**Introduction to Clustering with a focus on K-Means Clustering Algorithm**

[[Florence Cesa](https://medium.com/@florence.cesa?source=post_page-----607e167b2512--------------------------------)](https://medium.com/@florence.cesa?source=post_page-----607e167b2512--------------------------------)

[Florence Cesa](https://medium.com/@florence.cesa?source=post_page-----607e167b2512--------------------------------)

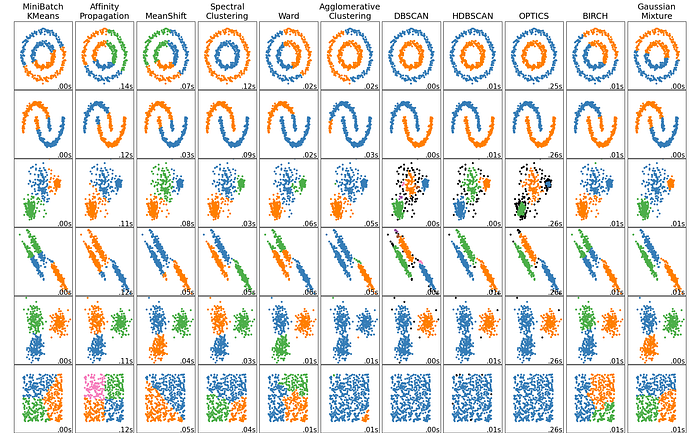
·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F4a73de5b6d09&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40florence.cesa%2Fintroduction-to-clustering-with-a-focus-on-k-means-clustering-algorithm-607e167b2512&user=Florence+Cesa&userId=4a73de5b6d09&source=post_page-4a73de5b6d09----607e167b2512---------------------post_header-----------)

8 min read

·

Dec 7

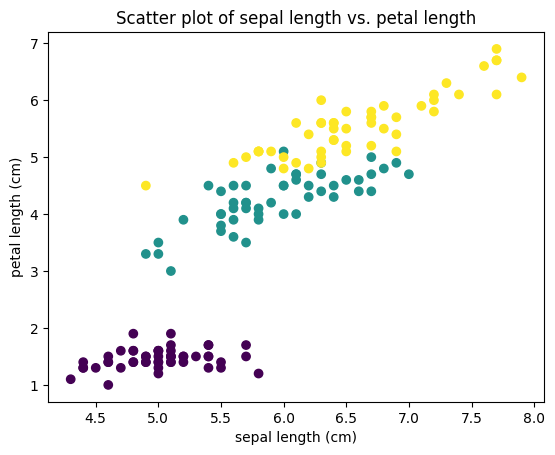


A comparison of the clustering algorithms in scikit-learn | [Source](https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods)

**Introduction**

Clustering is one of the two main type of machine learning: unsupervised machine learning ( when the data in the training dataset is unlabeled). The algorithms must use inferential methods to discover patterns, relationships, and correlations within the raw dataset. This trained model is then used to group similar data points together based on certain characteristics or features.

# Loading iris dataset  
from sklearn.datasets import load\_iris  
  
iris = load\_iris()  
data = iris.data  
  
# Import pyplot  
import matplotlib.pyplot as plt  
x = data[:,0]  
y = data[:,2]  
  
# Visualising dataset  
plt.scatter(x, y, c=iris.target)  
plt.title('Scatter plot of sepal length vs. petal length')  
plt.xlabel('sepal length (cm)')  
plt.ylabel('petal length (cm)')  
plt.show()



Before we delve deeper, here is a list of characteristics to determine if clustering algorithms are appropriate for your dataset:

1. *Do you know and understand the dataset you’re analysing?*
2. *Before running the clustering algorithm, you don’t have an exact idea of the nature of the subsets (clusters). Often, you don’t even know how many subsets are in the dataset before running the algorithm.*
3. *Are the subsets (clusters) determined only by the single dataset you’re analysing?*
4. *Is your goal to determine a model that describes the subsets in a single dataset and only this dataset?*

If you agree with all of the above, clustering algorithms are appropriate for your dataset.

**Key Concepts**

Clustering is based on calculating the similarity or distance measure to quantify between two observations. There are various geometric metrics that can be used to scale a multidimensional numeric data on an n-dimensional plot.

**Common geometric metrics include the following:**

* **Euclidean** which is a measure of the distance between points plotted on a Euclidean plane.
* **Manhattan** which is a measure of the distance between points where distance is calculated as the sum of the absolute value of the differences between two points’ Cartesian coordinates.
* **Minkowski distance** which is a generalisation of the Euclidean and Manhattan distance metricts.
* **Cosine similarity** which measures the similarity of two data points based on their orientation, as determined by calculating the cosine of the angle between them.

For nonnumeric data, **Jaccard distance** metric can be used which is an index that compares the number of features that two observations have in common. A numerical index is generated which quantifies the similarity between text strings.

**Popular Clustering Algorithms**

There are two main types of clustering algorithms:

1. **Partitional** algorithms that create only a single set of clusters.
2. **Hierarchical** algorithms that create separate sets of nested clusters, each in its own heirarchical level.

There are several clustering algorithms with different approach and characteristics such as kernel density estimation methods and neigbourhood algorithms. It’s common for data scientists to experiment with multiple algorithms, but the speed and robustness of the k-means algorithm make it a popular choice among data scientists.

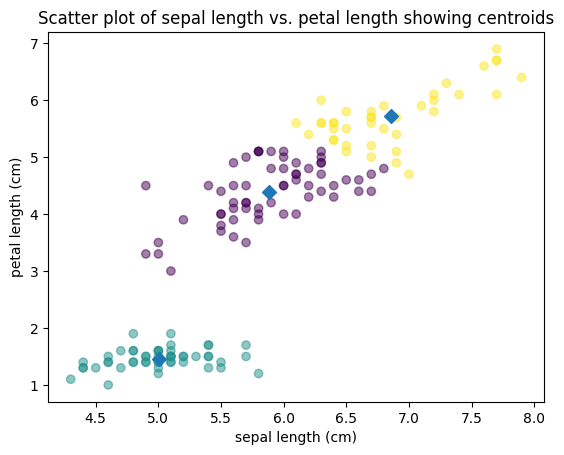
**K-Means Clustering**

K-Means is a simple, fast, partitioning algorithm that can be used to predict groupings within a dataset.

**Centroids**

To get a better understanding of how k-means clustering, let us familiarise ourselves with the concept of centroid. A centroid is the most representative point within a cluster group. With k-means, the number of centroids is represented by k, and k must be specified which the model should find as it generates prediction. The clusters are formed by calculating the nearest mean values to those centroids, measured by the Euclidean distance between observations.

# Import KMeans  
from sklearn.cluster import KMeans  
  
# Create KMeans instance with 3 clusters  
model = KMeans(n\_clusters=3, random\_state=0, n\_init='auto')  
  
# Use fit\_predict to fit model and obtain cluster labels  
labels = model.fit\_predict(data)  
  
# Scatter plot of xs and ys, using labels to define the colors  
xs = data[:,0]  
ys = data[:,2]  
plt.scatter(xs, ys, c=labels, alpha=0.5)  
  
# Assign the cluster centres  
centroids = model.cluster\_centers\_  
  
# Assign the columns of centroids  
centroids\_x = centroids[:,0]  
centroids\_y = centroids[:,2]  
  
# Make a scatter plot of centroids\_x and centroids\_y  
plt.scatter(centroids\_x, centroids\_y, marker='D', s=50)  
plt.title('Scatter plot of sepal length vs. petal length showing centroids')  
plt.xlabel('sepal length (cm)')  
plt.ylabel('petal length (cm)')  
plt.show()

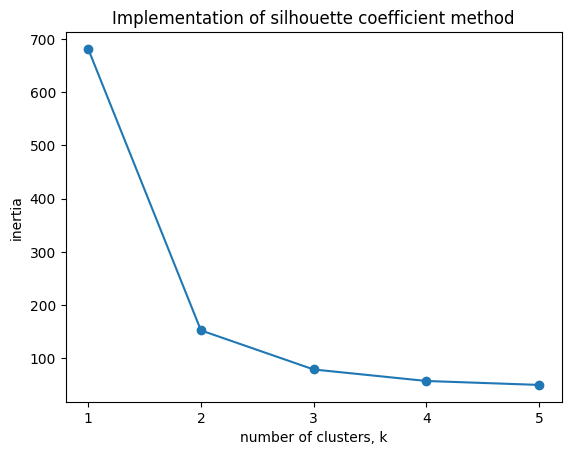


**Optimising k value**

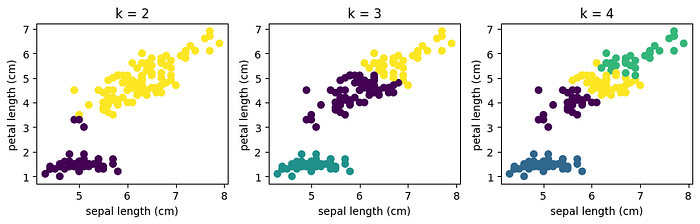
For a dataset that is 2- or 3-dimensional, it is possible to define the k value by looking at a scatterplot. It is also possible to pick the number of centroids based on subjective knowledge about the dataset. However, for a dataset that has more than three dimensions, there are computational methods to get the optimum k value.

The silhouette coefficient is one of the methods used to calculate the goodness of a clustering technique. It calculates the average distance of each point from all other points in a cluster and then compares that value with the average distance to every point in every other cluster.

import numpy as np  
ks = range(1, 6)  
inertias = []  
  
for k in ks:  
 # Create a KMeans instance with k clusters  
 model = KMeans(n\_clusters=k, random\_state=0, n\_init='auto')  
   
 # Fit model  
 model.fit(data)  
   
 # Append the inertia to the list  
 inertias.append(model.inertia\_)  
   
# Plot ks vs inertias  
plt.plot(ks, inertias, '-o')  
plt.title('Implementation of silhouette coefficient method')  
plt.xlabel('number of clusters, k')  
plt.ylabel('inertia')  
plt.xticks(ks)  
plt.show()

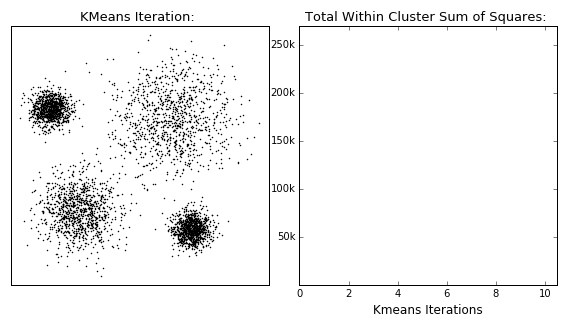


# Specify the range of k  
k\_values = range(2, 5)  
  
# Initialize figure for scatter plot  
plt.figure(figsize=(12, 8))  
  
# Iterate over different k  
for k in k\_values:  
 # Fit k-means  
 kmeans = KMeans(n\_clusters=k, random\_state=0, n\_init='auto')  
 kmeans.fit(data)  
   
 # Get cluster assignments  
 labels = kmeans.labels\_  
   
 # Scatter plot  
 plt.subplot(3, 4, k)  
 plt.scatter(data[:, 0], data[:, 2], c=labels, cmap='viridis', s=40)  
 plt.title(f'k = {k}')  
 plt.xlabel('sepal length (cm)')  
 plt.ylabel('petal length (cm)')  
  
plt.tight\_layout()  
plt.show()



**Mechanism of k-means algorithm**

K-Means algorithm places sample cluster centres on an n-dimensional plot and then evaluates whether moving them in any single direction would result in a new centre with higher density (with more observations closer to it). The centres are moved from regions of lower density to regions of higher density until all centres are within a region of local maximum density (a true centre of the cluster, where each cluster has a maximum number of data points closest to its cluster centre).

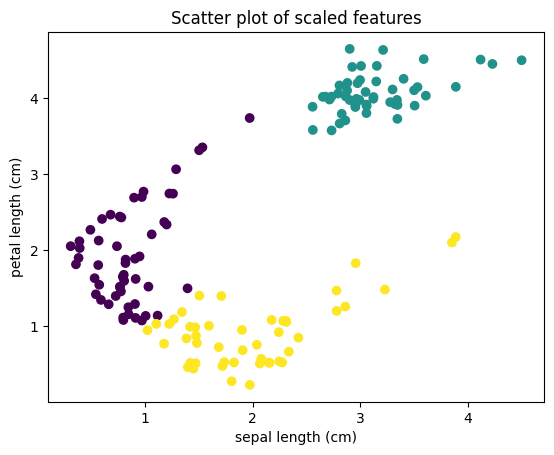


GIF illustration of K-Means algorithm and reassignment of centroids | [Source](https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/)

**Best practices and considerations:**

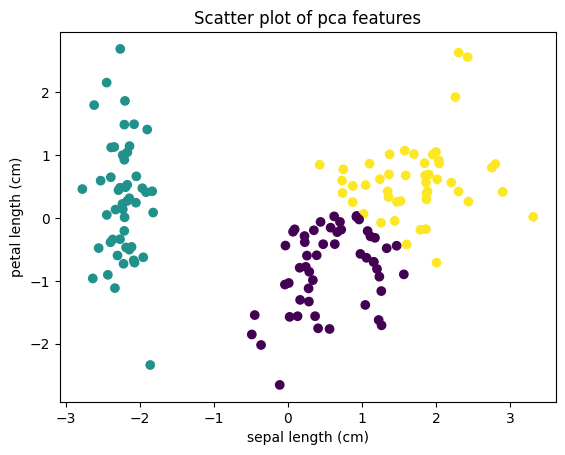
* **Scaling Features:** K-means is sensitive to the scale of the features. The features of a dataset are usually on different scales, the difference of scales can distort the results of Euclidean distance calculation. Standardising or normalising features ensures all dimensions contribute equally to the distance computation between observations.

# Perform necessary imports  
from sklearn.pipeline import make\_pipeline  
from sklearn.preprocessing import StandardScaler  
  
# Create scaler  
scaler = StandardScaler()  
  
# Create KMeans instance  
kmeans = KMeans(n\_clusters=3, random\_state=0, n\_init='auto')  
  
# Create pipeline  
pipeline = make\_pipeline(scaler, kmeans)  
  
# Fit the pipeline  
scaled\_data = pipeline.fit\_transform(data)  
  
# Calculate cluster labels  
labels = pipeline.predict(data)  
  
# Scatter plot  
plt.scatter(scaled\_data[:, 0], scaled\_data[:, 2], c=labels)  
plt.title(f'Scatter plot of scaled features')  
plt.xlabel('sepal length (cm)')  
plt.ylabel('petal length (cm)')  
plt.show()



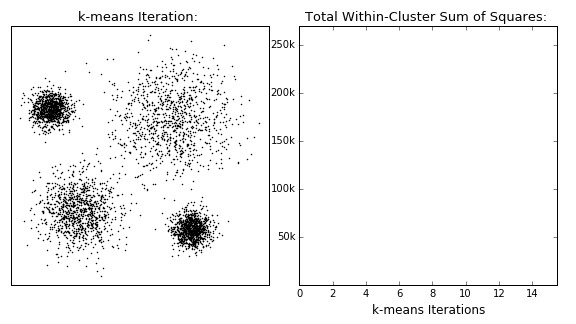
* **Dimensionality Reduction:** Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used with clustering algorithms. PCA finds relationships between features and transforms and reduces them to a set of principal components (directions in which the points vary the most). These components act as a synthetic, refined representation of the dataset, with the information redundancy, noise, and outliers stripped out.

# Perform necessary imports  
from sklearn.decomposition import PCA  
  
# Create PCA model with 2 components  
pca = PCA(n\_components=2)  
  
# Create scaler  
scaler = StandardScaler()  
  
# Fit scaler to data  
scaled\_data = scaler.fit\_transform(data)  
  
# Fit PCA to scaled data  
pca.fit(scaled\_data)  
  
# Transform scaled data  
pca\_features = pca.transform(scaled\_data)  
  
# Print shape of pca\_features  
print(pca\_features.shape)  
  
# Create KMeans   
kmeans = KMeans(n\_clusters=3, random\_state=0, n\_init='auto')  
  
# Fit KMeans to pca\_features  
kmeans.fit(pca\_features)  
  
# Calculate cluster labels  
labels = kmeans.predict(pca\_features)  
  
# Scatter plot  
plt.scatter(pca\_features[:, 0], pca\_features[:, 1], c=labels)  
plt.title(f'Scatter plot of pca features')  
plt.xlabel('sepal length (cm)')  
plt.ylabel('petal length (cm)')  
plt.show()



**Limitations of k-means clustering**

K-Means algorithm may produce incorrect results by placing cluster centres in areas of local minimum density. This happens when centres get lost in low-density regions (regions of the plot that have relatively few points plotted in them) and the algorithm-driven directional movement (the movement that’s meant to increase point density) starts to bounce and oscillate between distant clusters. In these cases, the centre gets caught in a low-density space that’s located between two high-point density zones resulting in erroneous clusters based around centres that converge in areas of low, local minimum density.



GIF illustration of K-Means algorithm converging in areas of low density | [Source](https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/)

**Conclusion**

When making predictions from data, grouping techniques can be a simple and powerful way to generate insights quickly. Clustering can be used across different domains due to its capability to make sense of and divide unlabeled data into subsets.

**References:**

1. Google (2023) What is Clustering? (Available at: <https://developers.google.com/machine-learning/clustering/overview>) (Accessed: 29th November 2023)
2. scikit-learn (2023) The Iris Dataset (Available at: <https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html#sphx-glr-auto-examples-datasets-plot-iris-dataset-py>) (Accessed: 29th November 2023)
3. GeeksforGeeks (2023) Clustering in Machine Learning (Available at: <https://www.geeksforgeeks.org/clustering-in-machine-learning/>) (Accessed: 3rd December 2023)
4. GeeksforGeeks (2023) K means Clustering — Introduction (Available at: <https://www.geeksforgeeks.org/k-means-clustering-introduction/?ref=lbp>) (Accessed: 3rd December 2023)
5. Wikipedia (2023) Euclidean distance (Available at: <https://en.wikipedia.org/wiki/Euclidean_distance>) (Accessed: 4th December 2023)
6. Wikipedia (2023) Jaccard index (Available at: <https://en.wikipedia.org/wiki/Jaccard_index>) (Accessed: 4th December 2023)
7. How To Learn Machine Learning (2023) Feature Scaling in Machine Learning: Why is it important? (<https://howtolearnmachinelearning.com/articles/feature_scaling_machine_learning/>) (Accessed: 5th December 2023)
8. GeeksforGeeks (2023) Principal Component Analysis (Available at <https://www.geeksforgeeks.org/principal-component-analysis-pca/>) (Accessed: 6th December 2023)