



# K-Means Clustering for Dummies: A Beginner's Guide



Amit Yadav

Follow

10 min read · Jul 14, 2024



52



I understand that learning data science can be really challenging...

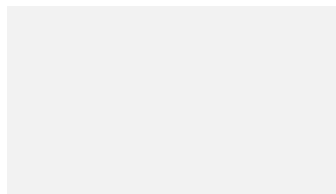
...especially when you are just starting out.

*But it doesn't have to be this way.*


That's why I spent weeks creating a [46-week Data Science Roadmap](#) with projects and study resources for getting your first data science job.

*Here's what it contains:*

## 1. Complete roadmap with study resources



## 2. 20+ practice problems for each week



Sign up to discover human stories that deepen your understanding of the world.

**Free**

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

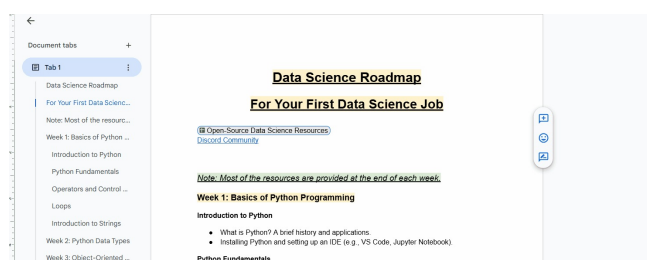
[Sign up for free](#)

**Membership**

- ✓ Read member-only stories
- ✓ Support writers you read most
- ✓ Earn money for your writing
- ✓ Listen to audio narrations
- ✓ Read offline with the Medium app

[Try for \\$5/month](#)

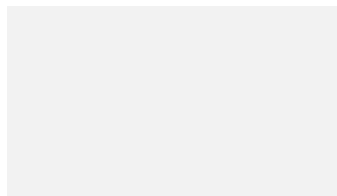
- Free courses
- Top GitHub repositories
- Free APIs
- List of data science communities to join
- Project ideas
- And much more...





If that's not enough, I've also added:

4. A **Discord community** to help our data scientist buddies get access to study resources, projects, and job referrals.



Like what you're seeing?

**[Click here to access everything!](#)**



Now, let's get back to the blog:

. . .

### **What is K-Means Clustering?**

To kick things off, let's talk about clustering in general. Clustering is a technique in data science where you group a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups.

It's like organizing your messy drawer at home; you group similar items together to make things easier to find.

Now, why is clustering important in data science? Clustering helps you uncover hidden patterns and structures in your data.

By grouping similar data points together, you can gain valuable insights and make more informed decisions.

Whether you're working with customer data, image data, or any other type of dataset, clustering can reveal meaningful patterns that might not be immediately obvious.

K-Means Clustering is one of the most popular and straightforward clustering algorithms out there. It's used to partition your data into K distinct clusters based on feature similarity.

Imagine you have a set of data points scattered across a graph. K-Means Clustering will group these points into K clusters by minimizing the variance within each cluster.

It's like finding the best way to organize a room with different types of items into specific zones.

### **Why Learn K-Means Clustering?**

Now, you might be wondering, why should you learn K-Means Clustering? There are several compelling reasons.

### **Practical Applications and Use Cases**

1. **Market Segmentation:** Imagine you work in marketing and have a large customer base. K-Means Clustering can help you segment your

customers into distinct groups based on their purchasing behavior. This allows you to tailor your marketing strategies to each segment, making your campaigns more effective.

2. **Image Compression:** If you're dealing with image data, K-Means Clustering can be used to reduce the number of colors in an image, thereby compressing it. This is particularly useful for web developers looking to optimize images for faster load times.
3. **Anomaly Detection:** In cybersecurity or fraud detection, K-Means can help identify unusual patterns in your data. By clustering normal behavior, you can more easily spot outliers that might indicate a security threat or fraudulent activity.
4. **Document Clustering:** If you're working with text data, K-Means can group similar documents together, making it easier to organize and retrieve information. This is especially useful in search engines and recommendation systems.

### **Benefits of Understanding K-Means for Data Analysis and Machine Learning**

1. **Simplicity and Efficiency:** K-Means Clustering is relatively simple to understand and implement, even if you're new to data science. It's computationally efficient, making it suitable for large datasets.
2. **Foundation for Other Algorithms:** Understanding K-Means provides a solid foundation for learning other, more complex clustering algorithms. It's like learning the basics of arithmetic before diving into calculus.
3. **Versatility:** K-Means is versatile and can be applied to various types of data and problems. Whether you're dealing with numerical data, categorical data, or even mixed types, K-Means can often be adapted to fit your needs.
4. **Enhancing Data Exploration:** By clustering your data, you can better understand its structure and distribution. This enhances your data exploration process, leading to more insightful analyses and better decision-making.

Learning K-Means Clustering helps you in grouping and understanding your data better but also opens doors to more advanced techniques and applications. So, let's dive in and explore how you can master this essential algorithm.

### **Step-by-Step Breakdown of the K-Means Algorithm**

Now, let's go through this step-by-step, as if we were clustering a simple set of points.

1. **Select the Number of Clusters (K):** Let's say you want to create 3 clusters ( $K=3$ ).
2. **Initialize Centroids:** Randomly select 3 points from your dataset as the initial centroids.
3. **Assign Data Points to Nearest Centroid:**
  - Calculate the distance from each data point to all centroids.
  - Assign each data point to the nearest centroid.
  - Now you have 3 clusters of data points.
1. **Update Centroids:**
  - For each cluster, calculate the mean position of all data points in that cluster.
  - Move the centroid to this new mean position.
1. **Repeat Until Convergence:**
  - Reassign all data points to the nearest centroid based on the updated centroids.

- Update the centroids again.
- Repeat this process until the centroids don't change much between iterations.

Here's a simple example to illustrate:

Imagine you have data points representing the locations of several coffee shops in a city. You want to group them into 3 clusters to find central locations for potential delivery hubs.

1. **Select K=3** and randomly pick 3 initial locations as centroids.
2. **Assign each coffee shop to the nearest centroid.** Now you have 3 groups.
3. **Calculate the mean location for each group** and move the centroids to these new locations.
4. **Reassign the coffee shops to the nearest new centroid** and update the centroids again.
5. **Continue this process** until the centroids stabilize.

By the end, you'll have 3 clusters of coffee shops, each with a central hub location that minimizes the distance to the shops in its cluster.

### Choosing the Right Number of Clusters

#### 1. The Elbow Method

Choosing the right number of clusters is crucial for effective K-Means Clustering. One popular method to determine this is the Elbow Method. Let's walk through what it is and how you can use it.

#### Explanation and How to Use It

The Elbow Method involves running the K-Means algorithm for a range of cluster numbers (say, 1 to 10) and computing the sum of squared distances from each point to its assigned centroid (known as inertia). As you increase the number of clusters, the inertia decreases because the clusters are smaller and tighter. However, there's a point where the rate of decrease sharply changes, forming an "elbow" in the graph. This point typically represents a good balance between cluster accuracy and computational efficiency.

#### Practical Example with a Dataset

Let's say you have a dataset of customer data from an e-commerce site, and you want to cluster customers based on their purchasing behavior.

1. **Run K-Means for different values of K** (e.g., 1 to 10).
2. **Plot the inertia for each value of K.**
3. **Look for the "elbow" in the plot.** The value of K at this point is your ideal number of clusters.

Here's how you can implement this in Python:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np

# Assuming you have your data in a variable called data
inertia = []
K = range(1, 11)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42).fit(data)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 4))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
```

```
plt.show()
```

## 2. Silhouette Score

Another method to determine the number of clusters is the Silhouette Score. This method measures how similar a data point is to its own cluster compared to other clusters.

### Explanation and Usage

The Silhouette Score ranges from -1 to 1. A higher score indicates that the data points are well-clustered. A score close to 1 means the points are appropriately clustered, a score around 0 indicates overlapping clusters, and a score less than 0 means the points are likely assigned to the wrong cluster.

### Practical Example

Using the same e-commerce customer dataset, you can calculate the Silhouette Score for different values of K and choose the one with the highest score.

Here's a Python implementation:

```
from sklearn.metrics import silhouette_score

silhouette_scores = []

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42).fit(data)
    score = silhouette_score(data, kmeans.labels_)
    silhouette_scores.append(score)

plt.figure(figsize=(8, 4))
plt.plot(K, silhouette_scores, 'bo-')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score For Optimal k')
plt.show()
```

## Implementing K-Means Clustering in Python

### Setting Up Your Environment

Before you start coding, you need to set up your Python environment. I recommend using Anaconda, which simplifies package management and deployment. You can install the necessary libraries using pip or conda.

```
pip install numpy pandas matplotlib seaborn scikit-learn
```

### Data Preparation

#### Importing and Cleaning Your Data

First, you need to import your data. Let's assume you have a CSV file with customer data.

```
import pandas as pd

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

# Inspect the first few rows
print(data.head())
```

### Feature Selection and Scaling

Select relevant features for clustering and scale them. Scaling ensures that

each feature contributes equally to the distance calculations.

```
from sklearn.preprocessing import StandardScaler

# Select features
features = data[['age', 'annual_income', 'spending_score']]

# Scale the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

## Coding the K-Means Algorithm

### Step-by-Step Code Implementation

Now, let's implement the K-Means algorithm using the scaled features.

```
from sklearn.cluster import KMeans

# Set the number of clusters
k = 3

# Initialize and fit the K-Means model
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(scaled_features)

# Get the cluster labels
labels = kmeans.labels_

# Add the cluster labels to the original dataframe
data['cluster'] = labels

print(data.head())
```

### Explanation of Each Part of the Code

1. **Import Libraries:** Import the necessary libraries.
2. **Load Data:** Read your dataset into a pandas DataFrame.
3. **Feature Selection:** Select the columns that are relevant for clustering.
4. **Scaling:** Standardize the features to have mean=0 and variance=1.
5. **Initialize and Fit K-Means:** Create a K-Means model with the desired number of clusters and fit it to your data.
6. **Assign Labels:** The model assigns each data point to a cluster, and these labels are added to your DataFrame.

### Visualizing the Clusters

Finally, visualize the clusters to understand the results better. You can use Matplotlib and Seaborn for this purpose.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a scatter plot of the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='annual_income', y='spending_score', hue='cluster',
               plt.title('Customer Segments')
               plt.xlabel('Annual Income')
               plt.ylabel('Spending Score')
               plt.show()
```

This visualization will help you see how the data points are grouped and if the clusters make sense.

By following these steps, you should be able to implement K-Means Clustering in Python, determine the optimal number of clusters, and visualize your results effectively.

## Practical Applications of K-Means Clustering

K-Means Clustering is a versatile tool with various practical applications in data science and business analytics. Here, we'll explore three common uses: market segmentation, image compression, and anomaly detection.

## Market Segmentation

### How Businesses Use K-Means to Segment Customers

Imagine you're running a business and you have a large customer base with diverse behaviors. K-Means Clustering can help you segment your customers into distinct groups based on their purchasing habits, demographics, or any other relevant features. This segmentation allows you to tailor your marketing strategies to each group, enhancing customer satisfaction and boosting sales.

#### Example:

Let's say you own an online retail store. You have data on customers' age, annual income, and spending score. By applying K-Means Clustering, you can create segments such as:

1. **Budget Shoppers:** Low income, high spending score.
2. **Mid-range Shoppers:** Medium income, medium spending score.
3. **High-end Shoppers:** High income, high spending score.

These insights enable you to create targeted marketing campaigns. For instance, you might offer discounts to budget shoppers or exclusive deals to high-end shoppers.

Here's how you might implement this in Python:

```
# Assuming you have your customer data loaded and scaled
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(scaled_features)

# Assigning the cluster labels to the original data
data['cluster'] = kmeans.labels_

# Example of a targeted marketing campaign
for cluster in data['cluster'].unique():
    cluster_data = data[data['cluster'] == cluster]
    print(f"Cluster {cluster} Summary:\n{cluster_data.describe()}\n")
```

## Image Compression

### Using K-Means for Reducing Image Sizes

K-Means Clustering can also be applied to image compression. By reducing the number of colors in an image, you can significantly decrease its size without losing much quality. The algorithm clusters similar colors together and replaces them with the centroid of the cluster.

#### Example:

Consider a 24-bit color image where each pixel is represented by three values (Red, Green, Blue). If the image has thousands of unique colors, you can use K-Means to reduce it to, say, 16 colors. This reduction can greatly compress the image size.

Here's how you might implement image compression using K-Means:

```
from skimage import io
import numpy as np

# Load the image
image = io.imread('your_image.jpg')
image = np.array(image, dtype=np.float64) / 255
```

```
# Reshape the image to be a list of pixels
w, h, d = image.shape
pixels = np.reshape(image, (w * h, d))

# Apply K-Means clustering to the pixels
kmeans = KMeans(n_clusters=16, random_state=42)
kmeans.fit(pixels)

# Replace each pixel with the centroid of its cluster
compressed_image = kmeans.cluster_centers_[kmeans.labels_]
compressed_image = np.reshape(compressed_image, (w, h, d))

# Display the compressed image
plt.figure(figsize=(10, 6))
plt.imshow(compressed_image)
plt.axis('off')
plt.show()
```

## Anomaly Detection

### Identifying Outliers in Data

K-Means Clustering can also be used for anomaly detection. In this context, anomalies are data points that do not fit well into any cluster, indicating they might be outliers or rare events.

#### Example:

Consider a financial institution monitoring transactions for fraudulent activity. By clustering the transactions based on various features (amount, frequency, location), you can identify anomalies that might indicate fraud.

Here's how you might implement anomaly detection with K-Means:

```
# Assuming you have your transaction data loaded and scaled
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(scaled_features)

# Calculate the distance of each point to its nearest centroid
distances = kmeans.transform(scaled_features).min(axis=1)

# Set a threshold for anomalies (e.g., top 5% of distances)
threshold = np.percentile(distances, 95)
anomalies = data[distances > threshold]

print(f"Anomalies detected:\n{anomalies}")
```

By identifying these anomalies, you can investigate further to determine if they represent fraudulent transactions or other significant events.

### Ending Note

In conclusion, K-Means Clustering is a powerful tool with diverse applications. Whether you're segmenting your market, compressing images, or detecting anomalies, understanding and applying K-Means Clustering can provide valuable insights and solutions to complex problems.

K Means Clustering

K Means

K Means Algorithm

👍 52



**Written by Amit Yadav**

1.2K Followers · 141 Following

Follow

Get Data Science Roadmap for Your First Data Science Job :  
<https://amit404.gumroad.com/l/ds-diary>

No responses yet



Write a response

What are your thoughts?



## More from Amit Yadav

 Amit Yadav

### Best Laptops For Data Science in 2025

I've found that laptops with Intel i7 or i9 processors, as well as AMD Ryzen 7 or 9, off...

Jun 23, 2024  229  4



 Amit Yadav

### How to Start Learning Machine Learning: A Practical Guide

The biggest lie in data science? That it takes years to get job-ready.

Apr 4  54  2



 In Biased-Algorithms by Amit Yadav

### SHAP Values Explained

I understand that learning data science can be really challenging...


Sep 20, 2024  20



 In Biased-Algorithms by Amit Yadav

### PageRank Algorithm Explained



I understand that learning data science can be really challenging...

Oct 4, 2024  4





See all from Amit Yadav

## Recommended from Medium


 In Stackademic by Shanoj 

### ML Algorithms for Clustering: K-Means, Hierarchical, & DBSCAN

Clustering algorithms are essential for data analysis and serve as a fundamental tool in...


 Nov 8, 2024  9



 In Top Python Libraries by ZHEMING XU

### How to visualize Decision Trees and Random Forest Trees?


Look at the trees with your eyes

 Mar 31  39



 Masego

### Making Sense of Categorical

 In Towards Explainable AI by Sandipan Paul

### Neural Network In SHORT

Survey Data: A K-Modes Clusterin...

When analysing survey responses, selecting the right clustering methodology is essentia...

Jan 24 3

A neural network is composed of layers of interconnected neurons that process...

Apr 9 69 1

In Python in Plain English by Mohamad Mahmood

KMeans Clustering From Scratch

with Animated Clustering Illustration

Oct 31, 2024 55 2

pritesh

Cluster Analysis in Python Part-3

Hello,

Nov 25, 2024

See more recommendations