

K-means clustering: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2020

Syntax

- Computing the Euclidean distance in Python:

```
def calculate_distance(vec1, vec2):  
    root_distance = 0  
  
    for x in range(0, len(vec1)):  
        difference = centroid[x] - player_values[x]  
        squared_difference = difference**2  
        root_distance += squared_difference  
  
    euclid_distance = math.sqrt(root_distance)  
  
    return euclid_distance
```

- Assigning observations to clusters:

```
def assign_to_cluster(row):  
    lowest_distance = -1  
    closest_cluster = -1  
  
    for cluster_id, centroid in centroids_dict.items():  
        df_row = [row['ppg'], row['atr']]  
        euclidean_distance = calculate_distance(centroid, df_row)  
  
        if lowest_distance == -1:  
            lowest_distance = euclidean_distance  
            closest_cluster = cluster_id  
        elif euclidean_distance < lowest_distance:  
            lowest_distance = euclidean_distance  
            closest_cluster = cluster_id  
  
    return closest_cluster
```

- Initializing the KMeans class from scikit-learn:

```
from sklearn.cluster import KMeans  
  
kmeans_model = KMeans(n_clusters=2, random_state=1)
```

Concepts

- Centroid-based clustering works well when the clusters resemble circles with centers.
- K-Means clustering is a popular centroid-based clustering algorithm. The K refers to the number of clusters we want to segment our data into. K-Means clustering is an iterative algorithm that switches between recalculating the centroid of each cluster and the items that belong to each cluster.
- Euclidean distance is the most common technique used in data science for measuring distance between vectors. The formula for distance in two dimensions is:

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2}$$

where q and p are the two vectors we are comparing.

- Example: If **q** is (5,2) and **p** is (3,1), the distance comes out to:
 $\sqrt{(5 - 3)^2 + (2 - 1)^2} = \sqrt{5} \approx 2.23607$.
- If clusters look like they don't move a lot after every iteration, this means two things:
 - K-Means clustering doesn't cause massive changes in the makeup of clusters between iterations, meaning that it will always converge and become stable.
 - Where we pick the initial centroids and how we assign elements to clusters initially matters a lot because K-Means clustering is conservative between iterations.
- To counteract the problems listed above, the **sklearn** implementation of K-Means clustering does some intelligent things like re-running the entire clustering process lots of times with random initial centroids so the final results are a little less biased.

Resources

- [Sklearn implementation of K-Means clustering](#)
- [Implementing K-Means clustering from scratch](#)

