# Big Data & Predictive Analytics
# Lecture notes on Clustering

School of Informatics
University of Leicester

## 1 Introduction

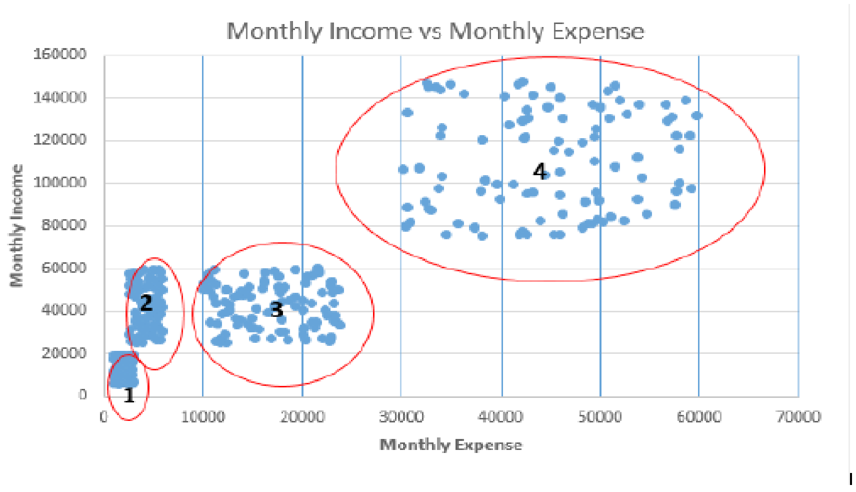
Clustering is the process of categorizing data points in clusters in which they are more similar to each other than the data points outside the cluster. Once the clusters are determined, we can identify the properties of the cluster and devise plans or strategies for each cluster. Clustering algorithms are *unsupervised* learning algorithms in the sense that we do not observe the outcome and we cannot train the data to recognise it. In other words, the training data has no ground truth labels to learn from. Yet, we would like to group similar data points into different clusters with possibly some overlap.

Clustering can basically be defined as follows:

- finding groups with a high similarity among each group members
- finding groups with a significant dissimilarity between the members of two different groups.

A key important concept in clustering is the notion of *similarity*, which we define later on. In a nutshell, it is a distance between individual members of a group that measures how close they are.

**How is clustering used?** Let us look at the plot of Monthly Income and Monthly Expense for a group of 400 people. As one can see, there are visible clusters of people whose earnings and expenses are different from people from other clusters, but are very similar to the people in the cluster they belong to:



In this plot, the visible clusters of the people can be identified based on their income and expense levels, as follows:

- Cluster 1 (low income, low expense): This has low income and low expense levels
- Cluster 2 (medium income, less expense): This has a medium level of income, but spend less.
- Cluster 3 (medium income, medium or high expense): This has medium levels of income, almost the same range as cluster 2, but they spend more than cluster 2.
- 4 (high income, high expense): This has a high level of income and a high level of expenses.

This analysis can be very useful if an organization is trying to target potential customers for their different range of products as they can target different clusters for different ranges of their products. Maybe, they can target the cluster 4 to sell their premium products and cluster 1 and 2 to sell their low-end products. This may result in higher conversion rates for the advertisement campaigns.

Usually it is difficult to assess the quality of a clustering algorithm but there are few properties of a good cluster, which can be listed as follows:

- Clusters should be identifiable and significant in size so that they can be classified as one.
- Points within a cluster should be compactly placed within themselves and there should be minimum overlap with points in the other clusters.
- Clusters should make business sense. The observations in the same cluster should exhibit similar properties when it comes to the business context.

Clustering can have a variety of applications. The following are some of the cases where clustering is used:

- Automatically grouping documents/web pages into topics. For instance, grouping news stories from today into categories

- Digital marketing wherein the aim is to find customers who think, behave, and make decisions on similar lines and reach out to them and persuade them in a tailor-made fashion.
- Clustering is used in seismology to find the expected epicenter of earthquakes and identify earthquake-prone zones.

One of the most popular clustering algorithm is the K-means wherein we specify the number of clusters we would like to get from the given dataset.

## 2 The K-Means Clustering Algorithm

As discussed earlier, a clustering algorithm relies on a measure of similarity or dissimilarity in order to cluster observations or individual data points.

### 2.1 Measuring similarity

If we consider the dataset as an n-dimensional space, where $n$ is the number of columns in the dataset, each data point will have n coordinates and we can calculate the mathematical distance between the points. A smaller distance implies a higher similarity and a cluster will contain points that are closer to each other.

There are many ways of calculating distances and different algorithms use different methods of calculating distance. The Euclidean distance is the most used distance. Let us denote each data point as $X_i$. Then, the Euclidean distance between the two points, $X_i$ and $X_j$, for a dataset having n-columns, is defined as follows:

$$d(X_i, X_j) = \sqrt{(X_{i,1} - X_{j,1})^2 + \ldots + (X_{i,n} - X_{j,n})^2}$$

The Manhattan distance is defined as

$$d(X_i, X_j) = |X_{i,1} - X_{j,1}| + \ldots + |X_{i,n} - X_{j,n}|$$

Depending on the context, we can also used the so-called Jaccard index as a measure of similarity or diversity between two sample sets. If we have two sample sets $X_i$ and $X_j$, then the Jaccard index $J(X_i, X_j)$ is defined as the cardinal (number of elements) of its intersection over that of its union.

$$d(X_i, X_j) = J(X_i, X_j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}$$

Once we have defined the distance between two points, we can calculate the distance between any two points and store them as a matrix. This matrix representing the distance between any two points (observations) in the dataset is called a *distance matrix.* This distance matrix, say $D$ is symmetric meaning its entries $D_{i,j} = D_{j,i}$ and that $D_{i,i} = 0$. The distance matrix is central to the clustering algorithm as it enables to determine the centers of each cluster.

## 2.2 The K-means algorithm

The K-means algorithm assumes the number of desired clusters is known in advance maybe based on the business context. It takes as input a dataset, say $m$ data points with $n$ columns. The centers of the clusters are updated by using the data points assigned to that cluster. It returns an array containing the cluster number to which each data point belongs to as well as the cluster centers.

The algorithm can be described as follows:

1. Input number of clusters. The number of clusters can be decided based on the business context and randomly chosen data points can be used to initialise the centers for the clusters.

2. Calculate the distance of each observation from each cluster. Assign all points to the closest cluster center assuming the distance of a data point from a cluster is well defined.

3. Change cluster centers $c_i$ to be in the middle of its points by using the following formula:

$$c_i = \frac{1}{C_i} \sum_{j=1}^{C_i} X_j$$

   wherein $C_i$ is the number of points in the cluster, and $X_k$ is a data point in the cluster.

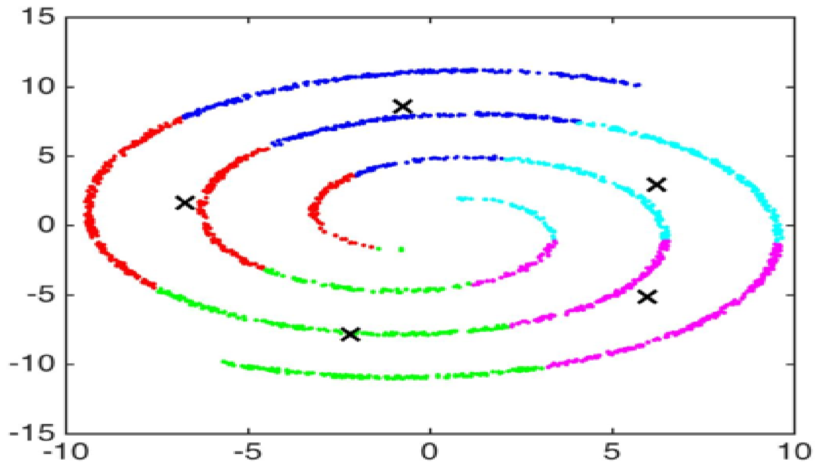
4. Repeat until convergence. This means none of the data points are being reassigned. In other words, that means that all the data points are already in the correct clusters and their distance to a cluster center can't be reduced further.

The goal of the K-means algorithm is to minimise the quantity

$$\mathrm{Cost}(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{m} \mathrm{min}_k \mathrm{d}(X_i, c_k)$$

$\mathrm{d}(X_i, c_k)$ is the distance between a point $X_i$ and the cluster center $c_k$. The function min implies we need to use the nearest cluster center. Ultimately, we want to minimize the total distances to the nearest cluster centers. In here, we are trying to find the cluster centers that minimize the total distance from all the points to their nearest cluster center. So the global objective is make local moves in order to optimize the total distance.

In the K-means algorithm, the cost gets lower at each step, but does it always converge to the globally optimal set of cluster assignments and reference points? The answer is no and that sometimes it needs replicates. Consider the following example:



We can find worse examples where K-means just doesn't optimize its objective function. Here it did a good job optimizing the objective, it's just that the objective was lousy for this dataset. Anyway, if you happen to start the reference points off in a bad spot sometimes, it just won't go to the right place. That's why most K-means code enables us to do replicates so we can start with different randomly chosen points to be the centers.

The goal of this algorithm is to attain a configuration of cluster centers and cluster data points so that the overall $J$ squared error function or J-score is minimized:

$$J = \sum_{i=1}^{k} \sum_{j=1}^{C_i} \text{dist}(X_j, c_i)^2$$

wherein, *dist* is the distance, $k$ is number of clusters, $C_i$ the number of points in the cluster, and $c_i$ is the centroid of the $i^{th}$ cluster. The J squared error function can be understood as the sum of the squared distance of points from their respective cluster centers. A smaller value of J implies tightly packed and homogeneous clusters. This also implies that most of the points have been placed in the right clusters.

Scikit-learn provides an implementation of the K-means algorithm. The following code fragment illustrates an example of using K-means for a given dataset `df_norm` that has been normalised.

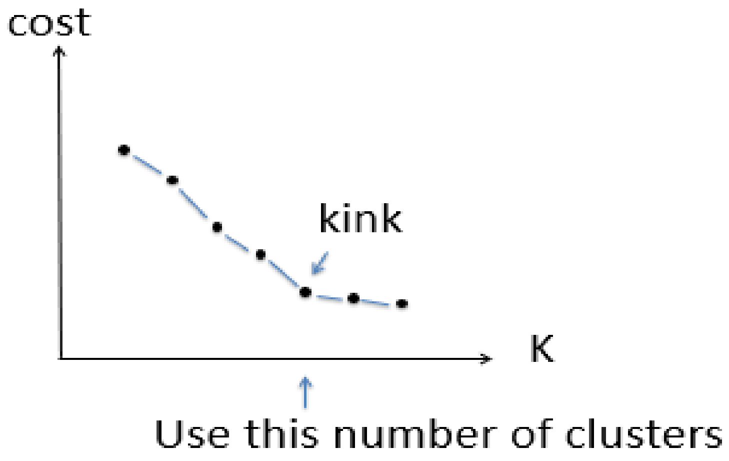
```
from sklearn.cluster import KMeans
from sklearn import datasets
model=KMeans(n_clusters=6)
model.fit(df_norm)
md=pd.Series(model.labels_)
df_norm['clust']=md
df_norm.head()
```

We will use Scikit-learn for the K-means algorithm in the Lab and/or tutorial.

## 3 Fine-tuning the clustering algorithm

Deciding the optimum value of K in the K-means algorithm is difficult. However, we can use the so-called Elbow method to decide what K should be. Ideally we have a well-understood business context that dictates the value of K.

One way of finding $k$ is to consider the total cost of the K-means for various values of $k$ and hopefully, we can observe a kink (kind of elbow) on the curve. The corresponding value of $k$ gives us the best number of clusters for the given dataset. The cost function can vary but a primary candidate can be the J-score.


cost
kink
K
Use this number of clusters