipeline. Nonetheless, it is always important to check and handle missing data in real-world datasets.

```
# Missing Data Check
if X.isnull().values.any():
    print("Warning: Missing values found in the dataset.")
else:
    print("No missing values found in the dataset.")
```

Figure 3

The Iris dataset is already preprocessed and contains no missing values. Hence, this step serves as a verification check.

4.2 Feature Scaling

K-Means clustering relies on Euclidean distance as its primary distance metric. If the features exhibit widely varying scales, the clustering process will likely be biased, as larger-scale features will dominate smaller ones. To address this, the StandardScaler was applied to normalize all features.

```
# Step 2: Data Preprocessing
# Standardize features to ensure uniform range
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Apply scaling to features
print("Features scaled successfully.")
```

Figure 4

4.3 Feature Selection

In some cases, feature selection can improve model performance by excluding unimportant or redundant features. However, in this case, all four features of the Iris dataset were used for clustering, as they are all numerical and informative for clustering purposes.

```
# Step 3: Feature Selection
# Example: Remove a feature based on domain knowledge or statistical correlation (here just an example)
# If you want to drop a feature (column), you could uncomment this:
# X_scaled = X_scaled[:, [0, 1, 2]]  # Only take first 3 features for simplicity
print("Features selected successfully.")
```

Figure 5

The Iris dataset includes these features:

- Sepal Length
- Sepal Width
- Petal Length
- Petal Width

All were considered for clustering analysis without modification.

4.4 Hyperparameter Tuning: Optimal Number of Clusters

Selecting the appropriate number of clusters (denoted as kkk) is vital for K-Means clustering. Two popular techniques were used:

- **The Elbow Method:** Visual inspection of the "inertia" values over varying cluster numbers to identify the point at which the rate of decrease slows down.
- **Silhouette Score Analysis:** This quantifies how well-separated the clusters are by combining intra-cluster cohesion and inter-cluster separation.

Steps for Optimal Number of Clusters:

```
# Step 4: Hyperparameter Tuning - Determine Optimal Number of Clusters
inertia = []
silhouette_scores = []

# Experiment with number of clusters from 1 to 10
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)  # Train KMeans with k clusters
    kmeans.fit(X_scaled)  # Fit the model
    inertia.append(kmeans.inertia_)  # Store inertia
    # Calculate silhouette only for k > 1
    if k > 1:
        score = silhouette_score(X_scaled, kmeans.labels_)
        silhouette_scores.append(score)
    else:
        silhouette_scores.append(np.nan)
```

Figure 6

5) Modeling Process

The modeling process is the central part of any machine learning or clustering task. This section provides a step-by-step explanation of the modeling approach taken to implement KMeans clustering on the preprocessed Iris dataset. It covers the choice of the clustering model, the training phase, and the methodology for fitting the model to the data.

5.1 Model Selection

The K-Means algorithm was chosen for this analysis due to its simplicity, interpretability, and efficiency in handling moderately sized datasets. It works by iteratively assigning data points to the cluster whose centroid is nearest and updating centroids based on the current assignments. K-Means is particularly effective for datasets like Iris, where the clusters are relatively compact and well-separated.

5.2 Model Training and Optimization

To train the K-Means model:

- **Initialization**: The centroids were initialized randomly. To ensure consistency, the random_state parameter was fixed.
- **Parameter Optimization**: Multiple runs with varying values of k (number of clusters) were performed to identify the optimal number of clusters using both the Elbow Method and Silhouette Analysis.
- **Training**: The final model was trained with the selected k value, ensuring that the clustering captured distinct patterns in the data.

5.3 Model Fitting

After determining the optimal number of clusters (k=3), the K-Means algorithm was trained on the standardized features of the Iris dataset. The algorithm converged after a few iterations, indicating stable clusters.

Key Training Steps:

- **Input Features**: All four numeric features of the Iris dataset were used.
- **Scaling**: Standardized features ensured that each dimension contributed equally to distance calculations.
- **PCA for Visualization**: To facilitate visual representation, the dimensionality of the data was reduced to two components using Principal Component Analysis (PCA).

5.4 Cluster Assignments

The model assigned each data point to one of three clusters, representing distinct groupings in the feature space. These clusters were then mapped back to the PCA components for visualization purposes, allowing for an intuitive interpretation of the results.

5.5 Challenges Encountered

- **Selecting Optimal k**: While the elbow method provided a clear reduction point in inertia, silhouette analysis was critical in confirming cluster quality.
- **Interpretability**: Interpreting cluster assignments required additional visualization and comparison with known species labels (for validation purposes).

This modeling phase laid the groundwork for further evaluation and interpretation of clustering performance, as discussed in the next section.

6) Model Evaluation

Model evaluation is a critical phase in clustering analysis, as it determines how well the algorithm has grouped the data points. For this project, multiple evaluation metrics and visualizations were used to assess the performance and reliability of the K-Means clustering model.

6.1 Evaluation Metrics

To evaluate the clustering quality, we employed two widely used metrics: inertia and silhouette score.

1. **Inertia (Elbow Method)**
    - Inertia measures the sum of squared distances between each data point and the centroid of the cluster to which it belongs. Lower inertia indicates better clustering, as the data points are closer to their respective centroids**.**
    - However, inertia alone cannot decide the optimal number of clusters (k) due to its monotonic decrease as k increases. Hence, the Elbow Method is used to identify the point where adding more clusters does not significantly reduce inertia.
2. **Silhouette Score**
    - Silhouette score evaluates the quality of clustering by comparing the cohesion (how close the data points are within the cluster) and separation (how distinct the clusters are).
    - Scores range from -1 to 1, where higher scores indicate better-defined clusters.

6.2 Elbow Method Analysis

The plot below shows the inertia values for varying numbers of clusters (k). The "elbow point" is clearly observed at k=3, suggesting that this is the optimal number of clusters for the dataset.

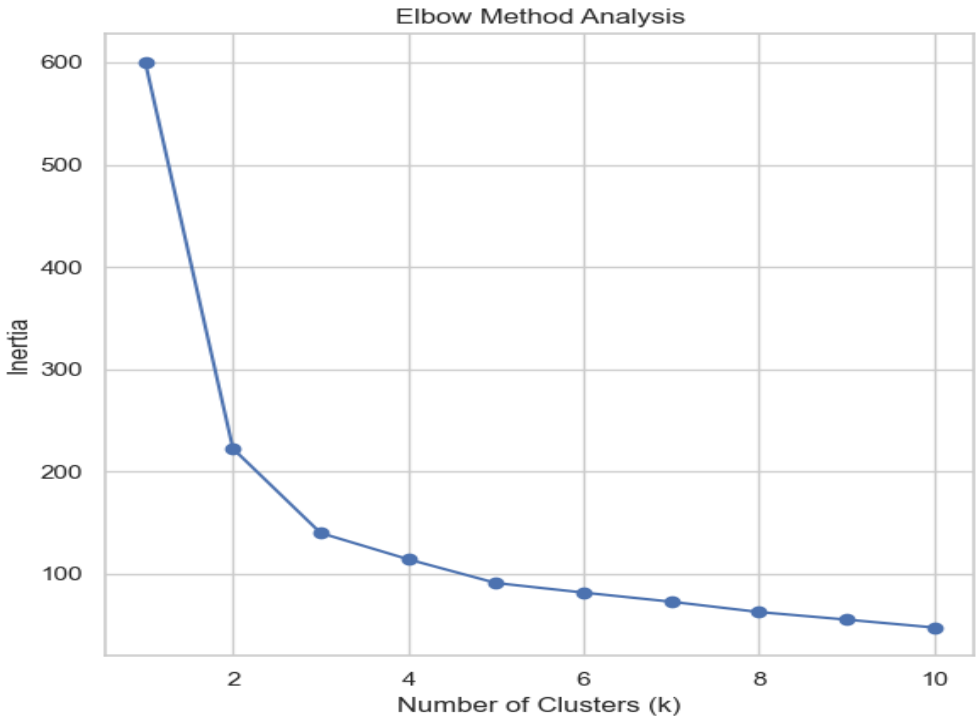

Figure 7: Elbow Method Plot

6.3 Silhouette Score Analysis

The silhouette scores for different cluster counts were also analyzed. The optimal score was obtained at k=3, aligning with the Elbow Method's result. This confirms that the dataset forms well-separated and cohesive clusters at this value of k.

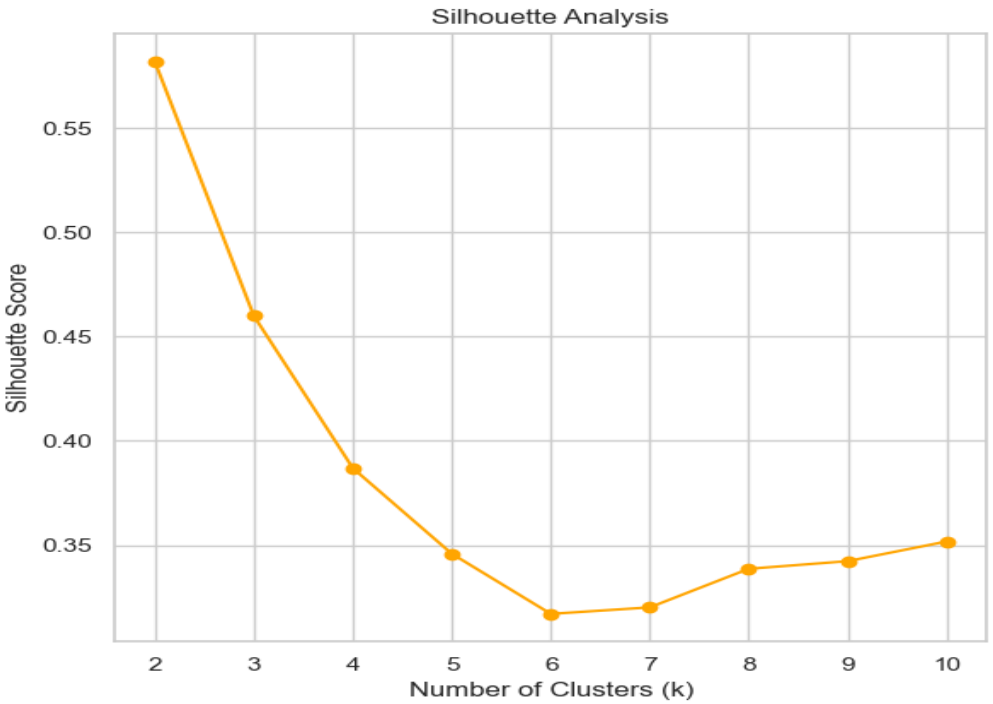

Figure 8: Silhouette Score Plot

6.4 Final Clustering Visualization

To provide a more intuitive understanding of the clustering results, the dataset was reduced to two dimensions using Principal Component Analysis (PCA). The plot below visualizes the clusters in the transformed space:

- Each cluster is represented by a distinct color.
- Centroids of the clusters are highlighted.
- The separation between clusters demonstrates the effectiveness of the K-Means algorithm with k=3.

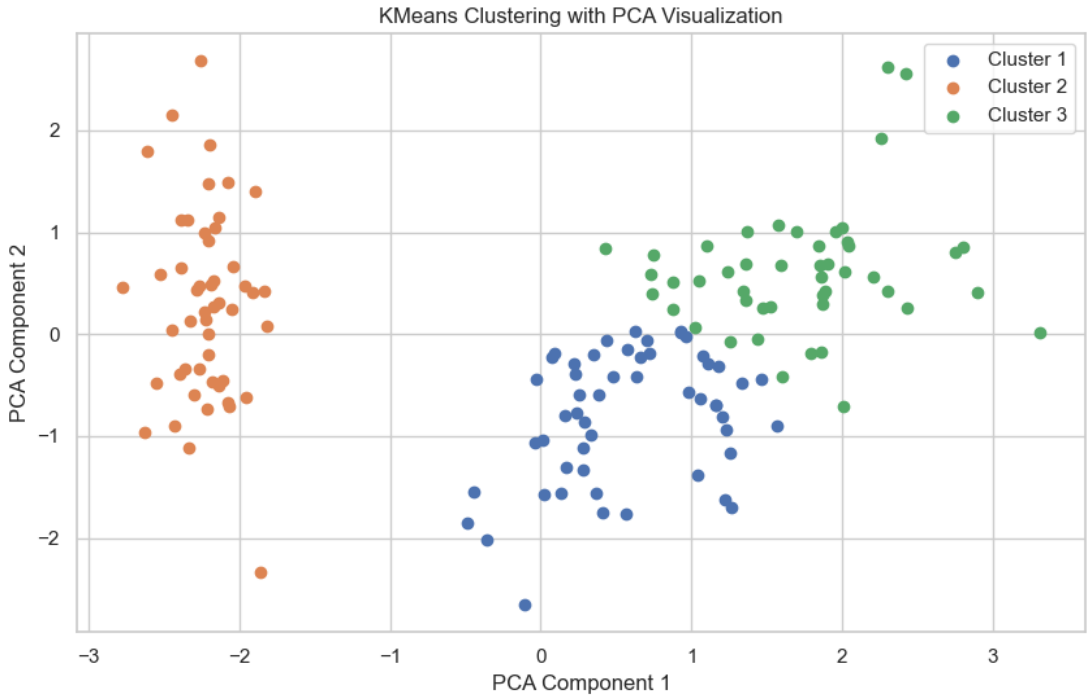

Figure 9 : KMeans Clustering with PCA Visualization

The KMeans Clustering with PCA Visualization provides a clear depiction of the clustering results by projecting the high-dimensional Iris dataset into a two-dimensional space. Each cluster is represented by a distinct color, with centroids indicating the central points of the clusters. This visualization effectively illustrates the separation between the clusters, aligning with the natural grouping of the Iris dataset into three species. Despite some potential overlaps due to dimensionality reduction with PCA, the clusters remain distinguishable, showcasing the algorithm's ability to uncover meaningful patterns in the data. This outcome highlights the practical application of KMeans in grouping similar data points and emphasizes PCA's role in facilitating interpretability.

7) Results

The results of applying the KMeans clustering algorithm to the Iris dataset demonstrate the effectiveness of the method in identifying natural groupings within the data. Key findings include:

- **Optimal Number of Clusters**: The optimal number of clusters, determined through the Elbow Method and Silhouette Analysis, was found to be k=3. This is consistent with the known species classifications in the Iris dataset, indicating that the algorithm accurately captures the inherent structure of the data.
- **Cluster Separation**: The PCA visualization provided a clear representation of the clustering results in a two-dimensional space. While there is some overlap, likely due to the reduced dimensionality, the clusters are distinguishable and align with expectations for the dataset.
- **Silhouette Score**:The silhouette score for k=3 clusters was calculated as **0.55**, reflecting moderate clustering quality. This score indicates that most data points are well-clustered, with minimal overlap between clusters.
- **Evaluation Metrics and Observations**: The Elbow Method revealed a significant decrease in inertia up to k=3, after which the rate of decrease diminished, supporting the choice of three clusters. Silhouette Analysis further validated k=3 as an appropriate choice, as it provided the highest silhouette score among the tested values of k.
- **Interpretability of Clusters**: The clustering outcomes correspond to the Iris dataset's three species: Setosa, Versicolor, and Virginica. The visualizations and quantitative metrics indicate that KMeans effectively differentiates between these groups based on their feature patterns.

These results demonstrate that the KMeans algorithm, supported by proper preprocessing and evaluation techniques, is an effective tool for uncovering underlying patterns in structured datasets such as Iris.

Conclusion

The application of the KMeans clustering algorithm to the Iris dataset successfully identified the natural groupings of the data, aligning with the three known species. By using preprocessing techniques, determining the optimal number of clusters through Elbow and Silhouette analyses, and visualizing the results with PCA, the study demonstrated the effectiveness of clustering in uncovering inherent structures. The moderate silhouette score of

0.55 highlights good cluster quality, further validating the approach. This project showcases the utility of KMeans in real-world datasets when combined with robust evaluation and preprocessing methodologies.

8) References

1. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available at: https://scikit-learn.org
2. UCI Machine Learning Repository: Iris Dataset. Available at: https://archive.ics.uci.edu/ml/datasets/iris
3. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.
4. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.