K-Means Clustering: technique in unsupervised machine learning

Student Number: 23079366
GitHub: https://github.com/Upender430/K-Means-clustering.git

---

1. Introduction

Clustering is one of the most powerful techniques in unsupervised machine learning. Unlike supervised learning, where we use labels to teach a model, clustering helps us find natural groupings in data without any labels.

Among the different clustering techniques, **K-Means** is one of the simplest and most popular. It divides data into k distinct clusters based on similarity. This technique is widely used in customer segmentation, image compression, pattern recognition, and more.

In this tutorial, we'll walk through:

- How K-Means works (with visual intuition)
- Applying K-Means to a 2D dataset
- Using K-Means to compress an image by reducing its colors

All code and examples are written in Python using Scikit-learn, NumPy, and Matplotlib.

---

2. How K-Means Works

K-Means follows a simple cycle of steps:

1. **Initialization**: Randomly choose k cluster centers (called centroids).
2. **Assignment**: Assign each data point to the nearest centroid.
3. **Update**: Recalculate the centroids as the mean of the points assigned to each cluster.
4. **Repeat**: Continue steps 2 and 3 until the centroids don't change significantly.

This process helps the algorithm converge to a solution where the intra-cluster distances are minimized.

K-Means starts with selecting k random centroids. Then, each data point is assigned to the closest centroid using Euclidean distance. After the assignment, the centroids are updated as the mean of the assigned points. These two steps—assignment and update—repeat until convergence.

What's really happening here is that the algorithm is trying to minimize the total distance within each cluster. The final result is a grouping where the data within a cluster is more similar to each other than to points in other clusters.

Choosing the Right Number of Clusters

One of the most common challenges is selecting the right number of clusters (k). A widely used approach is the Elbow Method. We plot the sum of squared errors (inertia) for different values of k and look for the "elbow point" where the reduction slows down.

Another method is the Silhouette Score, which measures how similar a point is to its own cluster compared to other clusters. Higher scores indicate better-defined clusters.

Mathematical Formulation of K-Means

K-Means aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. Mathematically, the goal is to minimize the following objective function:

$J = \sum\sum ||xi - \mu j||^2$

Where:
- xi is a data point,
- μj is the centroid of cluster j,
- $||xi - \mu j||^2$ is the squared Euclidean distance between the point and its assigned centroid.

The optimization is non-convex, meaning it can have multiple local minima. That's why different initial centroid positions can lead to different outcomes.

Understanding Inertia

Inertia is a measure of how internally coherent the clusters are. It calculates the sum of squared distances from each point to its assigned cluster center. Lower inertia indicates more compact clusters. However, inertia decreases with increasing k, which is why we use techniques like the elbow method to choose the optimal value of k.

Experiment: Testing Different Values of k

To further explore K-Means, I tested values of k from 1 to 10 using the Elbow Method. I plotted the inertia against k and noticed that after k=4, the drop in inertia was minimal. This helped me select a reasonable value for k.

This experiment was especially useful in understanding why selecting the right number of clusters is more of an art than an exact science. It depends on context, and sometimes domain knowledge is needed.

Experiment: Clustering Synthetic Data

To explore K-Means in a visual and intuitive way, I used **synthetic 2D data** generated using make_blobs from scikit-learn. This method creates data with clearly defined clusters, making it perfect for seeing how K-Means works in action.

When I set k=3, the algorithm effectively discovered the groupings and calculated the centroids. It was satisfying to see how the points were assigned based on distance, and how the cluster centers updated with each iteration.

This experiment helped me understand the core mechanics of K-Means—how it groups similar data even when there are no labels involved. It also showed how useful visualization can be for interpreting unsupervised learning results.

Comparison with Other Clustering Methods

Although K-Means is widely used, it's not the only clustering technique. I briefly compared it with DBSCAN and Hierarchical Clustering:

- DBSCAN is better at finding clusters of arbitrary shape and is robust to outliers.
- Hierarchical clustering builds a dendrogram and doesn't require k to be defined in advance.

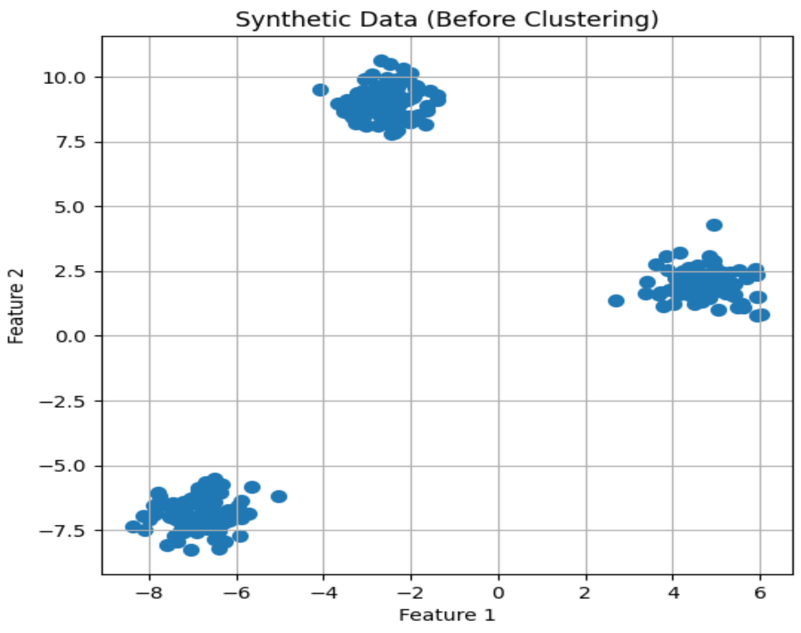
Each method has its strengths, but for simple, well-separated data, K-Means remains a great first choice.

3. Visual Demo on 2D Data

To get a hands-on feel for K-Means, we first create synthetic 2D data using make_blobs from Scikit-learn. This gives us clusters that are easy to visualize.

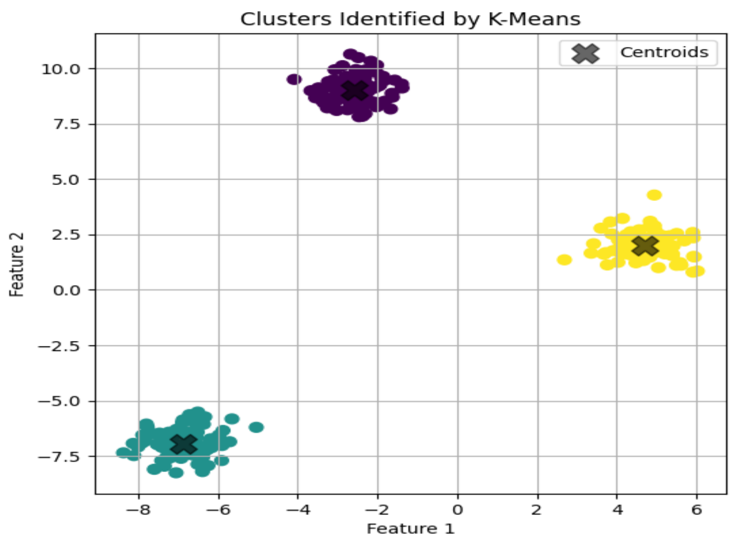
- **Before Clustering**

The plot below shows randomly generated data points that we'll cluster:



◆ After Clustering

Once we apply K-Means with k=3, the algorithm finds three clusters and marks their centers with black Xs:



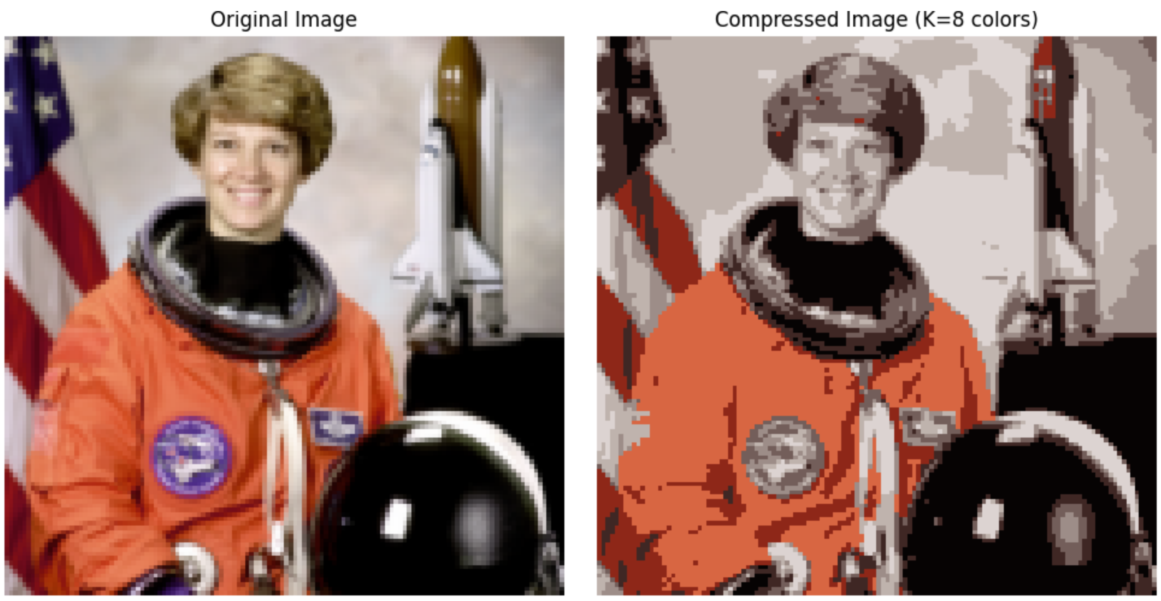
This visual result helps us understand how K-Means assigns groups based on distance.

---

4. Real-World Use Case: Image Compression

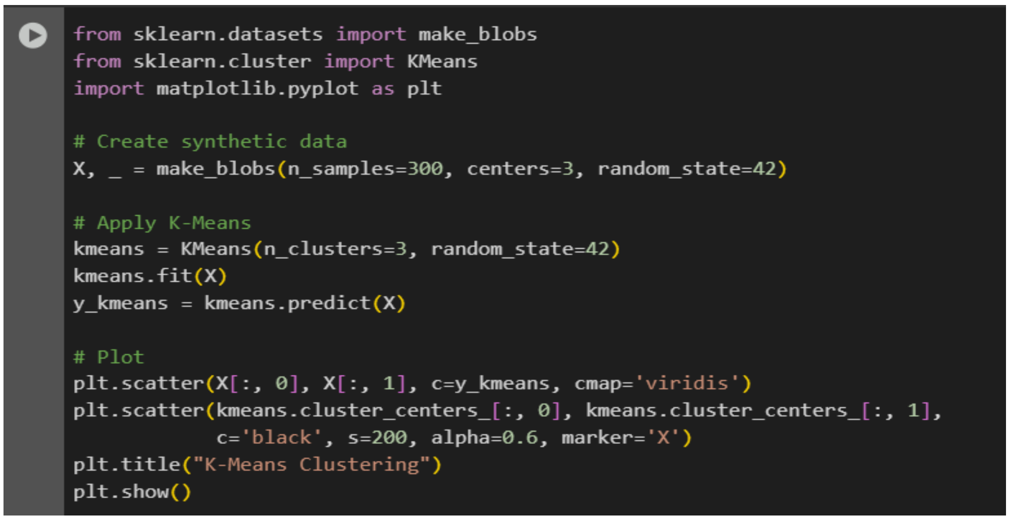
K-Means isn't just for toy datasets. It can be used to compress images by reducing the number of colors used.

We take a built-in image (the famous astronaut from skimage.data) and apply K-Means with k=8. This means the compressed version will use only 8 colors instead of the original 1000s.



The compressed image looks similar to the original but uses much less memory. This technique is useful in web development, game design, and mobile apps.

5. Code Snippet (K-Means in Action)



```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Create synthetic data
X, _ = make_blobs(n_samples=300, centers=3, random_state=42)

# Apply K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# Plot
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            c='black', s=200, alpha=0.6, marker='X')
plt.title("K-Means Clustering")
plt.show()
```

6. Pros and Cons of K-Means

Pros:

- Easy to implement and fast
- Works well with spherical clusters
- Scales to large datasets

Cons:

- Requires you to pre-define k
- Sensitive to initial centroid placement
- Struggles with complex or irregularly shaped clusters

Applications of K-Means

K-Means is used across different industries:

- Customer Segmentation: Grouping customers by spending habits or behavior.
- Image Compression: Reducing color space for storage efficiency.
- Document Clustering: Organizing similar articles or news based on keywords.
- Anomaly Detection: Finding points that don’t fit well into any cluster, like fraud detection.

Limitations and Improvements

Although K-Means is powerful, it has some limitations:

- It assumes clusters are spherical and evenly sized.
- It's sensitive to initial placement of centroids.
- Outliers can significantly affect clustering quality.

One improvement is K-Means++, which improves initialization. Other clustering methods like DBSCAN or hierarchical clustering can handle complex cluster shapes better.

Conclusion

K-Means might be simple, but its ability to uncover structure in unlabeled data makes it incredibly useful. Whether you're simplifying images, grouping users, or exploring datasets, K-Means offers a solid starting point.

References

- Scikit-learn documentation: https://scikit-learn.org/stable/modules/clustering.html#k-means
- Image Compression with K-Means: https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html
- K-Means Algorithm Explanation: https://en.wikipedia.org/wiki/K-means_clustering

---

Personal Reflection

Creating this tutorial helped me appreciate how even simple algorithms like K-Means can be incredibly powerful in practice. Visualizing the data before and after clustering made it easier to understand how the algorithm behaves and why distance matters. The image compression demo was especially exciting — seeing how unsupervised learning applies to real-world problems reinforced my learning and made it fun!