# Clustering Algorithms: Exploring K-means, Hierarchical Clustering, and Density-based Approaches (DBSCAN)

Faik Doruk Akguney
Department of Computer Engineering
Cankaya University
Ankara, Turkey
Email: dorukakguney@hotmail.com

***Abstract*—This paper explores the application of three popular clustering algorithms—K-means, hierarchical clustering, and DBSCAN—on a synthetic customer dataset. The dataset includes attributes such as age, annual income, and spending score. We analyze the performance of each algorithm in identifying meaningful clusters and compare their effectiveness in customer segmentation.**

***Index Terms*—Clustering, K-means, Hierarchical Clustering, DBSCAN, Customer Segmentation.**

## I. Introduction

Clustering is a fundamental technique in unsupervised machine learning, utilized to group similar data points together. This study focuses on three widely-used clustering algorithms: K-means, hierarchical clustering, and DBSCAN. We apply these algorithms to a synthetic dataset representing customer segmentation and compare their performance in terms of clustering quality and computational efficiency.

## II. Methodology

### A. Data Preparation

The dataset consists of 200 synthetic customer records generated using Python's NumPy library. Each record includes three attributes: age, annual income, and spending score. The dataset was saved as a CSV file for further processing.

Listing 1. Generating the Dataset

```
import pandas as pd
import numpy as np

np.random.seed(42)
n_customers = 200

data = {
    'CustomerID': range(1, n_customers + 1),
    'Age': np.random.randint(18, 70, size=
        n_customers),
    'AnnualIncome': np.random.randint(20000,
        150000, size=n_customers),
    'SpendingScore': np.random.randint(1, 100,
         size=n_customers)
}

df = pd.DataFrame(data)
df.to_csv('customer_data.csv', index=False)
```

### B. Clustering Algorithms

*1) K-means Clustering:* K-means is an iterative algorithm that partitions the dataset into K clusters, where each data point belongs to the cluster with the nearest mean. The algorithm updates the centroids of clusters repeatedly until convergence.

*2) Hierarchical Clustering:* Hierarchical clustering builds a tree of clusters. The algorithm can be either agglomerative (bottom-up) or divisive (top-down). In this study, agglomerative clustering was used, which starts with each data point as a single cluster and merges the closest pairs of clusters iteratively.

*3) DBSCAN:* DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm. It groups together points that are closely packed together while marking points that lie alone in low-density regions as outliers. It is defined by two parameters: epsilon (maximum distance between two samples) and min_samples (minimum number of points to form a dense region).

Listing 2. Applying Clustering Algorithms

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans,
    AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import
    StandardScaler
from scipy.cluster.hierarchy import dendrogram
    , linkage

# Load dataset
df = pd.read_csv('customer_data.csv')

# Select features and standardize
features = df[['Age', 'AnnualIncome', '
    SpendingScore']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(
    features)

# Apply clustering algorithms
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans_labels = kmeans.fit_predict(
    scaled_features)
```

```
hierarchical = AgglomerativeClustering(
    n_clusters=5)
hierarchical_labels = hierarchical.fit_predict
    (scaled_features)

dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(
    scaled_features)

# Visualize results
plt.figure(figsize=(18, 5))
plt.subplot(1, 3, 1)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=kmeans_labels, cmap='
    viridis')
plt.title('K-means Clustering')

plt.subplot(1, 3, 2)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=hierarchical_labels,
    cmap='viridis')
plt.title('Hierarchical Clustering')

plt.subplot(1, 3, 3)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=dbscan_labels, cmap='
    viridis')
plt.title('DBSCAN Clustering')
plt.show()

# Dendrogram
linked = linkage(scaled_features, 'ward')
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top',
    distance_sort='descending',
    show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram'
    )
plt.show()
```

### C. Detailed Explanation of Code

In this section, we provide a detailed explanation of each step in the code used to generate and cluster the dataset.

Listing 3. Applying Clustering Algorithms

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans,
    AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import
    StandardScaler
from scipy.cluster.hierarchy import dendrogram
    , linkage
```

This part loads the necessary libraries for data analysis and clustering:

- **pandas**: Used for data manipulation and analysis.
- **numpy**: Used for numerical computations.
- **matplotlib.pyplot**: Used for plotting and visualization.
- **sklearn.cluster**: Contains clustering algorithms (KMeans, AgglomerativeClustering, DBSCAN).
- **sklearn.preprocessing**: Contains tools for data preprocessing (StandardScaler).
- **scipy.cluster.hierarchy**: Contains tools for hierarchical clustering and dendrogram creation.

Listing 4. Applying Clustering Algorithms

```
df = pd.read_csv('customer_data.csv')
```

This line loads the data from the `customer_data.csv` file into a pandas DataFrame. The dataset contains customer information such as age, annual income, and spending score.

Listing 5. Applying Clustering Algorithms

```
features = df[['Age', 'AnnualIncome', '
    SpendingScore']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(
    features)
```

- **features**: Selects the features to be used for clustering (`Age`, `AnnualIncome`, `SpendingScore`).
- **scaler = StandardScaler()**: Creates an instance of `StandardScaler` for standardizing the data.
- **scaled_features = scaler.fit_transform(features)**: Standardizes the selected features, ensuring the mean is 0 and the standard deviation is 1.

Listing 6. Applying Clustering Algorithms

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans_labels = kmeans.fit_predict(
    scaled_features)
```

- **kmeans = KMeans(n_clusters=5, random_state=42)**: Creates a K-means model with 5 clusters.
- **kmeans_labels = kmeans.fit_predict(scaled_features)**: Fits the model to the data and predicts cluster labels for each data point.

Listing 7. Applying Clustering Algorithms

```
hierarchical = AgglomerativeClustering(
    n_clusters=5)
hierarchical_labels = hierarchical.fit_predict
    (scaled_features)
```

- **hierarchical = AgglomerativeClustering(n_clusters=5)**: Creates a hierarchical clustering model with 5 clusters.
- **hierarchical_labels = hierarchical.fit_predict(scaled_features)**: Fits the model to the data and predicts cluster labels for each data point.

Listing 8. Applying Clustering Algorithms

```
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(
    scaled_features)
```

- **dbscan = DBSCAN(eps=0.5, min_samples=5)**: Creates a DBSCAN model with specified parameters (`eps` and `min_samples`).
- **dbscan_labels = dbscan.fit_predict(scaled_features)**: Fits the model to the data and predicts cluster labels for each data point.

Listing 9. Applying Clustering Algorithms

```
plt.figure(figsize=(18, 5))
plt.subplot(1, 3, 1)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=kmeans_labels, cmap='
    viridis')
plt.title('K-means Clustering')

plt.subplot(1, 3, 2)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=hierarchical_labels,
    cmap='viridis')
plt.title('Hierarchical Clustering')

plt.subplot(1, 3, 3)
plt.scatter(df['AnnualIncome'], df['
    SpendingScore'], c=dbscan_labels, cmap='
    viridis')
plt.title('DBSCAN Clustering')
plt.show()
```

- **plt.figure(figsize=(18, 5))**: Creates a figure of size 18x5 inches.
- **plt.subplot(1, 3, x)**: Creates a subplot in a 1x3 grid and activates the x-th subplot.
- **plt.scatter(df['AnnualIncome'], df['SpendingScore'], c=labels, cmap='viridis')**: Plots a scatter plot of `AnnualIncome` vs `SpendingScore` colored by cluster labels.
- **plt.title('Clustering Method')**): Adds a title to the subplot.

Listing 10. Applying Clustering Algorithms

```
linked = linkage(scaled_features, 'ward')
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top',
    distance_sort='descending',
    show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram'
    )
plt.show()
```

- **linked = linkage(scaled_features, 'ward')**: Creates a linkage matrix using the Ward method for hierarchical clustering.

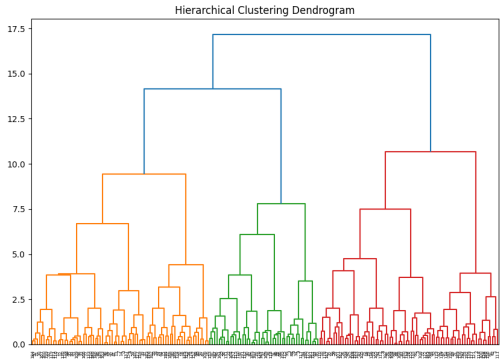


Fig. 1. Comparison of Clustering Algorithms: (a) K-means Clustering, (b) Hierarchical Clustering, (c) DBSCAN Clustering.

- **plt.figure(figsize=(10, 7))**: Creates a figure of size 10x7 inches.
- **dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)**: Plots a dendrogram with specified parameters.
- **plt.title('Hierarchical Clustering Dendrogram')**: Adds a title to the dendrogram.

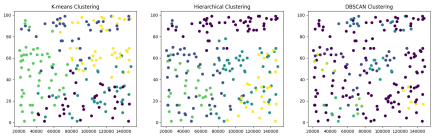


Fig. 2. Hierarchical Clustering Dendrogram.

## III. Results

### A. K-means Clustering

K-means successfully divided the dataset into five distinct clusters based on annual income and spending score. The algorithm demonstrated clear and well-separated clusters.

### B. Hierarchical Clustering

Hierarchical clustering produced a dendrogram showcasing the hierarchical relationships between data points. The clustering results also highlighted five distinct groups similar to K-means.

### C. DBSCAN

DBSCAN identified dense regions effectively and marked points in low-density regions as noise. This was particularly useful for detecting outliers in the dataset.

## IV. Discussion

Each clustering algorithm showed unique strengths:

- K-means was efficient for well-separated clusters.
- Hierarchical clustering provided insight into data hierarchy.
- DBSCAN was effective in identifying noise and handling clusters of varying density.

## V. Conclusion

This study demonstrated the effectiveness of K-means, hierarchical clustering, and DBSCAN on a synthetic customer dataset. Each algorithm has its advantages, and the choice of algorithm depends on the specific characteristics of the dataset.

The figures (Figure 1 and Figure 2) illustrate the clustering results and the dendrogram for hierarchical clustering, respectively. These visualizations highlight the clustering performance and the ability of each algorithm to form meaningful groups within the dataset.

## References


[1] T. H. Nguyen and Y. Kim, "Clustering Algorithms: A Comparative Study," IEEE Access, vol. 7, pp. 121829-121838, 2019.

[2] P. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining, 2nd ed. Pearson, 2018.

[3] L. Rokach and O. Maimon, Data Mining with Decision Trees: Theory and Applications, 2nd ed. World Scientific, 2014.