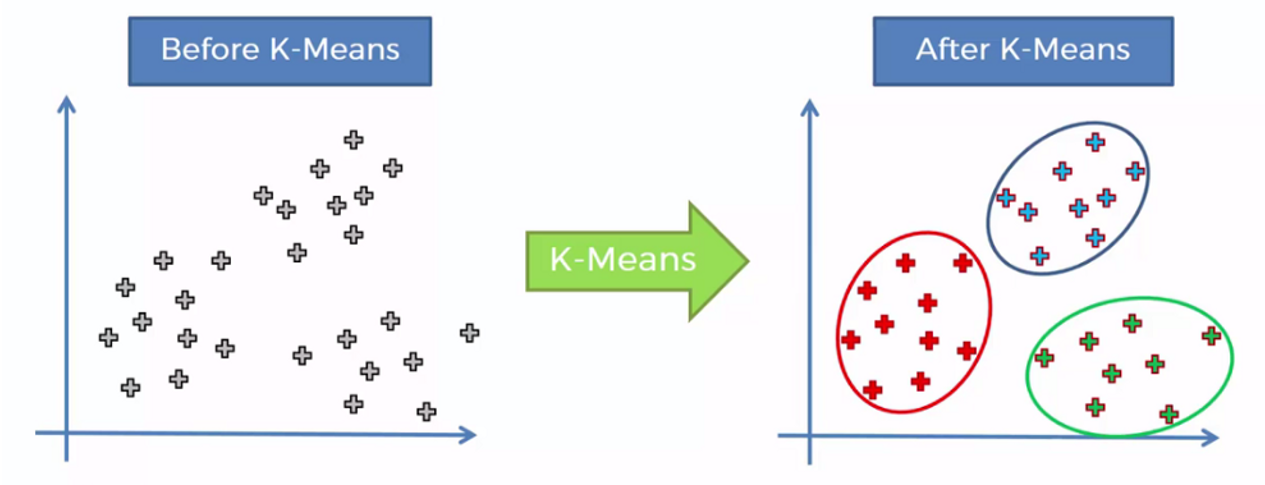
CUSTOMER SEGMENTATION USING Building The Customer Segmentation Model

As mentioned above, we are going to create a K-Means clustering algorithm to perform customer segmentation.

The goal of a K-Means clustering model is to segment all the data available into non-overlapping sub-groups that are distinct from each other.

Here is a simple visual representation of how K-Means clustering groups a dataset into different segments:



When building a clustering model, we need to decide how many segments we want to group the data into. This is achieved by [a heuristic called the elbow method](https://iopscience.iop.org/article/10.1088/1757-899X/336/1/012017).

We will create a loop and run the K-Means algorithm from 1 to 10 clusters. Then, we can plot model results for this range of values and select the elbow of the curve as the number of clusters to use.

Run the following lines of code to achieve this:

**import** matplotlib.pyplot **as** plt

**from** sklearn.datasets **import** make\_blobs

**from** sklearn.cluster **import** KMeans

**from** sklearn.metrics **import** silhouette\_score

**from** sklearn.decomposition **import** PCA

**from** mpl\_toolkits.mplot3d **import** Axes3D

SSE = []

**for** cluster **in** range(1,10):

kmeans = KMeans(n\_clusters = cluster, init='k-means++')

kmeans.fit(scaled\_features)

SSE.append(kmeans.inertia\_)

*# converting the results into a dataframe and plotting them*

frame = pd.DataFrame({'Cluster':range(1,10), 'SSE':SSE})

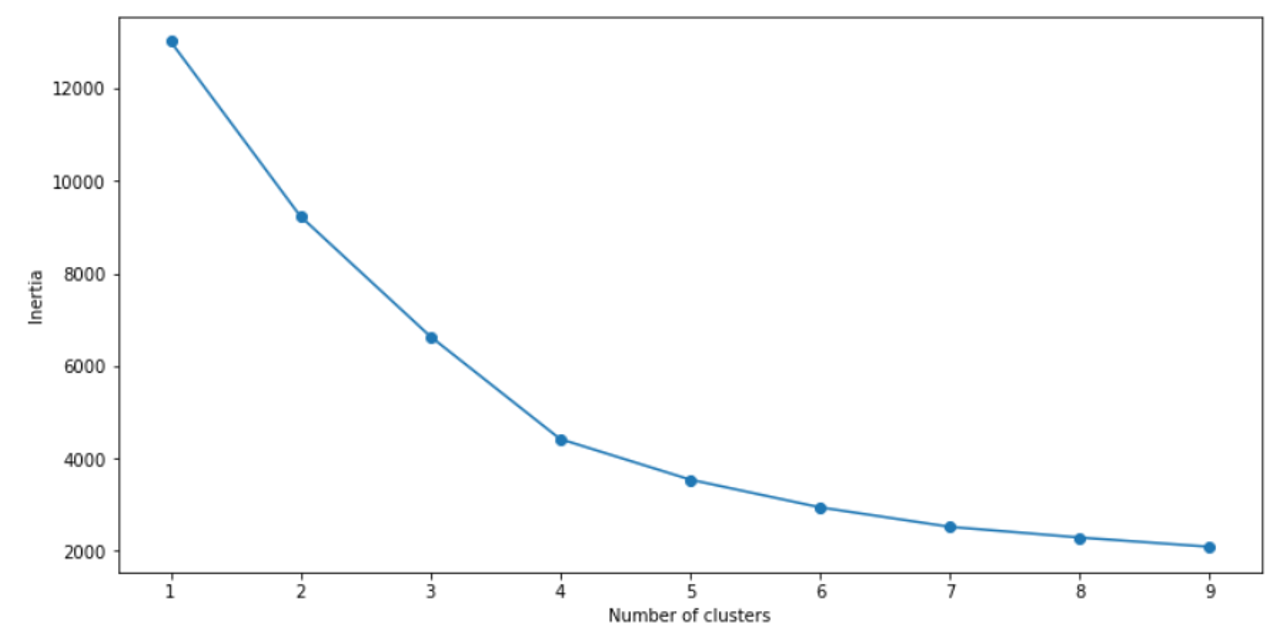
plt.figure(figsize=(12,6))

plt.plot(frame['Cluster'], frame['SSE'], marker='o')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

Here are the results of the lines of code above:



The “elbow” of this graph is the point of inflection on the curve, and in this case is at the 4-cluster mark.

This means that the optimal number of clusters to use in this K-Means algorithm is 4. Let’s now build the model with 4 clusters:

*# First, build a model with 4 clusters*

kmeans = KMeans( n\_clusters = 4, init='k-means++')

kmeans.fit(scaled\_features)

To evaluate the performance of this model, we will use a metric called the silhouette score. This is a coefficient value that ranges from -1 to +1. A higher silhouette score is indicative of a better model.

print(silhouette\_score(scaled\_features, kmeans.labels\_, metric='euclidean'))

The silhouette coefficient of this model is **0.44,**indicating reasonable cluster separation.