

Cluster Evaluation

CLUSTER EVALUATION

- **Intra-cluster cohesion (compactness):**
 - Cohesion measures how near the data points in a cluster are to the cluster centroid.
 - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation (isolation):**
 - Separation means that different cluster centroids should be far away from one another.

EVALUATION: INTERNAL QUALITY MEASURES

$sim(c_i, c_i)$ is the cluster intra similarity. It is a measure of how “cohesive” the cluster is.

If the centroids are not normalized it is the average similarity of the texts in the cluster.

The intra similarity of a clustering:

$$\Phi_{intra}(C) = \frac{1}{n} \sum_{c_i \in C} n_i \cdot sim(c_i, c_i)$$

which is the average similarity of the objects in the set to all objects in their respective clusters.

EVALUATION: INTERNAL QUALITY MEASURES

Similarly, the average similarity of all objects in each cluster to all the objects in the entire set may be calculated:

$$\Phi_{intra}(C) = \frac{1}{n} \sum_{c_i \in C} n_i \cdot sim(c_i, C)$$

This measure shows separation between the clusters.

EVALUATION: EXTERNAL QUALITY MEASURES

For each cluster and category:

➤ $p(i, j) = n_i^{(j)} / n_i$

➤ $r(i, j) = n_i^{(j)} / n_j$

$p(i, j)$, is the probability that an object drawn at random from cluster c_i belongs to category $k^{(j)}$.

$p(k^{(j)} \mid c_i)$, the probability that the object picked at random belongs to category $k^{(j)}$, given that it belongs to cluster c_i .

EVALUATION: INTERNAL QUALITY MEASURES

Purity

➤ **Cluster Purity:**

$$p(c_i) = \max_j p(i, j) = \max_j \frac{n_i}{n}$$

➤ **Clustering Purity:**

$$p(C) = \sum_i \frac{n_i}{n} \cdot p(c_i)$$

Purity disregards all other than the majority class in each cluster.

EVALUATION: INTERNAL QUALITY MEASURES

Entropy

➤ Entropy for a cluster:

$$E(c_i) = - \sum_j p(i, j) \log(p(i, j))$$

➤ Entropy for the Clustering:

$$E(C) = \sum_i \frac{n_i}{n} \cdot E(c_i)$$

EVALUATION: INTERNAL QUALITY MEASURES






Entropy measures the disorganization. A lower value is better.

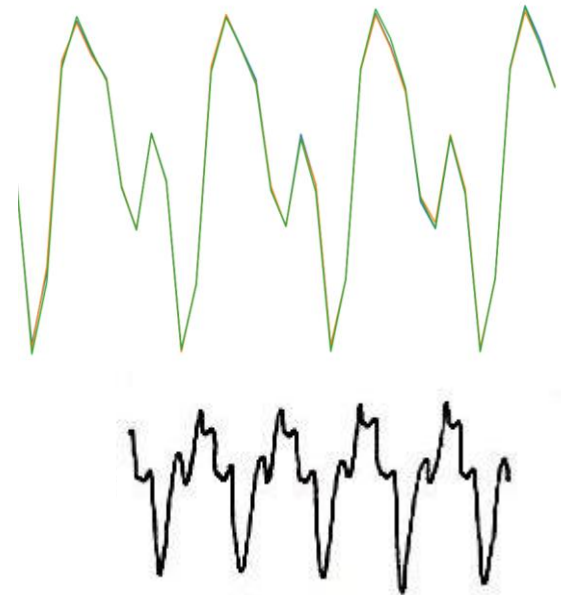
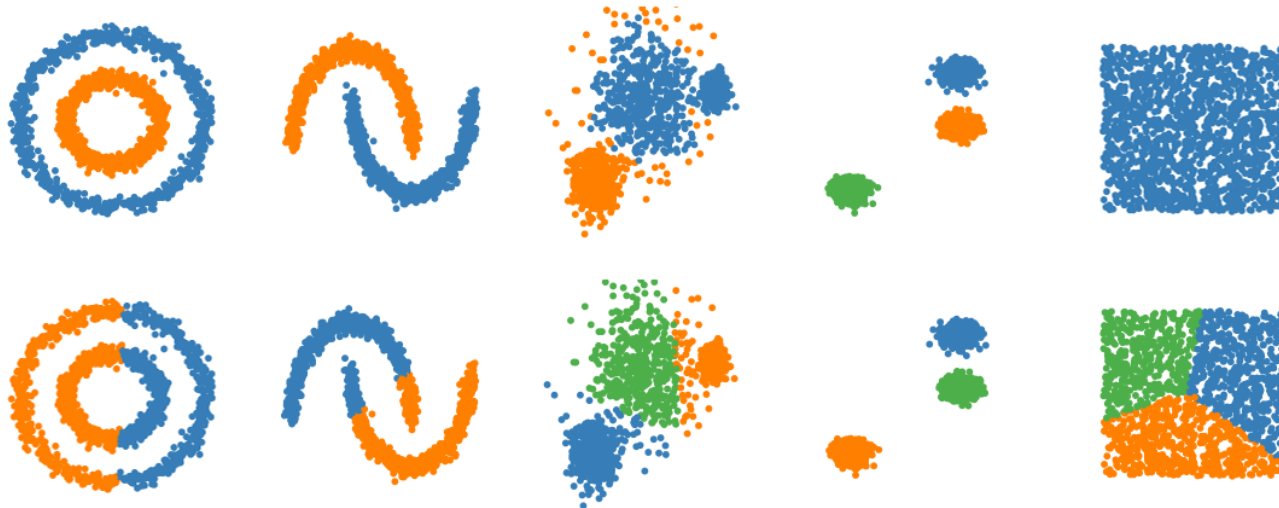
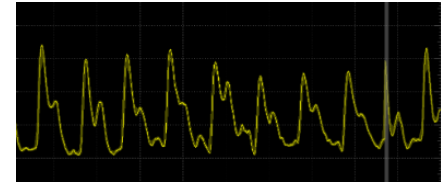
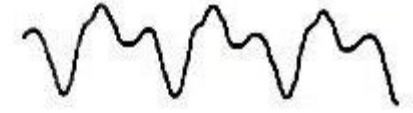
- $\min(E(c_i)) = 0$, when objects only from one category in the cluster.
- $\max(E(c_i)) = \log(\kappa)$, when equal number from all categories.

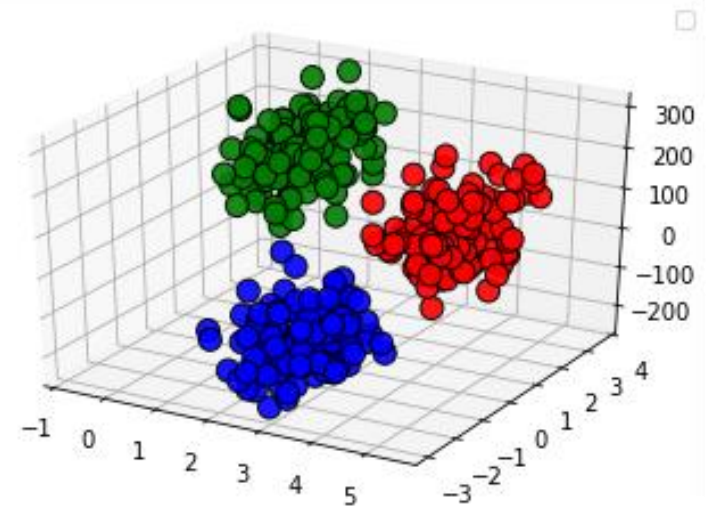
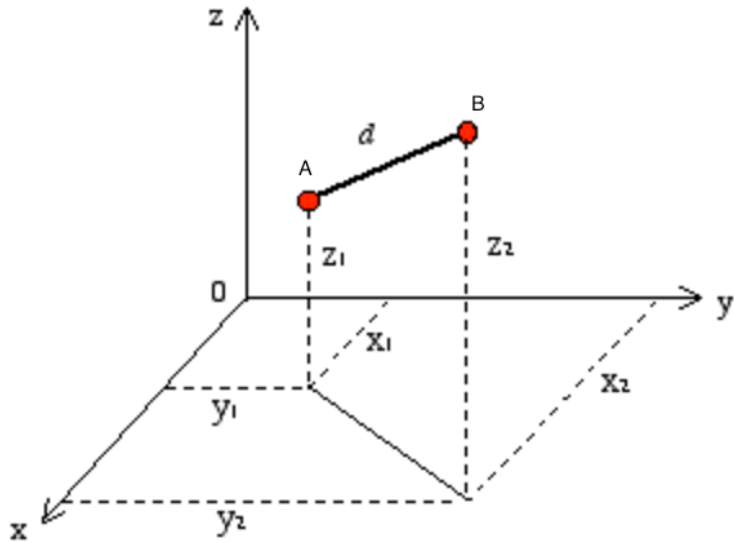
Normalized entropy: $NE(c_i) = E(c_i) / \log(\kappa)$.

Data application

Fields

- Image processing 
- Video 
- Sound 
- Text 
- Data signals 



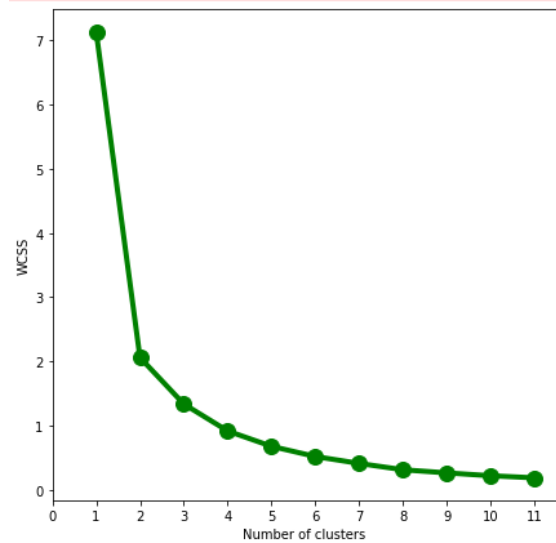


Euclidean distance for 3D points:

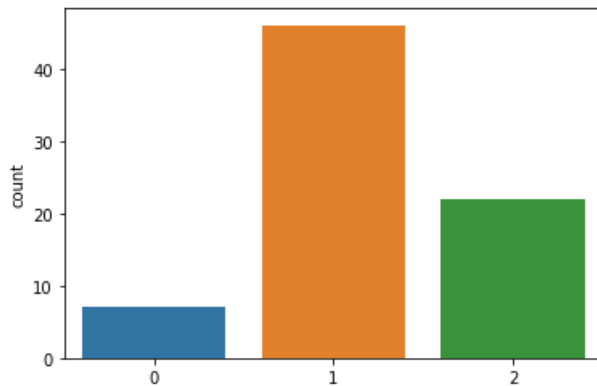
$$d = \sqrt{\sum_{k=1}^m \left((x_i^{(k)} - x_j^{(k)})^2 + (y_i^{(k)} - y_j^{(k)})^2 + (z_i^{(k)} - z_j^{(k)})^2 + (R_i^{(k)} - R_j^{(k)})^2 + (G_i^{(k)} - G_j^{(k)})^2 + (B_i^{(k)} - B_j^{(k)})^2 \right)}$$

R , G , and B represent the intensity values of individual pixels in the image, corresponding to the red, green, and blue colors.

k-means clustering and cluster visualization in 3D



(a)

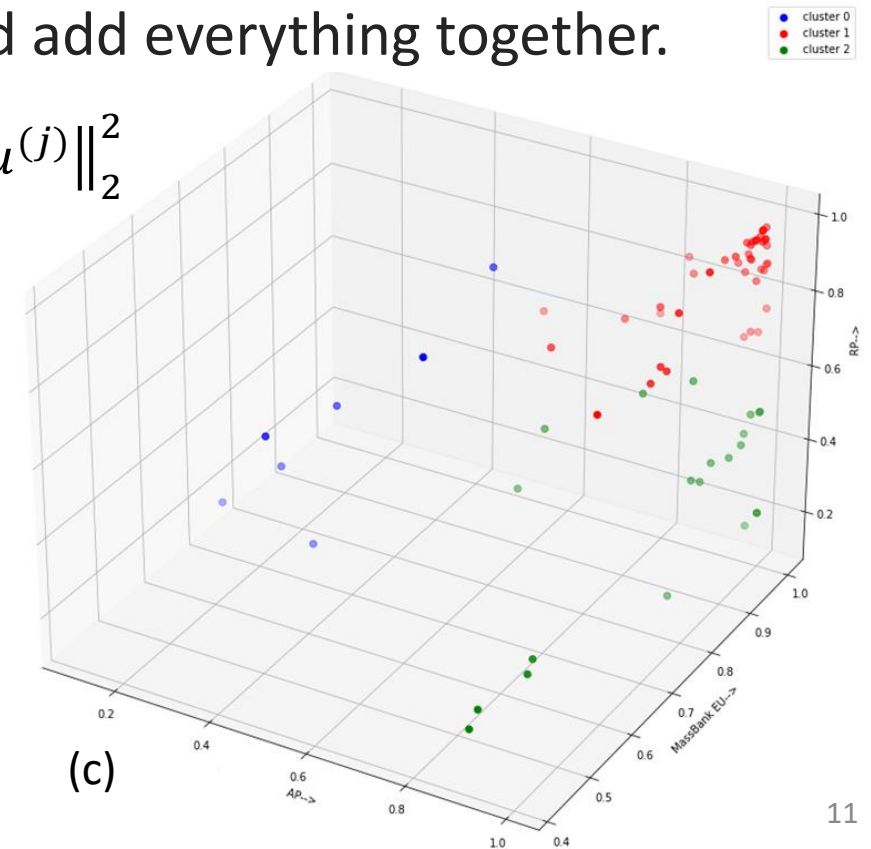


(b)

WCSS (within-cluster sum of squares):

It is the sum of the square of ***Euclidean distance*** between the centroid and all the cluster points of that cluster. In simple terms, it will calculate the distance between the centroid and each point belongs to that cluster and add everything together.

$$WSSE = \sum_{i=1}^n \sum_{j=1}^k w^{(i,j)} \|x^{(i)} - \mu^{(j)}\|_2^2$$



(c)

Figure:

(a) WCSS for clusters. (b) Number of values in each of 3 clusters. (c) Results of *k*-means algorithm for 3 clusters.

k-means clustering and cluster visualization in 3D

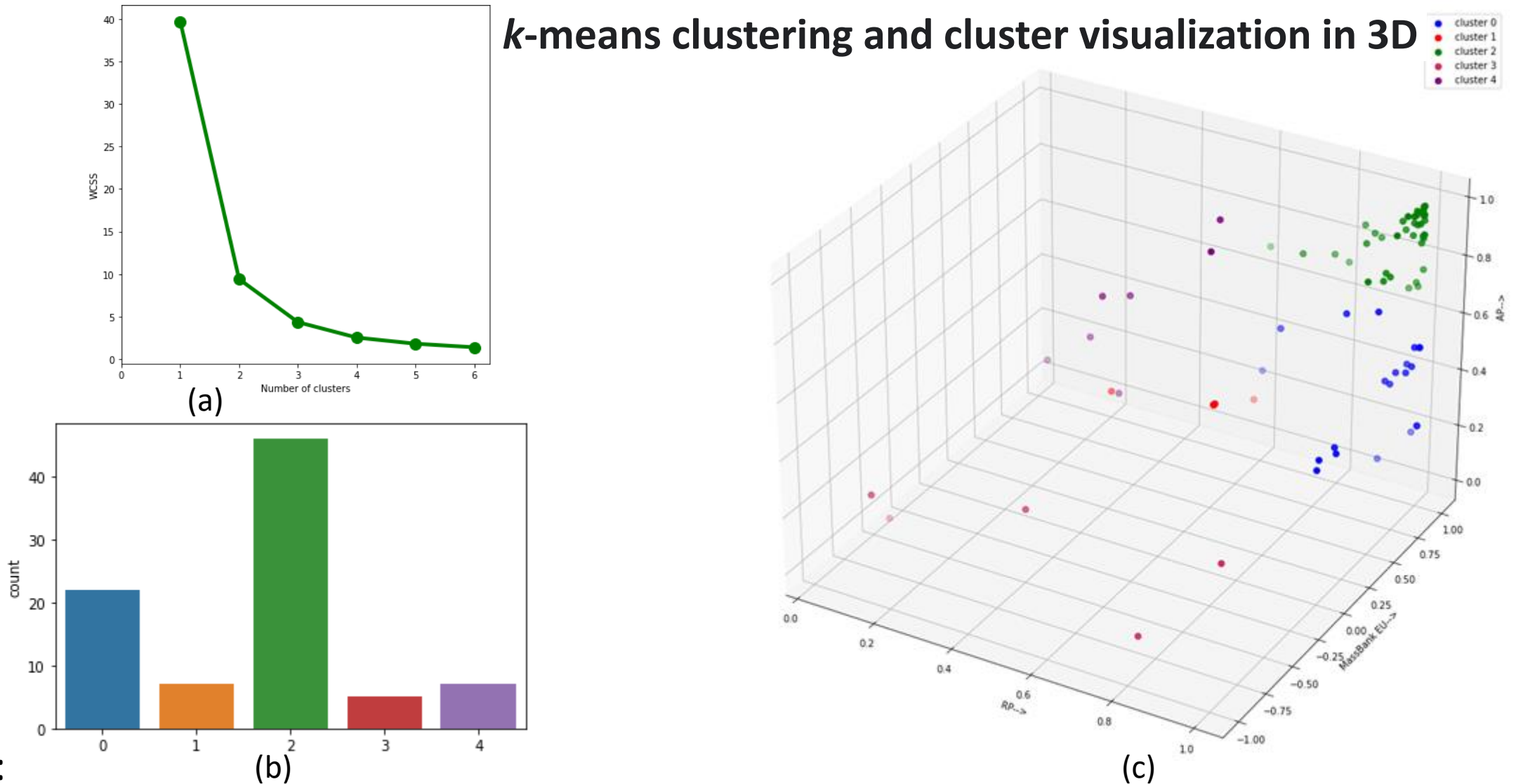


Figure:

(a) WCSS for optimal number of clusters. (b) Number of values in each cluster. (c) Results of *k*-means algorithm.

Choosing the Appropriate Number of Clusters

Two methods that are commonly used to evaluate the appropriate number of clusters:

1. The **elbow method** the visual representation of the WSS (within-cluster sum of squares) / Inertia plot / Sum of squared errors (SSE) / Scree plot method / WSEE.

2. The **silhouette coefficient**

exhibits a **peak** characteristic as compared to the gentle bend in the elbow method. This is easier to visualize and reason with.

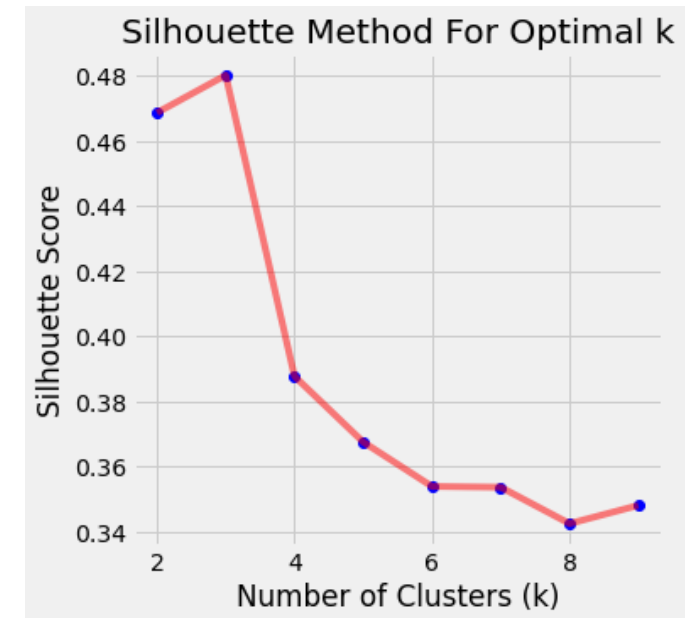
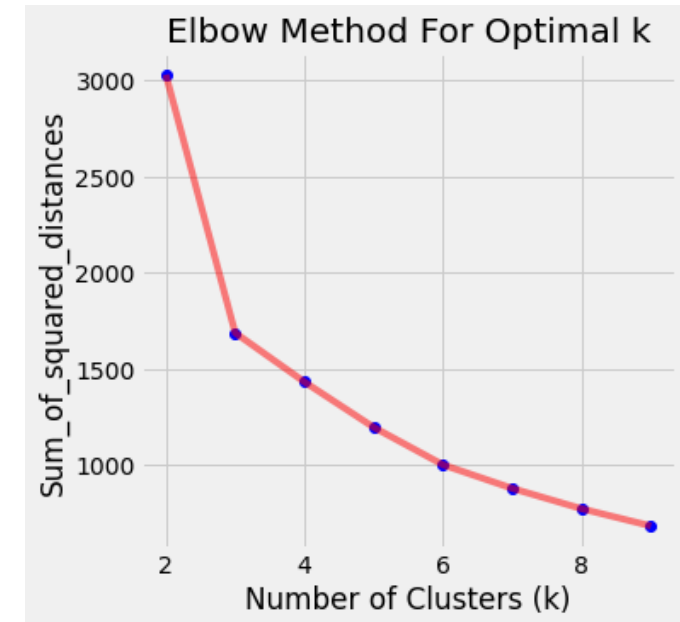
The **silhouette coefficient** quantifies how **close** (max 1) and **far away** (min -1) the data point is to other points in the cluster.

The *average silhouette coefficient* of all the samples is summarized into one score.

[K-Means Clustering in Python: Step-by-Step Example \(statology.org\)](https://statology.org/k-means-clustering-in-python-step-by-step-example/)

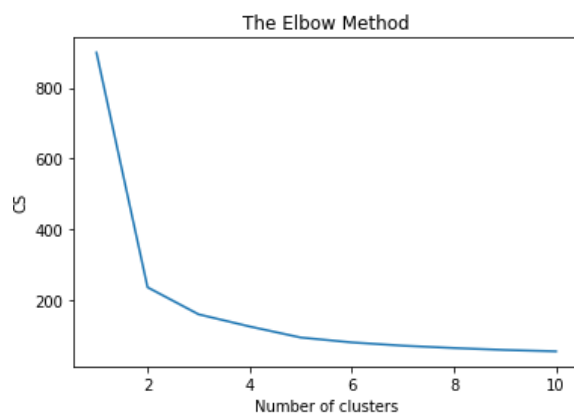
[K-Means Clustering in Python: A Practical Guide – Real Python](https://realpython.com/k-means-clustering-in-python/)

[Python Machine Learning - K-means \(w3schools.com\)](https://www.w3schools.com/python/python_machine_learning_kmeans.asp)



Use elbow method to find optimal number of clusters

```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```



```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)

labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))

print('Accuracy score: {0:0.2f}'.format(correct_labels/float(y.size)))
```

Result: 63 out of 7050 samples were correctly labeled.
Accuracy score: 0.01

K-Means model with 3 clusters

```
kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'.format(correct_labels/float(y.size)))
```

Result: 138 out of 7050 samples were correctly labeled.
Accuracy score: 0.02

K-Means model with 4 clusters

```
kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'.format(correct_labels/float(y.size)))
```

Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62