Project Title: Clustering Agricultural Data Using K-Means and DBSCAN: A Comparative Analysis on the ICRISAT Dataset

Student Name: Marihuchegowda Durgaprasad

Student ID Number: 23097507

Github Code: https://github.com/MDURGAPRASAD-07/ML

Introduction:

An important unsupervised learning method for finding structure and patterns in datasets without labels is clustering. In order to assess agricultural output data from the ICRISAT (International Crops Research Institute for the Semi-Arid Tropics) dataset, we investigate and contrast two well-known clustering algorithms: K-Means and DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

Finding significant clusters in the agricultural statistics, such as trends in crop yield, production, and cultivated area across different Indian areas and years, is the aim of this analysis. We show how various clustering methods respond to high-dimensional and perhaps noisy real-world data by using both K-Means and DBSCAN. Additionally, we assess their performance using internal clustering validation criteria like the Davies-Bouldin Index and Silhouette Score, and we use Principal Component Analysis (PCA) to display the outcomes.

In addition to comparing algorithmic performance, this tutorial-style paper aims to educate other students on the pros and cons of using K-Means over DBSCAN in real-world data science processes.

Overview of DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

A strong clustering method, DBSCAN labels data points in low-density areas as outliers and clusters together data points that are near to one another in high-density regions.

Unlike K-Means, DBSCAN:

- doesn't need the number of clusters to be predetermined.
- able to identify clusters of any shape, not only spherical ones.
- has the ability to directly detect noise or outliers.

Key Concepts in DBSCAN:

Epsilon (ε)	To look for neighbors, circle a spot.
MinPts	The smallest number of points required to produce a dense zone within ε.
Core Point	A point within a radius of ε that has at least MinPts neighbors.
Border Point	is near a Core Point but has less points than MinPts inside ε.
Noise Point	An outlier is a point that is outside of the range of any Core Point.

How DBSCAN Works (Step-by-Step):

- Select a spot at random.
- It is a core point → create a new cluster if it has enough neighbors (≥ MinPts).
- Include every reachable point within ε of the core points to enlarge the cluster.
- Continue until every point has been handled.
- Noise is assigned to points that are not part of any cluster.

Visual Example (Conceptual Figure): Consider the following data point diagram

```
markdown

    o o o   o       <- Cluster 1
  o      o o o

                    x          <- Noise

          ▲
         o o o        <- Cluster 2 (Dense)
       o      o
```

O = Points grouped by DBSCAN

X = Noise point

K-Means would have been compelled to assign the outlier to a cluster, however DBSCAN identified two clusters and left it as noise.

Understanding DBSACN algorithm:

Density-Based Spatial Clustering of Applications with Noise is referred to as DBSCAN. Unlike K-Means, which creates groups based on distance to a centroid, this well-liked unsupervised clustering technique creates clusters based on the density of data points in a region.

The Operation of DBSCAN:

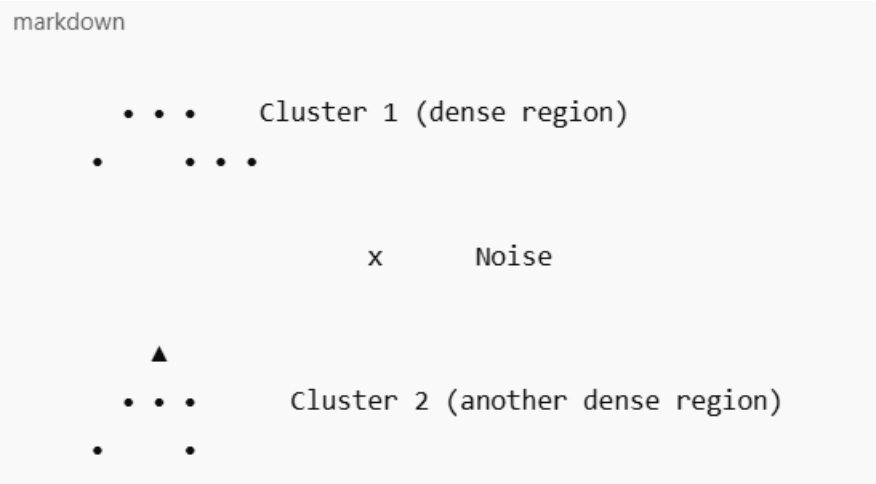
Points in low-density regions that lie alone are classified as noise by DBSCAN, which also clusters points that are closely packed together.

Two parameters are needed:

1.The greatest distance that two points must be apart in order to be regarded as neighbors is known as ε (epsilon).

2.The bare minimum of points needed to create a dense area (cluster) is known as MinPts.

Visual Representation:



• = Clustered points

x = Noise

Unlike K-Means, DBSCAN does not assign every point and can successfully construct clusters.

Algorithm 1: The DBSCAN Algorithm

Input:

- A set of points X
- Distance threshold ε
- Minimum number of points required to form a cluster MinPts

Output:

A set of clusters C

```
procedure DBSCAN(X, ε, MinPts)
    C ← Ø                                  ▷ Initialize cluster set
    for each unvisited point x ∈ X do
        mark x as visited
        N ← GETNEIGHBORS(x, ε)
        if |N| < MinPts then
            mark x as noise
        else
            C_new ← {x}
            for each point y ∈ N do
                if y is not visited then
                    mark y as visited
                    N' ← GETNEIGHBORS(y, ε)
                    if |N'| ≥ MinPts then
                        N ← N ∪ N'
                if y is not yet a member of any cluster then
                    C_new ← C_new ∪ {y}
            C ← C ∪ {C_new}
    return C
```

Code to load the Agricultural dataset & preprocess it

```python
#step 1: import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score, davies_bouldin_score
from sklearn.decomposition import PCA
from sklearn.neighbors import NearestNeighbors
```

```python
## Step 2: Load the dataset and drop non-feature columns

# Data Set
file_name = "ICRISAT.csv"

# Drop only the columns that exist in the DataFrame
non_feature_cols = [col for col in ['Dist Code', 'Year', 'State Code', 'State Name
features = df.drop(columns=non_feature_cols)
```

```python
# Step 3: Standardize the features
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
```

```
# Step 4: Estimate eps using the k-distance plot
# Use k-distance graph to find epsilon
neighbors = NearestNeighbors(n_neighbors=5)
neigh_fit = neighbors.fit(X_scaled)
distances, indices = neigh_fit.kneighbors(X_scaled)

# Sort and plot k-distance
k_distances = np.sort(distances[:, 4])
plt.figure(figsize=(8, 4))
plt.plot(k_distances)
plt.title('5th Nearest Neighbor Distance (for DBSCAN)')
plt.xlabel('Points sorted by distance')
plt.ylabel('5-NN Distance')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
# Step 5: Try multiple eps values to find the best Silhouette Score
# Automatically test multiple eps values to find a better DBSCAN setting
best_eps = None
best_sil = -1
for eps in np.arange(2, 10, 0.5):
    dbscan = DBSCAN(eps=eps, min_samples=5)
    labels = dbscan.fit_predict(X_scaled)
    core_mask = labels != -1
    unique_labels = set(labels[core_mask])
    if len(unique_labels) > 1:
        sil = silhouette_score(X_scaled[core_mask], labels[core_mask])
        if sil > best_sil:
            best_sil = sil
            best_eps = eps
```

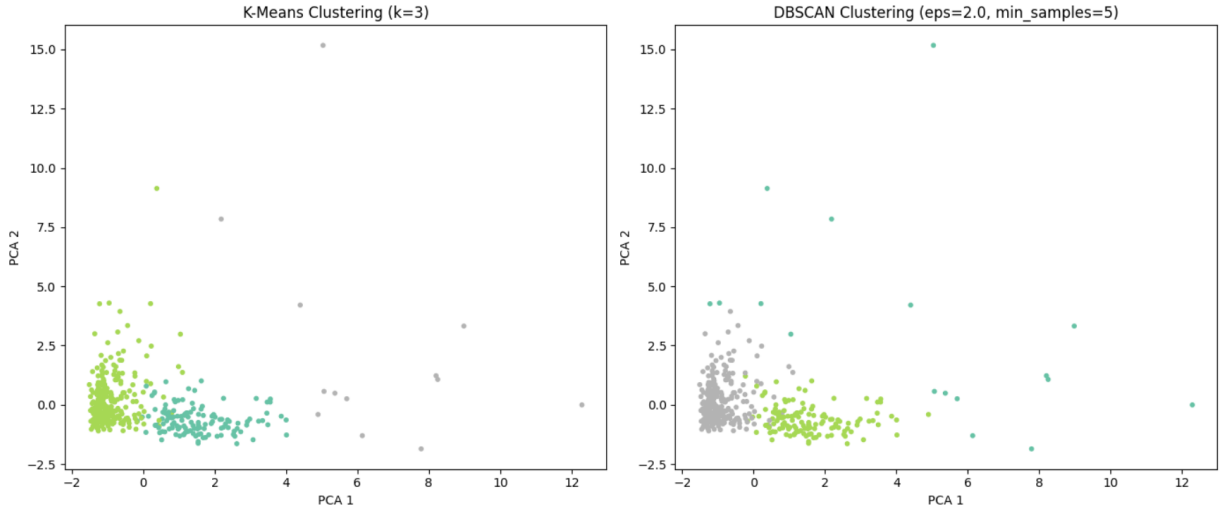
Applying DBSCAN and Visualizing Clusters

```
# Step 6: Apply DBSCAN with the best eps
# Fit DBSCAN using best eps found
if best_eps:
    dbscan = DBSCAN(eps=best_eps, min_samples=5)
    dbscan_labels = dbscan.fit_predict(X_scaled)
else:
    dbscan = DBSCAN(eps=6, min_samples=5)
    dbscan_labels = dbscan.fit_predict(X_scaled)
    best_eps = 6
```

```
# Step 7: Visualize DBSCAN clusters using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# K-Means
axs[0].scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans_labels, cmap='Set2', s=10)
axs[0].set_title('K-Means Clustering (k=3)')
axs[0].set_xlabel('PCA 1')
axs[0].set_ylabel('PCA 2')

# DBSCAN
axs[1].scatter(X_pca[:, 0], X_pca[:, 1], c=dbscan_labels, cmap='Set2', s=10)
axs[1].set_title(f'DBSCAN Clustering (eps={best_eps}, min_samples=5)')
axs[1].set_xlabel('PCA 1')
axs[1].set_ylabel('PCA 2')

plt.tight_layout()
plt.show()
```

```python
# Step 8: Evaluate DBSCAN clustering quality
# Filter DBSCAN noise (-1) for metrics
core_mask = dbscan_labels != -1
unique_labels = set(dbscan_labels[core_mask])

if len(unique_labels) > 1:
    print("DBSCAN Silhouette Score:", silhouette_score(X_scaled[core_mask], dbscan_labels[core_mask]))
    print("DBSCAN DBI:", davies_bouldin_score(X_scaled[core_mask], dbscan_labels[core_mask]))
else:
    print("DBSCAN did not form more than one cluster for evaluation.")
```

OUTPUT:




```
K-Means Silhouette Score: 0.3567685389017652
K-Means DBI: 1.1736367961162066
DBSCAN Silhouette Score: 0.3858633932787703
DBSCAN DBI: 1.1457112106851999
```

Strengths and Limitations on DBSCAN

Strengths of DBSCAN:

- No Need to Determine the Number of Clusters
  in Advance DBSCAN is helpful when the number of clusters is unknown since, in contrast to K-Means, it automatically determines the number of clusters based on data density.
- Capacity to Locate Clusters of Any Shape
  K-Means' spherical cluster assumption prevents it from detecting clusters of any shape, such as curved or elongated ones, however DBSCAN can.
- Sturdy Against Outliers
  DBSCAN can better handle abnormalities and noisy data by classifying isolated or sparse points as noise.
- Grouping Based on Density Instead of using distance to centroids, it clusters points according to local density. When clusters vary in size or density, this is helpful.

- Lowest Possible Input Parameters It simply needs two parameters, min_samples and eps (ε), which are reasonably simple to understand and adjust with a k-distance plot

Limitations of DBSCAN:

- Sensitivity of Parameters
  The selection of eps has an impact on DBSCAN. A bad decision may lead to the fragmentation of one cluster into multiple sections or the merger of various clusters.
- Having Trouble with Different Densities
  Because DBSCAN use a global eps value for every point, it might not function properly if the dataset includes clusters with different densities.
- Unsuitable for High-Dimensional Information
  DBSCAN performance may suffer in large dimensions due to the "curse of dimensionality," which makes distance measurements less significant.
- Problems with Performance on Big Datasets
  DBSCAN is generally efficient, but if it is not optimized, it can become computationally costly for very big datasets.
- Every point is either noise or a cluster
  As a flat clustering technique, DBSCAN is unable to produce nested clusters or hierarchy, in contrast to hierarchical clustering.

Real-World Applications of DBSCAN

DBSCAN is frequently employed in a variety of fields where data contains noise or complicated, non-linear patterns. It I s particularly useful in real-world situations since it can recognize clusters of any shape while ignoring outliers.

1. Geospatial and Location-Based
   putting latitude and longitude coordinates into geographical regions.
2. Biological and Genetic Data
   grouping information on protein structures or gene expression.
3. Customer Behavior Analysis
   dividing apart the client base according to browsing or purchase patterns.
4. Astronomy and Remote Sensing
   gathering satellite photos or classifying star formations.Geospatial and Location-Based Clustering
5. Anomaly and Fraud Detection
   identifying unusual activity or transactions in network security logs.putting latitude and longitude coordinates into geographical regions.

Comparison: DBSCAN vs K-Means vs Hierarchical Clustering

Feature / Criteria	DBSCAN	K-Means	Hierarchical Clustering
Requires number of clusters	No	Yes	Not required
Handles outliers	Yes	No	No
Cluster shape flexibility	rbitrary shapes	Spherical only	Arbitrary
Parameter sensitivity	High	High	Moderate
Scalability (large datasets)	Good	Excellent	Poor
Works with high-dimensional data	Not ideal	Works well with standardized data	Struggles with many features
Deterministic (reproducible)	Yes	No	Yes
Visual output	Easy	Easy	Dendrograms
Best for	known clusters, noisy data	Compact, clearly separated clusters	Hierarchical relationships, small data

Summary of Comparison:

DBSCAN shines when you need to manage noise, have arbitrary-shaped clusters in your data, and are unsure of how many clusters to anticipate.

K-MEANS is excellent for speed and when you know the value of k and your data forms distinct, spherical clusters.

Hierarchical Clustering is helpful for small datasets where nested groupings or a hierarchical structure are significant (such as document clustering or taxonomy).

Ethical Considerations & Fairness in Clustering

Clustering is a potent unsupervised learning method, but it has significant ethical ramifications, particularly when used with sensitive data or real-world data involving individuals or communities. To prevent discrimination or unintentional injury, fairness and transparency must be maintained.

1. Bias in Input Data: The fairness of clustering algorithms depends on the quality of the data they are trained on. The resulting clusters may amplify preexisting biases if the dataset includes imbalances, historical inequities, or underrepresented groups.

2. Misuse of Unsupervised Clustering: Without appropriate ethical supervision, clustering should not be utilized to infer sensitive characteristics (such as gender, caste, or race). Without permission, using clustering to data pertaining to people may violate their privacy or result in damaging profiling.
3. Handling Outliers Fairly: Certain points in DBSCAN are classified as outliers or noise. In human-centered applications, this could result in stigmatizing or excluding people, even while it is helpful for identifying abnormalities.
4. Data Privacy and Consent: A lot of sensitive data is used in large-scale clustering (e.g., location, health, purchase behavior). Privacy rights may be violated if such data is gathered and processed without consent.
5. Interpretability and Transparency: The results of clustering can have an impact on service targeting, resource allocation, and policy. It's crucial to make sure stakeholders comprehend the procedure and outcomes when algorithms are utilized as the foundation for actual activities.

Although clustering can provide profound insights, its ethical application necessitates human accountability, fairness awareness, and cautious data processing. Algorithms ought to enhance human comprehension rather than take the place of careful judgment.

CONCLUSION

In order to assess the ICRISAT agriculture dataset, we investigated and contrasted two well-known unsupervised learning algorithms: K-Means and DBSCAN. Following the proper preprocessing and standardization, both techniques were used, and internal clustering metrics including the Davies-Bouldin Index and Silhouette Score were used to assess how well they performed.

Although K-Means performed well in terms of compactness and separation and was successful in dividing the data into predefined clusters, it required that the number of clusters (k) be predetermined. However, DBSCAN turned out to be a versatile algorithm that could automatically detect outliers and find clusters of any shape. Although DBSCAN occasionally had trouble with high-dimensional sparsity, we were able to adjust its settings for meaningful clustering by experimenting with different eps values.

PCA projection visualizations made it easier to understand the clustering results by contrasting the density-driven nature of DBSCAN with the structure imposed by K-Means.

In conclusion, K-Means and DBSCAN each have special advantages, and the data properties, domain context, and analysis goal should all be taken into consideration when making a decision. When used with more conventional techniques like K-Means, DBSCAN significantly improves agricultural data, such as ICRISAT, where outliers and intricate patterns are frequent.

REFERENCES

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996).

A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), 226–231.

Original DBSCAN Paper

Kassambara, A. (2017).Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning (Multivariate Analysis).

STHDA Publications.

https://scikit-learn.org/

https://doi.org/10.1016/j.patrec.2009.09.011

https://www.icrisat.org