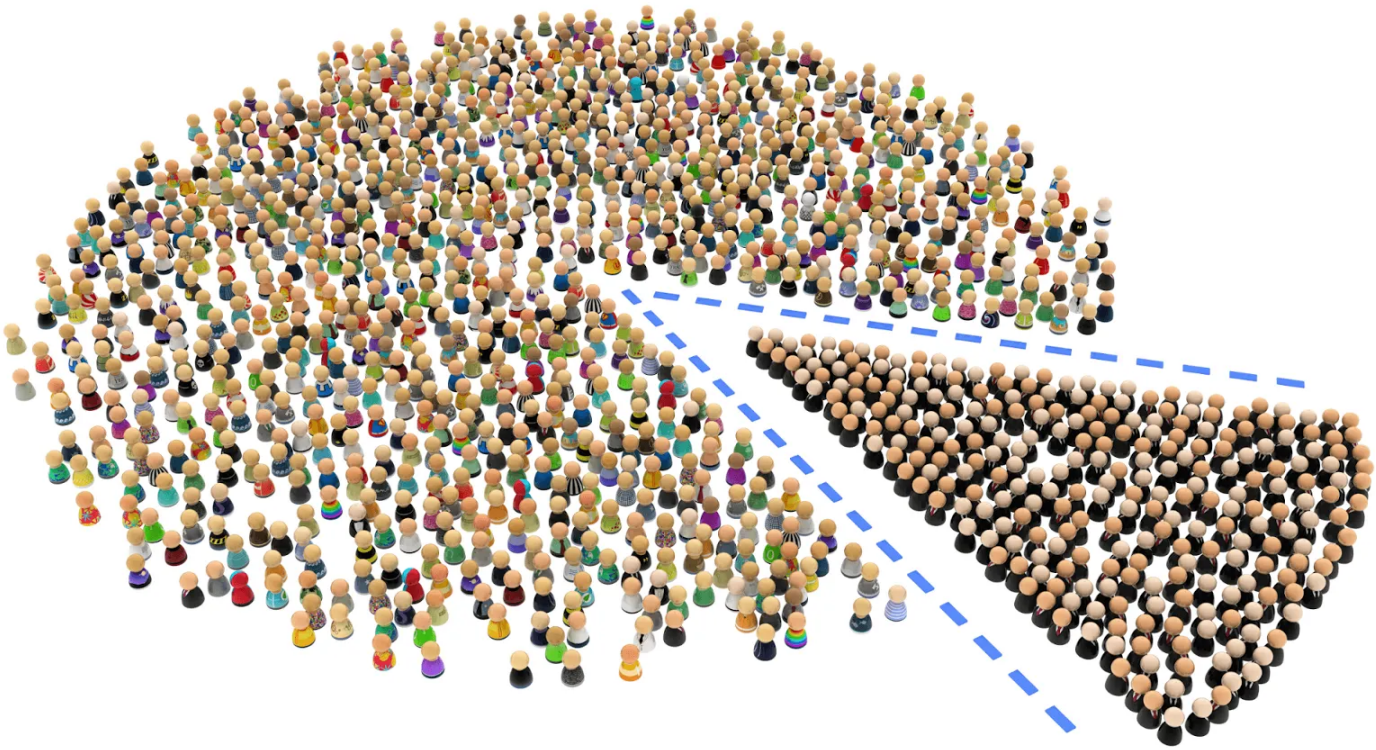
Step by Step to Understanding K-means Clustering and Implementation with sklearn

Arif R · Follow

Published in Data Folks Indonesia

6 min read · Oct 4, 2020

Simple explanation regarding K-means Clustering in Unsupervised Learning and simple practice with sklearn in python



source : How to Use Customer Segmentation in Google Analytics to Build Your Buyer Personal

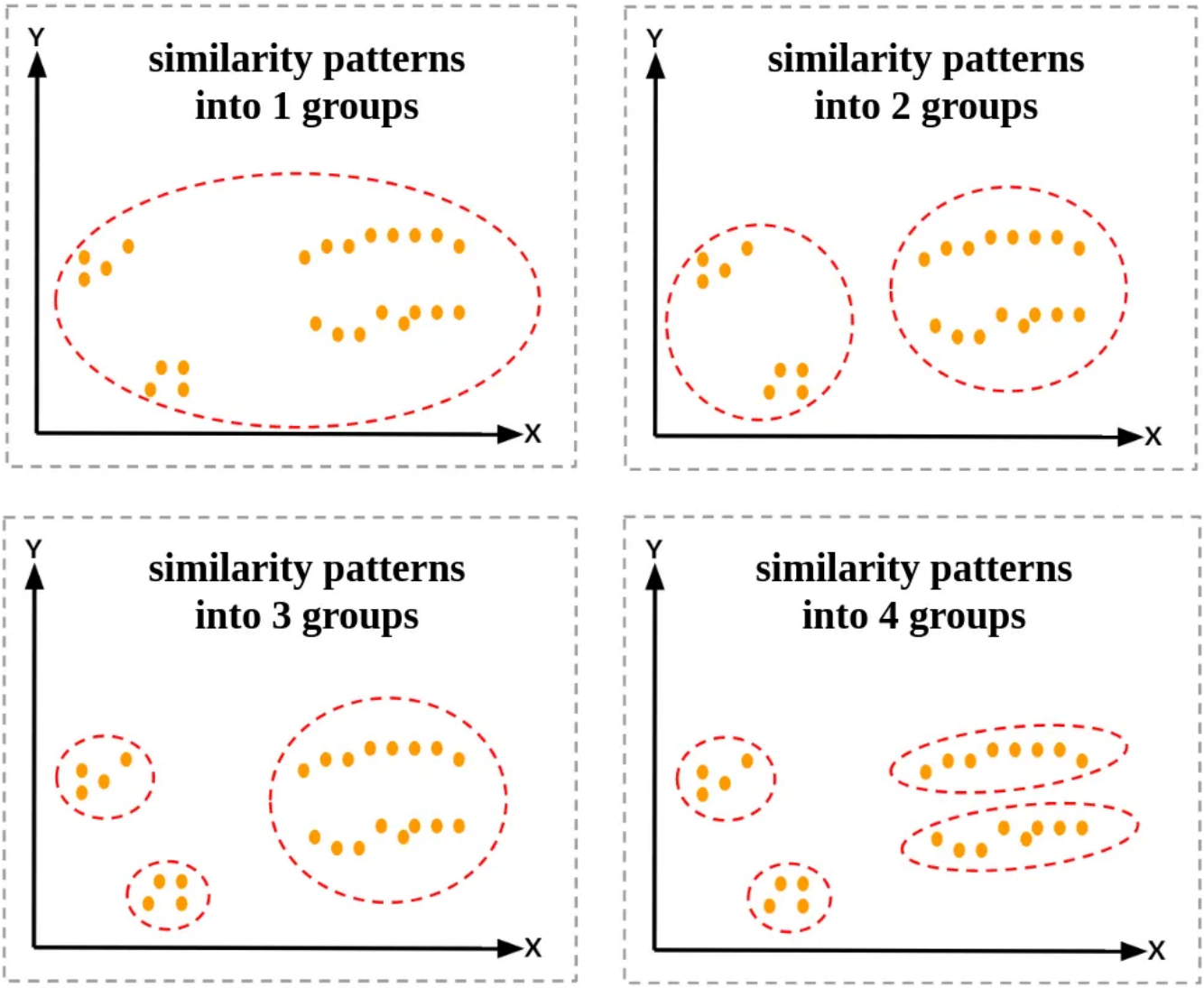
Previous article :

Machine Learning Explanation : Supervised Learning & Unsupervised Learning and Understanding Clustering in Unsupervised Learning

Remember Clustering Intuition ?

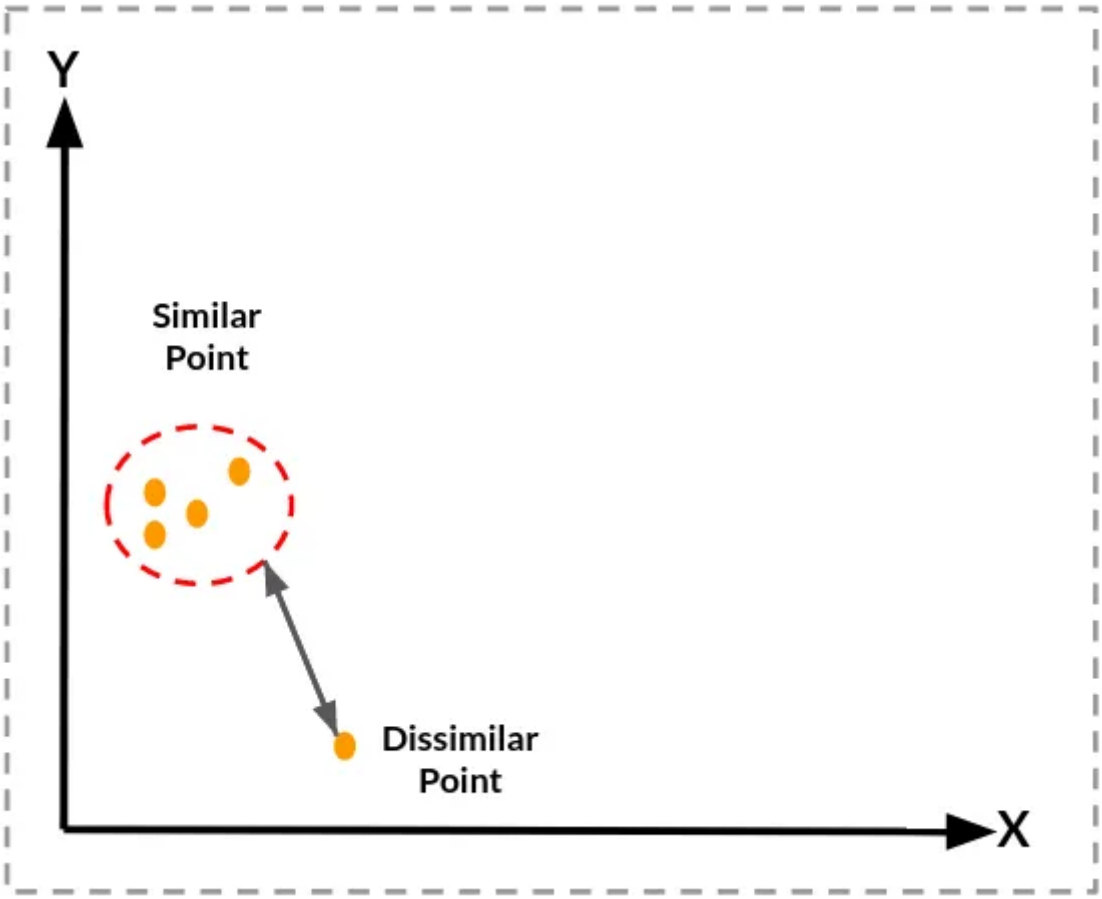
In the previous article, I was explained regarding Clustering Intuition.

> Clustering : grouping data based on similarity patterns based on distance



How do we know a point has the same group as another point?

> *As mentioned above : based on similarity patterns*

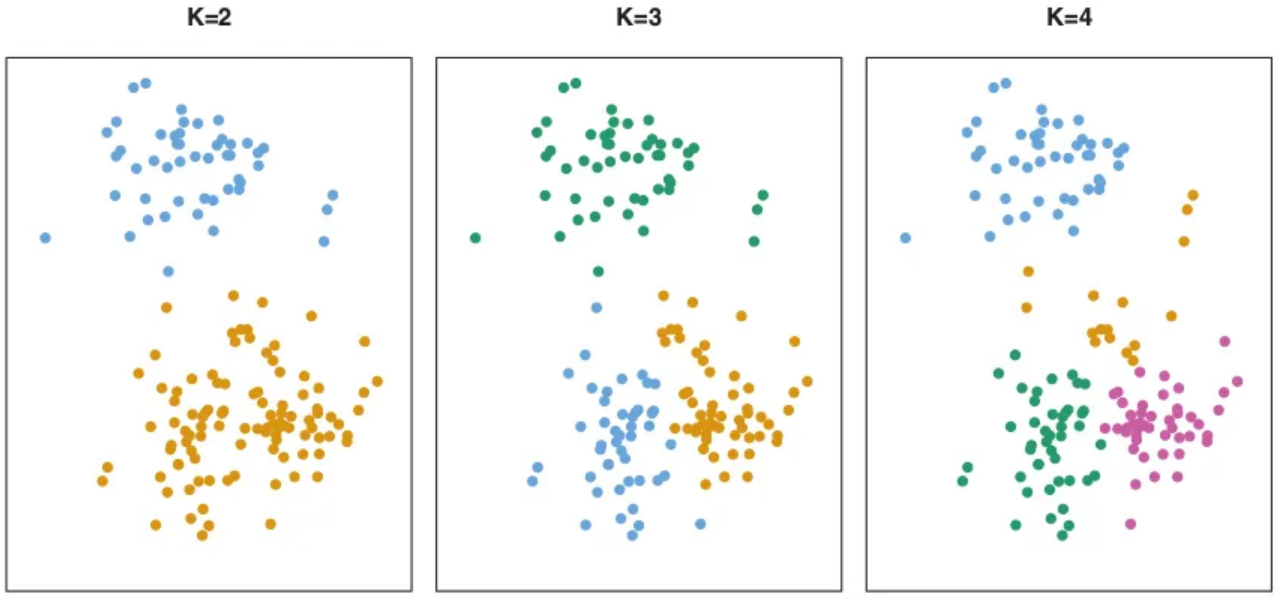


How to know the pattern similarity with K-means clustering method?

Before answering that, let's discuss what k-means is and how k-means does clustering

THEORY — What is K-means ?

K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters. To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters [1].



source : Intoduction to Statistical Learning with Applications in R page 387

The K-means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. The K-means algorithm aims to choose centroid that minimise the **inertia**, or **within-cluster sum-of-squares criterion** [2]:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

picture from scikit-learn — clustering

How does it work?

There are several steps to explain it. For instance, there is data provided below

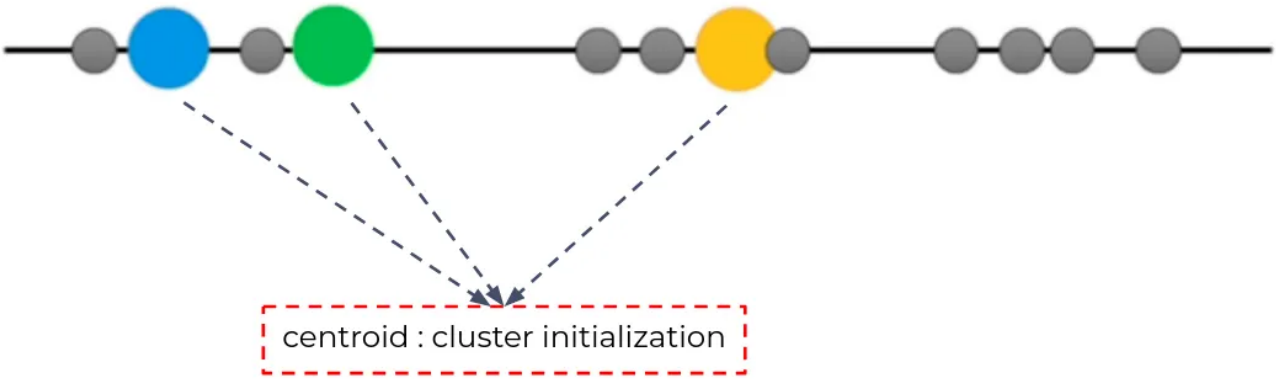
Here, I will explain step by step how k-means works

Step 1. Determine the value "K", the value "K" represents the number of clusters.

in this case, we'll select K=3. That is to say, we want to identify 3 clusters. **Is there any way to determine the value of K? yes there is, but we'll talk later about it.**

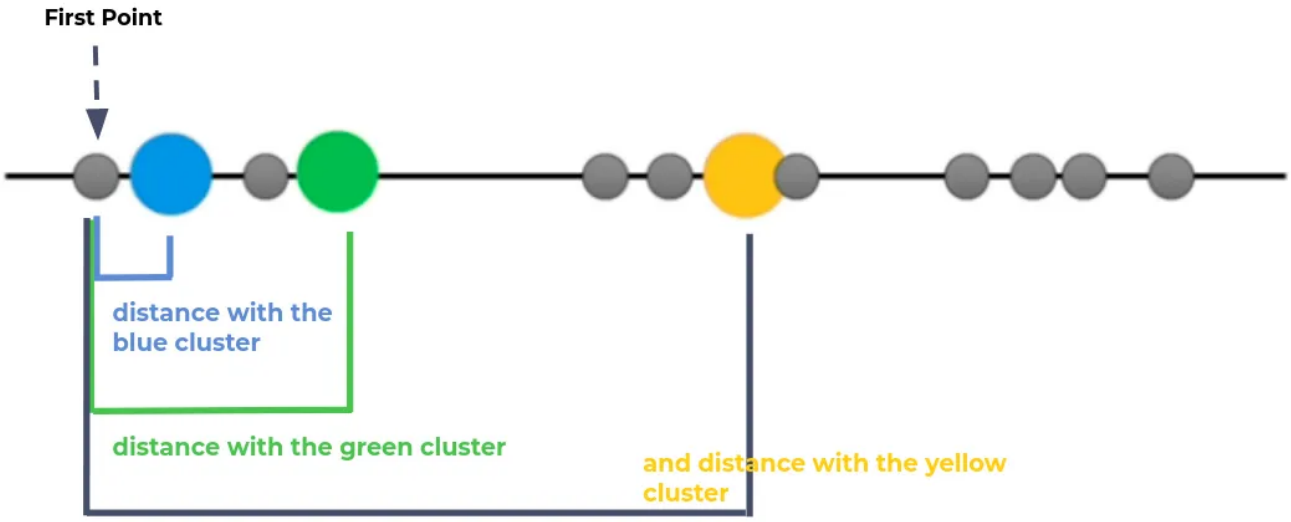
Step 2. Randomly select 3 distinct centroid (new data points as cluster initialization)

for example — attempts 1. “K” is equal 3 so there are 3 centroid, in which case it will be the cluster initialization



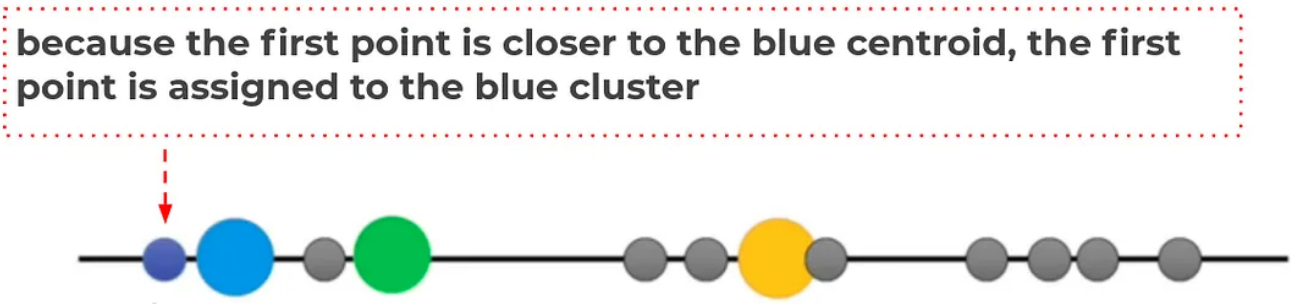
Step 3. Measure the distance (euclidean distance) between each point and the centroid

for example, measure the distance between first point and the centroid.

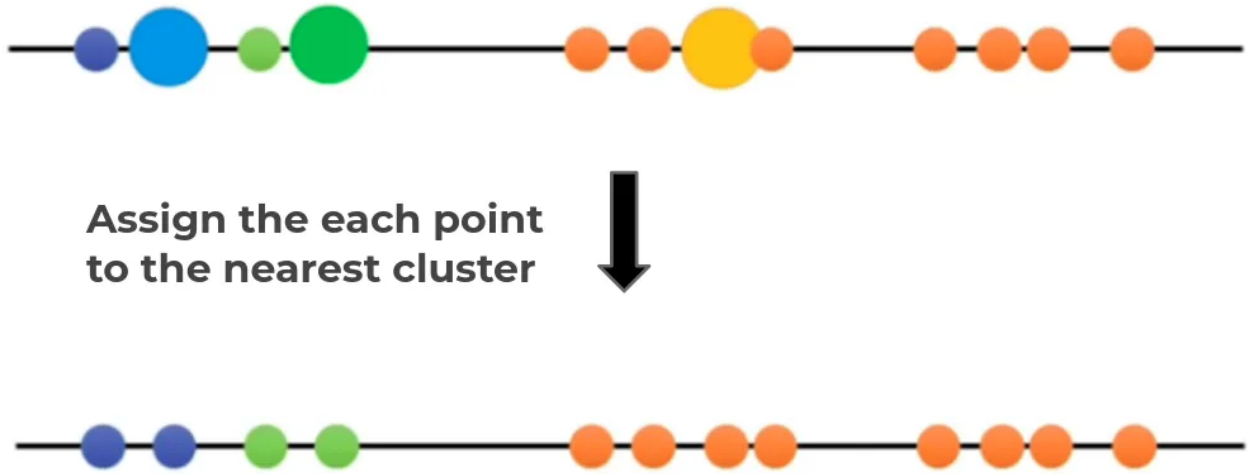


Step 4. Assign the each point to the nearest cluster

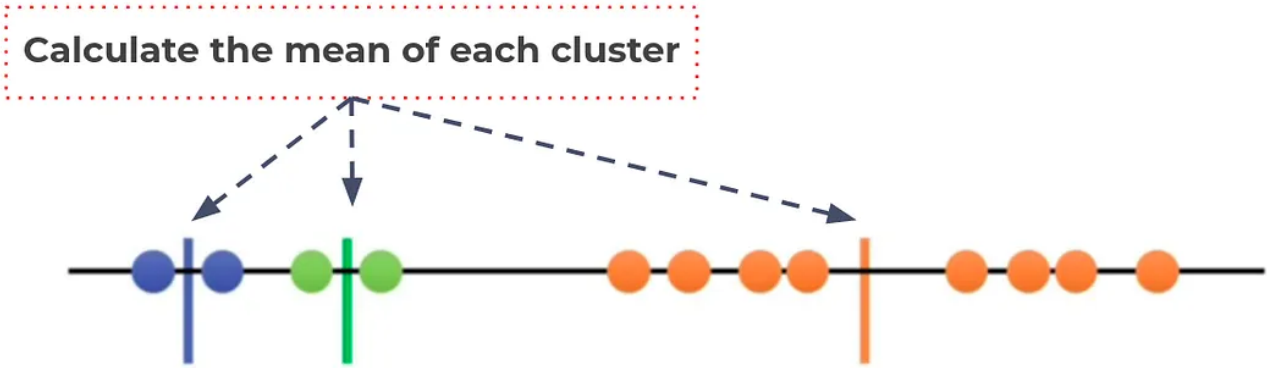
for example, measure the distance between first point and the centroid.



Do the same treatment for the other unlabeled point, until we get this

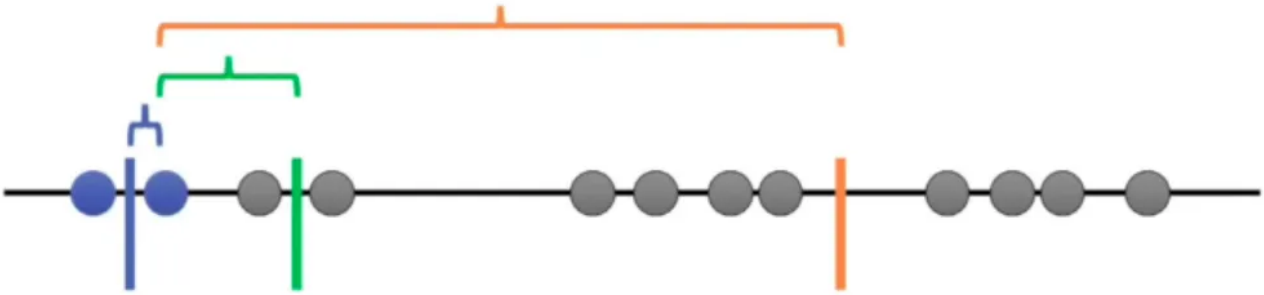


Step 5. Calculate the mean of each cluster as new centroid



Update the centroid with mean of each cluster

Step 6. Repeat step 3–5 with the new center of cluster



Repeat until stop:

- Convergence. (No further changes)

- Maximum number of iterations.

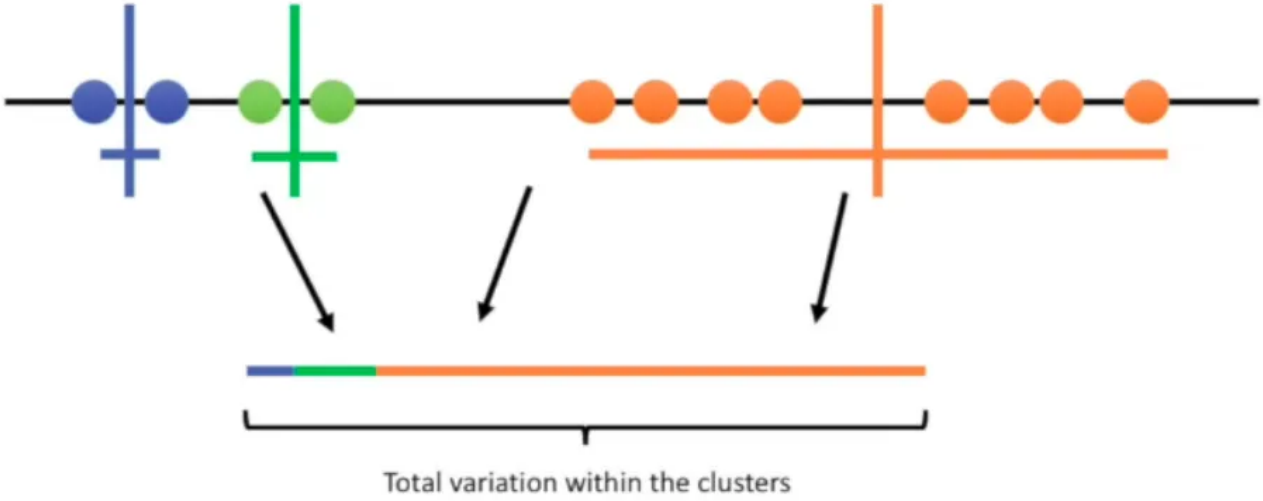
Since the clustering did not change at all during the last iteration, we're done.

Has this process been completed? of course not

Remember, the K-means algorithm aims to choose centroid that minimise the inertia, or within-cluster sum-of-squares criterion

So, how to assess the results of this clustering? let's continue

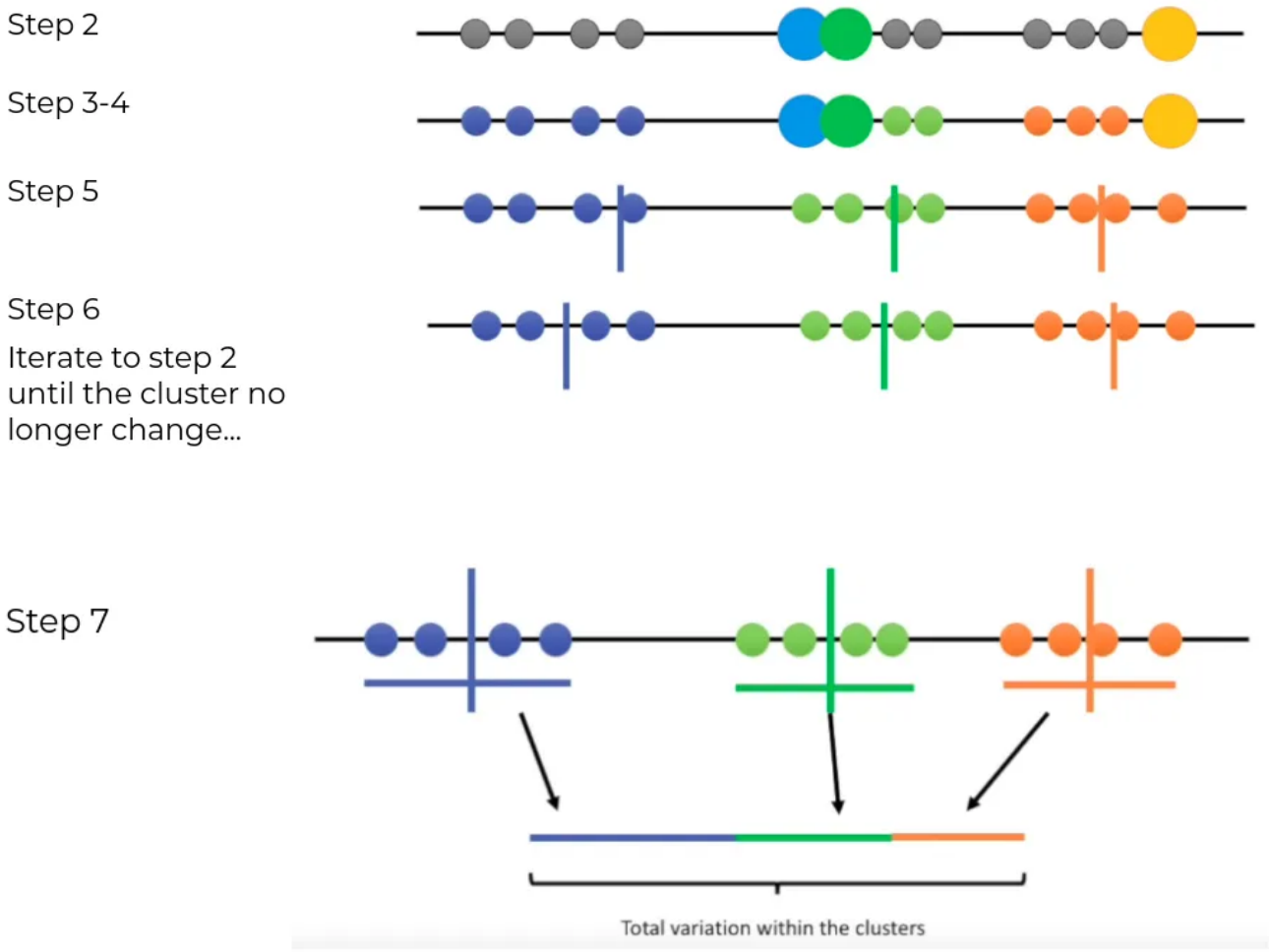
Step 7. Calculate the variance of each cluster



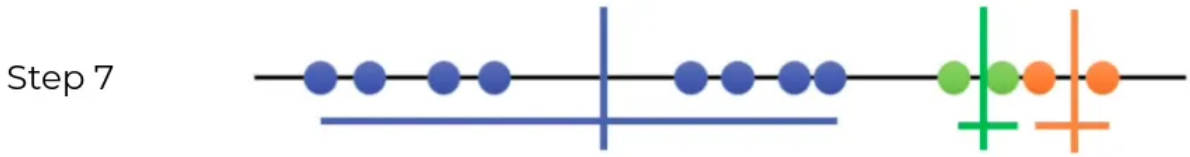
Since K-means clustering can't "see" the best clustering, it is only option is to keep track of these clusters, and their total variance, and do the whole thing over again with different starting points.

Step 8. Repeat step 2–7 until get the lowest sum of variance

For example — attempts 2 with different random centroid



For example — attempts 3 with different random centroid

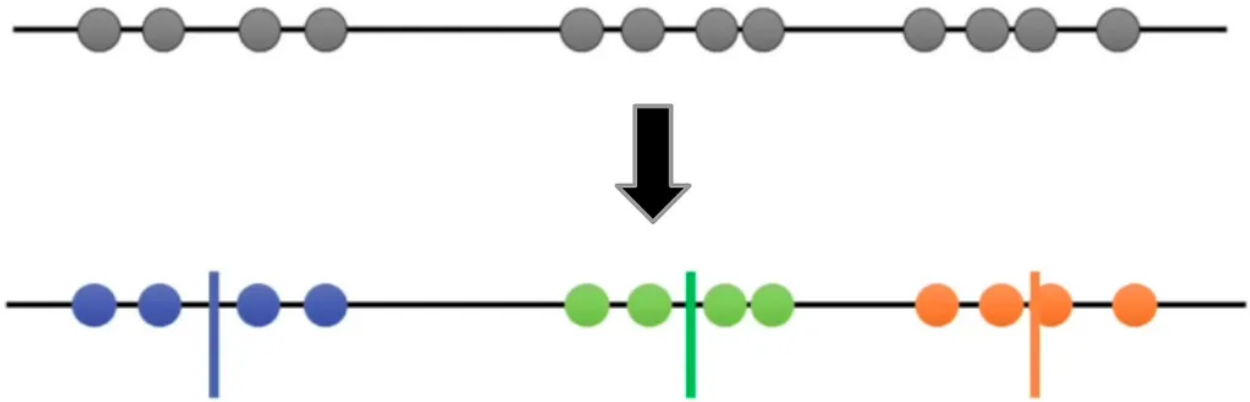


Repeat until stop:

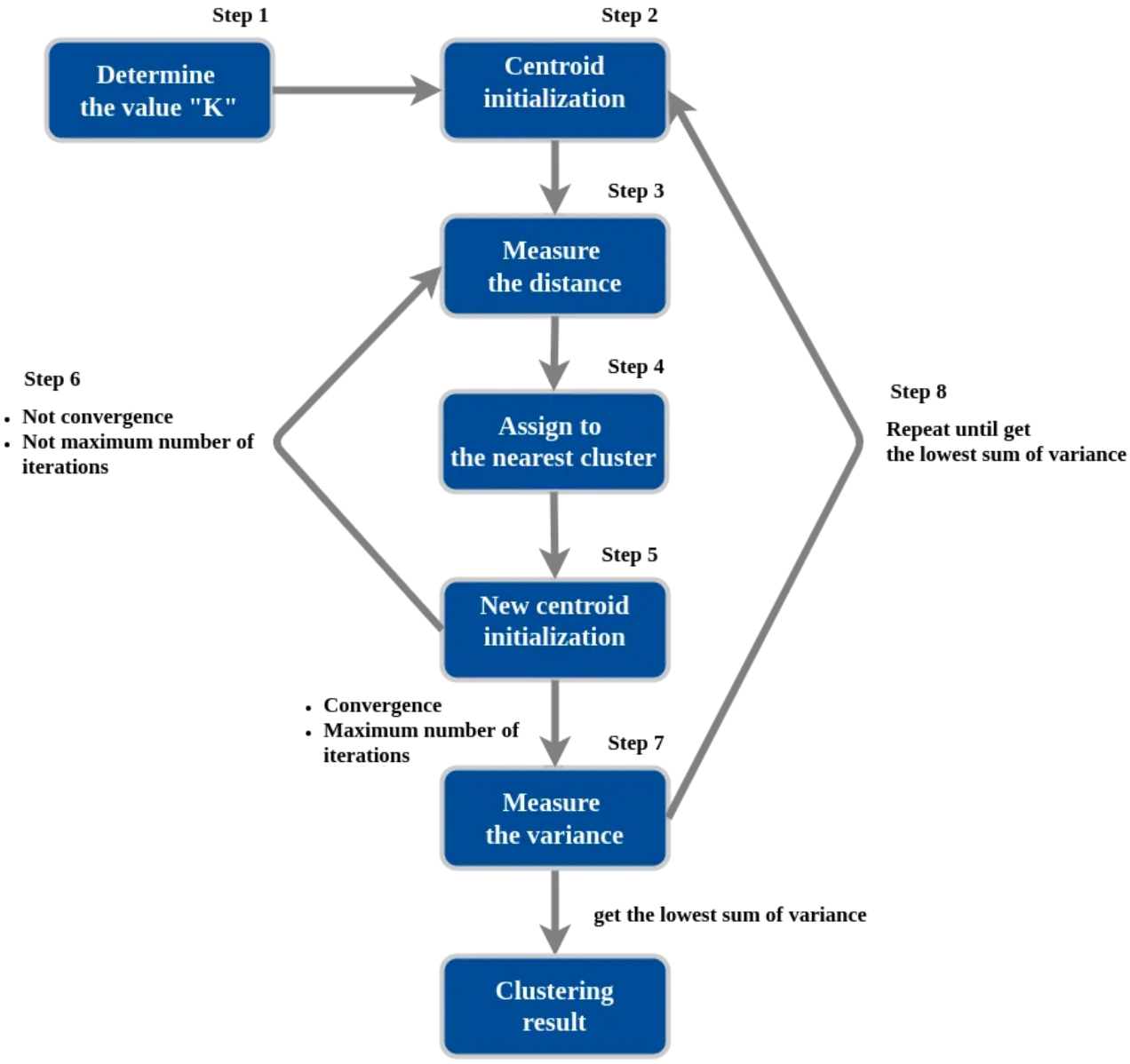
- **Until we get the lowest sum of variance and pick those cluster as our result**

Has this process been completed? Yes

The result of clustering is



K-means Clustering Overview

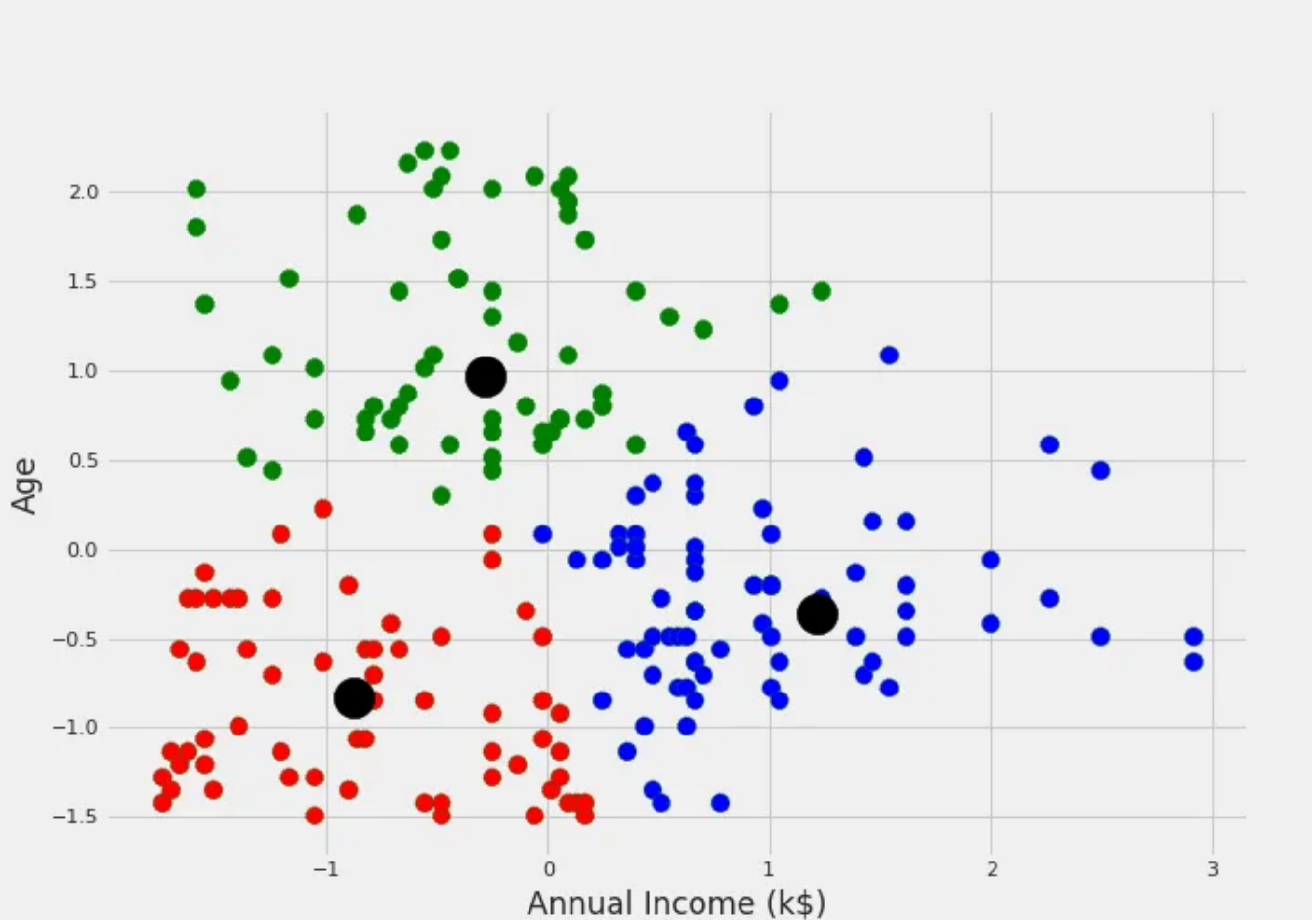


PRACTICE —How to implement K-means with sklearn in python?

The dataset is available --> here

Clustering with two feature

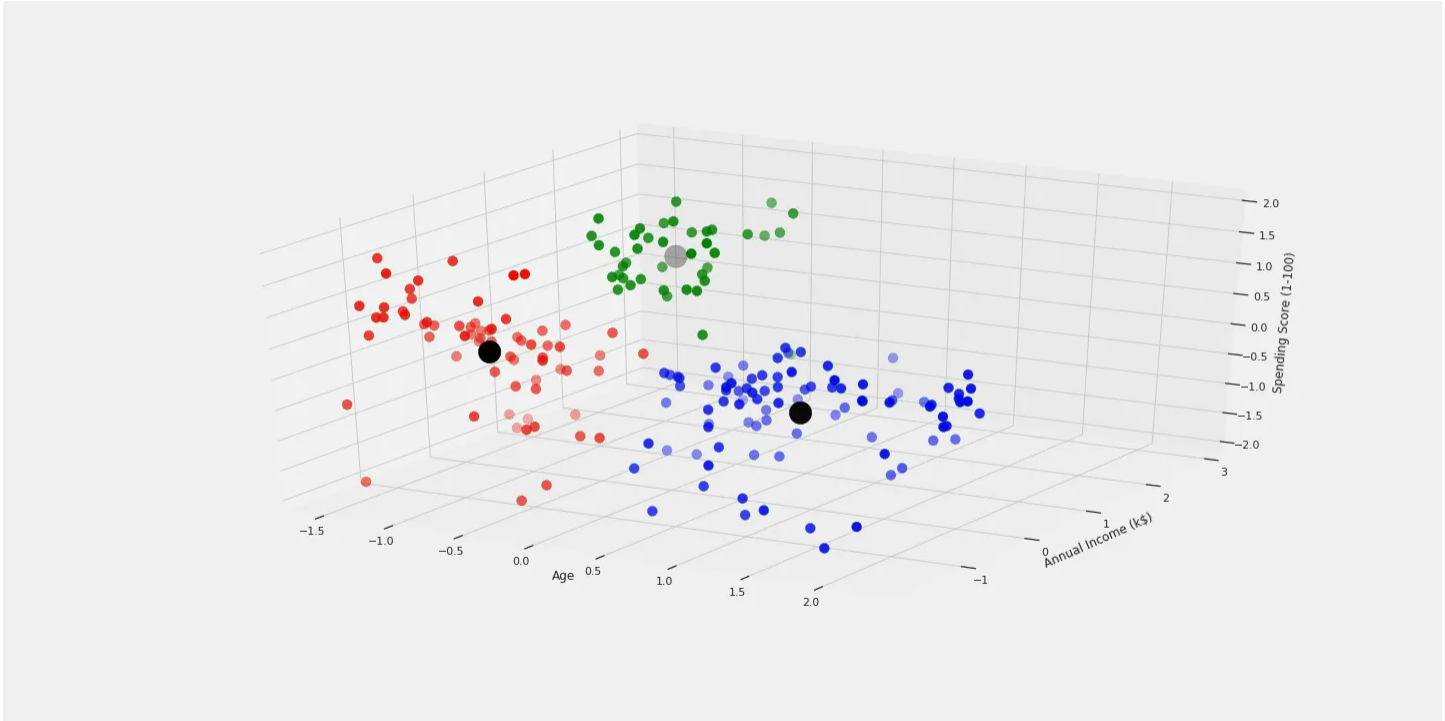
```
## Import Library
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

## Load Data
dfa = pd.read_csv("mall_customers.csv")
dfa = dfa[['Age','Annual Income (k$)']]
print('Total Row : ', len(dfa))

## Feature Scaling
sc_dfa = StandardScaler()
dfa_std = sc_dfa.fit_transform(dfa.astype(float))

## Clustering with KMeans
kmeans = KMeans(n_clusters=3, random_state=42).fit(dfa_std)
labels = kmeans.labels_

new_dfa = pd.DataFrame(data = dfa_std, columns = ['Age','Annual
Income (k$)'])
new_dfa['label_kmeans'] = labels

fig, ax = plt.subplots(figsize=(10,7))
plt.scatter(new_dfa["Annual Income (k$)"][new_dfa["label_kmeans"] ==
0], new_dfa["Age"][new_dfa["label_kmeans"] == 0],
            color = "blue", s=100, edgecolor='green',linestyle='--')

plt.scatter(new_dfa["Annual Income (k$)"][new_dfa["label_kmeans"] ==
1], new_dfa["Age"][new_dfa["label_kmeans"] == 1],
            color = "red", s=100, edgecolor='green',linestyle='--')

plt.scatter(new_dfa["Annual Income (k$)"][new_dfa["label_kmeans"] ==
2], new_dfa["Age"][new_dfa["label_kmeans"] == 2],
            color = "green", s=100, edgecolor='green',linestyle='--
')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=500);

ax.set_xlabel('Annual Income (k$)')
ax.set_ylabel('Age')
plt.show()
```



Clustering with three feature

```
## Load Data
dfa = pd.read_csv("mall_customers.csv")
dfa = dfa[['Age','Annual Income (k$)','Spending Score (1-100)']]
print('Total Row : ', len(dfa))

## Feature Scaling
sc_dfa = StandardScaler()
dfa_std = sc_dfa.fit_transform(dfa.astype(float))

## Clustering with KMeans
kmeans = KMeans(n_clusters=3, random_state=42).fit(dfa_std)
labels = kmeans.labels_

new_dfa = pd.DataFrame(data = dfa_std, columns = ['Age','Annual
Income (k$)','Spending Score (1-100)'])
new_dfa['label_kmeans'] = labels

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(new_dfa.Age[new_dfa.label_kmeans == 0], new_dfa["Annual
Income (k$)"][new_dfa.label_kmeans == 0], new_dfa["Spending Score
(1-100)"][new_dfa.label_kmeans == 0], c='blue', s=100,
edgecolor='green',linestyle=' – ')
ax.scatter(new_dfa.Age[new_dfa.label_kmeans == 1], new_dfa["Annual
Income (k$)"][new_dfa.label_kmeans == 1], new_dfa["Spending Score
(1-100)"][new_dfa.label_kmeans == 1], c='red', s=100,
```

```
edgecolor='green',linestyle=' – ')
ax.scatter(new_dfa.Age[new_dfa.label_kmeans == 2], new_dfa["Annual
Income (k$)"][new_dfa.label_kmeans == 2], new_dfa["Spending Score
(1-100)"][new_dfa.label_kmeans == 2], c='green', s=100,
edgecolor='green',linestyle=' – ')
ax.scatter(new_dfa.Age[new_dfa.label_kmeans == 3], new_dfa["Annual
Income (k$)"][new_dfa.label_kmeans == 3], new_dfa["Spending Score
(1-100)"][new_dfa.label_kmeans == 3], c='orange', s=100,
edgecolor='green',linestyle=' – ')
ax.scatter(new_dfa.Age[new_dfa.label_kmeans == 4], new_dfa["Annual
Income (k$)"][new_dfa.label_kmeans == 4], new_dfa["Spending Score
(1-100)"][new_dfa.label_kmeans == 4], c='purple', s=100,
edgecolor='green',linestyle=' – ')

centers = kmeans.cluster_centers_
ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2], c='black',
s=500);

plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



Linkedin arif romadhan

My Website : https://komuternak.com/

Reference