Finding optimal number of clusters

Silhouette analysis

$$silhouette\ score = \frac{p - q}{max(p, q)}$$

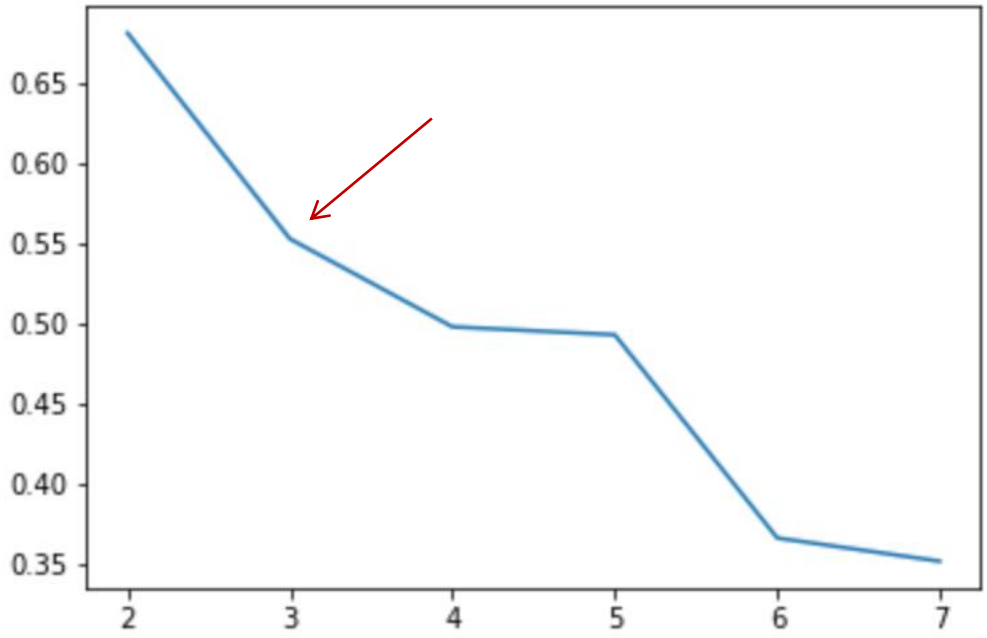
p is the mean distance to the points in the nearest cluster that the data point is not a part of

q is the mean intra-cluster distance to all the points in its own cluster

- The value of the silhouette score lies between -1 to 1.
- A score closer to 1 indicates that the data point is very similar to other data points in the cluster,
- A score closer to -1 indicates that the data point is not similar to the data points in its cluster.

```
from sklearn.cluster import KMeans
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas  as pd
from sklearn.metrics import silhouette_score
data = datasets.load_iris()
X = data.data
y = data.target

sse_ = []
for k in range(2, 8):
    kmeans = KMeans(n_clusters=k).fit(X)
    sse_.append([k, silhouette_score(X, kmeans.labels_)])
```

```
plt.plot(pd.DataFrame(sse_)[0], pd.DataFrame(sse_)[1]);
```



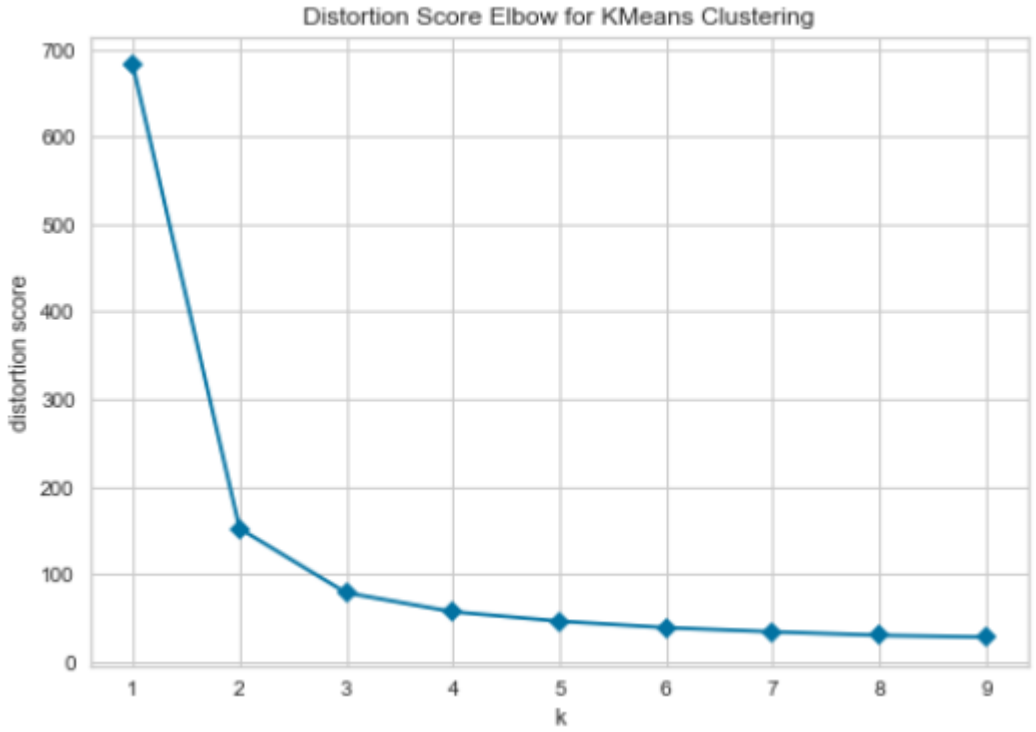
From the above diagram, we try to identify the first angle from the left. We see that the first angle, corresponds to number **3**. That means that the method evaluates that the optimal number of clusters is **3**.

Elbow method

1st approach

```
from sklearn.cluster import KMeans

from sklearn import datasets

from yellowbrick.cluster import KElbowVisualizer

data = datasets.load_iris()

X = data.data

y = data.target


# Instantiate the clustering model and visualizer

model = KMeans()

visualizer = KElbowVisualizer(model, k=(1,10), timings=False ,locate_elbow=False)

visualizer.fit(X)        # Fit the data to the visualizer
```

visualizer.show() # Finalize and render the figure

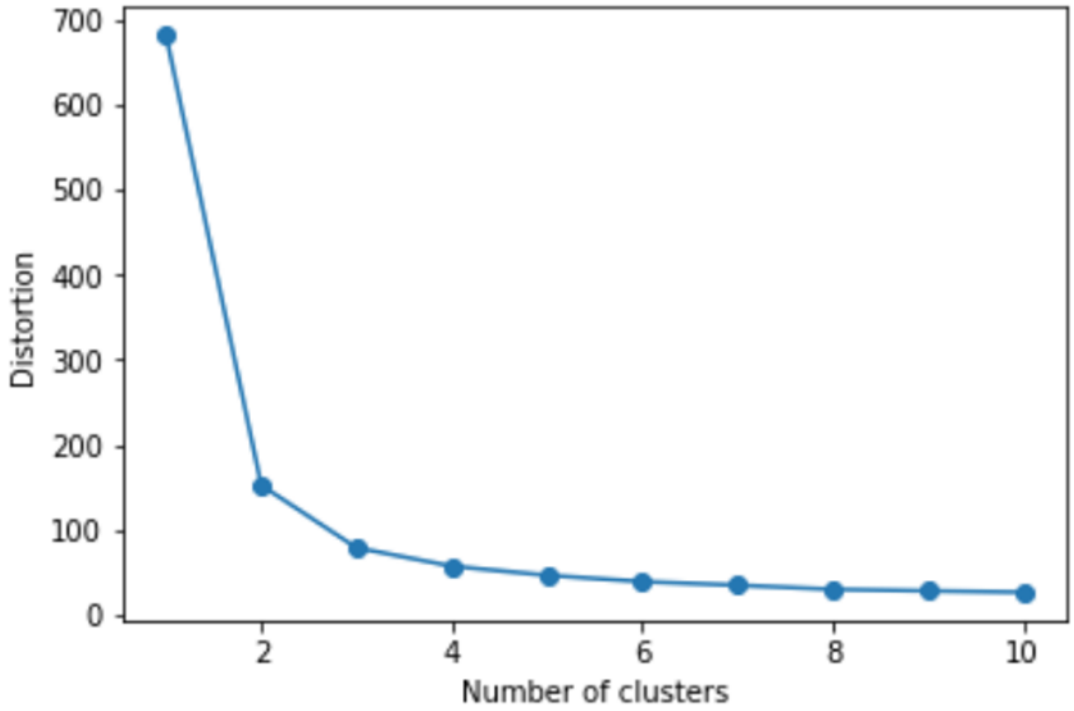


2nd approach

The elbow method is a useful graphical tool to estimate the optimal number of clusters k for a given task. Intuitively, we can say that, if k increases, the within-cluster SSE ("distortion") will decrease. This is because the samples will be closer to the centroids they are assigned to.

```
from sklearn.cluster import KMeans

from sklearn import datasets

import matplotlib.pyplot as plt


data = datasets.load_iris()

X = data.data

y = data.target


# calculate distortion for a range of number of cluster

distortions = []

for i in range(1, 11):
```

```
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(X)
    distortions.append(km.inertia_)

plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```



From the above diagram we try to identify the “elbow”. In this case, the elbow is located at number 3. That means that the method evaluate that the optimal number of clusters is 3.