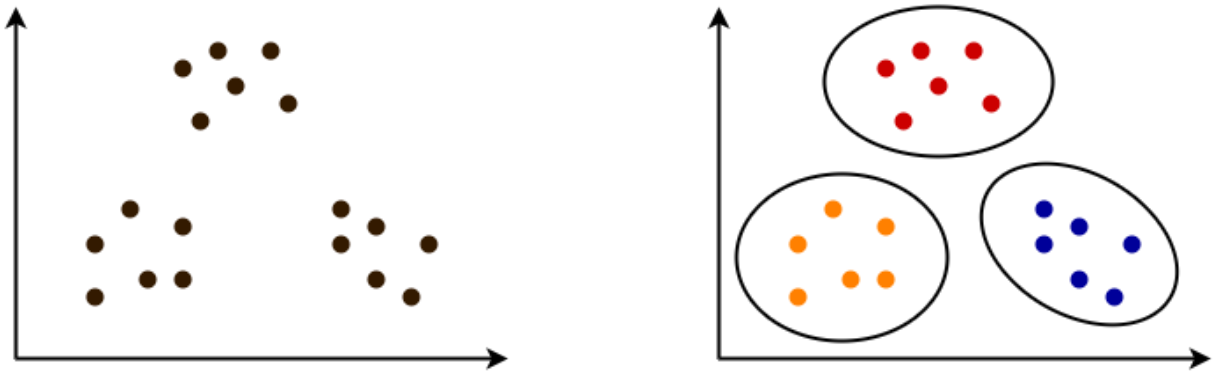
K-Means Clustering

It was proposed by Stuart Lloyd at the Bell Labs in 1957 as a technique for pulse-code modulation, but it was only published outside of the company in 1982, in a paper titled _“Least square quantization in PCM”._By then, in 1965, Edward W. Forgy had published virtually the same algorithm, so K-Means is sometimes referred to as Lloyd-Forgy.

K-Means is a clustering approach in which the data is grouped into K distinct non-overlapping clusters based on their distances from the K centres. The value of **K** needs to be specified first and then the algorithm assigns the points to exactly one cluster.



Before K-Means

After K-Means

Theory

The theory discussed above can be mathematically expressed as:

- Let C1, C2, Ck be the K clusters
- Then we can write: $C_1 U C_2 U C_3 U \dots U C_k$ ={1,2,3,…,n} i.e., each datapoint has been assigned to a cluster.
- Also,

$$C_k \cap C_{k'} = \emptyset \text{ for all } k \neq k'.$$

  This means that the clusters are non-overlapping.
- The idea behind the K-Means clustering approach is that the within-cluster variation amongst the point should be minimum. The within-cluster variance is denoted by: W(Ck).Hence, according to the statement above, we need to minimize this variance for all the clusters. Mathematically it can be written as:

$$\underset{C_1,\dots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}.$$

- The next step is to define the criterion for measuring the within-cluster variance. Generally, the criterion is the Euclidean distance between two data points.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

- The above formula says that we are calculating the distances between all the point in a cluster, then we are repeating it for all the K clusters(That's why two summation signs) and then we are dividing it by the number of observation in the clusters (Ck is the number of observations in the Kth cluster) to calculate the average.

So, ultimately our goal is to minimize:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}.$$

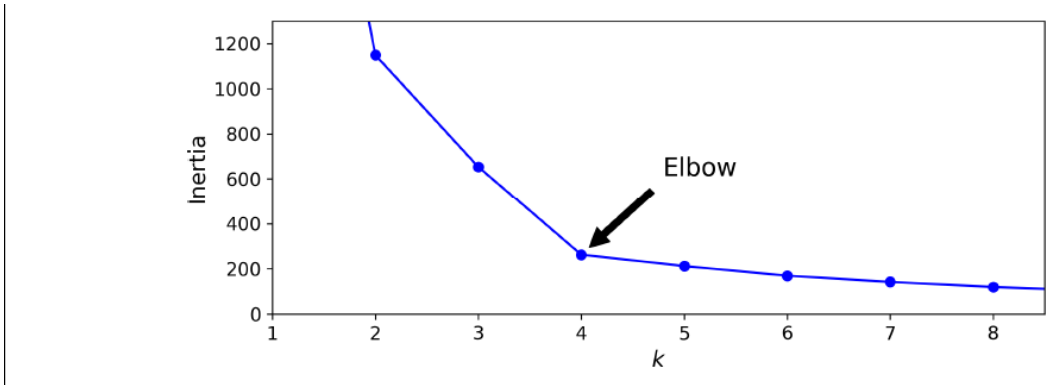
Algorithm:

1. Randomly assign K centres.
2. Calculate the distance of all the points from all the K centres and allocate the points to cluster based on the shortest distance. The model's *inertia* is the mean squared distance between each instance and its closest centroid. The goal is to have a model with the lowes intertia.
3. Once all the points are assigned to clusters, recompute the centroids.
4. Repeat the steps 2 and 3 until the locations of the centroids stop changing and the cluster allocation of the points becomes constant.

As we saw earlier, we need to provide the value of K beforehand. But the question is how to get a good value of K. An optimum value of K is obtained using the Elbow Method.

The Elbow-Method

This method is based on the relationship between the within-cluster sum of squared distances(WCSS Or Inertia) and the number of clusters. It is observed that first with an increase in the number of clusters WCSS decreases steeply and then after a certain number of clusters the drop in WCSS is not that prominent. The point after which the graph between WCSS and the number of clusters becomes comparatively smother is termed as the elbow and the number of cluster at that point are the optimum number of clusters as even after increasing the clusters after that point the variation is not decreasing by much i.e., we have accounted for almost all the dissimilarity in the data. An elbow-curve looks like:



Hands On Example

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
dataset=pd.read_csv('Mall_Customers.csv')
dataset.head()
```

Out[2]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

- Here I only Consider Annual Income (k$) and Spending Score (1-100) you can use as many data you want

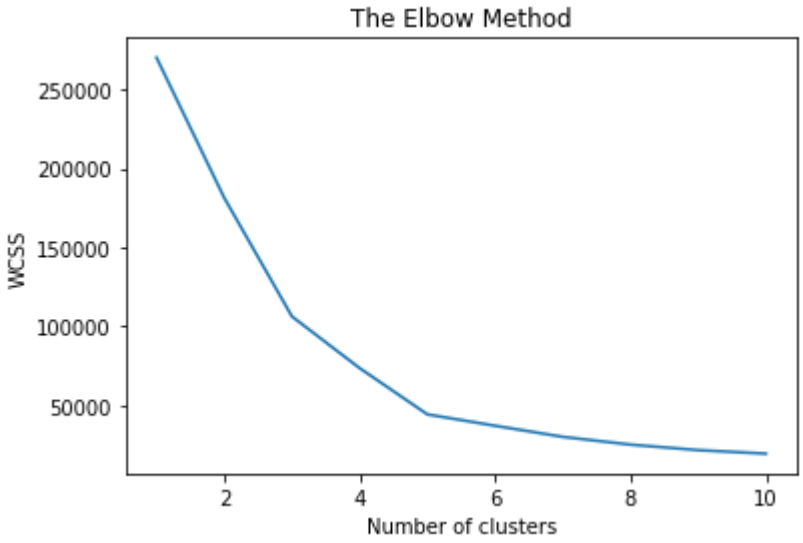
In [3]:
```
#dataset
X=dataset.iloc[:,3:]
X.head()
```

Out[3]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

Elbow Method to find Appropriate K-Value

In [4]:
```
#elbow method Find appropriate K-value
from sklearn.cluster import KMeans
wcss=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



- Here you see clearly at point 5 the curve is down so here i choose 5 is my K_value

Fitting K-Means to the dataset

In [5]:
```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

In [6]:
```
y_kmeans
```

Out[6]:
```
array([3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,
       3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 1,
       3, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 4, 2, 1, 2, 4, 2, 4, 2,
       1, 2, 4, 2, 4, 2, 4, 2, 4, 2, 1, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,
       4, 2])
```

In [7]:
```
# It predicts the cluster number to which the datapoint belongs to
test=kmeans.predict([[15, #Annual Income (k$)
                      39 #Spending Score (1-100)
                     ]])
print("it belongs to ",test[0],"cluster")
```

```
it belongs to  3 cluster
```

In [8]:
```
# Looking at the points which belong to Cluster0
X[y_kmeans==0].head() #it show in 0 number cluster which are include
```

Out[8]:

	Annual Income (k$)	Spending Score (1-100)
1	15	81
3	16	77
5	17	76
7	18	94
9	19	72

In [9]:
```
# Visualising the clusters
plt.scatter(X[y_kmeans == 0]['Annual Income (k$)'], X[y_kmeans == 0]['Spending Sc
plt.scatter(X[y_kmeans == 1]['Annual Income (k$)'], X[y_kmeans == 1]['Spending Sc
plt.scatter(X[y_kmeans == 2]['Annual Income (k$)'], X[y_kmeans == 2]['Spending Sc
plt.scatter(X[y_kmeans == 3]['Annual Income (k$)'], X[y_kmeans == 3]['Spending Sc
plt.scatter(X[y_kmeans == 4]['Annual Income (k$)'], X[y_kmeans == 4]['Spending Sc
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

K-Means Advantages :

1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.

2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-Means Disadvantages :

1) Difficult to predict K-Value.

2) With global cluster, it didn't work well.

3) Different initial partitions can result in different final clusters.

4) It does not work well with clusters (in the original data) of Different size and Different density