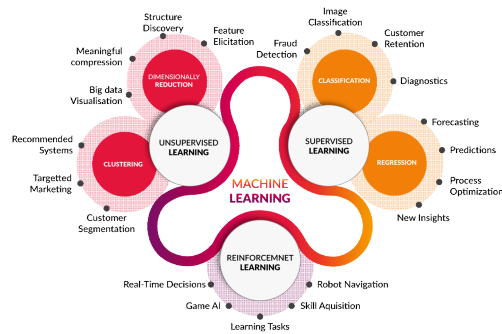
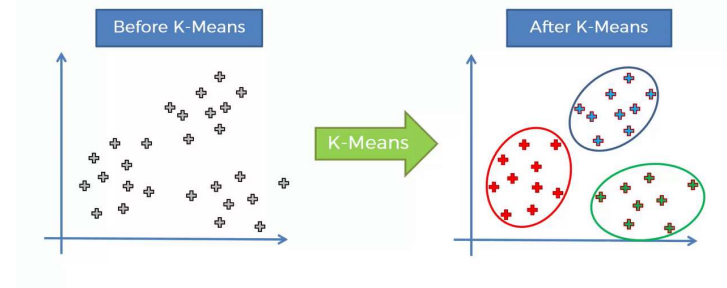


Clustering

Machine Learning
Dr. Adnan Abid

What K-Means does for you



How did it do that ?

STEP 1: Choose the number K of clusters



STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



STEP 3: Assign each data point to the closest centroid → That forms K clusters



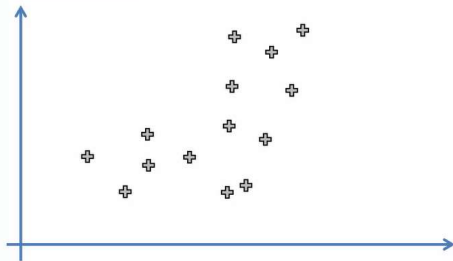
STEP 4: Compute and place the new centroid of each cluster



STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

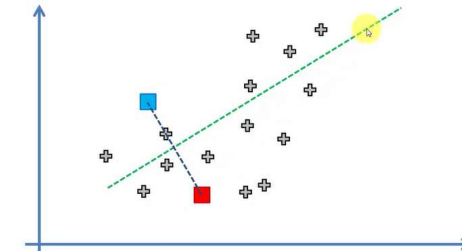
K-Means algorithm

STEP 1: Choose the number K of clusters: $K = 2$



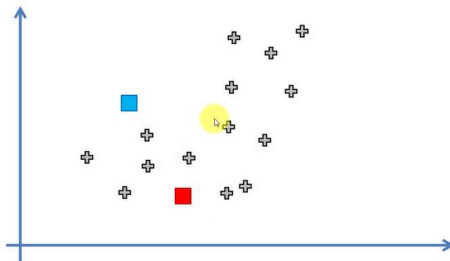
K-Means algorithm

STEP 3: Assign each data point to the closest centroid → That forms K cluster:



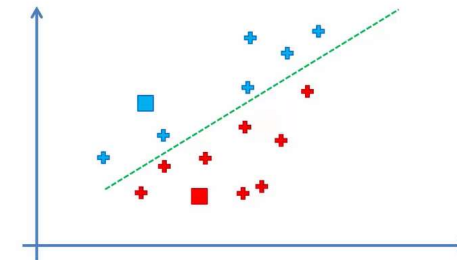
K-Means algorithm

STEP 3: Assign each data point to the closest centroid → That forms K clusters



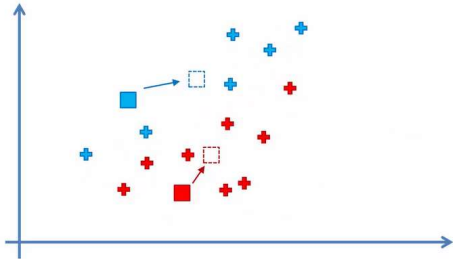
K-Means algorithm

STEP 3: Assign each data point to the closest centroid → That forms K clusters



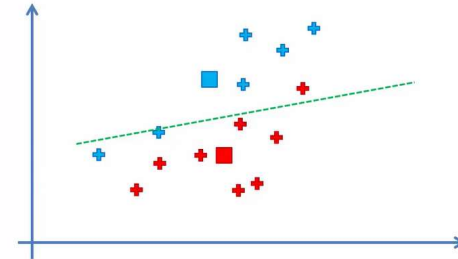
K-Means algorithm

STEP 4: Compute and place the new centroid of each cluster



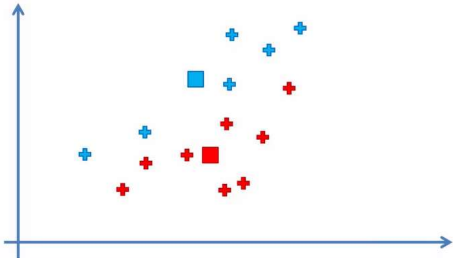
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



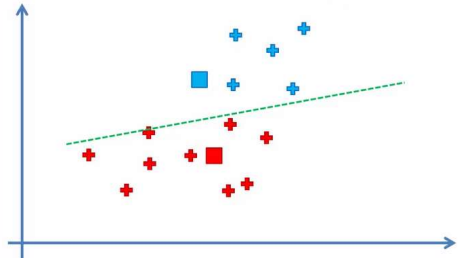
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



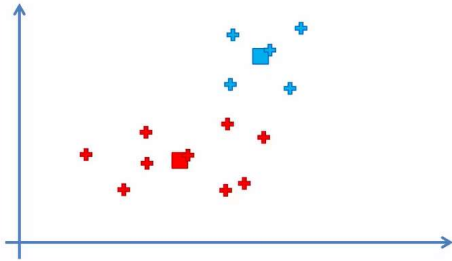
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



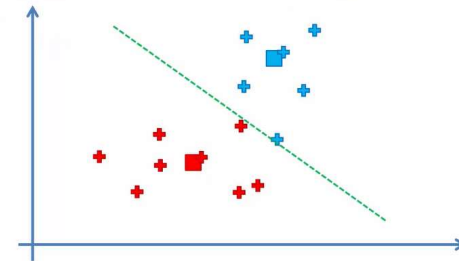
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



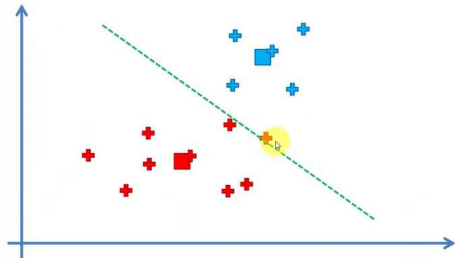
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



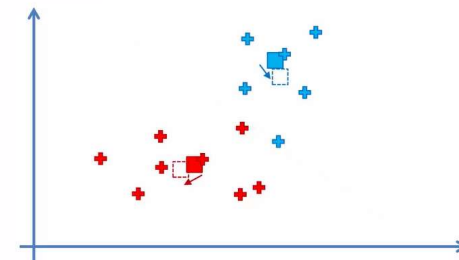
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



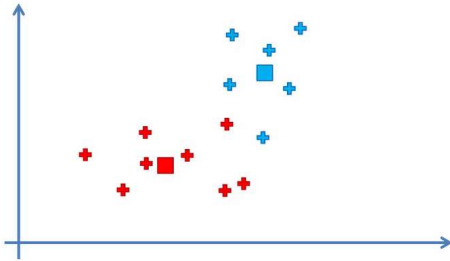
K-Means algorithm

STEP 4: Compute and place the new centroid of each cluster



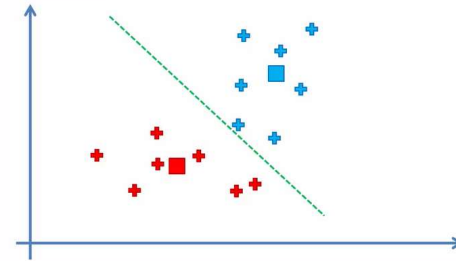
K-Means algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



K-Means algorithm

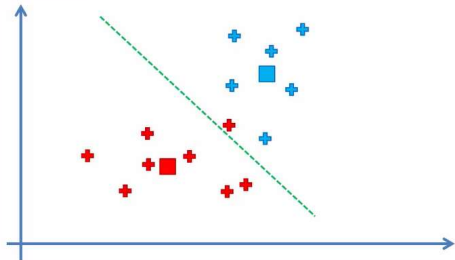
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



Arturata Windees

K-Means algorithm

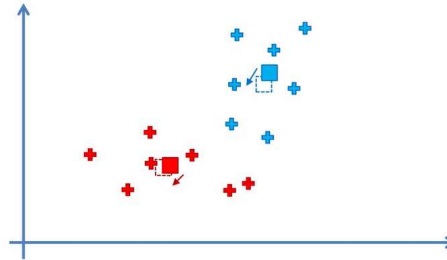
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



Arturata Windees

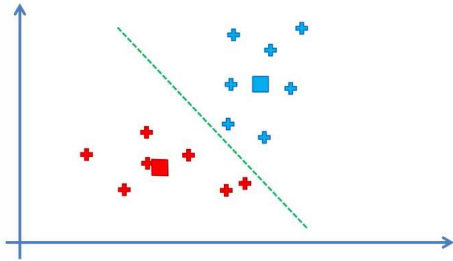
K-Means algorithm

STEP 4: Compute and place the new centroid of each cluster



K-Means algorithm

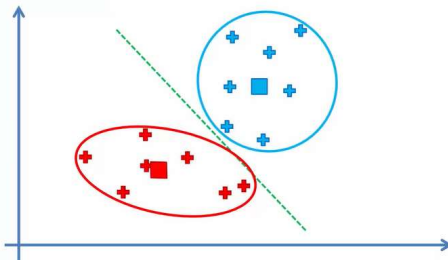
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



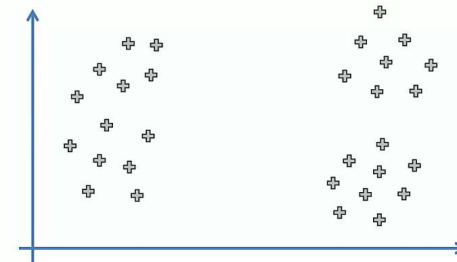
K-Means Intuition: Random Initialization Trap

K-Means algorithm

FIN: Your Model Is Ready

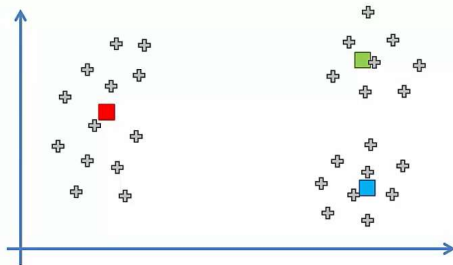


Random Initialization Trap



If we choose $K = 3$ clusters...

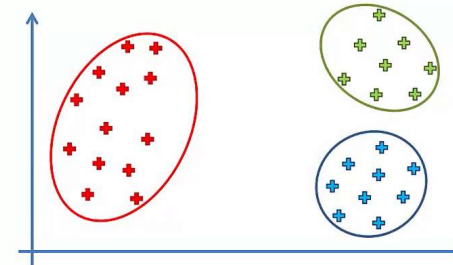
Random Initialization Trap



...this correct random initialisation would lead us to...

Activate Windows

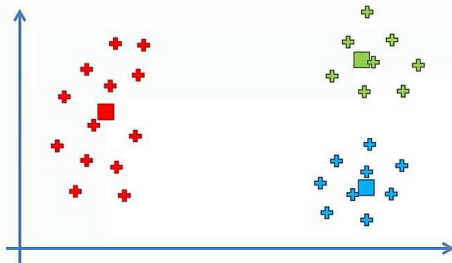
Random Initialization Trap



...the following three clusters

Activate Windows

Random Initialization Trap



...the following three clusters

Activate Windows

Random Initialization Trap

But what would happen if we had a bad random initialisation ?

Random Initialization Trap

STEP 1: Choose the number K of clusters



STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



STEP 3: Assign each data point to the closest centroid → That forms K clusters



STEP 4: Compute and place the new centroid of each cluster



STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

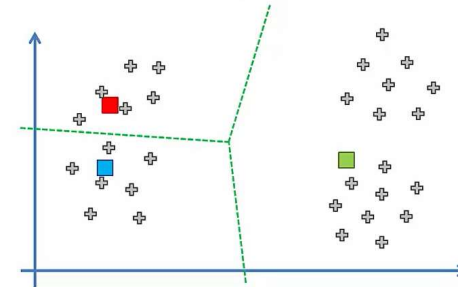


Your Model is Ready

Activate Window

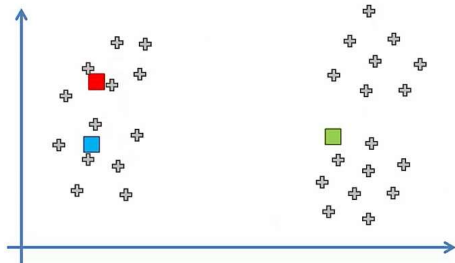
Random Initialization Trap

STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



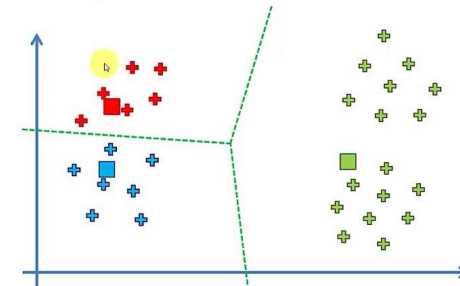
Random Initialization Trap

STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



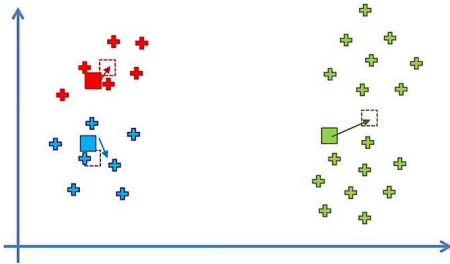
Random Initialization Trap

STEP 3: Assign each data point to the closest centroid → That forms K clusters



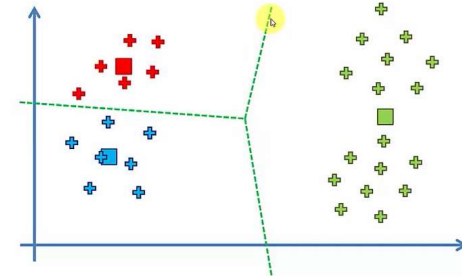
Random Initialization Trap

STEP 3: Assign each data point to the closest centroid → That forms K clusters



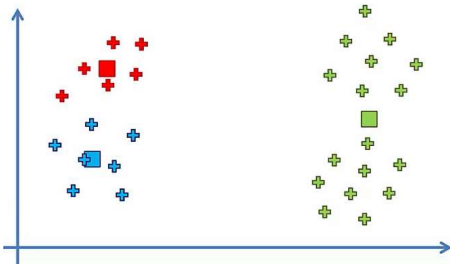
Random Initialization Trap

STEP 5: Reassign each data point to the new closest centroid. If any reassignment took place, go to STEP 4, otherwise go to FIN.



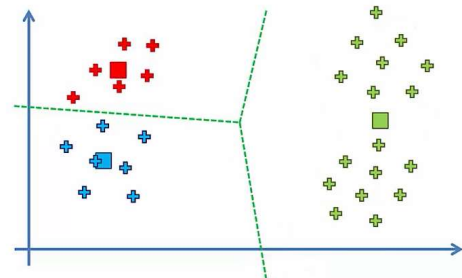
Random Initialization Trap

STEP 4: Compute and place the new centroid of each cluster



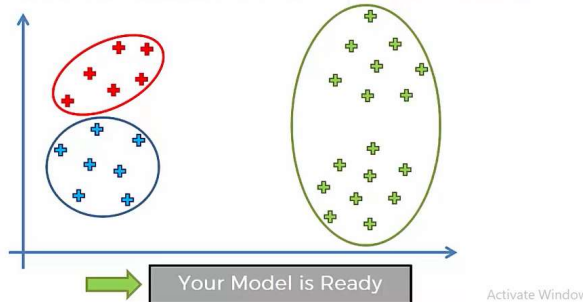
Random Initialization Trap

STEP 5: Reassign each data point to the new closest centroid. If any reassignment took place, go to STEP 4, otherwise go to FIN.



Random Initialization Trap

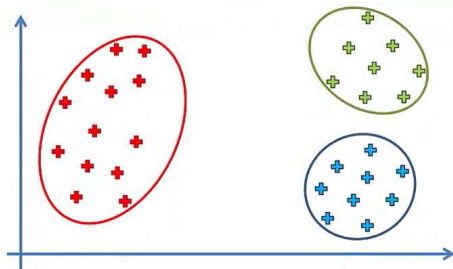
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



Random Initialization Trap

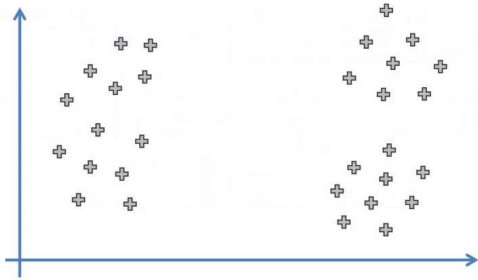


Random Initialization Trap



K-Means Intuition:
Choosing the right number
of clusters

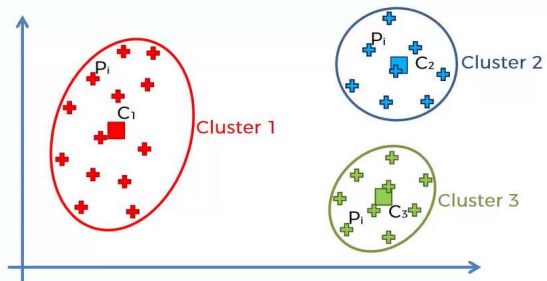
Choosing the right number of clusters



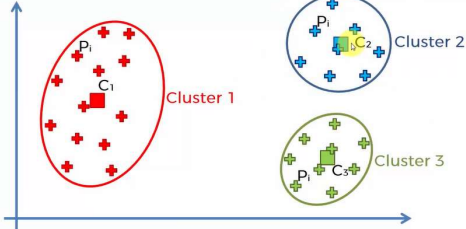
Choosing the right number of clusters

$$WCSS = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

Choosing the right number of clusters

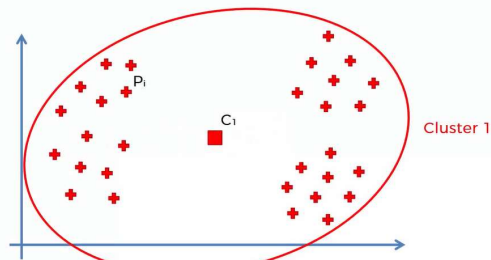


Choosing the right number of clusters



$$WCSS = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

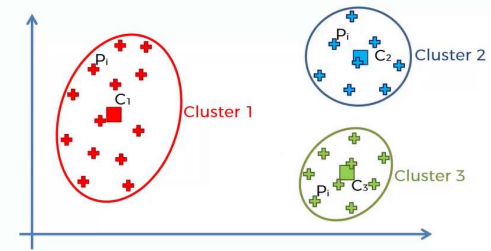
Choosing the right number of clusters



$$WCSS = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2$$

Activate Windows
Go to Settings to activate Windows.

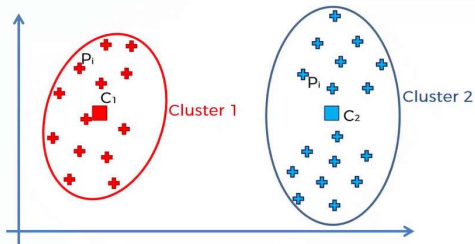
Choosing the right number of clusters



$$WCSS = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

Activate Windows
Go to Settings to activate Windows.

Choosing the right number of clusters

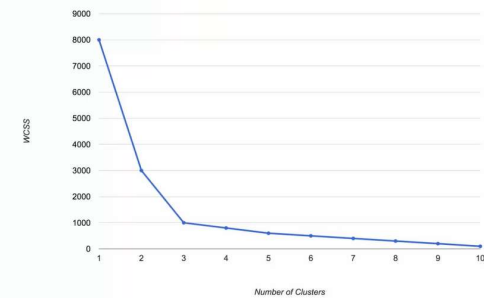


$$WCSS = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2$$

Activate Windows
Go to Settings to activate Windows.

Choosing the right number of clusters

The Elbow Method



Activate Windows
Go to Settings to activate Windows.

Dataset - DataFrame

Index	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
1	1	Male	19	15	39
2	2	Male	21	15	81
3	3	Female	28	16	6
4	4	Female	23	16	77
5	5	Female	31	17	40
6	6	Female	22	17	76
7	7	Female	35	18	6
8	8	Female	23	18	54
9	9	Male	34	19	3
10	10	Female	38	19	72
11	11	Male	67	19	14
12	12	Female	35	19	89
13	13	Female	36	28	15

We have only chosen 2 attributes from the data
Income
SpendingScore

This will help in plotting the results in 2D
Otherwise, all attributes seem to be essential for decision making

We have only chosen 2 attributes from the data
Income
SpendingScore

This will help in plotting the results in 2D
Otherwise, all attributes seem to be essential for decision making

We have only chosen 2 attributes from the data
Income
SpendingScore

This will help in plotting the results in 2D
Otherwise, all attributes seem to be essential for decision making

The screenshot shows the Spyder Python 3.5.3 IDE. The editor displays the `kmeans.py` script. The variable explorer shows the following variables:

Name	Type	Size	Value
X	DataFrame	(200, 5)	array([[15, 39], [15, 81], ...])
dataset	DataFrame	(200, 5)	Column names: CustomerID, Gender, Age, Annual Income (k\$), Spending Score (1-100)
i	int	1	10
wcss	list	10	[269981.28888888893, 181363.59595959596, 186, ...]
y_kmeans	int32	(200,)	array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, ...])

The console shows the execution of the script, including the Elbow Method plot. The plot shows the WCSS (Within-Cluster Sum of Squares) on the y-axis (ranging from 0 to 30000) and the Number of clusters on the x-axis (ranging from 1 to 10). The curve starts high and decreases sharply, then levels off around 4 clusters.

Max_iter → maximum number of iterations (if k-means does not converge)
N_init → Number of times random initial points will be chosen by k-means++

The screenshot shows the Spyder Python 3.5.3 IDE. The editor displays the `kmeans.py` script. The variable explorer shows the following variables:

Name	Type	Size	Value
X	DataFrame	(200, 5)	array([[15, 39], [15, 81], ...])
dataset	DataFrame	(200, 5)	Column names: CustomerID, Gender, Age, Annual Income (k\$), Spending Score (1-100)
i	int	1	10
wcss	list	10	[269981.28888888893, 181363.59595959596, 186, ...]
y_kmeans	int32	(200,)	array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, ...])

The console shows the execution of the script, including the Clusters of clients plot. The plot shows the Spending Score (1-100) on the y-axis (ranging from -20 to 120) and the Annual Income (k\$) on the x-axis (ranging from 0 to 120). The data points are colored according to their cluster assignment: Cluster 1 (red), Cluster 2 (blue), Cluster 3 (green), Cluster 4 (yellow), Cluster 5 (purple), and Centroids (black dots).

The screenshot shows the Spyder Python 3.5.3 IDE. The editor displays the `kmeans.py` script. The variable explorer shows the following variables:

Name	Type	Size	Value
X	DataFrame	(200, 5)	array([[15, 39], [15, 81], ...])
dataset	DataFrame	(200, 5)	Column names: CustomerID, Gender, Age, Annual Income (k\$), Spending Score (1-100)
i	int	1	10
wcss	list	10	[269981.28888888893, 181363.59595959596, 186, ...]
y_kmeans	int32	(200,)	array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, ...])

The console shows the execution of the script, including the Clusters of clients plot. The plot shows the Spending Score (1-100) on the y-axis (ranging from -20 to 120) and the Annual Income (k\$) on the x-axis (ranging from 0 to 120). The data points are colored according to their cluster assignment: Cluster 1 (red), Cluster 2 (blue), Cluster 3 (green), Cluster 4 (yellow), Cluster 5 (purple), and Centroids (black dots).

The screenshot shows the Spyder Python 3.5.3 IDE. The editor displays the `kmeans.py` script. The variable explorer shows the following variables:

Name	Type	Size	Value
X	DataFrame	(200, 5)	array([[15, 39], [15, 81], ...])
dataset	DataFrame	(200, 5)	Column names: CustomerID, Gender, Age, Annual Income (k\$), Spending Score (1-100)
i	int	1	10
wcss	list	10	[269981.28888888893, 181363.59595959596, 186, ...]
y_kmeans	int32	(200,)	array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, ...])

The console shows the execution of the script, including the Clusters of clients plot. The plot shows the Spending Score (1-100) on the y-axis (ranging from -20 to 120) and the Annual Income (k\$) on the x-axis (ranging from 0 to 120). The data points are colored according to their cluster assignment: Cluster 1 (red), Cluster 2 (blue), Cluster 3 (green), Cluster 4 (yellow), Cluster 5 (purple), and Centroids (black dots).