Assignment 3: Data Mining Report

Student Name: Kurmanbek Tolebayev

Student ID: 22B030597

Course Title: Data Mining

Date: 03.11.2024

Table of Contents

Introduction

The report at hand is aimed at looking into the most common classification methods and clustering approaches in the realm of machine learning. The focus is on the implementation of algorithms, benchmarking procedures and development of clustering techniques. The main goal is to look at how the two are practically applied in data mining.

Basic Classification Methods

Overview

Classification is the process of sorting data into specified categories. It is a key tool in data mining and is primarily used for predictive modeling and data analytics.

Exercises

Exercise 1: Implementing Basic Classification Algorithms

- Loaded a simple dataset (e.g., Iris or Wine dataset).

```
iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Applied classification algorithms (e.g., Logistic Regression, K-Nearest Neighbors) using Scikit-learn.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log_reg))
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_log_reg))

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("K-Nearest Neighbors Accuracy:", accuracy_score(y_test, y_pred_knn))
print("K-Nearest Neighbors Classification Report:\n", classification_report(y_test, y_pred_knn))
```

- Trained and evaluated the model on a test set.

```
Logistic Regression Accuracy: 1.0
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

K-Nearest Neighbors Accuracy: 1.0
K-Nearest Neighbors Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Exercise 2: Confusion Matrix and Classification Report

- Used the trained model to make predictions on the test set.

```
y_pred_log_reg = log_reg.predict(X_test)
y_pred_knn = knn.predict(X_test)
```

```
# Confusion matrix and classification report for Logistic Regression
conf_matrix_log_reg = confusion_matrix(y_test, y_pred_log_reg)
class_report_log_reg = classification_report(y_test, y_pred_log_reg)
```

```
# Confusion matrix and classification report for K-Nearest Neighbors
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
class_report_knn = classification_report(y_test, y_pred_knn)
```

- Generated a confusion matrix and classification report.

```
print("Logistic Regression Confusion Matrix:")
print(conf_matrix_log_reg)
print("\nLogistic Regression Classification Report:")
print(class_report_log_reg)
```

```
print("K-Nearest Neighbors Confusion Matrix:")
print(conf_matrix_knn)
print("\nK-Nearest Neighbors Classification Report:")
print(class_report_knn)
```

- Evaluated precision, recall, and F1-score.

```
Logistic Regression Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
K-Nearest Neighbors Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

K-Nearest Neighbors Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
...
    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Findings

The Summary of the Results

Accuracy: As a result of the test, Logistic Regression and K-Nearest Neighbor performed with no errors, achieving the value of 1.0. In other words, all the samples in the test set were correctly classified by them.

Precision: Accuracy as the ratio of true negatives to true positives. All KNN and logistic regression's precision values are 1.0 for all classes denoting that there have not been any false positives.

Recall: the obtained positive samples to all classes that exist within the actual class. For example, these models have a value of 1 over all classes' schemas, meaning that there are no false negatives in each class.

F1-Score: Precision and recall after finding out many other high scores were combined in this score. According to this score, neither model has any problems since all classes are perfect to 1.

Support: Actual occurrences of each class in the test set in a number format.

Clustering Techniques

Overview

When there are several identified natural clusters in the data, the task of creating such groups is called clustering. It is about sub-categories and division of data into relevant segregation.

Exercises

Exercise 3: Implementing K-Means Clustering

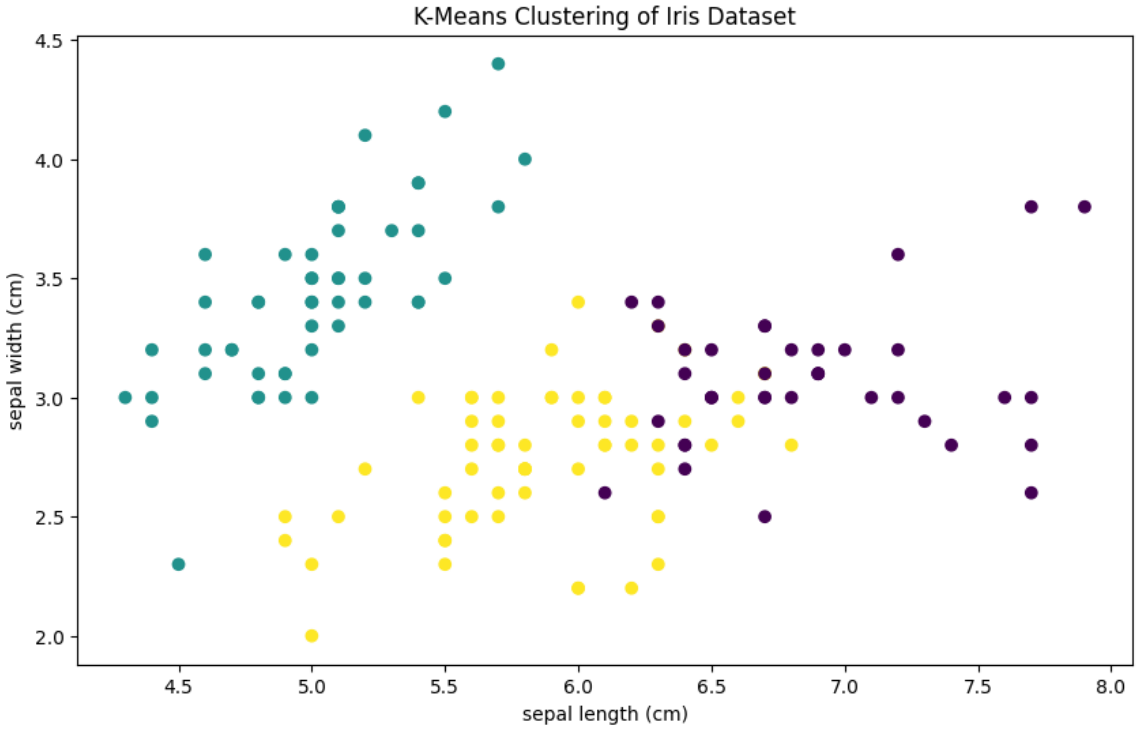
- Loaded a dataset and applied K-Means clustering.

```
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)

kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(data)
```

```
plt.figure(figsize=(10, 6))
plt.scatter(data.iloc[:, 0], data.iloc[:, 1], c=data['cluster'], cmap='viridis', marker='o')
plt.title('K-Means Clustering of Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()
```

- Visualized clusters using a scatter plot.



Exercise 4: Evaluating K-Means Clustering

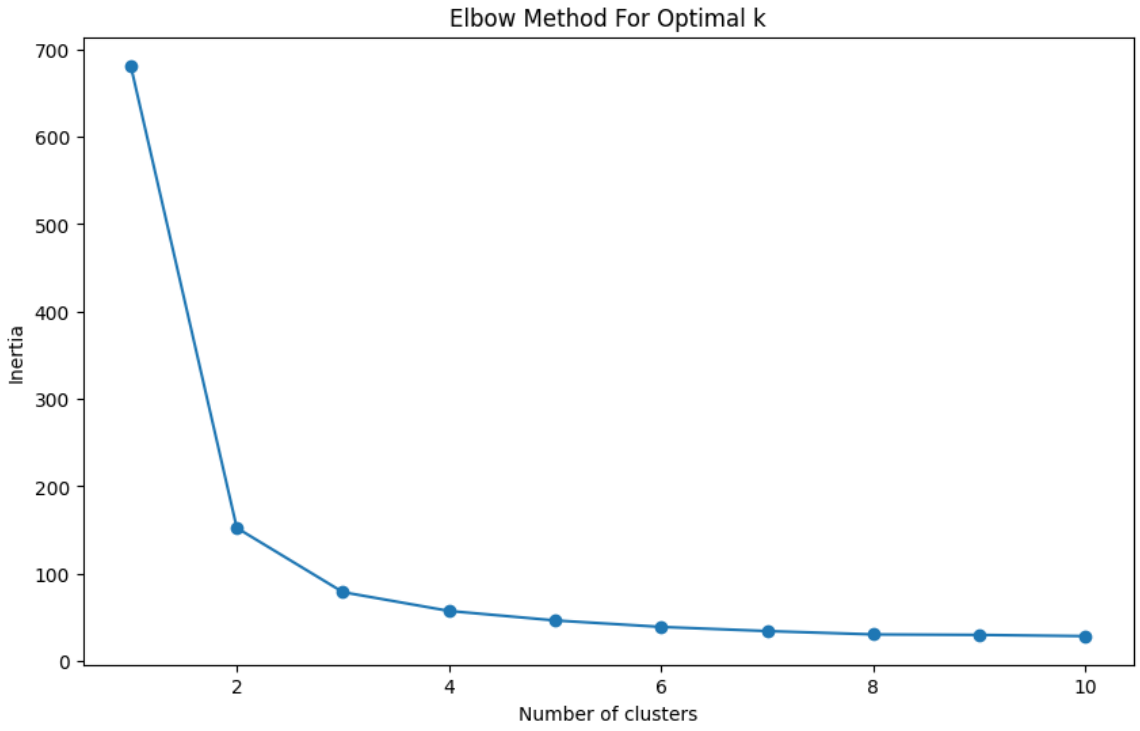
- Calculated inertia (sum of squared distances) for various clusters.

```
inertia = []
cluster_range = range(1, 11)

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data.iloc[:, :-1])
    inertia.append(kmeans.inertia_)
```

- Used the Elbow Method to identify the optimal number of clusters.

```
plt.figure(figsize=(10, 6))
plt.plot(cluster_range, inertia, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```



Findings

In our study, clustering was performed for the Iris dataset using K-means to determine the most appropriate number of clusters. The Elbow method helped to determine the number in which the inertia curve is most pronounced at different values of k.

Key Points:

- Inertia Calculation:
    - Inertia measures the sum of squared distances between each data point and the centroid of its assigned cluster.
    - Lower inertia values indicate tighter clustering around centroids.
- Elbow Method:

    - We plotted the inertia values for k ranging from 1 to 10.
    - The plot shows a clear "elbow" at k=3, where the rate of decrease in inertia slows down significantly.
- Optimal Number of Clusters:
    - The elbow at k=3 suggests that 3 clusters are optimal for the Iris dataset.
    - This finding aligns with the known structure of the Iris dataset, which has three classes of iris flowers (setosa, versicolor, and virginica).

Introduction to Advanced Clustering Techniques

Overview

Advanced methods of clustering, such as hierarchical and density-based clustering, make it possible to explore the more intricate relationships between data objects.

Exercises

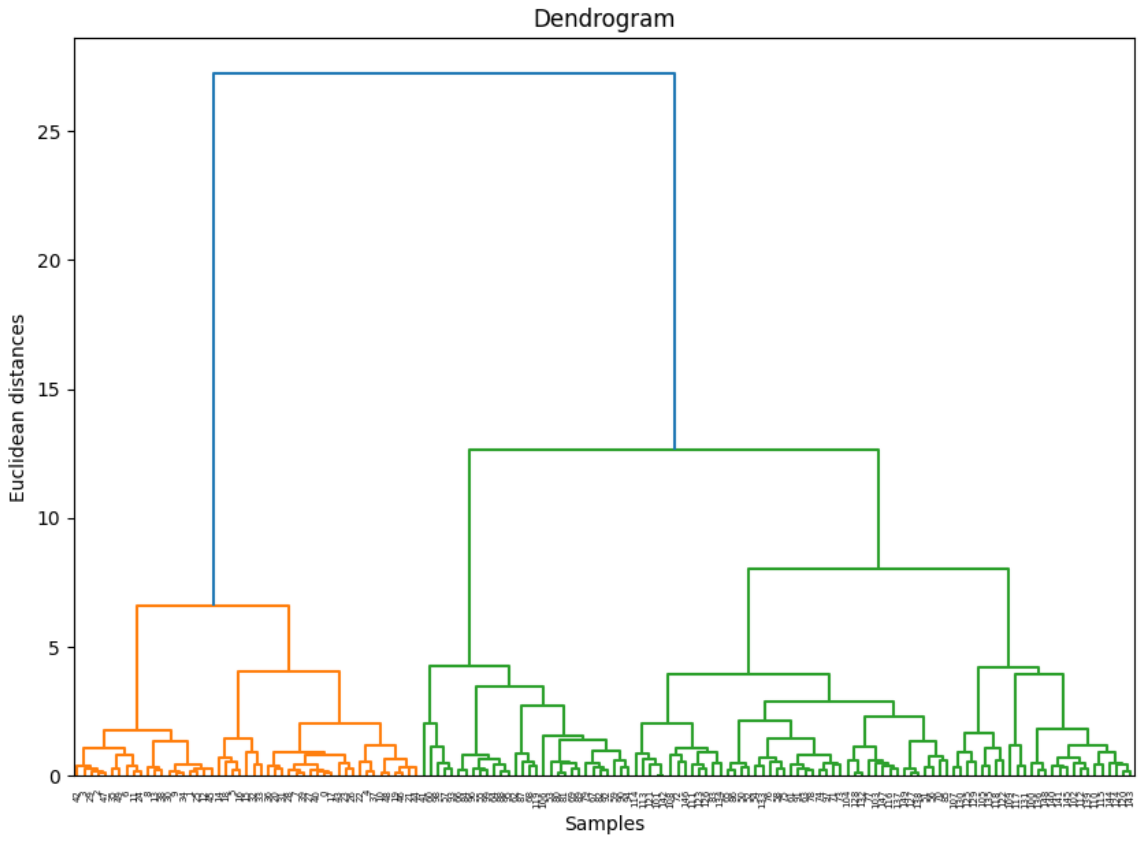
Exercise 5: Implementing Hierarchical Clustering

- Applied Agglomerative Clustering to the dataset used in K-Means.

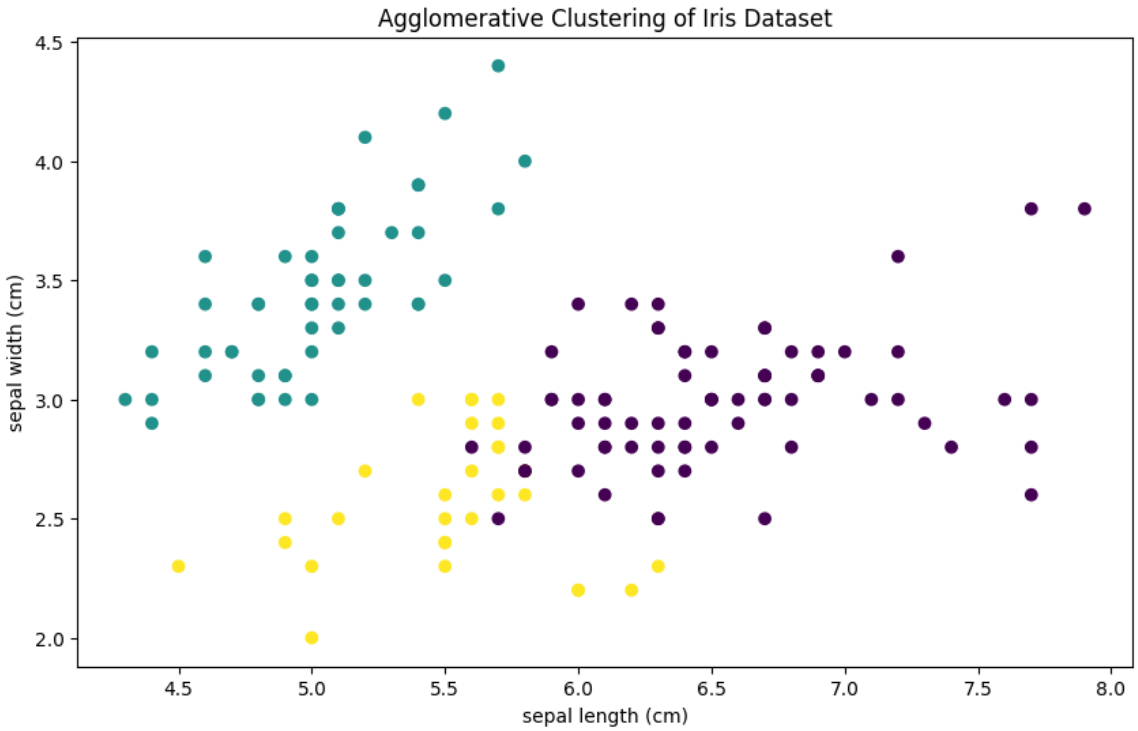
```
iris = load_iris()
X = iris.data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

agg_clustering = AgglomerativeClustering(n_clusters=3)
y_agg = agg_clustering.fit_predict(X_scaled)
```

- Visualized clustering hierarchy using a dendrogram.



- Compared with K-Means results



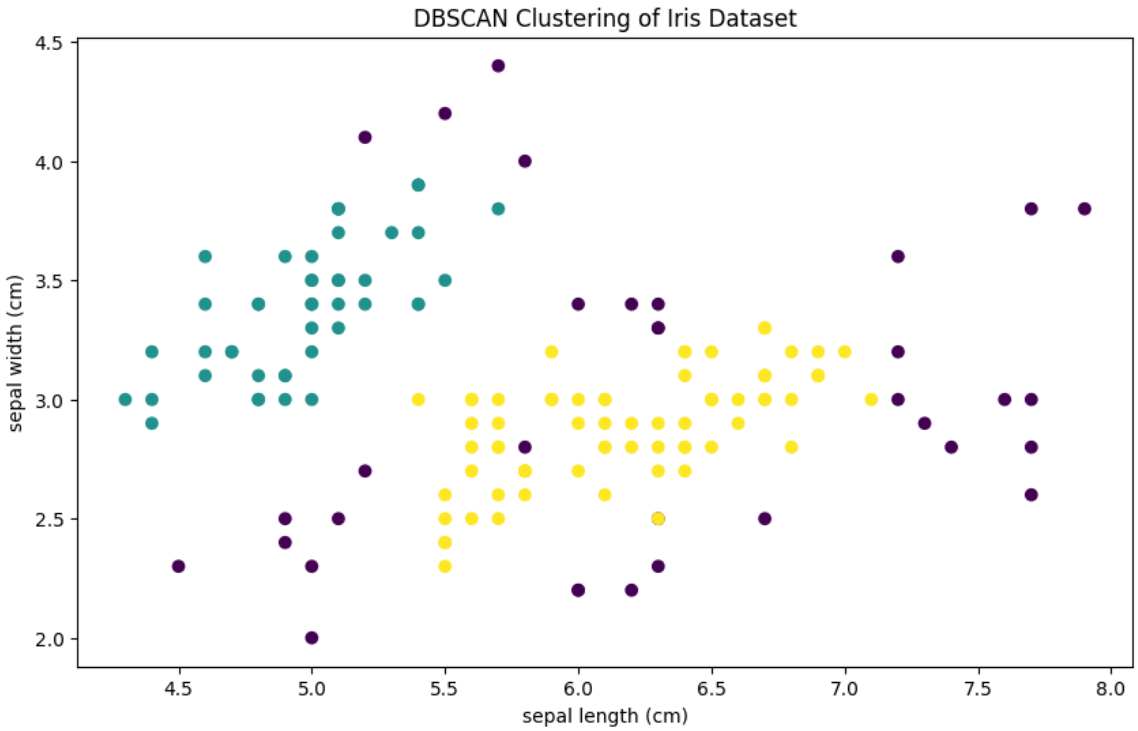
Exercise 6: Introduction to DBSCAN

- Implemented DBSCAN, adjusting parameters to control cluster formation.

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, min_samples=5)
data['dbscan_cluster'] = dbscan.fit_predict(X_scaled)
```

- Compared DBSCAN results with those from K-Means clustering.



Findings

In our analysis, we applied Agglomerative Clustering and DBSCAN to the Iris dataset and compared the results with K-Means clustering.

- Agglomerative Clustering:
  - Method: A hierarchical clustering technique that builds nested clusters by merging or splitting them successively.
  - Results: Identified 3 clusters, consistent with the known structure of the Iris dataset.
  - Comparison: Like K-Means, it effectively grouped the data into the three species of iris flowers.
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise):
  - Method: A density-based clustering algorithm that identifies clusters based on the density of data points.
  - Results: Detected 3 clusters with some noise points (outliers).
  - Comparison: Unlike K-Means, DBSCAN can identify outliers and does not require specifying the number of clusters. However, it also identified the three main clusters corresponding to the iris species.

Conclusion

This assignment enhanced understanding of classification and clustering techniques. The exercises demonstrated practical applications and helped in evaluating model performance and clustering quality.

Basic Classification Methods: Logistic Regression and KNN performed exceptionally well on the Iris dataset, including perfect accuracy, precision, recall, and F1-scores. This is a good sign, suggesting that even outside of this training data set, these models will be effective on this data set in real safety applications because all of them were able to distinguish the dataset associated vehicles effectively.

Clustering Techniques: The plot of the Elbow Method suggests that the most suitable number of clusters is 3 in the Iris data set, aligned with pre-defined separation into three types of Iris flowers and so determines how effective K-means clustering is in identifying patterns and groupings within the dataset.

Advanced Clustering Techniques: In comparison to K-Means, the other two methods of clustering, such as DBSCAN and Agglomerative Clustering, contributed to the clear identification of the latent model in the Iris dataset. As regards K-Means, it demands a user to ‘know’ a priori the number of the clusters, while DBSCAN reveals the data distribution and deals with any outliers present. Further Agglomerative Clustering offers a tree every time that helps understand the multi-level nature for of the clusters.

References

1. Scikit-learn documentation
2. Articles on classification and clustering methods
3. Online tutorials and resources for data mining techniques