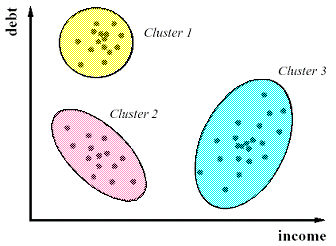
# INTRODUCTION

Clustering is the process of using machine learning and algorithms to identify how different types of data are related and creating new segments based on those relationships. Clustering finds the relationship between data points so they can be segmented.

Cluster analysis is unsupervised machine learning task of dividing the population or data points into a number of groups such that data points in the same groups are similar.

Following figure is an example of finding clusters of US population based on their income and debt



That means the observations (data points) within a cluster are homogenous and when compared to other clusters are heterogenous.

Suppose, you are the head of a retail supply chain and wish to understand preferences of your costumers to scale up your business. Is it possible for you to look at details of each costumer and devise a unique business strategy for each one of them? Definitely not. But what you can do is to cluster all of your costumers into say 10 groups based on their purchasing habits and use a separate strategy for costumers in each of these 10 groups. And this is what we call clustering.

## SEGMENTATION

Segmenting is the process of putting customers into groups based on similarities, and clustering is the process of finding similarities in customers so that they can be grouped, and therefore segmented.

When you segment you know who to target. If I’m selling an expensive little black dress, I want to target women who have a high annual household income. In this case I’m defining the limits of the group. Women. With annual incomes over a hundred thousand dollars who have purchased similar items in that product category. Still there might be many other factors that influence the decision to buy the dress. It might not be possible to manually study all the features and find relations between them This is where clustering comes in picture.

Segmentation is always possible, even in an extremely homogeneous collection of items. You just decide where you will cut between the groups whereas finding clusters in this extremely homogeneous collection is impossible, since by definition there are no density differences, and hence no clusters to find.

## APPLICATIONS OF CLUSTERING

Clustering has a large no. of applications spread across various domains. Some of the most popular applications of clustering are:

* Recommendation engines
* Market segmentation
* Social network analysis
* Search result grouping
* Medical imaging
* Image segmentation
* Anomaly detection

## SUPERVISED LEARNING WITH CLUSTERING

Clustering is an unsupervised machine learning approach, but can it be used to improve the accuracy of supervised machine learning algorithms as well by clustering the data points into similar groups and using these cluster labels as independent variables in the supervised machine learning algorithm.

# CLUSTERING ALGORITHMS CLASSIFICATION

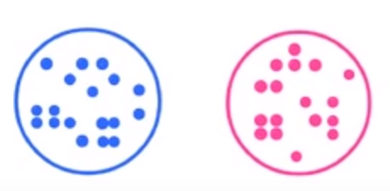
Clustering algorithms may be classified as listed below:

* Exclusive Clustering
* Overlapping Clustering / Soft clustering
* Hierarchical Clustering
* Probabilistic Clustering
* Density based clustering

## EXCLUSIVE CLUSTERING

Exclusive clustering (aka partition clustering) is a type of hard Clustering where data points are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster.

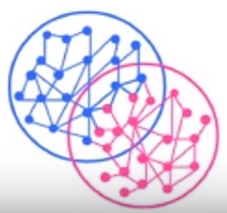
* Hard clustering
* Data points / items belongs exclusively to one cluster.
* For example: K- means clustering



## OVERLAPPING CLUSTERING

In Overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership.

* Soft clustering
* Data points belong to multiple clusters.
* For example: Fuzzy C-means clustering



## PROBABILISTIC CLUSTERING

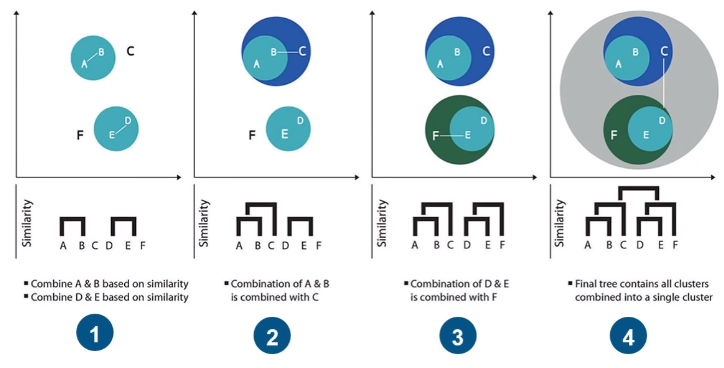
By considering uncertainty in the attributes common methods cannot be applicable in data clustering. Elements with predefined probabilistic distributions for their attributes are clustered using Probabilistic clustering. Example Mixture of Gaussians clustering.

Instead of putting each data point into a particular cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 2 clusters.

## HIERARCHAL CLUSTERING

A hierarchical clustering algorithm is based on the union between the two nearest clusters. These models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away.

Hierarchal clustering produces a set of nested clusters organised as a hierarchical tree. It can be visualized as a dendrogram as shown below:



There are two types of hierarchical clustering:

1. Agglomerative
2. Divisive

### AGGLOMERATIVE

Data points are clustered using a bottom-up approach starting with individual data points. Start with classifying all data points into separate clusters & then aggregating them as the distance decreases.

Agglomerative clustering tries to link each data point, by a distance measure, to its nearest neighbour, creating a cluster. Reiterating the algorithm using different linkage methods, the algorithm gathers all the available points into a rapidly diminishing number of clusters, until in the end all the points reunite into a single group.

If your dataset doesn’t contain too many observations, it’s worth trying agglomerative clustering with all the combinations of linkage and distance and then comparing the results carefully.

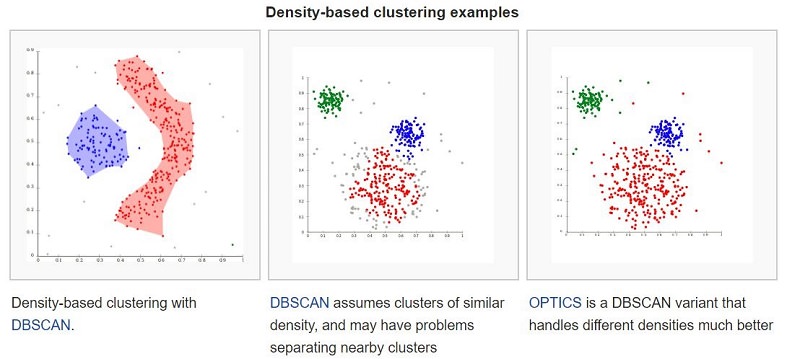
### DIVISIVE

In divisive, top-down approach is followed where all the data points are treated as one big cluster and the clustering process involves dividing the one big cluster into several small clusters.

All data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective.

## DENSITY BASED CLUSTERING

These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.



The most popular density-based clustering method is DBSCAN. It features a well-defined cluster model called "density-reachability". Similar to linkage-based clustering, it is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion.

A cluster consists of all density-connected objects plus all objects that are within these objects' range.

Another interesting property of DBSCAN is that its complexity is fairly low. It requires a linear number of range queries on the database – and that it will discover essentially the same results in each run, therefore there is no need to run it multiple times.

OPTICS is a generalization of DBSCAN that removes the need to choose an appropriate value for the range parameter and produces a hierarchical result related to that of linkage clustering.

The key drawback of DBSCAN and OPTICS is that they expect some kind of density drop to detect cluster borders. On a data set consisting of mixtures of Gaussians, these algorithms are nearly always outperformed by methods such as EM clustering that are able to precisely model this kind of data.

# HIERARCHICAL LINKAGE METHODS

There are three linkage methods:

1. Ward
2. Complete
3. Average

## WARD

Tends to look for spherical clusters, very cohesive inside and extremely differentiated from other groups. Another nice characteristic is that the method tends to find clusters of similar size. It works only with the Euclidean distance.

## COMPLETE

Links clusters using their furthest observations, that is, their most dissimilar data points. Consequently, clusters created using this method tend to be comprised of highly similar observations, making the resulting groups quite compact.

## AVERAGE

Links clusters using their centroids and ignoring their boundaries. The method creates larger groups than the complete method. In addition, the clusters can be different sizes and shapes, contrary to the Ward’s solutions. Consequently, this average, multipurpose, approach sees successful use in the field of biological sciences.

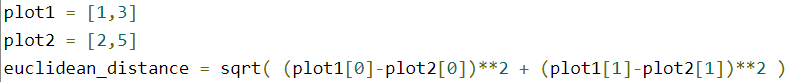
# HIERARCHICAL DISTANCE METRICS

There are also three distance metrics:

1. Euclidean
2. Manhattan
3. Cosine

## EUCLIDEAN (L2)

There are various ways to compute distance on a plane, many of which you can use here, but the most accepted version is Euclidean Distance Euclidean distance is:



## MANHATTAN (L1)

Similar to Euclidean, but the distance is calculated by summing the absolute value of the difference between the dimensions. In a map, if the Euclidean distance is the shortest route between two points, the Manhattan distance implies moving straight, first along one axis and then along the other — as a car in the city would, reaching a destination by driving along city blocks.

## COSINE

A good choice when there are too many variables and you worry that some variable may not be significant. Cosine distance reduces noise by taking the shape of the variables, more than their values, into account. It tends to associate observations that have the same maximum and minimum variables, regardless of their effective value.

# STEPS TO PERFORM CLUSTER ANALYSIS

We can follow below steps while doing a cluster analysis.

## CONFIRM DATA IS METRIC

While one can cluster data even if they are not metric, many of the statistical methods available for clustering require that the data are so: this means not only that all data are numbers, but also that the numbers have an actual numerical meaning, that is, 1 is less than 2, which is less than 3 etc. The main reason for this is that one needs to define distances between observations (see step 4 below), and often (“black box” mathematical) distances (e.g. the “Euclidian distance”) are defined only with metric data.

However, one could potentially define distances also for non-metric data. For example, if our data are names of people, one could simply define the distance between two people to be 0 when these people have the same name and 1 otherwise - one can easily think of generalizations.

## HYPOTHESIS BUILDING

This is the most crucial step of the whole exercise. Try to identify all possible variables that can help segment the portfolio regardless of its availability.

For example, a US bank X wants to understand the profile of its customer base to build targeted campaigns. Let’s try to come up with a list for this example.

* Customer balance with bank X
* Number of transaction done in last 1/3/6/12 months
* Balance change in last 1/3/6/12 months
* Demographics of the customer
* Customer total balance with all US banks

The list is just for illustrative purpose. In real scenario this list will be much longer.

## INITIAL SHORTLIST OF VARIABLE

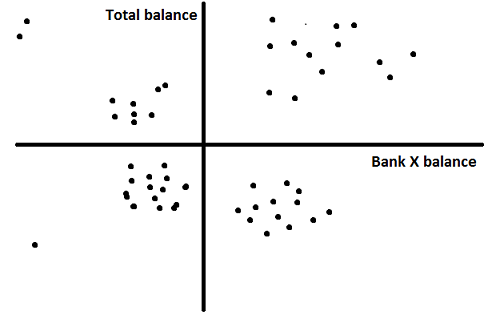
The decision about which variables to use for clustering is a critically important decision that will have a big impact on the clustering solution. So, we need to think carefully about the variables we will choose for clustering. Good exploratory research that gives us a good sense of what variables may distinguish people or products or assets or regions is critical. Clearly this is a step where a lot of contextual knowledge, creativity, and experimentation/iterations are needed.

Moreover, we often use only a few of the data attributes for segmentation (the segmentation attributes) and use some of the remaining ones (the profiling attributes) only to profile the clusters

Let’s say, for the current example we have only data for Customer balance with bank X and Customer total balance with all US banks (total balance).

## VISUALIZE THE DATASET

It is very important to know the population spread across the selected variable before starting any analysis. For the current scenario, the exercise becomes simpler as the number of selected variables is only 2. Following is a scatter plot between total balance and Bank X balance (origin taken as mean of both the variables).



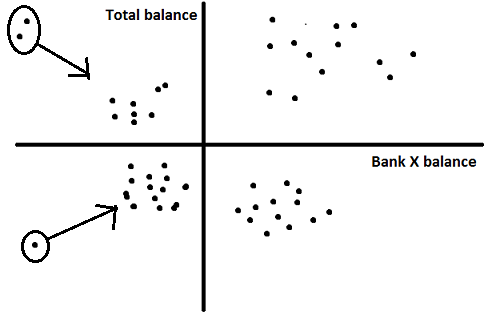
This visualization helps me to identify clusters which I can expect after the final analysis. Here, we can see there are four clear clusters in four quadrants. We can expect the same result in the final solution.

## DATA CLEANING

Cluster analysis is very sensitive to outliers. It is very important to clean data on all variables taken into consideration. There are two industry standard ways to do this exercise:

* Remove the outliers: (Not recommended in case the total data-points are low in number) We remove the data-points beyond mean +/- 3\*standard deviation.
* Capping and flouring of variables: (Recommended approach) We cap and flour all data-points at 1 and 99 percentiles.

Let’s choose second approach



## VARIABLE CLUSTERING

This step is performed to cluster variables capturing similar attributes in data. And choosing only one variable from each variable cluster will not drop the separation drastically compared to considering all variables. Remember, the idea is to take minimum number of variables to justify the separation to make the analysis easier and less time consuming. You can simply use Proc VARCLUS to generate these clusters.

## DEFINE SIMILARITY MEASURE

Remember that the goal of clustering and segmentation is to group observations based on how similar they are. It is therefore crucial that we have a good understanding of what makes two observations (e.g. customers, products, companies, assets, investments, etc) “similar”.

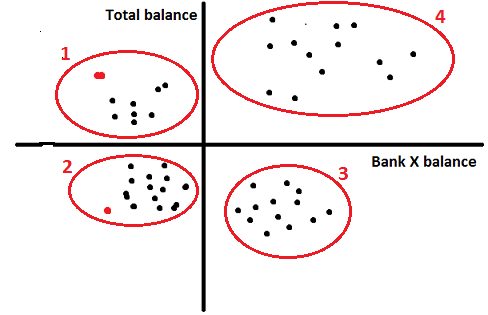
Most statistical methods for clustering and segmentation use common mathematical measures of distance. Typical measures are, for example, the Euclidean distance or the Manhattan distance.

## VISUALIZE PAIR-WISE DISTANCES

Having defined what we mean “two observations are similar”, the next step is to get a first understanding of the data through visualizing for example individual attributes as well as the pairwise distances (using various distance metrics) between the observations. If there are indeed multiple segments in our data, some of these plots should show “mountains and valleys”, with the mountains being potential segments.

## CLUSTERING

We can use any of the two-technique discussed in the article depending on the number of observations. k-means is used for a bigger sample (k=4 (which is apparent from the above figure).



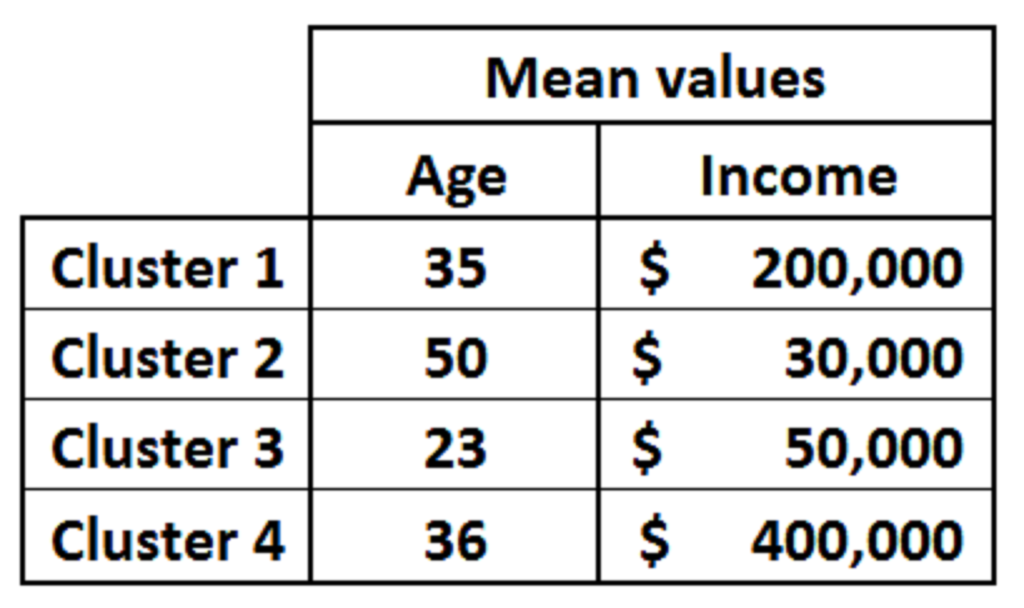
As we can see, the algorithm found 4 clusters which were already apparent in the visualization. In most business cases the number of variables will be much larger and such visualization won’t be possible.

## CONVERGENCE OF CLUSTERS

A good cluster analysis has all clusters with population between 5-30% of the overall base. Say, my total number of customers for bank X is 10000. The minimum and maximum size of any cluster should be 500 and 3000. If any of the cluster is beyond the limit than repeat the procedure with additional number of variables. We will discuss in detail about other convergence criterion in the next article.

## PROFILING OF THE CLUSTERS

After validating the convergence of cluster analysis, we need to identify behaviour of each cluster. Let’s say we map age and income to each of the four clusters and get following results



Now is the time to build story around each cluster. Let’s take any two cluster and analyze.

Cluster 1: (High Potential Low balance customer) These customers do have high balance in aggregate but low balance with bank X. Hence, they are high potential customer with low current balance. Also, the average salary is on a higher side which validates our hypothesis of customer being high potential.

Cluster 3: (High Potential high balance customers) Even though the salary and total balance in aggregate is on a lower side, we see a lower average age. This indicates that the customer has a high potential to increase their balance with bank X.

# GOOD CLUSTER ANALYSIS

There are two basic requirements of a good cluster analysis:

1. **DATA-POINTS WITHIN SAME CLUSTER SHARE SIMILAR PROFILE:** Statistical method to judge this criterion is simply checking the standard deviation for each input variable in each cluster. A perfect separation in case of cluster analysis is rarely achieved. Hence, even one standard deviation distance between two cluster means is considered to be a good separation.
2. **WELL SPREAD PROPORTION OF DATA-POINTS AMONG CLUSTERS:** There are no industry standards for this requirement. But a minimum of 5% and maximum of 35% of the total population can be assumed as a safe range for each cluster.

Whenever stuck with bad results for any modelling technique, the best way to solve it is not trying different solution but to explore what could have gone wrong. Let’s try to do the same in this case. Following are some of the hypothesis around what could have gone wrong.

1. Outliers still present: Try making outlier definition stricter. Try capping and flouring instead of removing the outlier.
2. Presence of overshadowing variable: This kind of variable generally become visible in the FASTCLUS output, in the significance table. If the significance index of a particular variable is exceptionally high, try replacing it with the next best variable. This next best variable can be found at the VARCLUS step.

What if the problem still remains? What does this indicate?

It indicates that the population is too homogeneous to be segmented. This problem is the hardest nut to crack. Try incorporating more data points to start with. If the problem still remains, we need to do clustering at a finer level.

The two techniques (K-means clustering & Hierarchical Clustering) have opposite pros and cons and hence can be used together to complement each other.

Following is the technique I finally used to get reasonable and actionable clusters:

1. Use outlier removal and overshadowing variable removal technique
2. Use k-means method to get many granular clusters: Build number of granular clusters. For n number of final clusters required, I put k=3 to 4 times n.
3. Use Hierarchical Clustering to club these granular clusters: Plug the mean values of the granular clusters as individual data-points to hierarchical clustering. This will give a line diagram as output. Choose the combinations which will not lead a cluster to go above 35% and make sure all the clusters come above 5%.

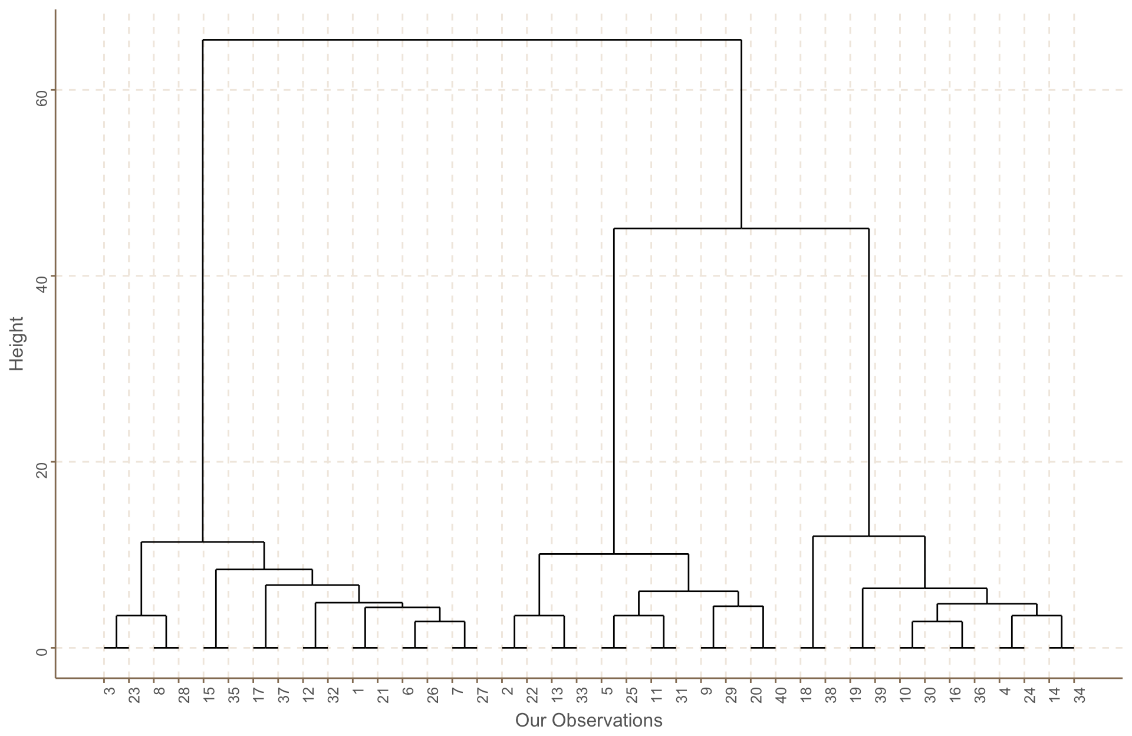
The output of this step is a set of clusters complying to both the constraints of a good clustering analysis.

# NUMBER OF CLUSTERS

There are many statistical methods for clustering and segmentation. In practice one may use various approaches and then eventually select the solution that is statistically robust, interpretable, and actionable - among other criteria.

Like all clustering methods Kmeans Clustering and the Hierarchical Clustering also require that we decided how to measure the distance/similarity between our observations. The only difference to highlight is that Kmeans requires the user to define how many clusters to create, while Hierarchical Clustering does not.

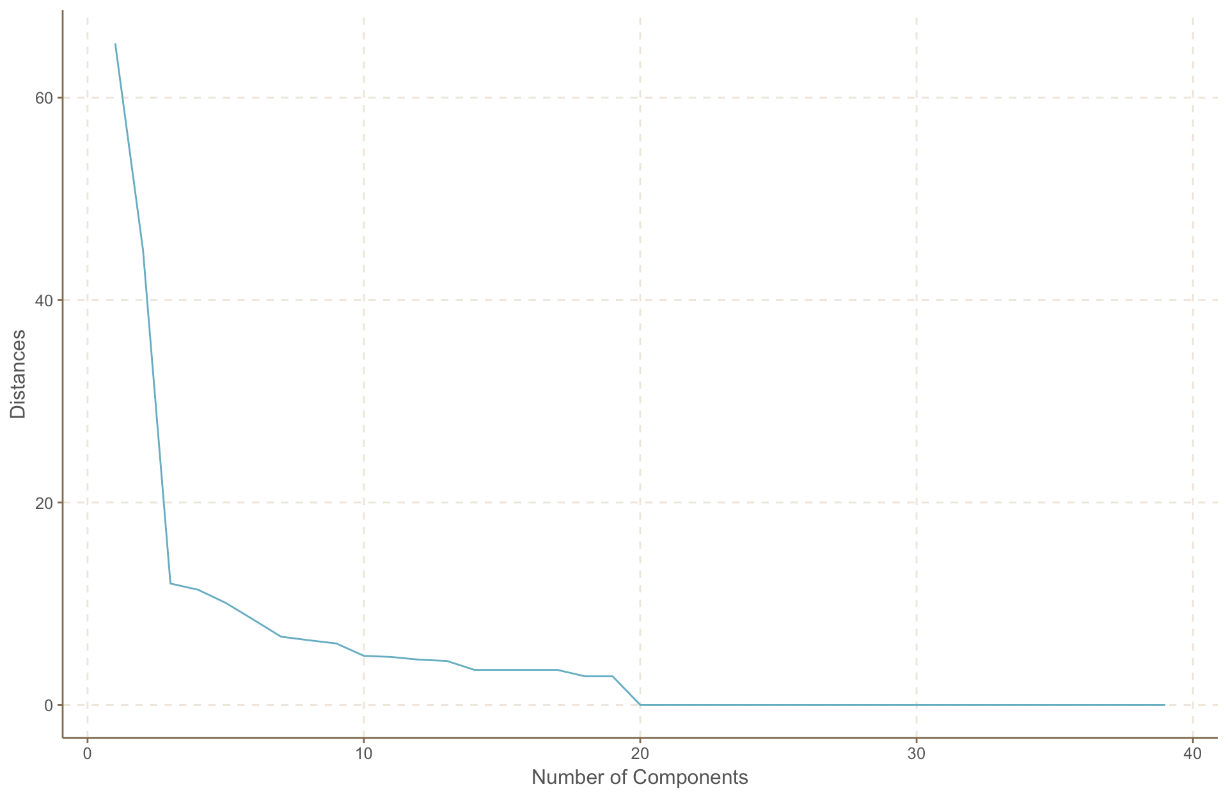
Let’s fist use the Hierarchical Clustering method, as we do not know for now how many clusters there are in our data. Hierarchical clustering is a method that also helps us visualise how the data may be clustered together. It generates a plot called the Dendrogram which is often helpful for visualization - but should be used with care.



The Dendrogram indicates how this clustering method works: observations are “grouped together”, starting from pairs of individual observations which are the closest to each other, and merging smaller groups into larger ones depending on which groups are closest to each other. Eventually all our data are merged into one segment. The heights of the branches of the tree indicate how different the clusters merged at that level of the tree are. Longer lines indicate that the clusters below are very different. As expected, the heights of the tree branches increase as we traverse the tree from the end leaves to the tree root: the method merges data points/groups from the closest ones to the furthest ones.

Dendrograms are a helpful visualization tool for segmentation, even if the number of observations is very large - the tree typically grows logarithmically with the number of datapoints. However, they can be very misleading. Notice that once two data points are merged into the same segment they remain in the same segment throughout the tree. This “rigidity” of the Hierarchical Clustering method may lead to segmentations which are suboptimal in many ways. However, the dendrograms are useful in practice to help us get some understanding of the data, including the potential number of segments we have in the data. Moreover, there are various ways to construct the dendrograms, not only depending on the distance metric we defined in the earlier steps above, but also depending on how the data are aggregated into clusters.

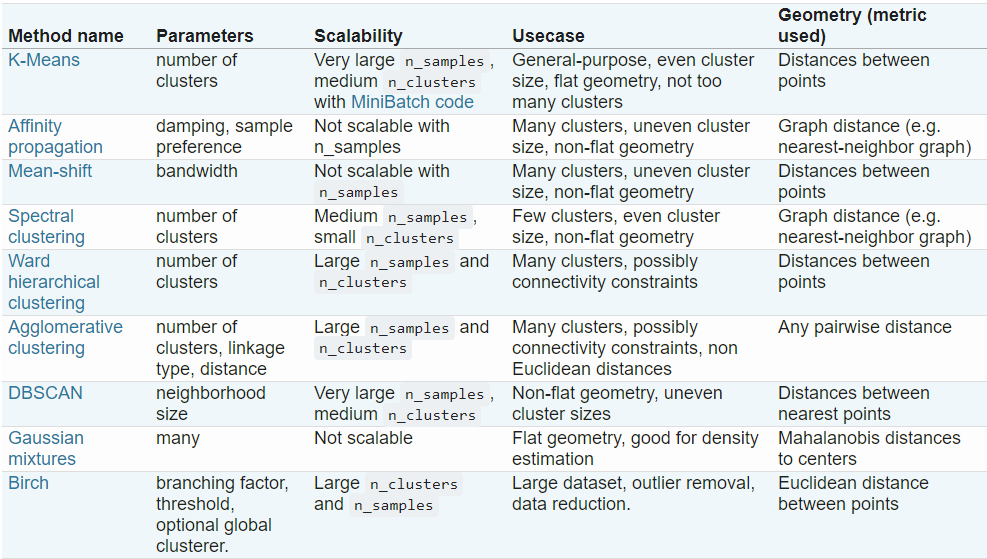
We can also plot the “distances” travelled before we need to merge any of the lower and smaller in size clusters into larger ones - the heights of the tree branches that link the clusters as we traverse the tree from its leaves to its root. If we have n observations, this plot has n-1 numbers.



As a rule of thumb, one can select the number of clusters as the “elbow” of this plot: this is the place in the tree where, if we traverse the tree from the leaves to its root, we need to make the “longest jump” before we merge further the segments at that tree level. Of course, the actual number of segments can be very different from what this rule of thumb may indicate: in practice we explore different numbers of segments, possibly starting with what a hierarchical clustering dendrogram may indicate, and eventually we select the final segmentation solution using both statistical and qualitative criteria.

# ALGORITHMS IN SKLEARN

Each clustering algorithm comes in two variants: a class, that implements the fit method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the labels\_ attribute.



# K-MEANS CLUSTER

The KMeans algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

The k-means algorithm divides a set of samples into K disjoint clusters, each described by the mean of the samples in the cluster. The means are commonly called the cluster “centroids”. The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum of squared criterion.

Inertia, or the within-cluster sum of squares criterion, can be recognized as a measure of how internally coherent clusters are. It suffers from various drawbacks:

* Inertia makes the assumption that clusters are convex and isotropic which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.
* Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called “curse of dimensionality”). Running a dimensionality reduction algorithm such as PCA prior to k-means clustering can alleviate this problem and speed up the computations.

Kmeans algorithm is good in capturing structure of the data if clusters have a spherical-like shape. It always tries to construct a nice spherical shape around the centroid. That means, the minute the clusters have a complicated geometric shape, KMeans does a poor job in clustering the data.

Note: Isotropy is uniformity in all orientations. Anisotropy is also used to describe situations where properties vary systematically, dependent on direction.

Kmeans clustering does not produce good results for following:

* Incorrect number of clusters specified.
* Anisotropic distributed data and elongated data.
* Different variance.
* Unevenly sized blobs.
* If dataset contains high numbers of features.

K means does not perform well with elongated shapes, we can use hierarchal clustering (single linkage for such issue).

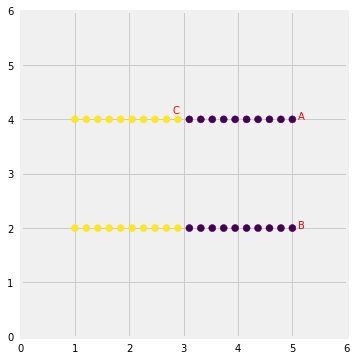
## ASSUMPTIONS

If any one of these 3 assumptions are violated, then k-means will fail.

* Kmeans assume the variance of the distribution of each attribute (variable) is spherical;
* All variables have the same variance;
* The prior probability for all k clusters are the same, i.e. each cluster has roughly equal number of observations.

## NON-SPHERICAL CLUSTERS

KMeans algorithm doesn’t let data points that are equally far-away from each other share the same cluster even though they obviously belong to the same cluster. Below is an example of data points on two different horizontal lines that illustrates how KMeans tries to group half of the data points of each horizontal line together.



Kmeans considers the point ‘B’ closer to point ‘A’ than point ‘C’ since they have non-spherical shape. Therefore, points ‘A’ and ‘B’ will be in the same cluster but point ‘C’ will be in a different cluster. Note the Single Linkage hierarchical clustering method gets this right because it doesn’t separate similar points.

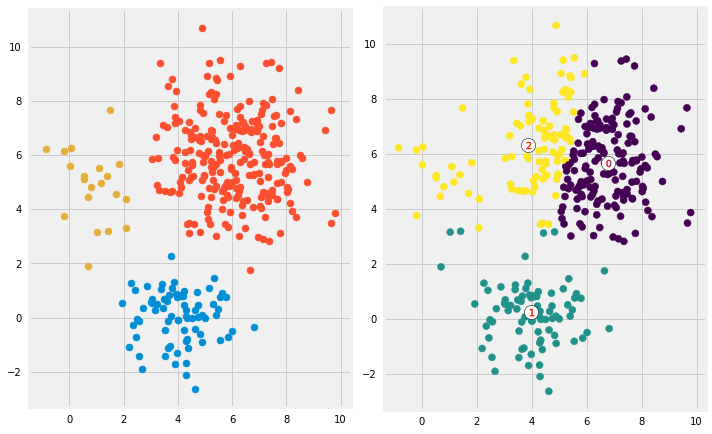
## CLUSTERS WITH UNEQUAL MEAN AND VARIANCE

Clusters with multivariate normal distributions with different means and standard deviations.

Each group was generated from different multivariate normal distribution (different mean/standard deviation). One group will have a lot more data points than the other two combined.

Left fig shows the datapoints in the original distributions and right figure shows the clusters obtained after applying Kmeans clustering.

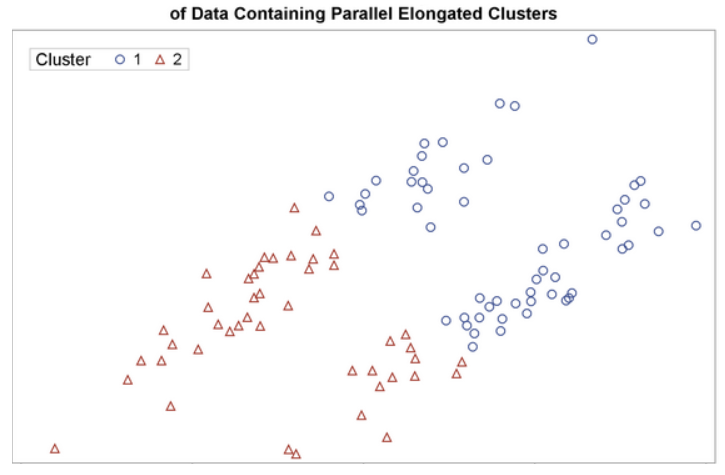
Looks like KMeans couldn’t figure out the clusters correctly. Since it tries to minimize the within-cluster variation, it gives more weight to bigger clusters than smaller ones. In other words, data points in smaller clusters may be left away from the centroid in order to focus more on the larger cluster.



We can use Gaussian Mixture Modeling (aka: EM Clustering) if all points should belong to a cluster and there are no outliers and noise.

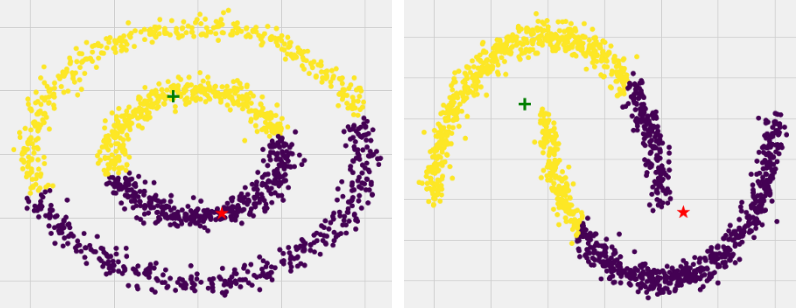
## ELONGATED CLUSTERS

Kmeans does not perform well with elongated clusters as shown below. We can use hierarchal clustering instead.

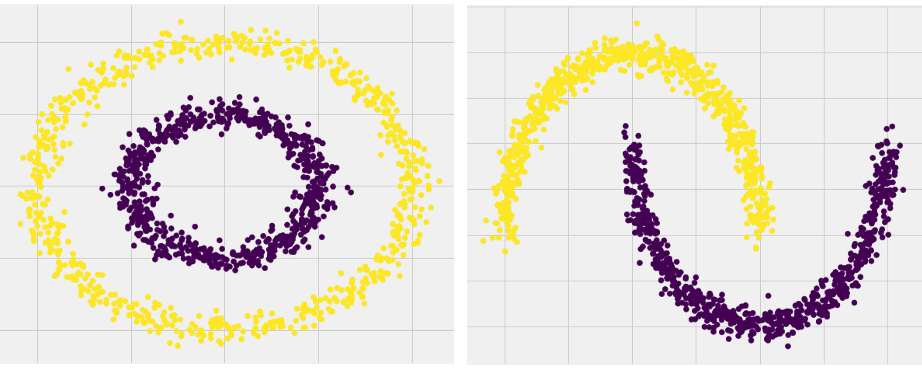


## CLUSTER INSIDE CLUSTER

Datapoints that have complicated geometric shapes such as moons and circles within each other.



As expected, KMeans couldn’t figure out the correct clusters for both datasets. However, we can help KMeans perfectly cluster these kinds of datasets if we use kernel methods. The idea is we transform to higher dimensional representation that make the data linearly separable (the same idea that we use in SVMs). Different kinds of algorithms work very well in such scenarios such as Spectral Clustering.



## IMPORTANT POINTS

Below are the main takeaways:

* Scale/standardize the data when applying KMeans algorithm.
* Elbow method in selecting number of clusters doesn’t usually work because the error function is monotonically decreasing for all K’s.
* Kmeans gives more weight to the bigger clusters.
* Kmeans assumes spherical shapes of clusters (with radius equal to the distance between the centroid and the furthest data point) and doesn’t work well when clusters are in different shapes such as elliptical clusters.
* If there is overlapping between clusters, KMeans doesn’t have an intrinsic measure for uncertainty for the examples belong to the overlapping region in order to determine for which cluster to assign each data point.
* Kmeans may still cluster the data even if it can’t be clustered such as data that comes from uniform distributions.

## EVALUATION METHODS

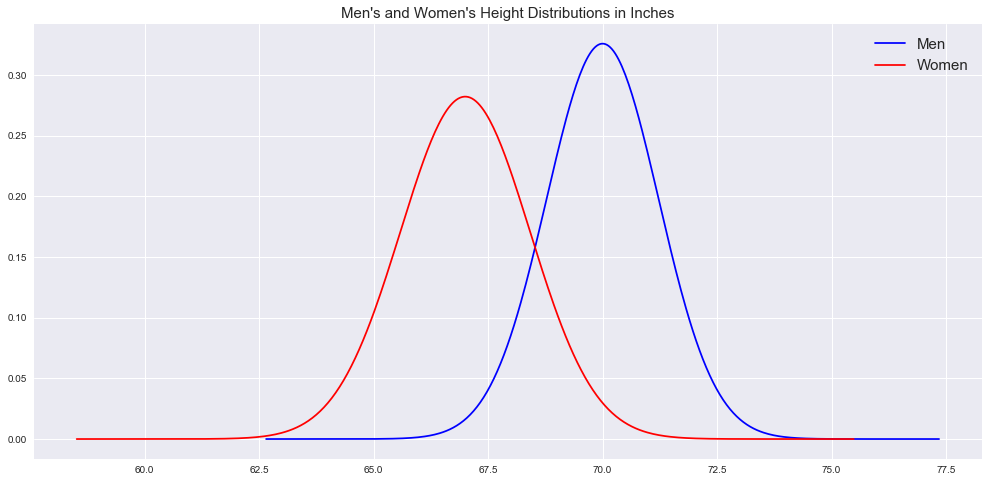
Contrary to supervised learning where we have the ground truth to evaluate the model's performance, clustering analysis doesn't have a solid evaluation metric that we can use to evaluate the outcome of different clustering algorithms. Moreover, since KMeans requires k as an input and doesn't learn it from data, there is no right answer in terms of the number of clusters that we should have in any problem. Sometimes domain knowledge and intuition may help but usually that is not the case. In the cluster-predict methodology, we can evaluate how well the models are performing based on different K clusters since clusters are used in the downstream modeling.

Two metrics that may give us some intuition about k:

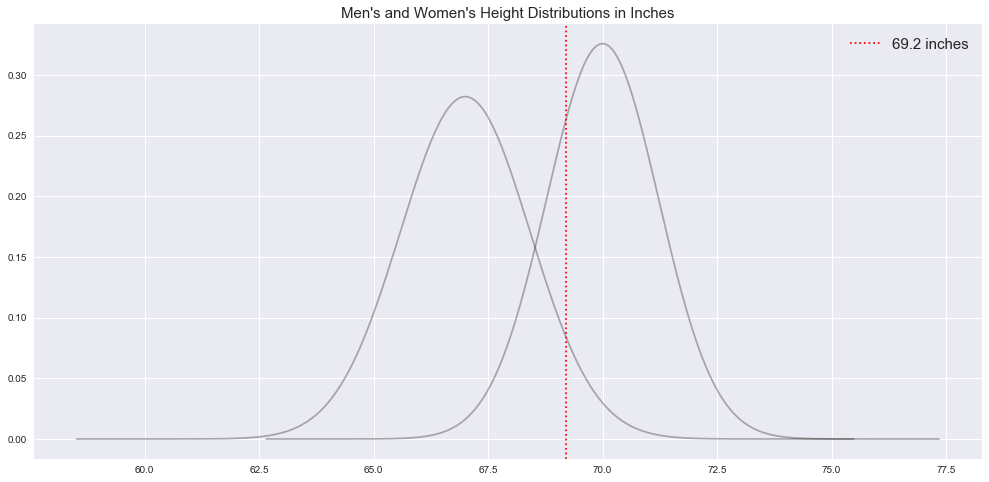
* Elbow method
* Silhouette analysis

# GAUSSIAN MIXTURE MODELS (GMMS)

A classic example of how real-life data can exhibit a normal (Gaussian) distribution with non-uniform variance is height by gender. Both men and women have approximately normally distributed heights, but their means and variances differ.



But what if all we were given was a data set of heights and asked to determine the person's gender it came from? Sounds like a clustering problem, doesn't it? Say we randomly sample a height from our data and it's 69.2 inches:



Gaussian mixture models, or GMMs, work differently than K-means in that they allow for directional variation (although still assume approximate normality) and are a "soft clustering" method. Based on the above plot, we can see that 69.2 inches has a higher probability of coming from the male height distribution but there is also a probability that it could be a woman's height.

GMMs fit a number of normal distributions to our data set by estimating their parameters using what's called expectation maximization. This is a two-step iterative algorithm (in some ways similar to K-means):

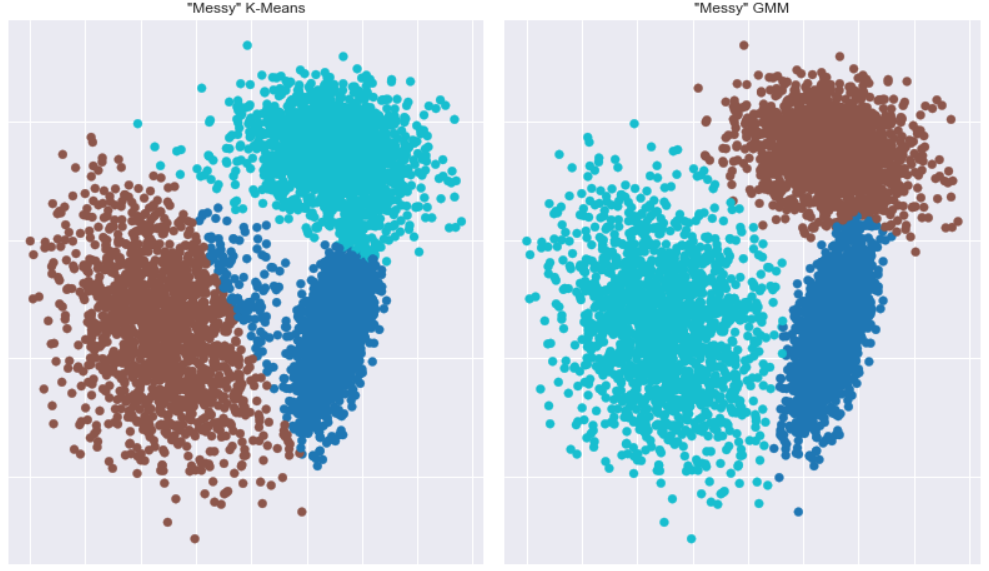
* **EXPECTATION**

Generate a number of distributions with reasonable parameters (mean and variance) based on the given data, then "ask" every data point how likely is it to fall within each. As with K-means, you need to specify the number of clusters — or in this case, we call them "components" — a priori.

* **MAXIMIZATION**

Iterate and update our distribution parameters to maximize the data points' likelihood of being assigned to the most probable cluster.

Our "messy" data is normally distributed, but not isotropic (i.e. "anisotropic" which implies diff variance in diff directions). Furthermore, the "messy" clusters aren't well separated, so K-means has a hard time judging which centroid each data point is closest to.



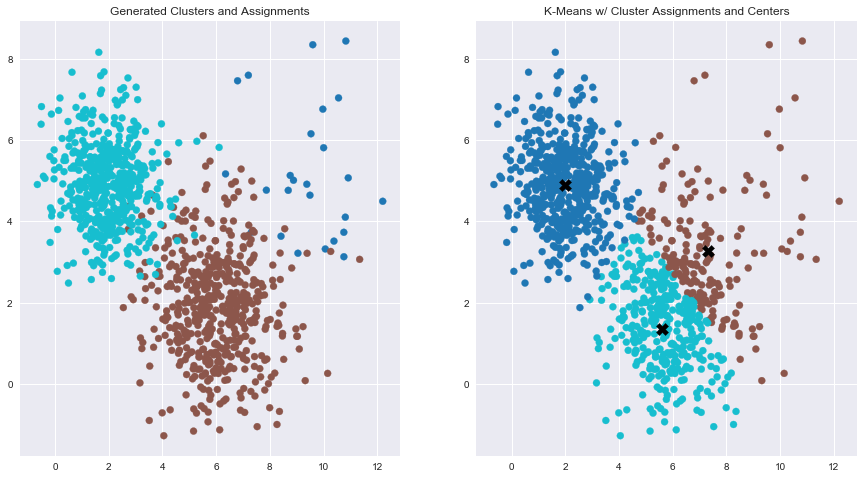
GMM clustering looks much better.

Pros: GMM allows data to vary anisotropically and provides probability estimates of cluster membership rather than "hard labeling" data points like K-means.

Cons: GMM still assumes normal distributions across dimensions and requires the number of components/clusters are specified a priori.

# DBSCAN

What would happen if cluster membership was imbalanced?



On the left side is the raw data generated, so n = 500 for both the turquoise and brown clusters and n = 25 for the blue cluster — clearly unbalanced. If we try to cluster our data points using K-means again under the assumption that there are three clusters, we see what happens in the right plot. Not pretty!

Since the K-means algorithm is trying to minimize the sum or squared differences within each cluster, our centroids are naturally "pulled" towards larger, denser clusters. If the number of clusters is not known a priori, we can estimate it using the silhouette score to determine the best number. This brings up yet another consideration: What if some data points don't belong in any cluster? What if they are outliers or noise? Both K-means and GMMs include every data point in a cluster no matter how far away it is from the nearest centroid.

Density-based clustering methods are one way of addressing these situations. The most popular method is density-based spatial clustering of applications with noise (DBSCAN), which differs from K-means in a few important ways:

* DBSCAN does not require the analyst to select the number of clusters a priori — the algorithm determines this based on the parameters it's given.
* It excels at clustering non-spherical data.
* It does not force every data point into a cluster — if a data point lies outside of the specified starting parameters, it will be classified as "noise" and not included in any resulting cluster.

Rather than randomly initialize centroids within our 2D data set and shuffle them into the best position, DBSCAN looks at each data point individually. DBSCAN relies on two parameters:

Rather than randomly initialize centroids within our 2D data set and shuffle them into the best position, DBSCAN looks at each data point individually. DBSCAN relies on two parameters:

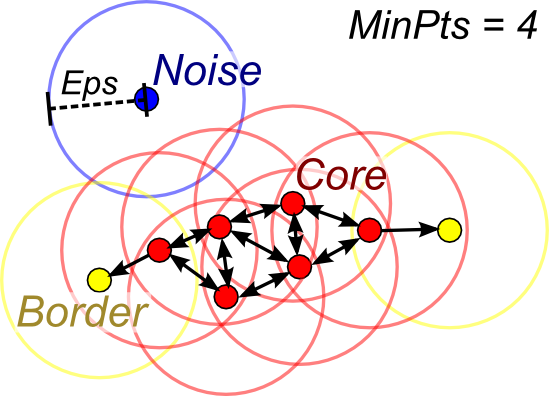
* **EPSILON, OR "EPS"**

A measure of radial distance extending from a data point. A larger epsilon means a larger distance from a data point is considered when determining if another data point should be considered in its "neighbourhood" and vice versa.

* **MINIMUM POINTS**

The number of other data points within a data point's "neighbourhood" for it to be considered a "core" data point.

Consider below figure:



The image above shows how DBSCAN parameters work. The circles from each data point represent the epsilon distance. For a data point to be considered "core" and therefore "clusterable," it needs to have the minimum number of data points fall within its epsilon radius (in this example, that number is four including the data point itself). "Border" points are also in the core clusters and form their edges. The blue data point has neither the minimum number of data points in its neighbourhood nor does it fall within a core data point's neighbourhood and is thus labeled as "noise."

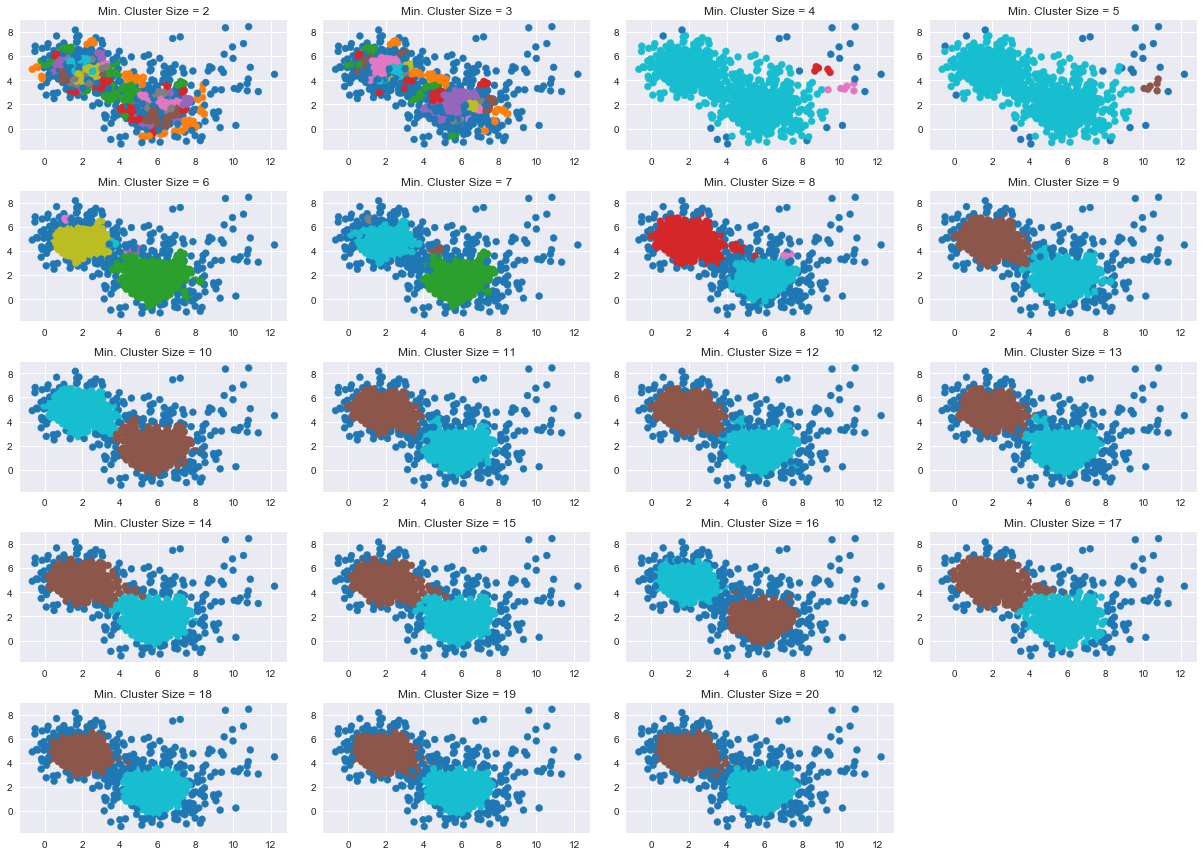
A major drawback with DBSCAN is that it's sensitive to the epsilon and minimum points parameters you give it. By setting "eps" at four vs. five could completely stop DBSCAN from working appropriately.

# HDBSCAN (HIERARCHICAL DENSITY-BASED SPATIAL CLUSTERING)

An evolved version of DBSCAN, called "HDBSCAN" (the H for "hierarchical"), attempts to allow for clusters of differing variances and densities. HDBSCAN really only requires us to provide one parameter: minimum cluster size. Think of it like hierarchical/agglomerative clustering (that other clustering method you may have been taught in stats class): Every data point starts as part of its own cluster and iteratively clusters with the next nearest data points until all data points are clustered together. The minimum cluster size parameter allows us to toss out clusters below this threshold.

It's more intuitive from a business standpoint to determine what size a cluster/segment needs to be in order for it to be considered "actionable" (i.e., you aren't going to create a targeted marketing campaign for five customers but you might for 50,000).

Let's see how HDBSCAN handles our unbalanced/noisy data with a variety of minimum cluster size parameters:



The image above shows the minimum cluster size parameter from two to 20. You can see with smaller cluster sizes that nearly every data point is considered in its own cluster. Not very helpful. Yet, as we enlarge this number, we begin to see something more useful. Around a minimum cluster size of nine, two distinct clusters begin to form. Gradually increasing this value, we see that clusters get smaller with more and more data points labeled as noise (the darker blue coloured ones).

Pros: Does not require the number of clusters to be set a priori and allows for noisy data.

Cons: Relies on the density of data points so sparse data sets may not work well and may lose efficiency on higher dimensional data sets.

# REFERENCES

<https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>