Assignment №3

4th year student - Orisbay Eldana

Data Mining

15.10.2024

Introduction

- The relevance of classification and clustering in machine learning lies in their ability to analyze and make sense of data in different ways. **Classification** is crucial for tasks that require making predictions based on labeled data, allowing systems to automate decision-making processes, such as diagnosing diseases or categorizing emails as spam. **Clustering**, on the other hand, is essential for uncovering hidden patterns and relationships in unlabeled data, enabling insights like customer segmentation or anomaly detection. Together, these techniques help solve a wide range of real-world problems by organizing, categorizing, and identifying patterns in both structured and unstructured data.
- Objectives:
    - Explore Basic Classification Techniques**:** Implement and evaluate Logistic Regression and K-Nearest Neighbors (KNN) using Scikit-learn, focusing on accuracy, precision, recall, and F1-score.
    - Apply Clustering Techniques: Implement K-Means clustering, evaluate its performance using the Elbow Method, and compare results with other clustering methods like hierarchical clustering and DBSCAN.
    - Understand and Compare Advanced Clustering: Explore hierarchical clustering and DBSCAN, compare their results with K-Means, and analyze their effectiveness in unsupervised learning.
    - Enhance Machine Learning Knowledge**:** Gain hands-on experience with classification, clustering, and model evaluation techniques using real-world datasets
- Report Structure
  Title Page

    - Title of the Report
    - Student Name
    - Course Title
    - Date

  Table of Contents

    - Introduction
    - Basic Classification Methods
        - Overview
        - Exercises
        - Findings
    - Clustering Techniques
        - Overview
        - Exercises
        - Findings
    - Introduction to Advanced Clustering Techniques
        - Overview
        - Exercises
        - Findings
    - Conclusion
    - References

Basic Classification Methods

- **Overview:** Classification algorithms organize and understand complex datasets in machine learning. These algorithms are essential for categorizing data into classes or labels, automating decision-making and pattern identification. The importance of data is that classification makes data easier to locate and retrieve.
- **Exercises:**
    - I used the Iris dataset, which is readily available in Scikit-learn, where X holds the feature matrix, i.e., the measurements for the flowers (sepal length, petal length, etc.). y holds the labels (or target), which are the species of the flowers (0, 1, or 2 corresponding to setosa, versicolor, and virginica).
    - **train_test_split()**: This function splits the dataset into two parts:
      Training set: Used to train the machine learning model (70% of the data).
      Test set: Used to evaluate the model after training (30% of the data).
    - **LogisticRegression()**: Initializes a Logistic Regression model. **fit()**: Trains the Logistic Regression model on the training data.
    - **predict()**: Uses the trained Logistic Regression model to predict the labels of the test set.
      accuracy_score(): Computes the accuracy by comparing the predicted labels (y_pred) with the actual test labels (y_test).**Accuracy** measures how many predictions the model got correct out of the total number of predictions.

```
# Exercise 2: Confusion Matrix and Classification Report
# Orisbay Eldana
```

```
# Use the trained model to make predictions on the test set.
# Orisbay Eldana
y_pred_log_reg = log_reg.predict(X_test)
```

```
# Generate a confusion matrix and classification report using Scikit-learn.
# Orisbay Eldana
print(confusion_matrix(y_test, y_pred_log_reg))
print(classification_report(y_test, y_pred_log_reg))
```

```
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
# Discuss the results, focusing on precision, recall, and F1-score.
# Orisbay Eldana
# Precision: Measures how many of the predicted positive cases were actually positive.
# Recall: Measures how many of the actual positive cases were correctly predicted.
# F1-score: The harmonic mean of precision and recall, giving a single measure of a model's accuracy.
```

    - **confusion_matrix()**: Creates a confusion matrix, which provides a summary of prediction results. It shows the number of correct and incorrect predictions for each class.

- **classification_report()**: Generates a detailed report of precision, recall, F1-score, and support for each class. This gives a clearer picture of how well the model is performing, especially for imbalanced datasets (although the Iris dataset is balanced).

```
# Exercise 2: Confusion Matrix and Classification Report
# Orisbay Eldana
```

```
# Use the trained model to make predictions on the test set.
# Orisbay Eldana
y_pred_log_reg = log_reg.predict(X_test)
```

```
# Generate a confusion matrix and classification report using Scikit-learn.
# Orisbay Eldana
print(confusion_matrix(y_test, y_pred_log_reg))
print(classification_report(y_test, y_pred_log_reg))
```

```
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
# Discuss the results, focusing on precision, recall, and F1-score.
# Orisbay Eldana
# Precision: Measures how many of the predicted positive cases were actually positive.
# Recall: Measures how many of the actual positive cases were correctly predicted.
# F1-score: The harmonic mean of precision and recall, giving a single measure of a model's accuracy.
```

- **Findings:** Confusion Matrix:The model performed best on setosa with 100% accuracy.There were some misclassifications between versicolor and virginica, but overall, the model performed well on all classes.

  Classification Report**:** Precision**:** High for setosa (1.00), moderate for versicolor (0.82) and virginica (0.71). Recall: High for setosa (1.00) and virginica (0.83), moderate for versicolor (0.82). F1-score: 1.00 for setosa, 0.82 for versicolor, and 0.77 for virginica.

Clustering Techniques

- **Overview:** Clustering is an unsupervised machine learning technique that groups objects, data points, or observations into clusters or categories based on their similarities or patterns. Some common applications for clustering: Market segmentation, Social network analysis, Search result grouping, Medical imaging, Image segmentation.
- **Exercises:**
  - Loading Data**:** The Iris dataset was loaded using load_iris().
  - Fitting K-Means: The KMeans model was initialized with 3 clusters (since we expect 3 species in the Iris dataset) and fitted on the data (X).
  - Cluster Labels: After fitting the model, the cluster labels were retrieved using kmeans.labels_, which assigns each data point to one of the clusters.
  - Cluster Centers: The centroids of the clusters were obtained using kmeans.cluster_centers_.

- Visualization:
  A scatter plot was created with the first two features of the Iris data (X[:, 0] and X[:, 1]) to visualize the clustered data points.
  The cluster centers were overlaid in red 'X' markers to highlight the center of each cluster.

```
# 2. Clustering Techniques
# Orisbay Eldana
```

```
# Exercise 3: Implementing K-Means Clustering
# Orisbay Eldana
```

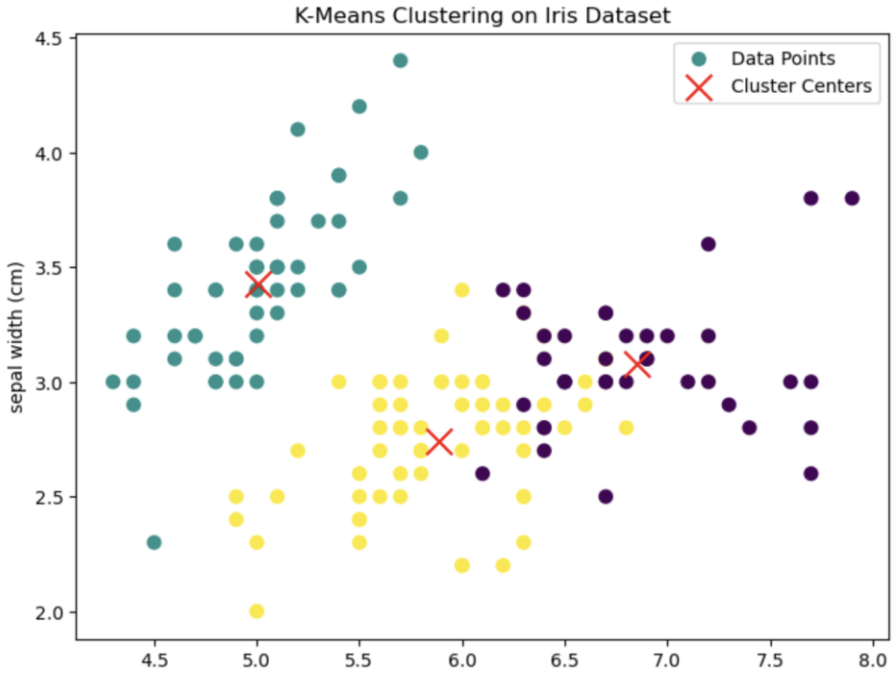
```
# Load a dataset (e.g., customer data or a toy dataset).
# Orisbay Eldana
iris = load_iris()
X = iris.data
```

```
# Use K-Means to group the data into clusters.
# Orisbay Eldana
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

```
cluster_labels = kmeans.labels_  # Correctly define cluster labels
cluster_centers = kmeans.cluster_centers_
```

```
# Visualize the clusters using a scatter plot.
# Orisbay Eldana
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, cmap='viridis', marker='o', s=50, label='Data Points')
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='red', marker='x', s=200, label='Cluster Centers')
plt.title('K-Means Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.legend()
plt.show()
```



- Inertia Calculation: The model was run for a range of cluster numbers (from 1 to 10). For each number of clusters, the inertia (sum of squared distances of samples to their nearest cluster center) was calculated.

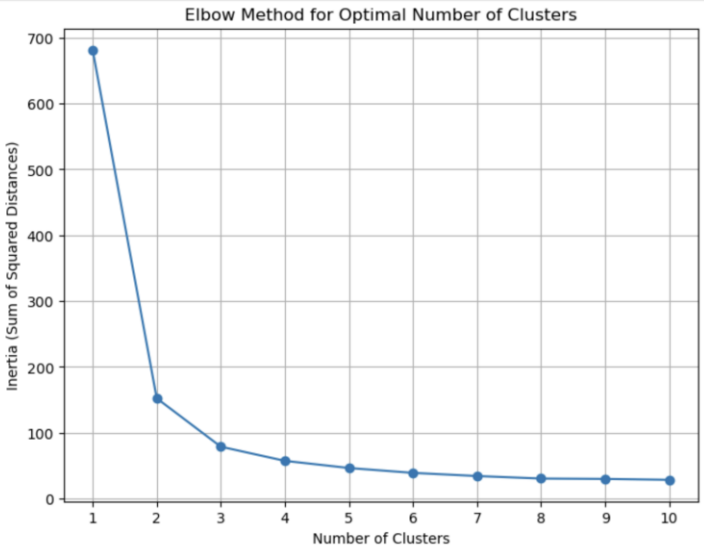
- Plotting: The inertia values were plotted against the number of clusters.

```
# Exercise 4: Evaluating K-Means Clustering
# Orisbay Eldana
```

```
# Calculate the inertia (sum of squared distances of samples to their closest cluster center).
# Orisbay Eldana
inertia_values = []
cluster_range = range(1, 11)  # Try 1 to 10 clusters
```

```
# Experiment with different numbers of clusters and plot the inertia to find the optimal number of clusters (Elbow Method).
# Orisbay Eldana
for n_clusters in cluster_range:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X)
    inertia_values.append(kmeans.inertia_)
```

```
plt.figure(figsize=(8, 6))
plt.plot(cluster_range, inertia_values, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia (Sum of Squared Distances)')
plt.xticks(cluster_range)
plt.grid(True)
plt.show()
```



[104]:
```
# Discuss the results.
# Orisbay Eldana

# Inertia: As we increase the number of clusters, inertia decreases since more clusters mean data points are closer to their cluster centers.
# However, after a certain point, the decrease in inertia becomes negligible.

# Elbow Method: From the inertia plot, the "elbow" point indicates the optimal number of clusters.
# This is the point where adding more clusters does not significantly improve the fit, as inertia reduction flattens out.
```

- **Findings:** The clusters are fairly well-separated, with data points grouped around their cluster centers. However, some overlap exists, particularly between versicolor and virginica. Overall, K-Means does a good job of capturing the natural divisions in the Iris dataset, though some misclassifications may occur. Elbow Method**:** The elbow point at 3 clusters shows that adding more clusters does not significantly improve the model, confirming that 3 is the optimal number of clusters for this dataset.

Introduction to Advanced Clustering Techniques

- **Overview:** Hierarchical clustering is a method that builds a hierarchy of clusters by either merging small clusters (agglomerative) or splitting large clusters (divisive). It creates a tree-like structure called a dendrogram that visually represents the nested groupings of data points. Density-Based Spatial Clustering of Applications with Noise

(DBSCAN) groups together points that are close to each other based on a distance threshold and a minimum number of points required to form a cluster.

- **Exercises:**
    - AgglomerativeClustering(n_clusters=3) was used to fit the data.
    - The fit_predict() function was applied to generate cluster labels.
    - A **dendrogram** was plotted using linkage(X, method='ward'), showing how the data points are merged into clusters.

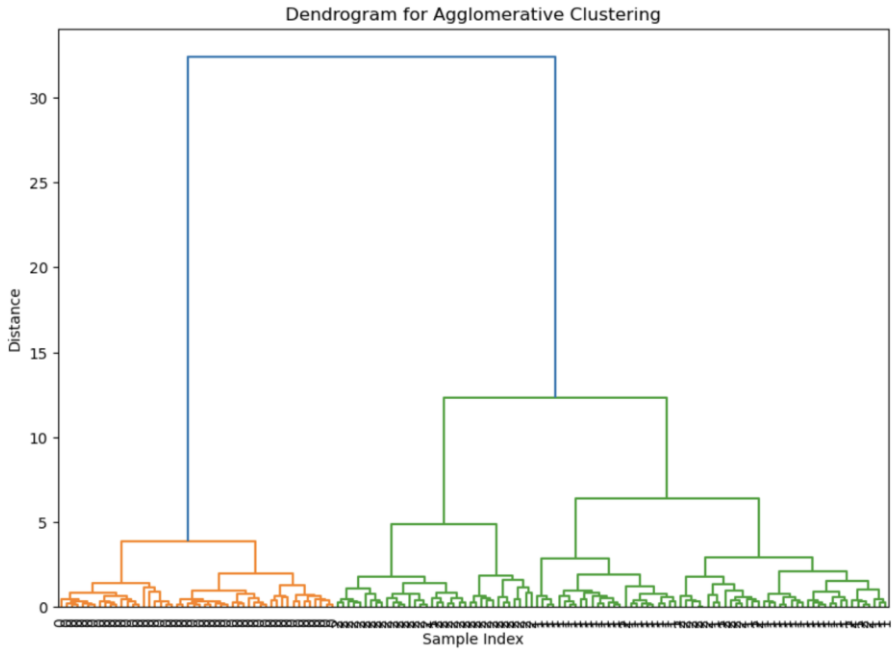
```
# 3. Introduction to Advanced Clustering Techniques
```

```
# Exercise 5: Implementing Hierarchical Clustering
# Orisbay Eldana
```

```
# Apply Agglomerative Clustering to the same dataset used in K-Means.
# Orisbay Eldana
agg_clustering = AgglomerativeClustering(n_clusters=3)
agg_labels = agg_clustering.fit_predict(X)
```

```
# Visualize the results using a dendrogram to illustrate the clustering hierarchy.
# Orisbay Eldana
linked = linkage(X, method='ward')  # 'ward' minimizes variance between clusters

plt.figure(figsize=(10, 7))
dendrogram(linked, labels=iris.target, leaf_rotation=90, leaf_font_size=10)
plt.title('Dendrogram for Agglomerative Clustering')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()
```



[133]:
```
# Compare the results with those from K-Means.
# Orisbay Eldana

# Hierarchical Clustering is based on grouping similar data points step by step.
# The dendrogram shows how clusters are merged at each step, and you can visually inspect how different samples cluster together.

# Comparison with K-Means: K-Means tends to partition the data into predefined clusters,
# while hierarchical clustering does not require a fixed number of clusters at the start,
# allowing you to observe different levels of clustering.
```

    - DBSCAN(eps=0.5, min_samples=5) was used to fit the data, where eps is the distance threshold and min_samples is the minimum number of points to form a cluster.

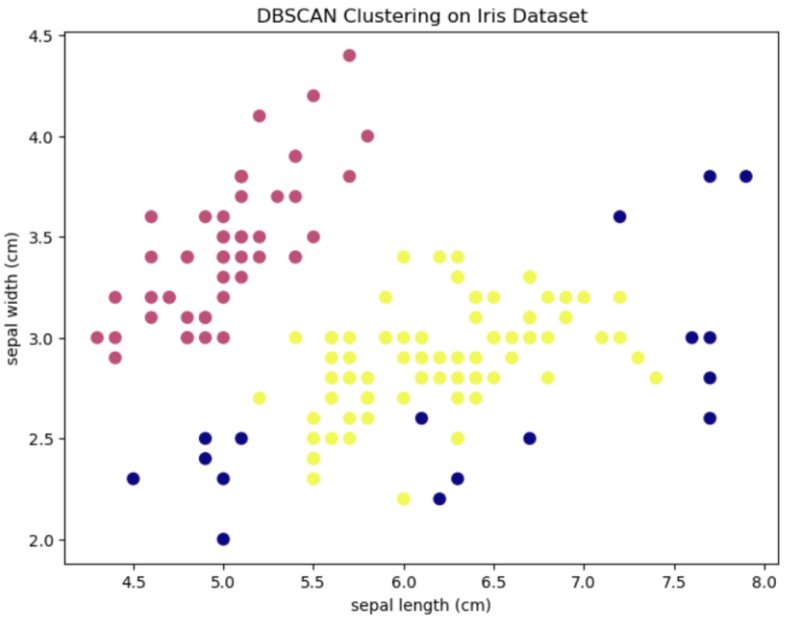
- The fit_predict() function was applied to generate cluster labels.
- A scatter plot was created to visualize the clusters, with the colors representing different cluster labels.

```
# Exercise 6: Introduction to DBSCAN
# Orisbay Eldana
```

```
# Implement DBSCAN on the dataset and visualize the clusters.
# Orisbay Eldana
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X)

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=dbscan_labels, cmap='plasma', marker='o', s=50)
plt.title('DBSCAN Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()

print(f"DBSCAN Labels: {np.unique(dbscan_labels)}")
```




```
DBSCAN Labels: [-1  0  1]
```

[149]:
```
# Discuss the parameters (eps and min_samples) and their effect on clustering results.
# Orisbay Eldana
# DBSCAN can handle clusters of arbitrary shapes and identify noise (outliers) by assigning them the label -1.
# This is different from K-Means, which assumes spherical clusters and forces all data points into clusters.

# Effect of Parameters:
# Increasing eps leads to larger clusters, as more points will be included in a neighborhood.
# Reducing min_samples makes it easier to form clusters, potentially detecting more clusters with smaller core points.
```

[147]:
```
# Compare the performance of DBSCAN with K-Means.
# Orisbay Eldana
# K-Means is sensitive to the initial number of clusters and works best with spherical clusters,
# while DBSCAN is better suited for identifying clusters of varying densities and shapes.
# DBSCAN also detects outliers, which K-Means does not handle explicitly.
```

- **Findings:** Comparison with K-Means: K-Means tends to partition the data into predefined clusters, while hierarchical clustering does not require a fixed number of clusters at the start, allowing you to observe different levels of clustering.

  K-Means is sensitive to the initial number of clusters and works best with spherical clusters, while DBSCAN is better suited for identifying clusters of varying densities and shapes.DBSCAN also detects outliers, which K-Means does not handle explicitly.

Conclusion

- Basic Classification Methods**:**
  - Logistic Regression & KNN**:** These are fundamental algorithms for supervised learning. Logistic Regression is a linear model for categorical outcomes, while KNN assigns labels based on nearby data points.
  - Model Evaluation**:** Using metrics like precision, recall, and F1-score provides a deeper understanding of model performance beyond accuracy.

  Clustering Techniques**:**
  - K-Means**:** A popular clustering algorithm that groups data based on proximity to centroids. The Elbow Method helps find the optimal number of clusters by analyzing inertia.
  - Hierarchical Clustering**:** Builds a hierarchy of clusters, visualized by dendrograms, useful for finding clusters at different levels.
  - DBSCAN: A density-based algorithm that identifies clusters based on dense regions, effective for noisy datasets.

  Comparison of Clustering Algorithms**:**
  - K-Means vs. Hierarchical: K-Means is efficient but assumes spherical clusters, while hierarchical clustering is more flexible but computationally intensive.
  - K-Means vs. DBSCAN: DBSCAN works well for clusters of varying shapes and sizes and is less sensitive to noise, unlike K-Means.

- **Classification Techniques**:
  Classification algorithms like Logistic Regression or KNN are extensively used in healthcare for diagnosing diseases based on patient data. For example, Logistic Regression can help predict whether a patient has diabetes based on factors like age, weight, blood sugar levels, etc.
  Retailers use classification models to categorize customers into different segments, such as frequent buyers, casual shoppers, or first-time visitors. This segmentation allows for targeted marketing strategies.
  Clustering Techniques:
  Hierarchical clustering can be used to group similar products or users in recommendation systems. For example, users who exhibit similar behaviors can be clustered together to recommend products or services that other users in the same cluster have liked.
  Clustering methods are used in urban planning and city infrastructure development to group similar neighborhoods based on data such as traffic patterns, housing prices, and crime rates. This helps governments make data-driven decisions on resource allocation and urban development.

References

- https://www.digitalguardian.com/blog/what-data-classification-data-classification-definition#:~:text=Classification%20makes%20data%20easier%20to,%2C%20security%2C%20and%20regulatory%20compliance.