**Elbow Method vs Silhouette Co-efficient in Determining the Number of Clusters**

*Author: Adria Binte Habib*

In this article, I am going to write about the optimal method for figuring out number of clusters in k-means clustering. K-means clustering algorithm has a specific parameter called ‘K’ for detecting number of clusters. Other algorithms, for example, hierarchical clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure) do not require the parameter ‘K’ because these are not centroid based algorithms. [1]

There are several numbers of methods for figuring out number of clusters which work differently. Today I will discuss about Elbow Method and Silhouette Method and compare between them.

I will discuss the whole procedure using an example [dataset](https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand) of clustering from UCI repository.

Firstly, we need to preprocess the dataset.

#import libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

# Importing the dataset

dataset = pd.read\_csv('Live.csv')

dataset=dataset.drop(['Column1','Column2','Column3','Column4'],

axis=1)

#Label Encoding

lb\_make = LabelEncoder()

dataset['status\_type']= lb\_make.fit\_transform(dataset['status\_type'])

lb\_2 = LabelEncoder()

dataset['status\_published']= lb\_2.fit\_transform(dataset['status\_published'])

X = dataset.values

# Splitting the dataset into the Training set and Test set

X\_train, X\_test = train\_test\_split(X,test\_size = 0.2, random\_state = 0)

If you don’t know about data preprocessing, you can check this out [Data Preprocessing](https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/)

**Elbow Method**

The concept of Elbow method comes from the structure of arm. However, depending on the value of parameter ‘metric’ the structure of elbow method may change. At first, k-means clustering algorithm is applied on the dataset for k number of clusters (I used k = 2 to 15) to find groups in data in unlabeled data. [2]

After that, an average score has been calculated according to the parameter ‘metric’. The default value for metric parameter is ‘distortion’ which calculates the sum of squared distances from the assigned centroids. [3]

When the graphs are plotted for the ‘distortion’ value, it gives a structure of arm showing the elbow and it denotes the number of clusters, as shown in Figure 1.

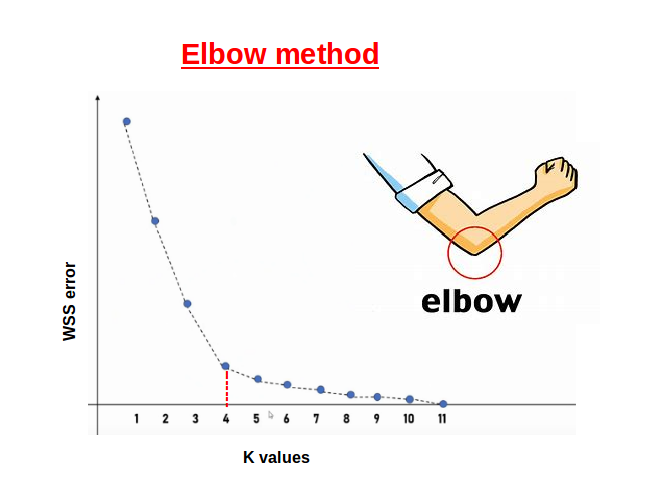


Figure 1: Elbow method using distortion

However, I used another value of this ‘metric’ parameter called ‘calinski\_harabasz’. This parameter calculates the ratio of scatterings between and within the clusters [3] which gives better assumptions than others. The structure of the elbow looks up side down for this parameter.

Finally, the average scores have been plotted on a graph (k vs. average calinski\_harabasz score) using yellowbrick visualizer [3], shown in Figure 2.

#Kmeans Clustering

range\_n\_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

for n\_clusters in range\_n\_clusters:

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X\_train)

visualizer= KElbowVisualizer(clusterer,k=(2,15), metric =

'calinski\_harabasz',locate\_elbow =

False, timings= False)

visualizer.fit(X\_train) # Fit the data to the visualizer

visualizer.show() # Finalize and render the figure

From Figure 2, you can see that the elbow points at k = 11. So, it can be said that for this approach, the number of optimal clusters are 11.

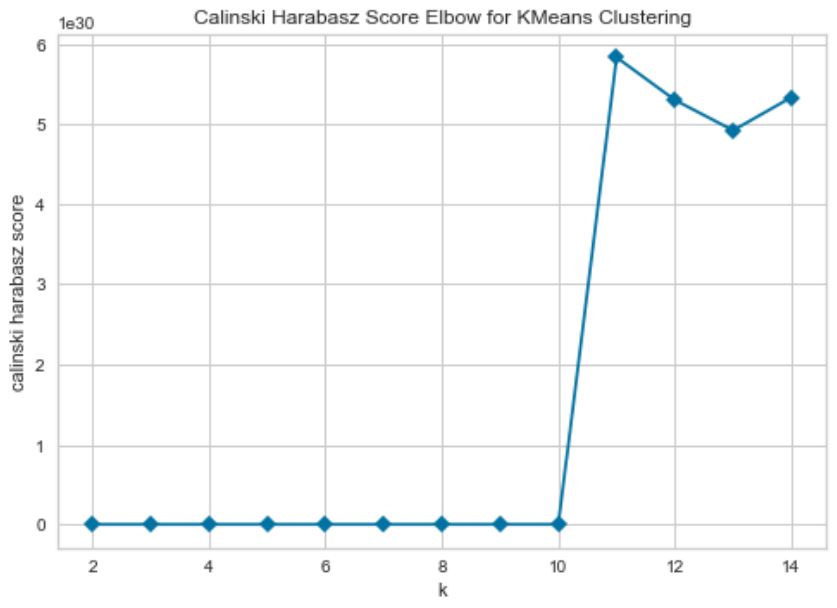


Figure 2: Elbow method using Calinski \_Harabasz

**Sillhouette Score Method**

The silhouette plot displays a measure, ranging [-1, 1] where [4],

+1: Means clusters are well apart from each other and clearly distinguished

0: Means clusters are indifferent or we can say that the distance between clusters is not significant

-1: Means the clusters are assigned in the wrong way

Now let’s see, how this silhouette score is being calculated,

The formula for a single silhouette coefficient is,

Where, a = mean intra-cluster distance and b = mean nearest-cluster distance

After calculating each silhouette coefficient, an average score has been calculated which lies in the range of [-1, +1]. This average silhouette score defines the number of optimal clusters.

Let’s dive into the code,

#import libraries

from sklearn.metrics import silhouette\_score

from yellowbrick.cluster import SilhouetteVisualizer

#Kmeans Clustering

range\_n\_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

for n\_clusters in range\_n\_clusters:

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X\_train)

# The silhouette\_score gives the average value for all the samples.

# This gives a perspective into the density and separation of the formed

# clusters

silhouette\_avg = silhouette\_score(X\_train, cluster\_labels)

print("For n\_clusters =", n\_clusters,

"The average silhouette\_score is :", silhouette\_avg)

visualizer = SilhouetteVisualizer(clusterer, colors='yellowbrick', is\_fitted= 'auto')

visualizer.fit(X\_train) # Fit the data to the visualizer

visualizer.show() # Finalize and render the figure

**OUTPUT:**

For n\_clusters = 2 The average silhouette\_score is : 0.9419743880621418

For n\_clusters = 3 The average silhouette\_score is : 0.8925568467675032

For n\_clusters = 4 The average silhouette\_score is : 0.8854468255579183

For n\_clusters = 5 The average silhouette\_score is : 0.8859344049988384

For n\_clusters = 6 The average silhouette\_score is : 0.896222949688388

For n\_clusters = 7 The average silhouette\_score is : 0.9531228433846561

For n\_clusters = 8 The average silhouette\_score is : 0.9882303235394505

For n\_clusters = 9 The average silhouette\_score is : 0.9942722572401562

For n\_clusters = 10 The average silhouette\_score is : 0.9860105575225317

**For n\_clusters = 11 The average silhouette\_score is : 0.9999999922837097**

For n\_clusters = 12 The average silhouette\_score is : 0.906560275971653

For n\_clusters = 13 The average silhouette\_score is : 0.6440635368489311

For n\_clusters = 14 The average silhouette\_score is : 0.7165267612201155

For n\_clusters = 15 The average silhouette\_score is : 0.6440635368489311

From the above output, it can be seen that the highest silhouette score got for the number of clusters 11. However, in Figure 3, I plotted the silhouette score graph for 12 clusters as after crossing n\_clusters = 12, the silhouette score started decreasing which means that the clusters are assigned in the wrong way. Since, we know the higher the silhouette score, the chances get higher to the optimal one, so it can be said that the probable number of clusters should be 11.

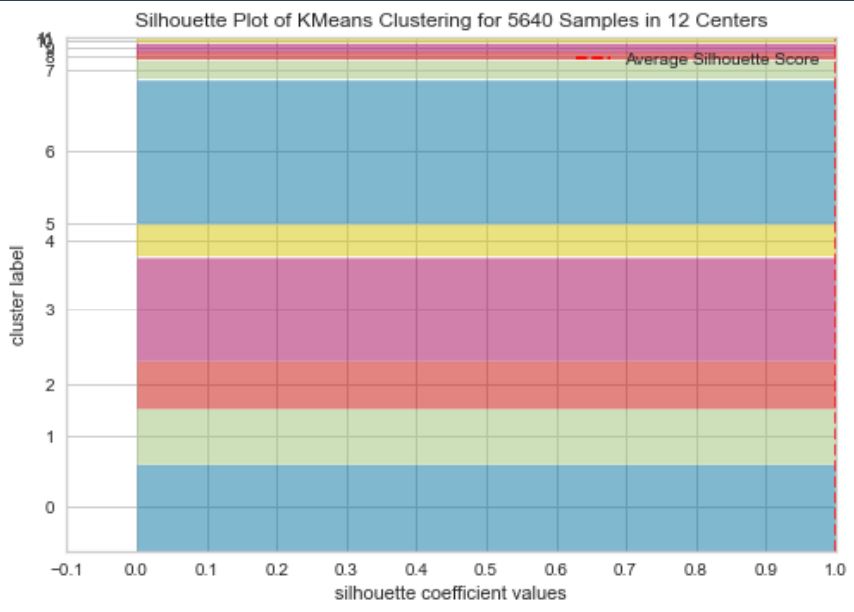


Figure 3: Silhouette plot

Now we should discuss which method is more appropriate.

However, we should always keep it in mind that the elbow method does not work well if the data is not very clustered. If you don’t see a skewed graph then you will not be able to find the number of clusters [3]. Moreover, the skewed graph does not always give the correct answer. If there are lots of duplicate data then there is possibility that elbow method may not give proper output. In the case of overlapping data, the silhouette coefficient works better, since it identifies the duplicate data.

On the other hand, in the case of silhouette score, defining the number of clusters does not depend on how skewed the graph is. It depends on the silhouette score, the closer it to +1 the chances get higher to become it to optimal one.

So, it can be said that the efficacy of the elbow method depends on the nature of the dataset. If the pattern of the relevant dataset is favorable then the elbow method works well. On the other hand, the silhouette score does not depend on the nature of dataset. Because Silhouette is a distance-based method, the mean distances between the intra-cluster objects and the nearest cluster is used for finding out the silhouette score.

However, in the only case the silhouette score does not work when you have only one cluster because silhouette score demands minimum 2 clusters or more than that [5]. **Moreover, it can simply be said that if there is only one cluster in the dataset then the dataset is not fitted for clustering.**

**……………………………**

**Reference**

[1] Wikipedia contributors. (2021, February 8). *Determining the number of clusters in a data set*.Wikipedia.https://en.wikipedia.org/wiki/Determining\_the\_number\_of\_clusters\_in\_a\_data\_set

[2] *fw\_error\_www*. (n.d.). Oracle. https://blogs.oracle.com/datascience/introduction-to-k-means-clustering

[3] *Elbow Method — Yellowbrick v1.3.post1 documentation*. (n.d.). Yellowbrick. <https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>

[4] Bhardwaj, A. (2020, May 28). *Silhouette Coefficient - Towards Data Science*. Medium. <https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c>

[5] *sklearn.metrics.silhouette\_score — scikit-learn 0.24.1 documentation*. (n.d.). Scikit-Learn.Org. <https://scikitlearn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html>