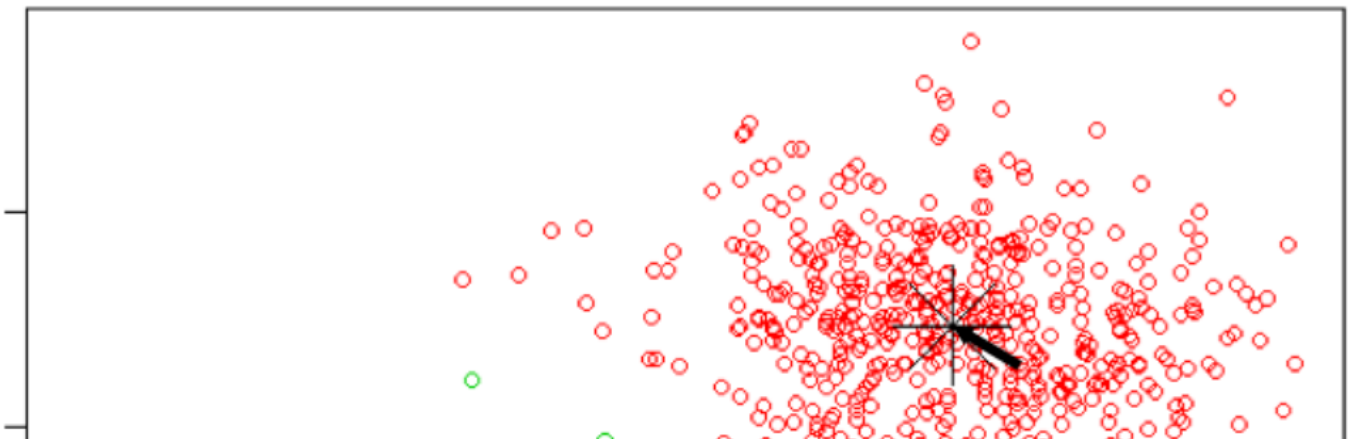
DATA SCIENCE

Silhouette Analysis vs Elbow Method vs Davies-Bouldin Index: Selecting the optimal number of clusters for KMeans clustering

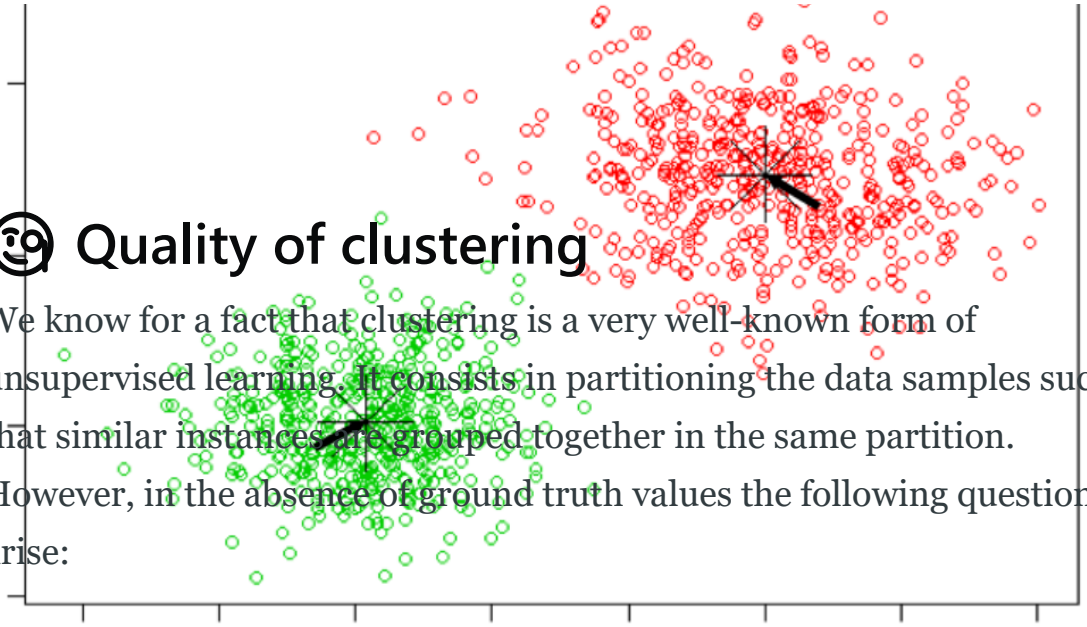
GEORGIOS DRAKOS

4 MAR 2020 • 7 MIN READ



In this article, I compare three well-known techniques for validating the quality of clustering: the **Davies-Bouldin Index,** the **Silhouette Score** and the **Elbow Method**. All the aforementioned techniques are used for determining the optimal number of clusters. If you are not familiar with clustering techniques please do read my previous article where I implemented the K-means algorithm from scratch in Python (9 lines).

🖉 Table of Contents



🧐 Quality of clustering

We know for a fact that clustering is a very well-known form of unsupervised learning. It consists in partitioning the data samples such that similar instances are grouped together in the same partition. However, in the absence of ground truth values the following questions arise:

- How do we know the correct k (number of clusters)?
- Usually we have a vague idea (less than 20) but there can be many choices of k which look good.
- How do we choose when the data is 4 dimensional or more when it is not use to plot them?

🔍 Access clustering

The process of accessing the quality of clustering can be a tricky task. For that reason, different measures do exist to determine the quality of clustering. To name a few:

- Silhouette Analysis
- The Elbow method
- Davies-Bouldin Index

The silhouette score is a measure of the average similarity of the objects within a cluster and their distance to the other objects in the other clusters.

For each data point i, we first define:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$

which represent the average distance of the point i to all the other points that belongs to the same cluster Ci. Large `a(i)` implies that the data point i is dissimilar to its cluster.In other words, if the point i belongs to the zero cluster this is the average distance of point i with all the other points of the zero cluster.

Secondly, we define:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

which represent the average distance of the point i to all the other points in the next nearest cluster. Large `b(i)` implies that the data point i is dissimilar to its neighbouring cluster.

This involves two steps:

- Find all the average distance of the point i with all the other

- Take the minimum of those averages.

then of 2 etc. Once you have calculated all those averages distances you take the minimum.

Finally, we define the silhouette score of a data point i as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Please note the following:

- $-1 \leqq s(i) \leqq 1$
- If $s(i) \approx 1$, then $a(i) \ll b(i)$ the point is in the correct cluster; it is much closer to its own cluster than to its neighbour.
- If $s(i) \approx -1$, then $a(i) \gg b(i)$ the point should be in another cluster; it is must closer to its neighbour than to its assigned cluster.
- s(i) = 0 if |Ci | = 1 (i.e., the silhouette score of a point in a single-element cluster is 0).
- If s(i) ≈ 0, the point could be in another cluster.

$$S = \frac{1}{N} \sum_{i} s_i$$

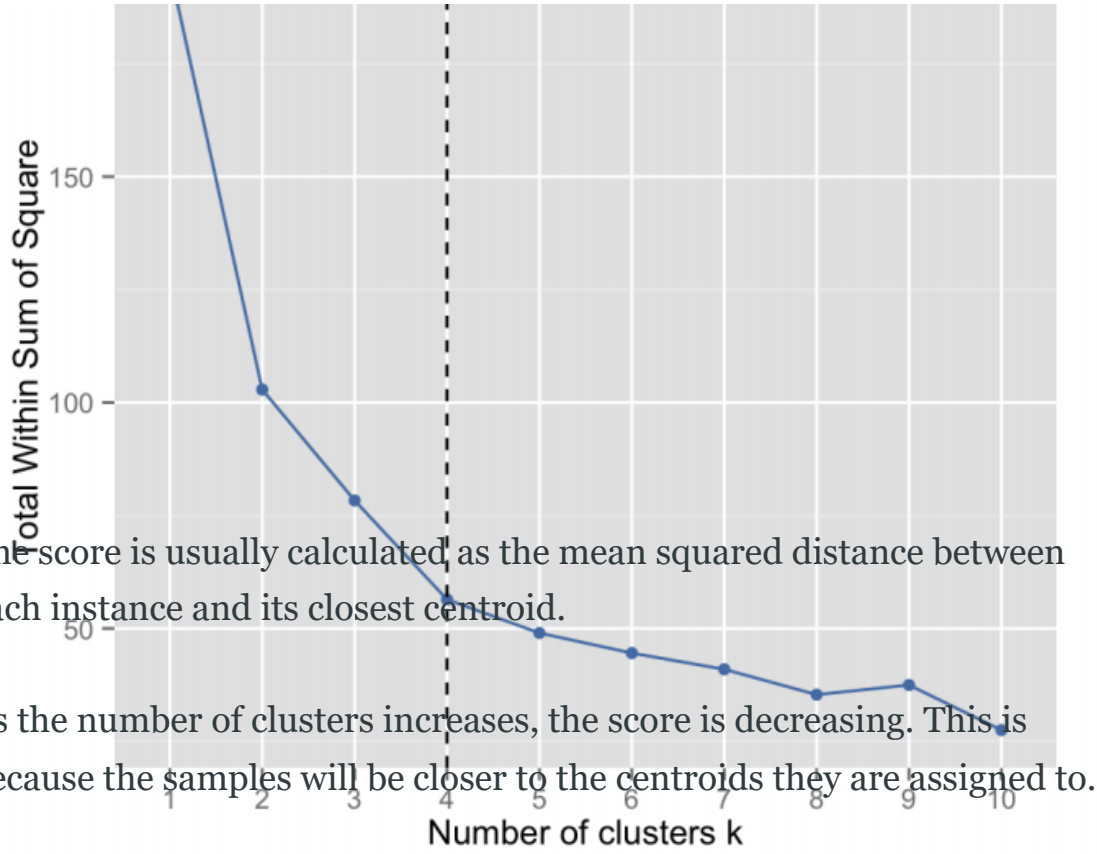
When using the the silhouette analysis we are normally looking for a value of k that provides:

- High average silhouette scores.
- No clusters with a maximum silhouette score less than the average score as this can indicate poor clustering and a bad choice of k.
- Well clustered data tends to have clusters where the silhouettes are similar in size and the silhouettes of each cluster member are similar

---

Elbow Method

The most common approach for selecting the optimal number of clusters is the so-called "elbow method". A simple (but not very robust) visual method to estimate the optimal number of clusters k. It involves running the algorithm multiple times with an increasing number of cluster choice and then plotting a clustering score (the within-cluster SSE - "**distortion**") as a function of the number of clusters.



The score is usually calculated as the mean squared distance between each instance and its closest centroid.

As the number of clusters increases, the score is decreasing. This is because the samples will be closer to the centroids they are assigned to.

The idea behind the elbow method is to identify the value of k where the score begins to decrease most rapidly before the curve reached a plateau.However, it can get confusing sometimes.

Davies-Bouldin Index

The Davies–Bouldin (DB) criterion is based on a ratio between **"within-cluster"** and **"between-cluster"** distances:

$$\mathrm{DB}(\mathcal{C}) = \frac{1}{k}\sum_{i=1}^{k} \max_{j \le k, j \neq i} D_{ij}, \;\; k = |\mathcal{C}|,$$

- `Dij` is the "within-to-between cluster distance ratio" for the ith and jth clusters.

$$D_{ij} = \frac{(a_i + a_j)}{d_{ij}},$$

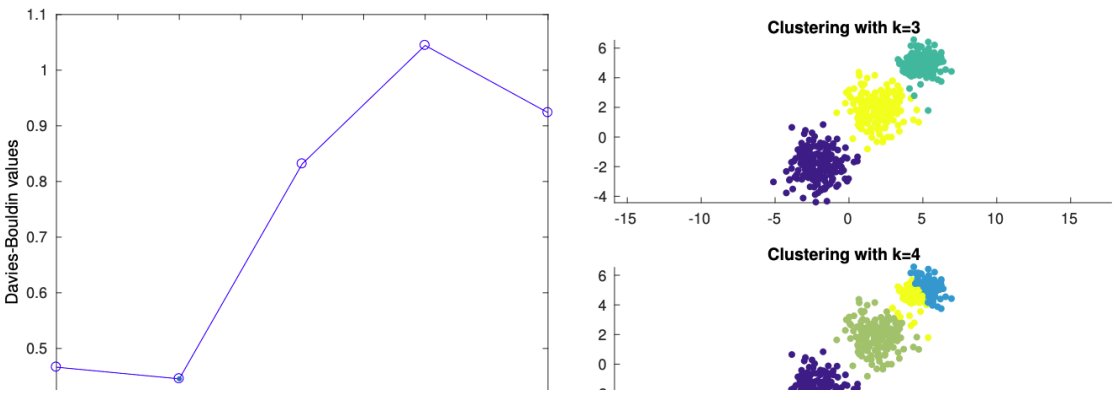
where `d- i` is the average distance between every data point in cluster i and its centroid, similar for `d- j`. `dij` is the Euclidean distance between the centroids of the two clusters.

If two clusters are close together (small `dij`) but have a large spread (large `d-i+ d- j`) then this ratio will be large, indicating that these clusters are not very distinct.

- `max Dij` denotes the "worst-case within-to-between cluster ratio" for cluster i

For example, assuming that we have four clusters 0,1, 2 and 3 we calculate D between cluster 0 and 1, 0 and 2, 0 and 3 and we take the maximum of those values. We do exactly the same for cluster 1,2 and 3. We then calculate the average of those maximum values.

Remember the optimal clustering has the smallest Davies–Bouldin index value.



Davies-Bouldin Index vs Silhouette Analysis vs Elbow Method

I have drawn the following conclusions for these techniques:

- Silhouette is the most computationally demanding.
- Silhouette analysis is the most informative.
- Try all! Before deciding which is the best.

For example, the **Davies-Bouldin Index** evaluates **intra-cluster similarity and inter-cluster differences** while the Silhouette score measure the distance between each data point, the centroid of the cluster it was assigned to and the closest centroid belonging to another cluster. Depending of what criterion is better for your case select the appropriate metric.

The best way to evaluate the results of your clustering efforts is to start by actually **examining -- human inspection -- the clusters formed and making a determination based on an understanding of what the data represents, what a cluster represents, and what the clustering is intended to achieve.**

Python Implementation

Let's cluster a dataset and evaluate its performance by using the above techniques. Below you can screenshots of a working example. The full notebook is available on my GitHub page.

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

[2]:
```
df = pd.read_csv('cluster_validation_data.txt', sep=",", header=None)
df.head()
```

[2]:

	0	1
0	2.510076	2.159303
1	3.739776	0.974175
2	-0.142930	2.960866
3	2.817929	2.268013
4	2.302407	2.119619

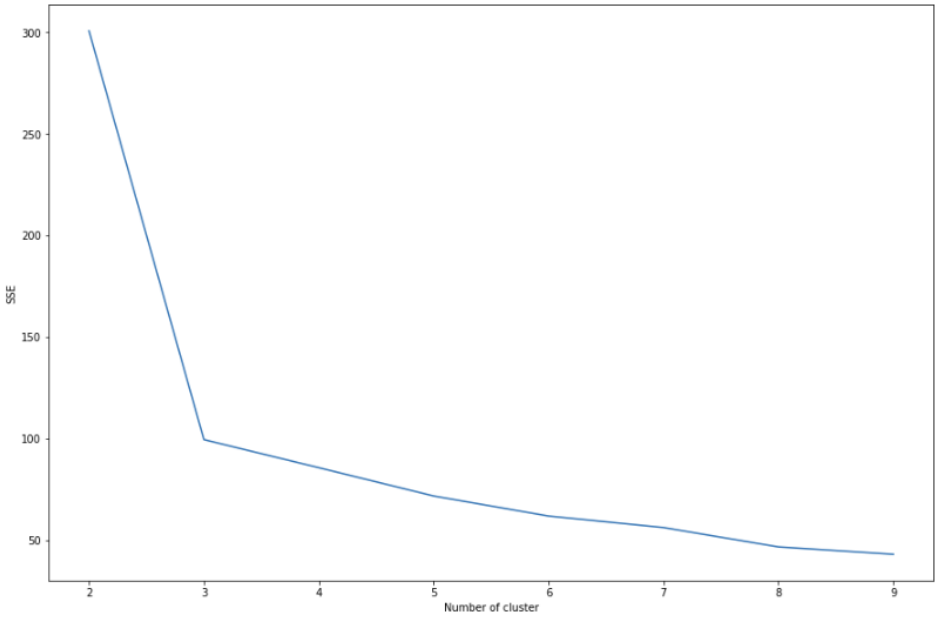
[3]:
```
# normalize data
X = df.values
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)
```

[4]:
```
sse,db,slc = {}, {}, {}
```

[ ]:

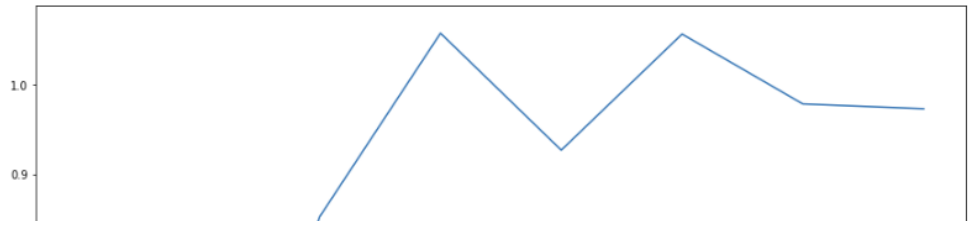
The Elbow method

[5]:
```
plt.figure(figsize=(15,10))
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.show()
```



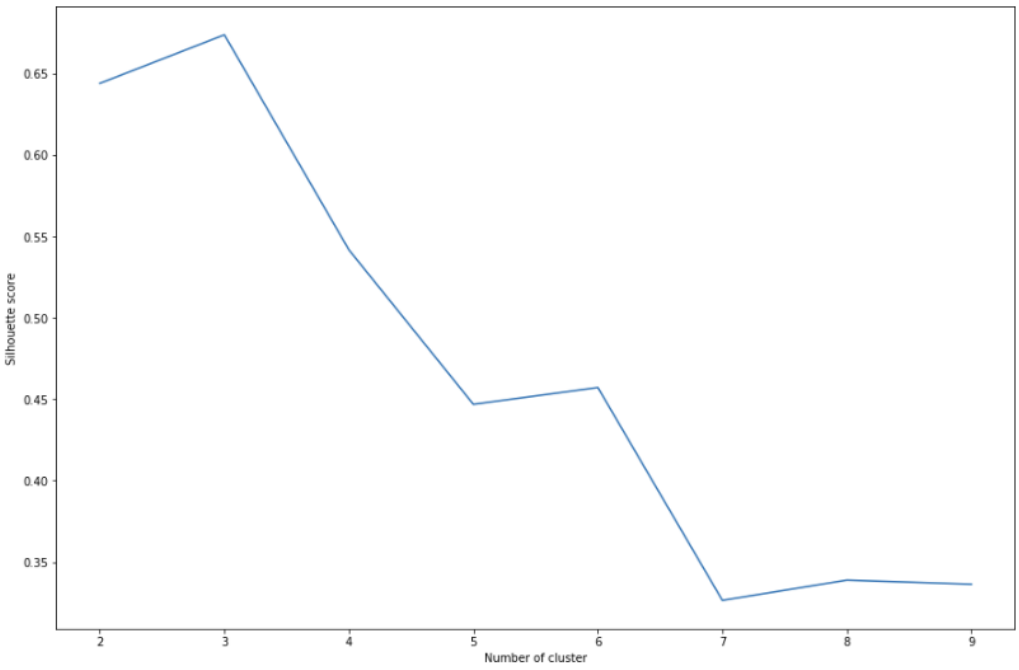
[ ]:

```
plt.ylabel("Davies-Bouldin values")
plt.show()
```



Silhouette Analysis

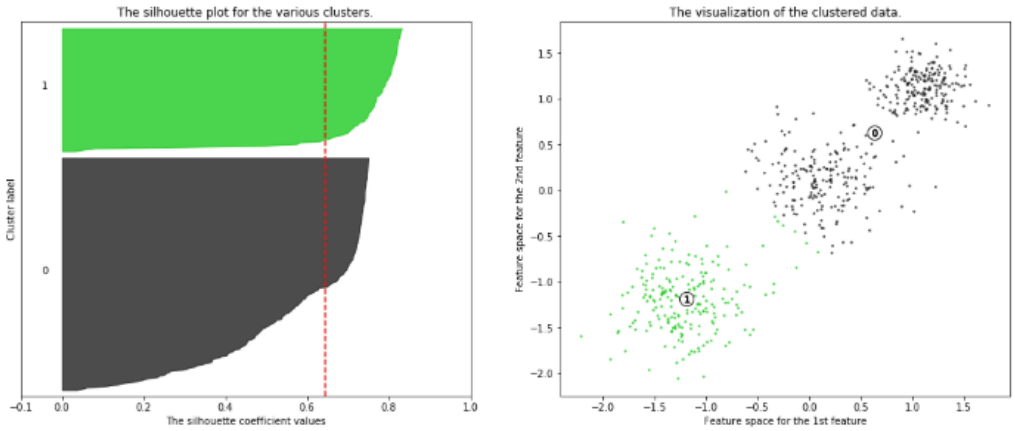
```
plt.figure(figsize=(15,10))
plt.plot(list(slc.keys()), list(slc.values()))
plt.xlabel("Number of cluster")
plt.ylabel("Silhouette score")
plt.show()
```



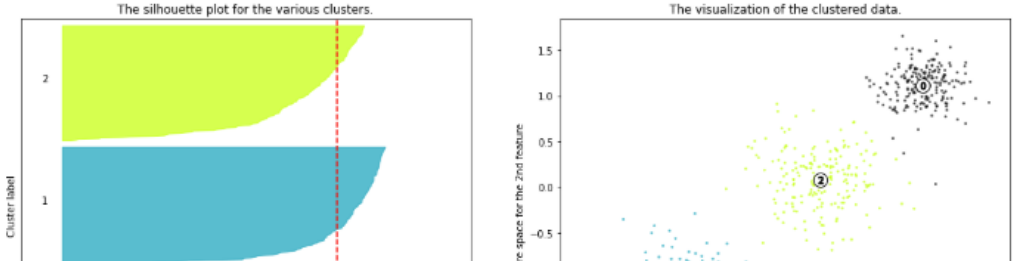
```
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])
    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters,max_iter=1000, random_state=10)
    cluster_labels = clusterer.fit_predict(X)
    # 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, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)
    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples
```

```
For n_clusters = 2 The average silhouette_score is : 0.6439937852306329
For n_clusters = 3 The average silhouette_score is : 0.6738163984313349
For n_clusters = 4 The average silhouette_score is : 0.5417842653985825
For n_clusters = 5 The average silhouette_score is : 0.4468613992497237
For n_clusters = 6 The average silhouette_score is : 0.45717363050117127
For n_clusters = 7 The average silhouette_score is : 0.32643665817784345
For n_clusters = 8 The average silhouette_score is : 0.33887140393048576
For n_clusters = 9 The average silhouette_score is : 0.3363570072965369
```

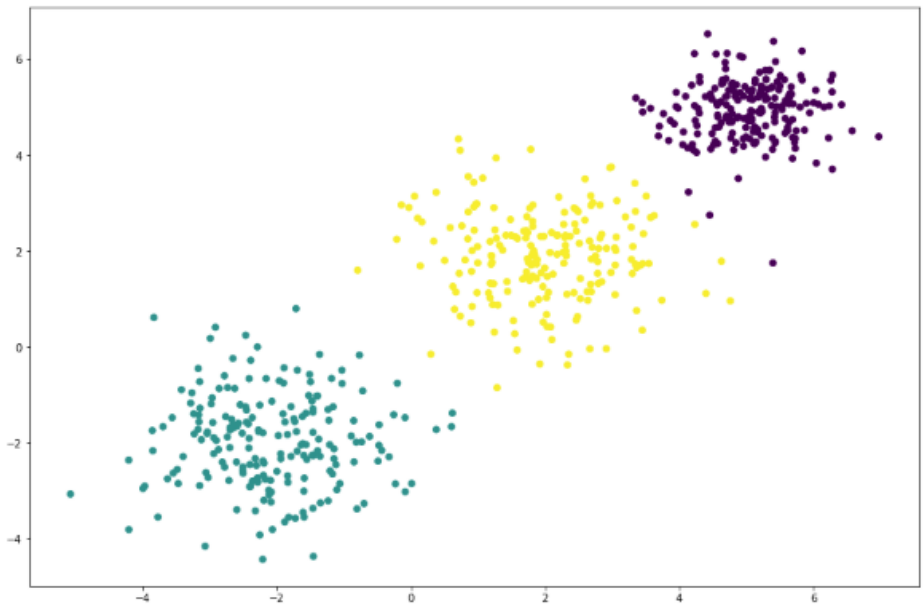
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



```
plt.scatter(X[:,0],X[:,1],c=labels)
plt.show()
```



🚀 For people who like video courses and want to **kick-start a career in data science** today, I highly recommend the below video course from Udacity:

📚 While for book lovers:

- **"Python for Data Analysis"** by Wes McKinney**,** best known for creating the Pandas project.
- **"Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow"** by Aurelien Geron, currently ranking first in

- **"Deep Learning"** by **Ian Goodfellow** research scientist at OpenAI.

🤘 Conclusion

This brings us to the end of this article. Hope you become aware of the different techniques for validating the quality of clustering.

. . .

Thanks for reading; if you liked this article, please consider subscribing to my blog. That way I get to know that my work is valuable to you and also notify you for future articles.

💪💪💪💪 As always keep studying, keep creating 🔥🔥🔥🔥

🔘 References

- https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

MORE IN **DATA SCIENCE**

Mlxtend: Feature Selection Tutorial
11 Mar 2020 – 4 min read

Implementation of K-means from scratch in Python (9 lines)
26 Feb 2020 – 4 min read

How to create and add a conda environment as Jupyter Kernel?
19 Feb 2020 – 3 min read

See all 30 posts →

DATA SCIENCE

Mlxtend: Feature Selection Tutorial

In this article, I present Mlxtend (machine learning extensions), a Python library of useful tools for the day-to-day data science tasks. To showcase its strength I use the library to

GEORGIOS DRAKOS
11 MAR 2020 • 4 MIN READ

You've successfully subscribed to GDCoder!

DATA SCIENCE

Implementation of K-means from scratch in Python (9 lines)

Last week, I was asked to implement the K-Means clustering algorithm from scratch in python as part of my MSc Data Science Degree Apprenticeship from the University of Exeter. In

GEORGIOS DRAKOS
26 FEB 2020 • 4 MIN READ