Python Machine Learning In Biology:

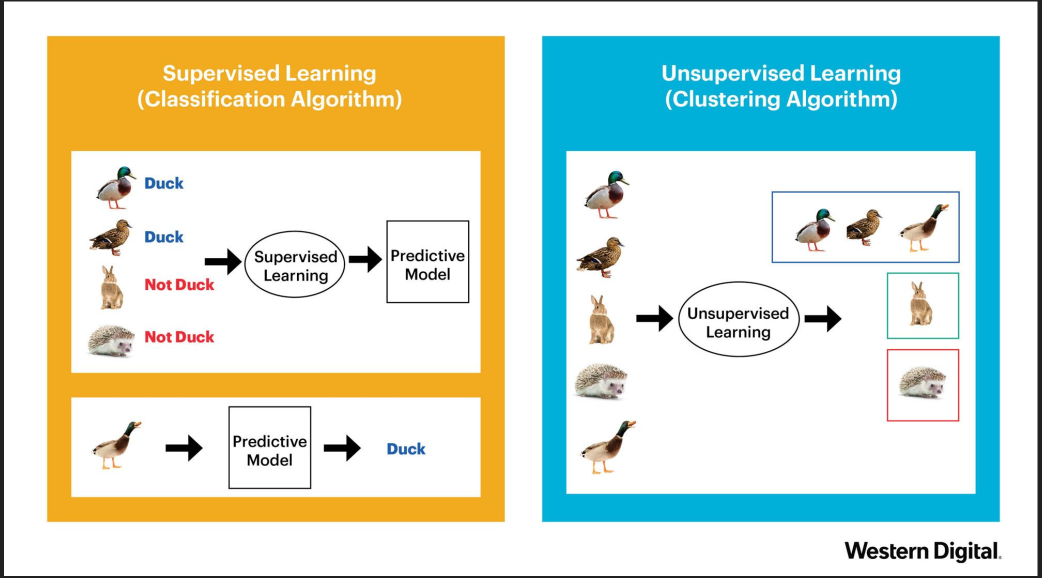
Unsupervised Learning Tour

Nichole Bennett

Supervised vs. Unsupervised Learning

Supervised --> Data has labels

Unsupervised --> Data doesn't have labels

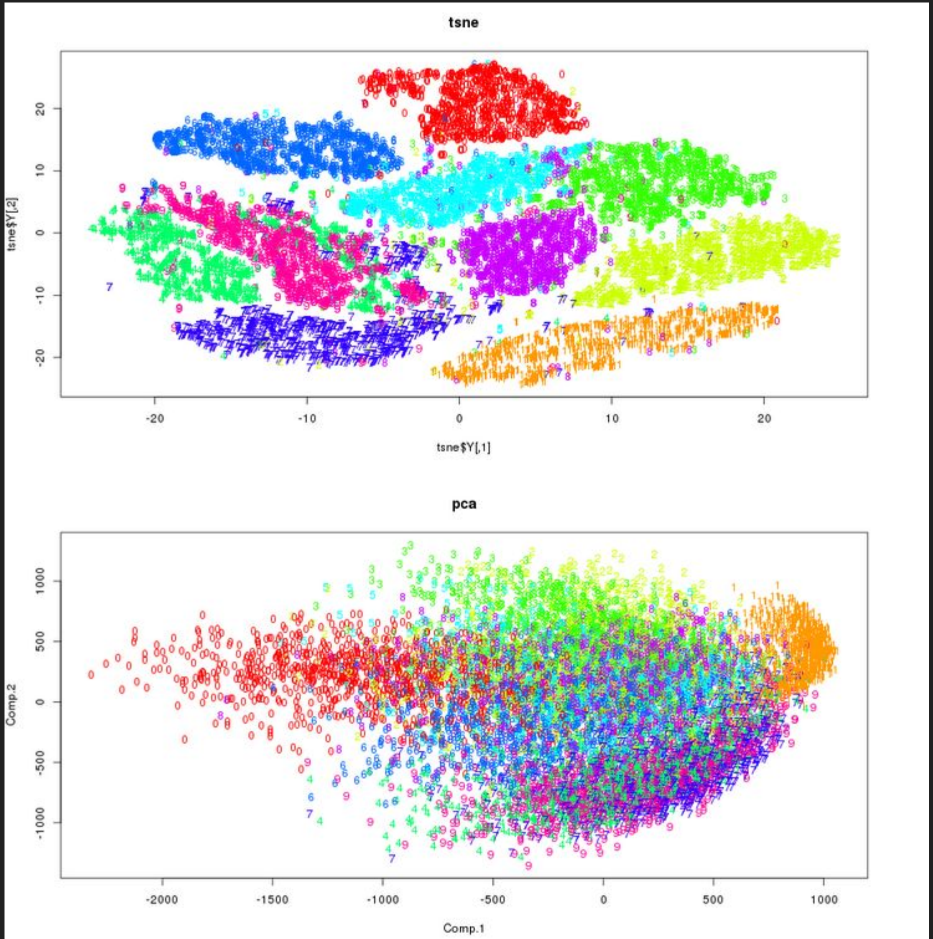


Unsupervised Learning

The goal is to find interesting structure or information in our data.

Applications:

- Visualize the structure of complex data
- Density estimates to predict probabilities of events
- Compress and summarize data
- Extract features for supervised learning tasks
- Discover important clusters or outliers



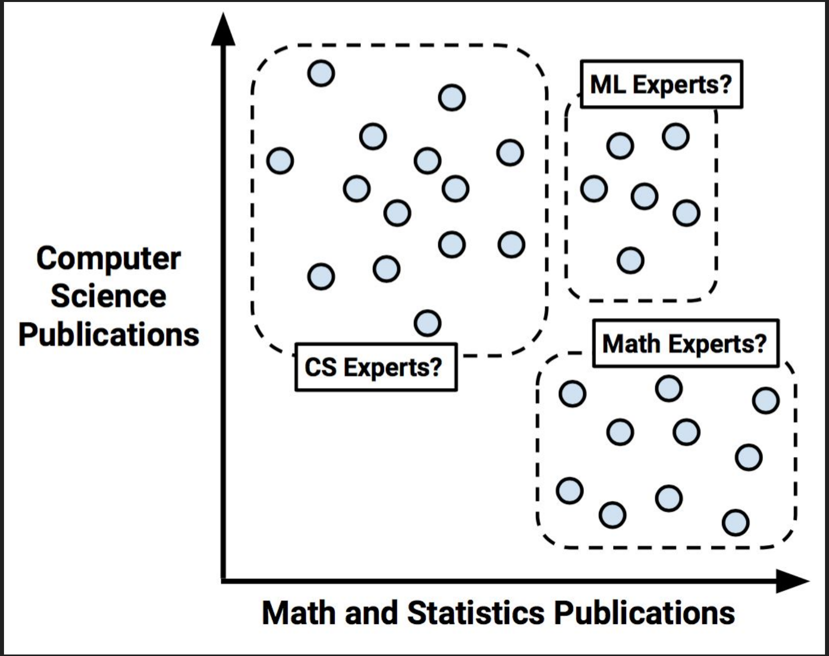
Two Main Types of Unsupervised Learning

Transformations

- Processes that extract or compose information

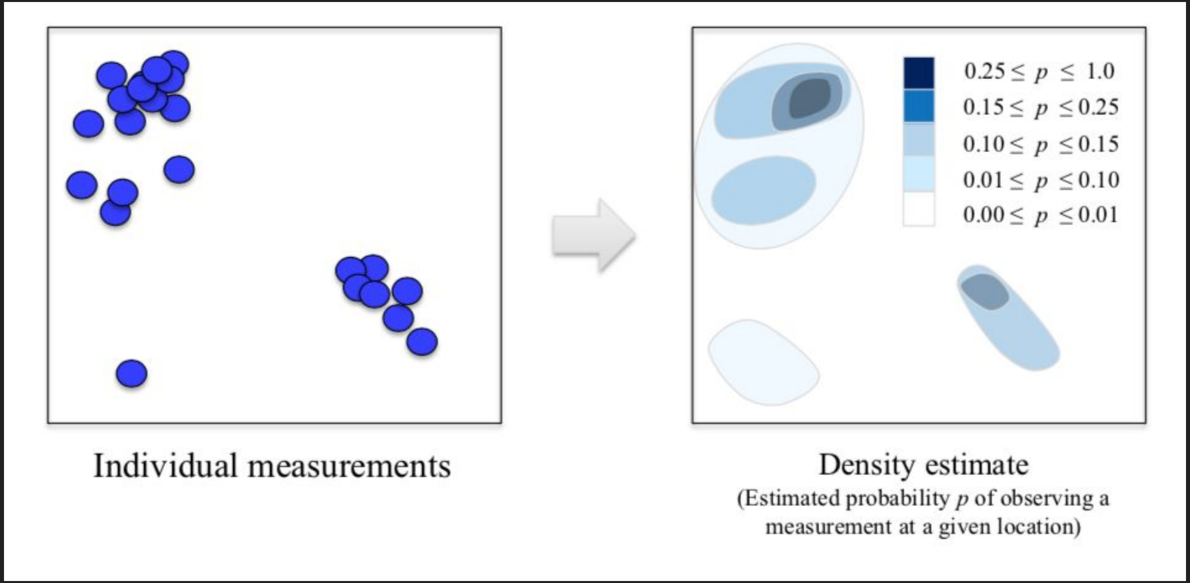
Clustering

- Find groups in data
- Assign data points to groups

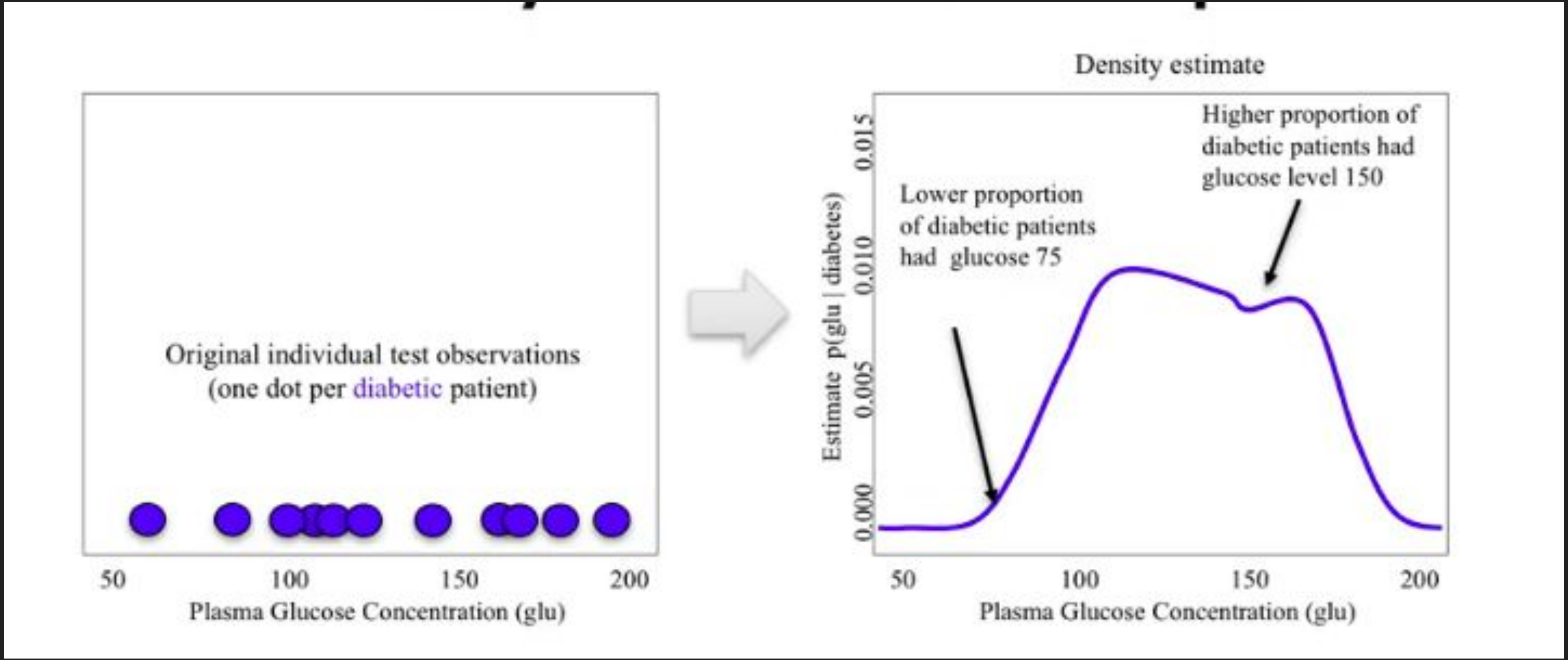

ML Experts?
Computer Science Publications
CS Experts?
Math Experts?
Math and Statistics Publications

Transformation: Density Estimation Visualization

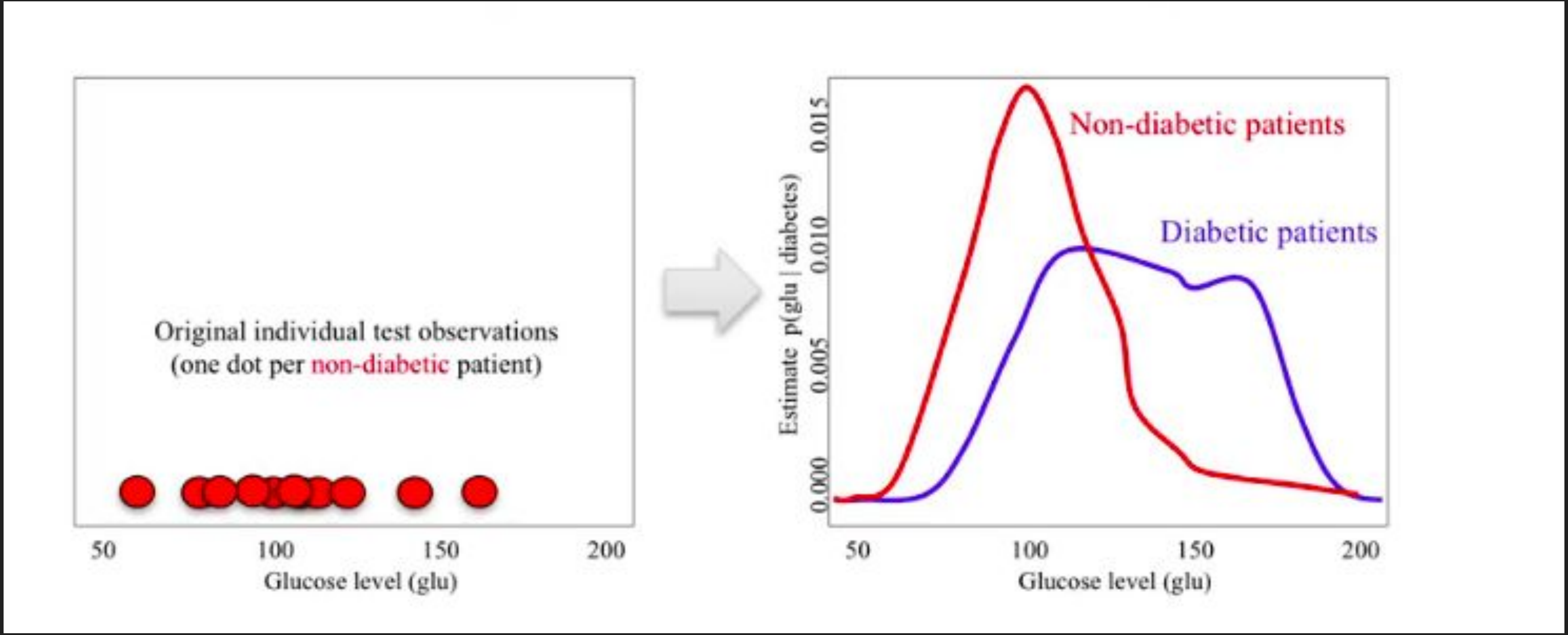
Have a set of measurements scattered throughout an area, want to create a smooth version that gives a general estimate for how likely it would be to observe a particular measurement in some area of space



Density Estimation Example

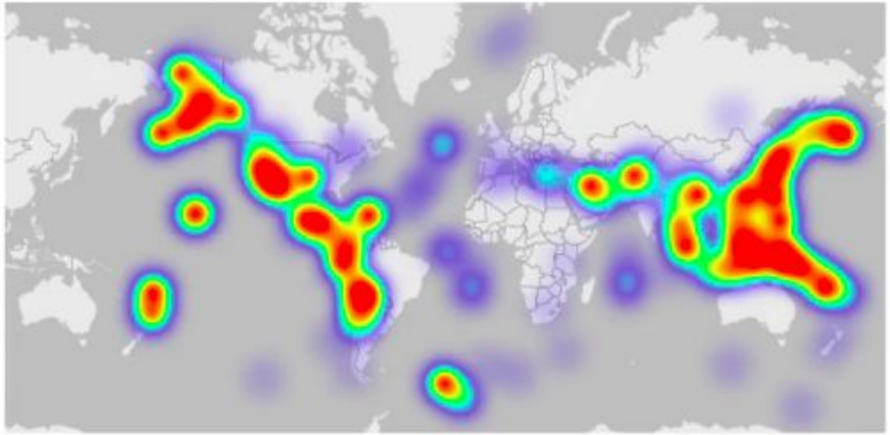


Density Estimation Example



Density Estimation

- Can be used to provide features for classification or regression
- Calculates a continuous probability density over the feature space, given a set of discrete samples in that feature space

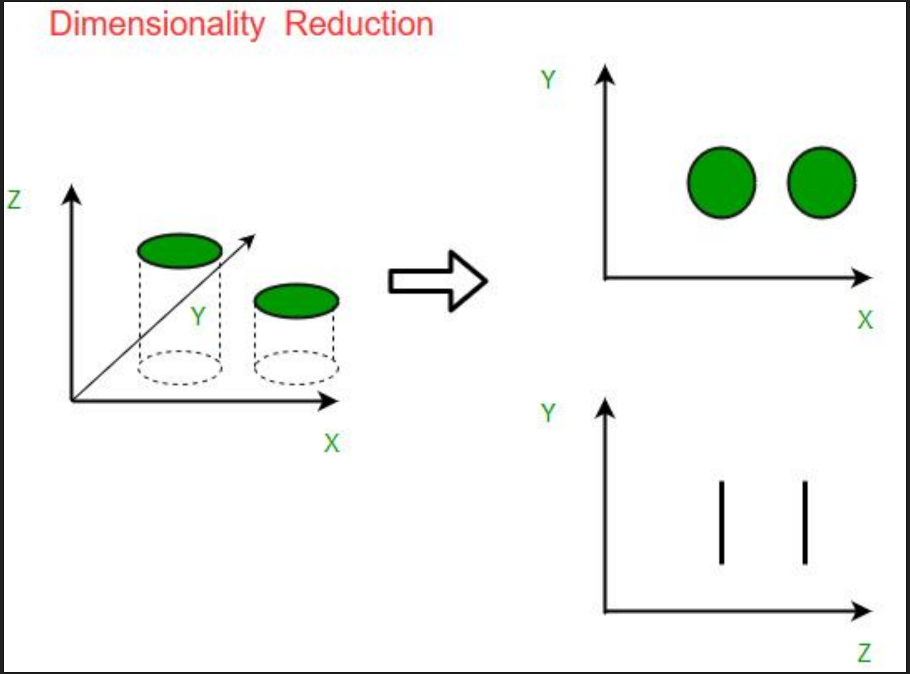


Recent global earthquake activity (U.S. Geological Survey data)

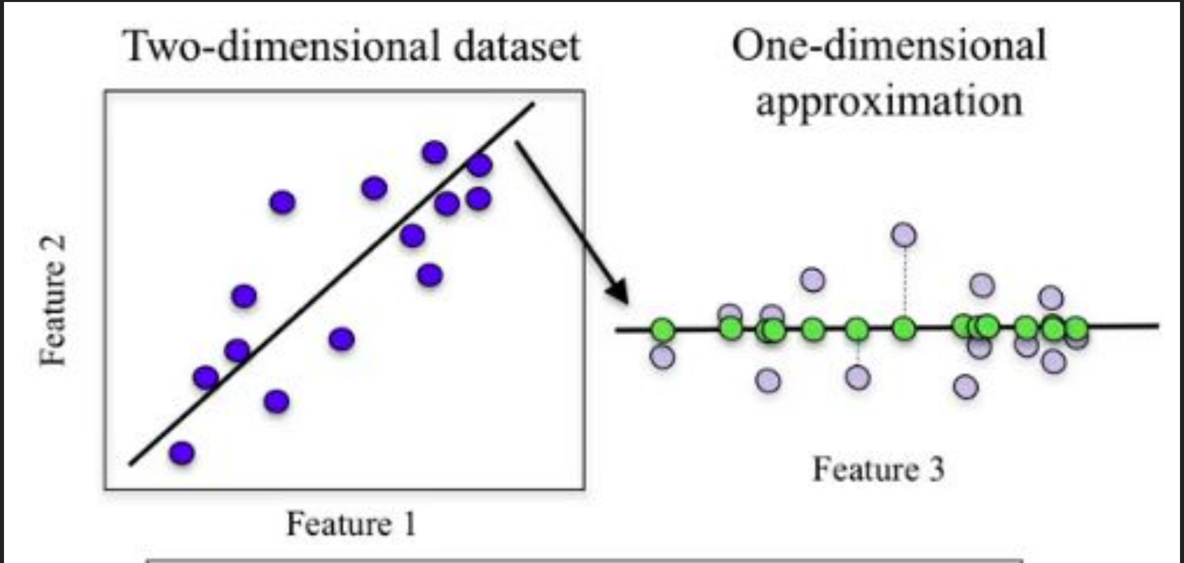
Source: http://www.digital-geography.com/csv-heatmap-leaflet/

Transformation: Dimensionality Reduction

- Find an approximate version of the dataset with less features
- Used for exploring or visualizing a dataset to understand groupings or relationships
- Often visualized as a 2-D scatterplot
- Also used for compression and finding features for supervised learning

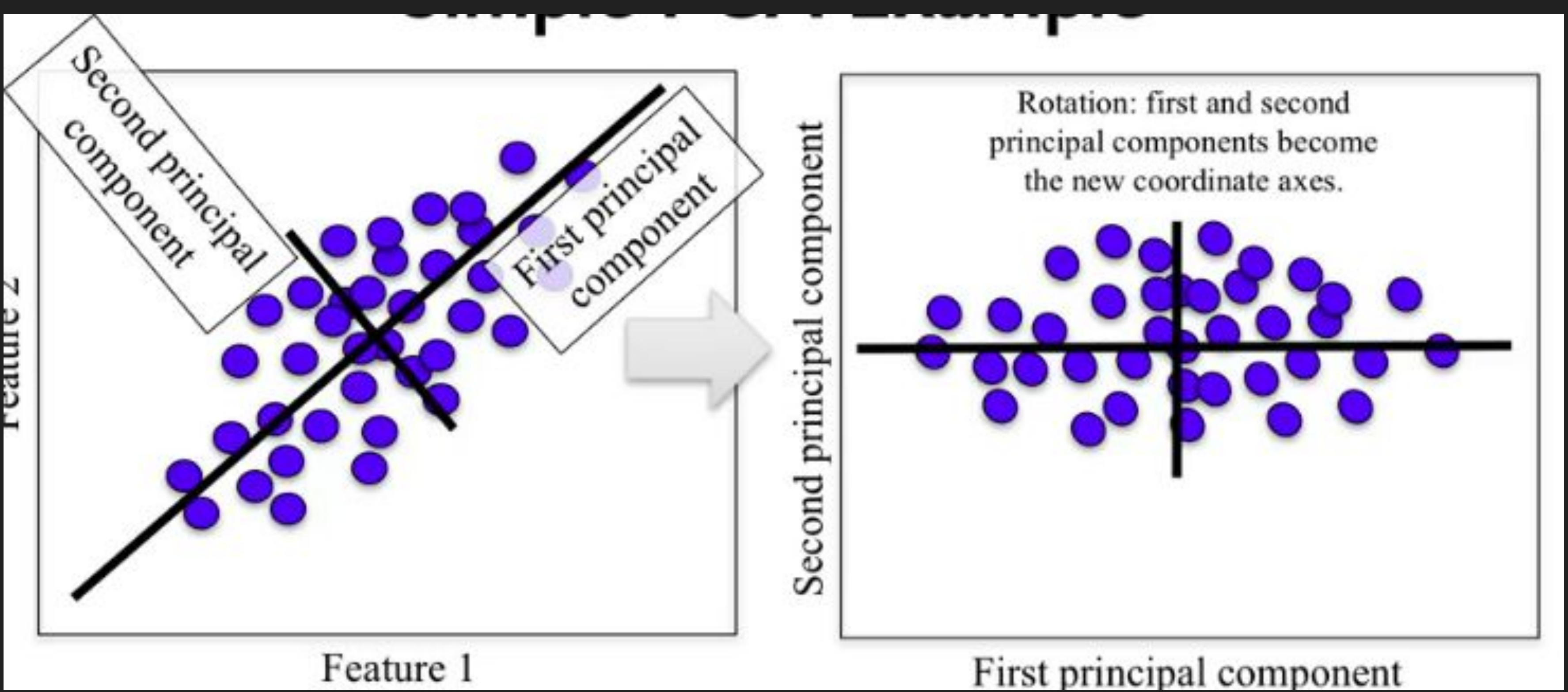


Dimensionality Reduction



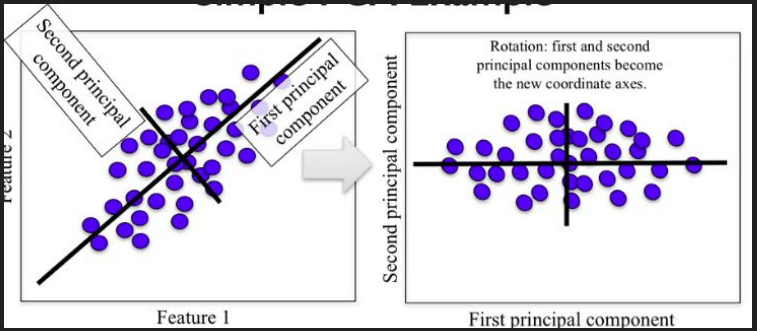
The one-dimensional approximation is obtained by projecting the original points onto the diagonal line and using their position on that line as the new single feature.

Principal Components Analysis

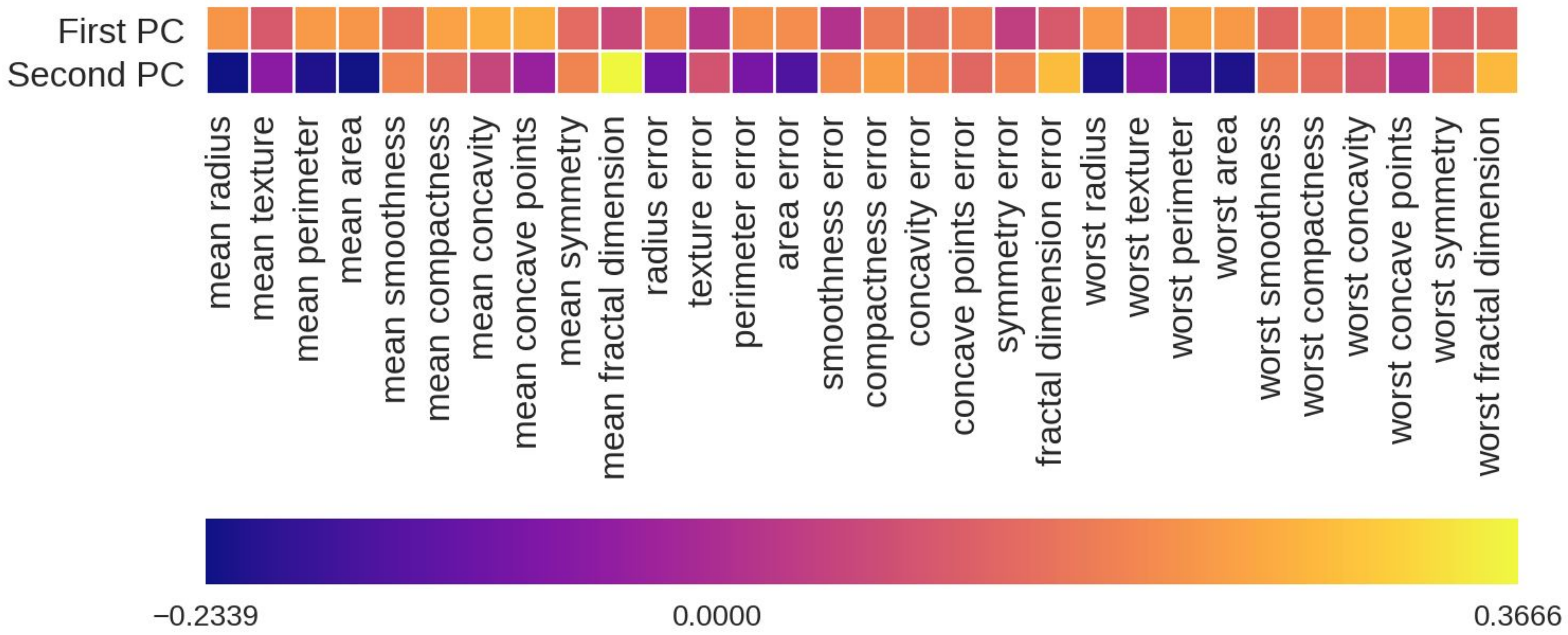

Second principal component
First principal component
Feature 2
Feature 1
Rotation: first and second principal components become the new coordinate axes.
Second principal component
First principal component

Principal Components Analysis

- Take the cloud of original data and find the rotation of it so the dimensionals are statistically uncorrelated
- Drops all but the most relevant information to capture most of the variation in the original dataset
- With > 2-D, rinse and repeat until you have the number of desired principal components
- PCA finds the best possible characteristics, that summarises the classes of a feature.
- The most challenging part of PCA is interpreting the components.


Second principal component
First principal component
Feature 2
Feature 1
Rotation: first and second principal components become the new coordinate axes.
Second principal component
First principal component

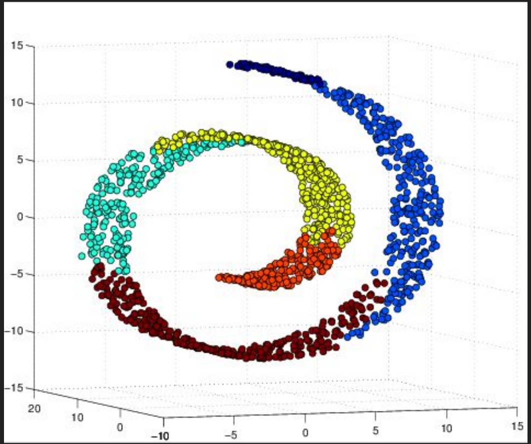
Principal Components Analysis

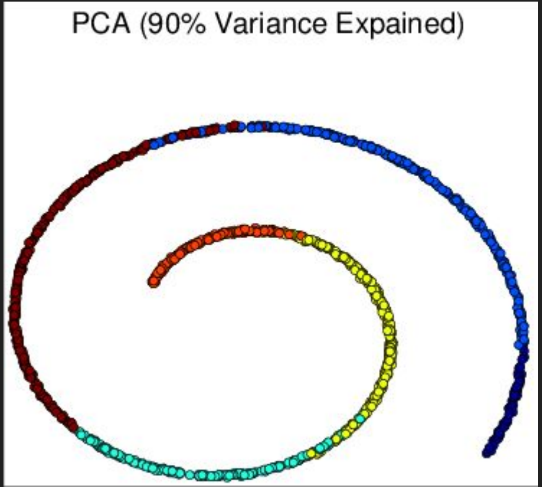


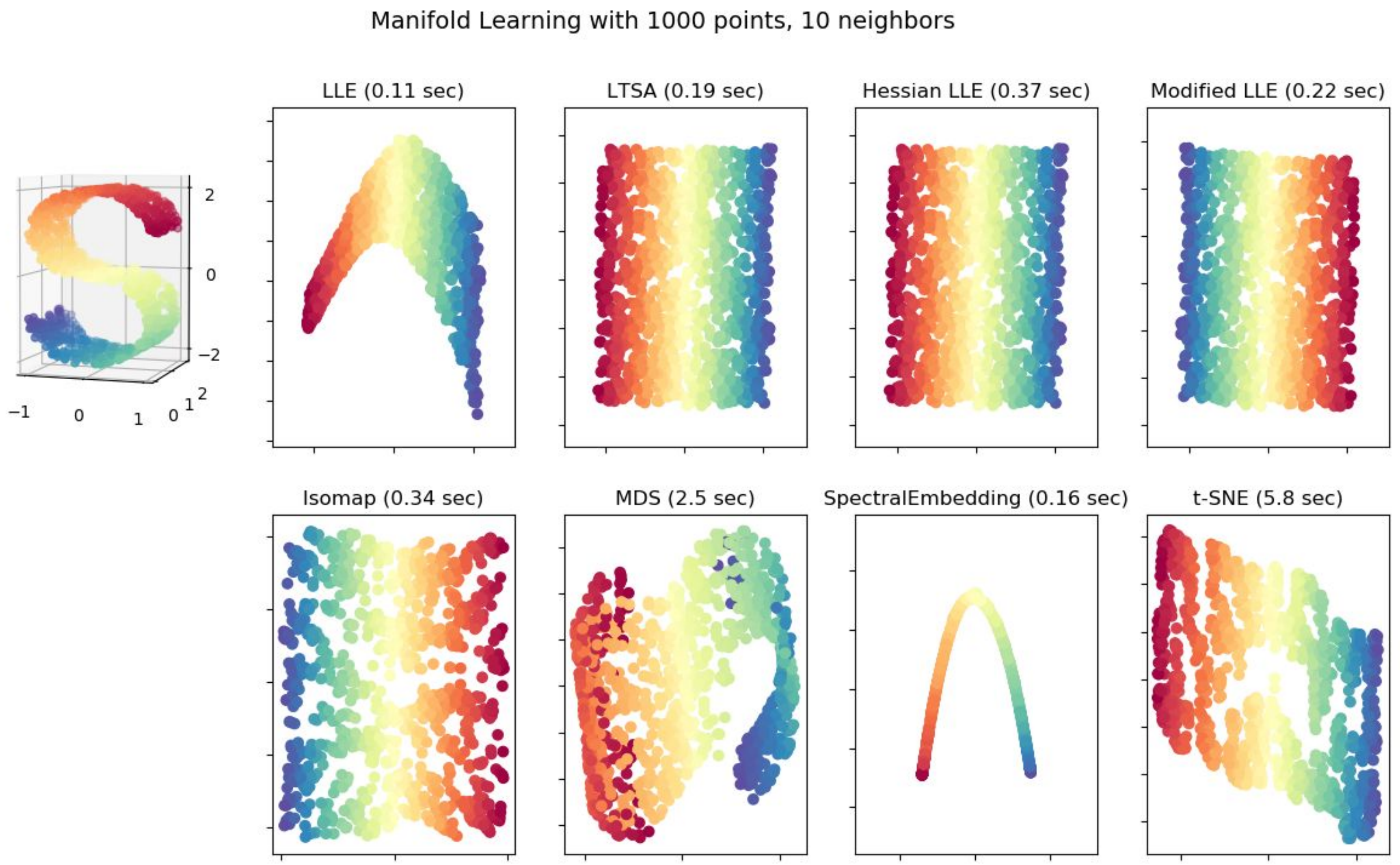
Manifold Learning Algorithms

PCA is a good initial tool, but it may not be able to find more subtle groupings that produce better visualizations for more complex data.

Manifold Learning Algorithms are a family of unsupervised learning algorithms that are good at finding low-dimensional structure in high-dimensional data and are useful for visualizations

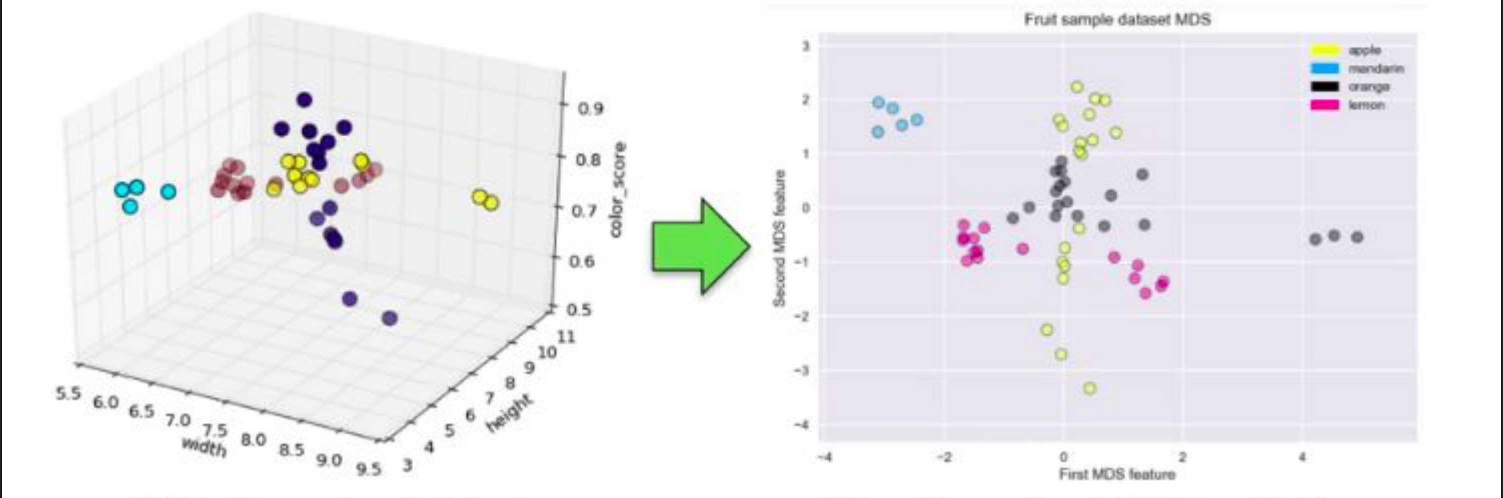





Manifold Learning with 1000 points, 10 neighbors
2
0
−2
−1
0
1
0
1
2
LLE (0.11 sec)
LTSA (0.19 sec)
Hessian LLE (0.37 sec)
Modified LLE (0.22 sec)
Isomap (0.34 sec)
MDS (2.5 sec)
SpectralEmbedding (0.16 sec)
t-SNE (5.8 sec)

Multidimensional Scaling (MDS)

- Visualize high-dimensional data and project it onto a lower-dimension
- Preserves distance information



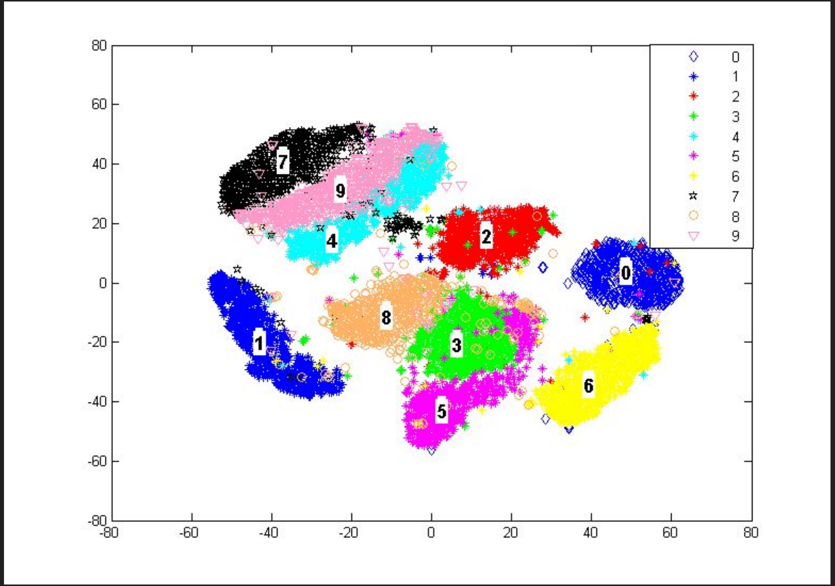
High-dimensional dataset Two-dimensional MDS projection

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Powerful manifold learning method

- Finds the 2-D representation by projecting and preserves neighbor distance information

- Works better on data with more well-defined local structure



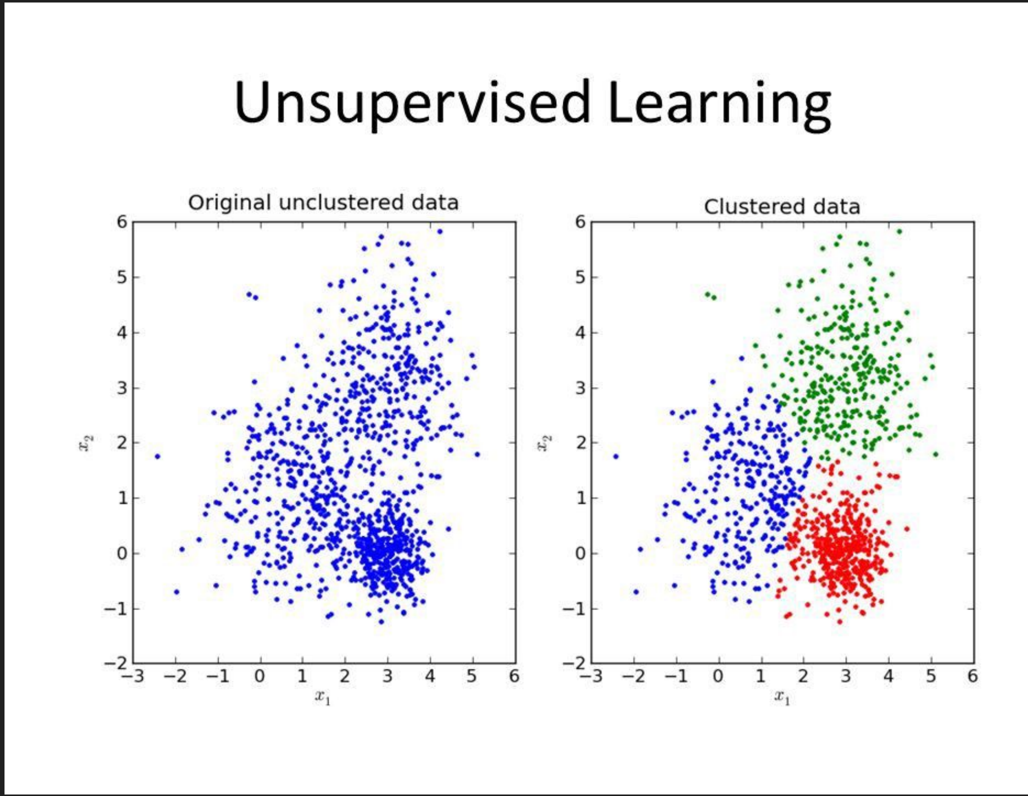
https://distill.pub/2016/misread-tsne/#citation

Clustering

Data points in the same cluster are ‘close’ or ‘similar’ in some way

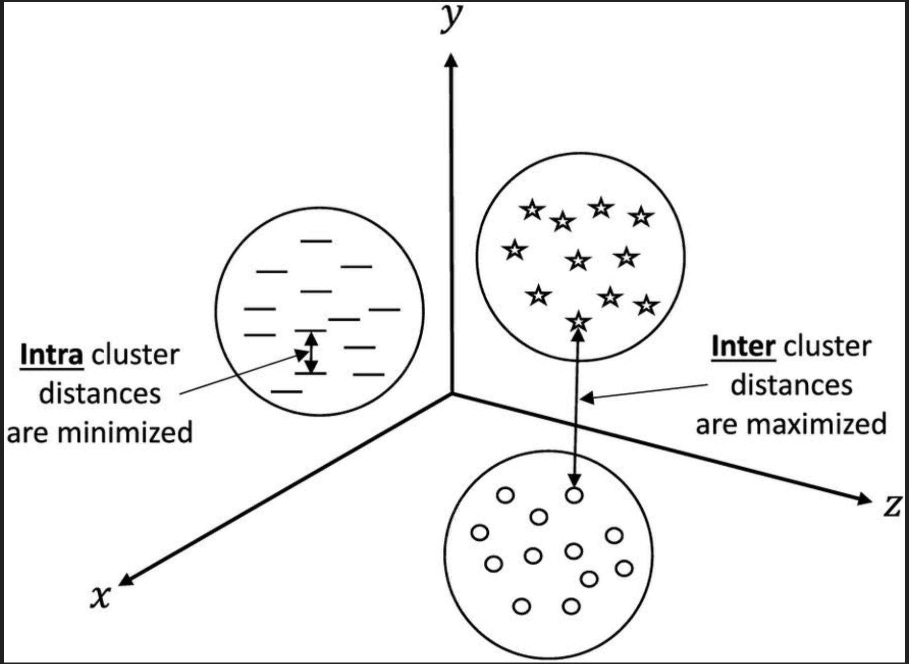
Different clusters should be ‘far apart’ or ‘different

(important to scale data)



Clustering

The goal is to minimize intra-cluster distance and maximize inter-cluster distance


y
x
z
Intra cluster distances are minimized
Inter cluster distances are maximized

Hard vs. soft clustering algorithms

Hard clustering: each sample assigned to exactly one cluster

Soft clustering (aka fuzzy clustering): each sample assigned to one or more cluster, and cluster membership expressed as a probability

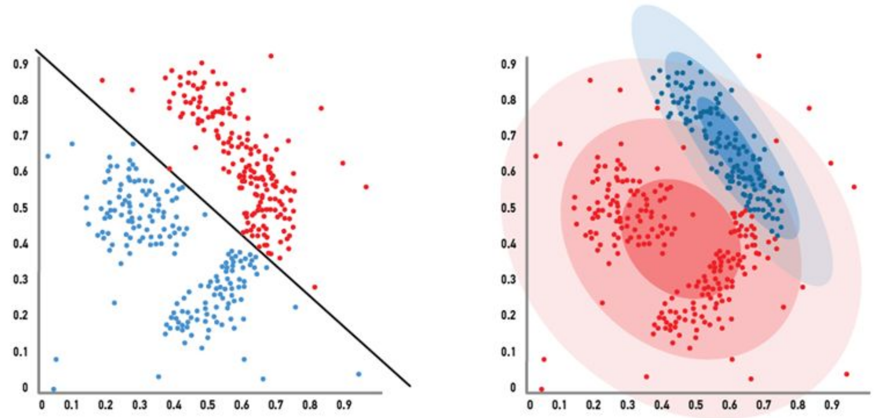


Figure 2. Classical clustering (left) and fuzzy clustering (right)

Clustering Methods

- K-Means (mean centroids)
- Hierarchical (nested clusters by merging or splitting successively)
- DBSCAN (density based)
- Affinity Propagation (graph based approach to let points 'vote' on their preferred 'exemplar')
- Mean Shift (can find number of clusters)
- Spectral Clustering
- Agglomerative Clustering (suite of algorithms all based on applying the same criteria/characteristics of one cluster to others)

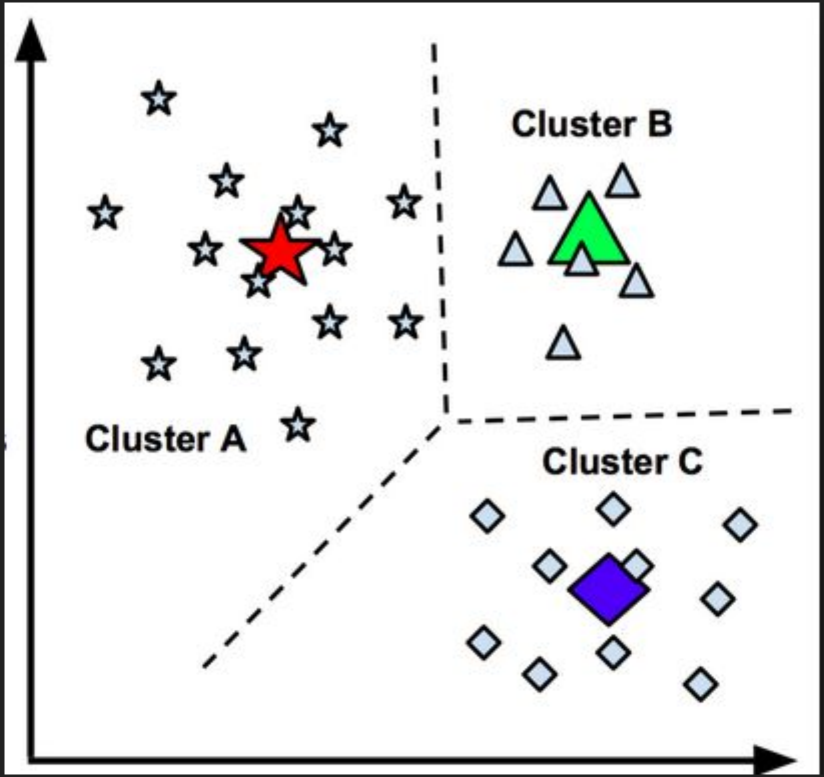
K-Means Clustering

Popular clustering algorithm

K is the number of clusters (chosen in advance)

Means refers to the mean points of the K clusters (known as centroids)

Goal is to partition the data into set such that the total sum of squared distances from each point to the mean point of the cluster is minimized.

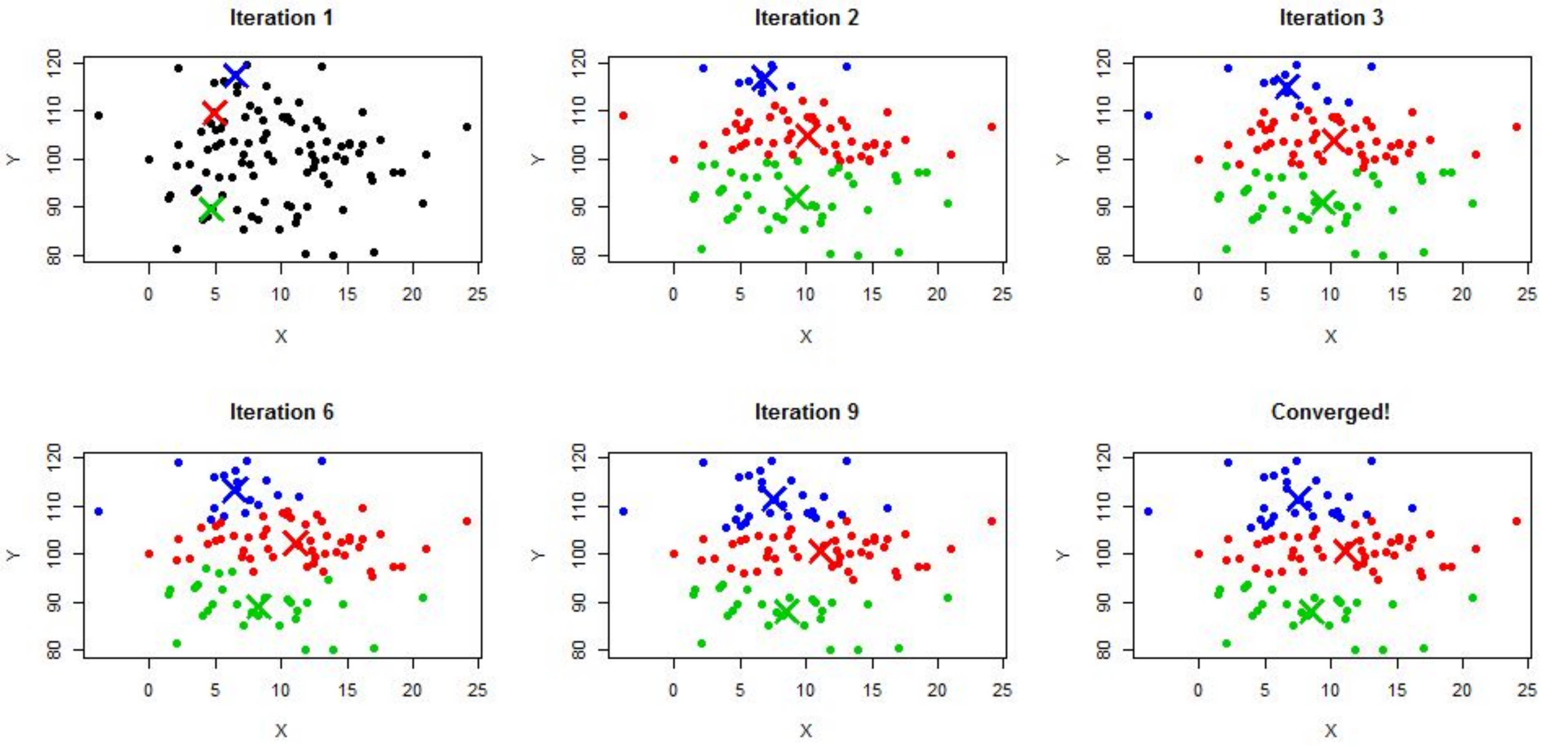


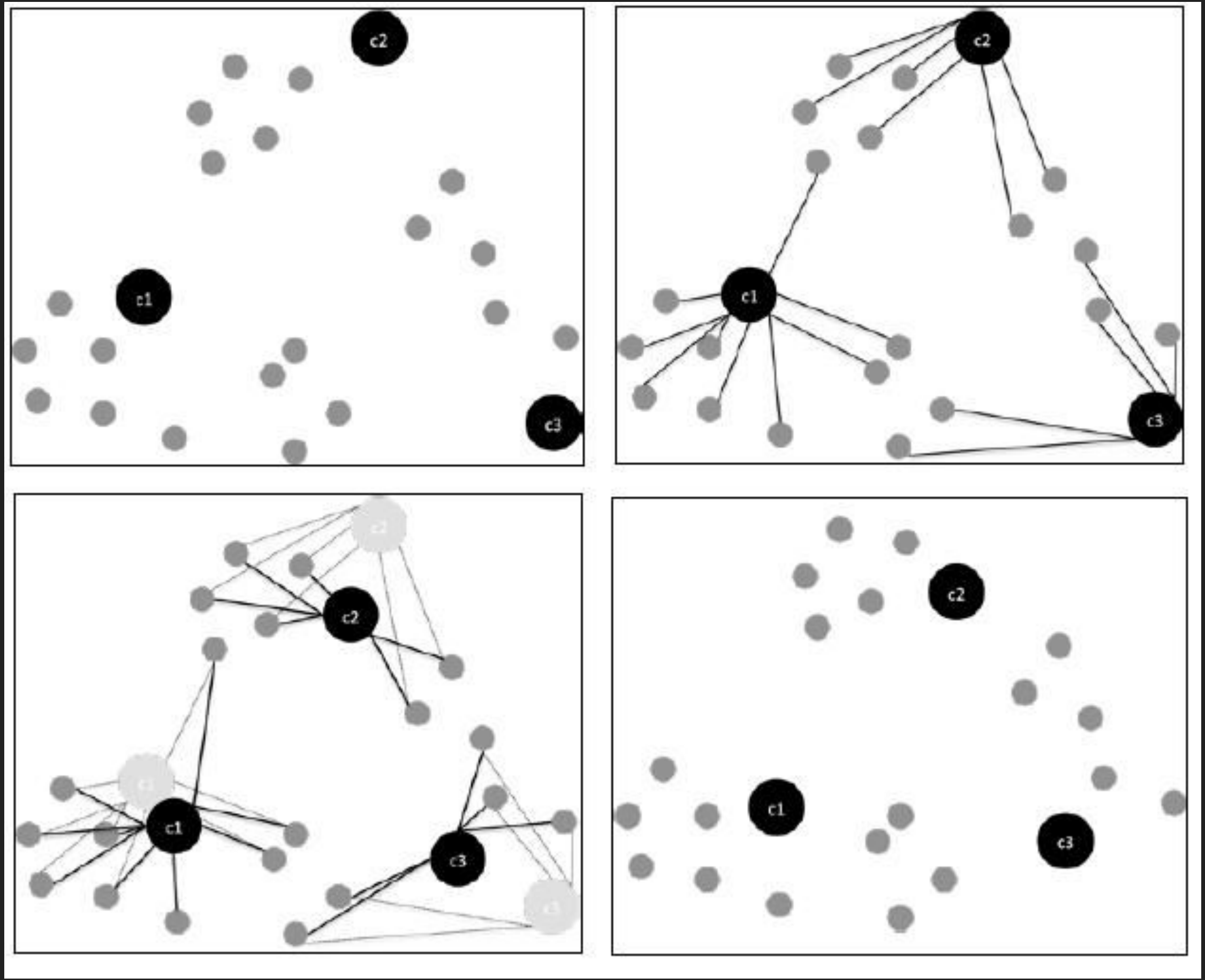
K-Means Living Demo

K-mean algorithm

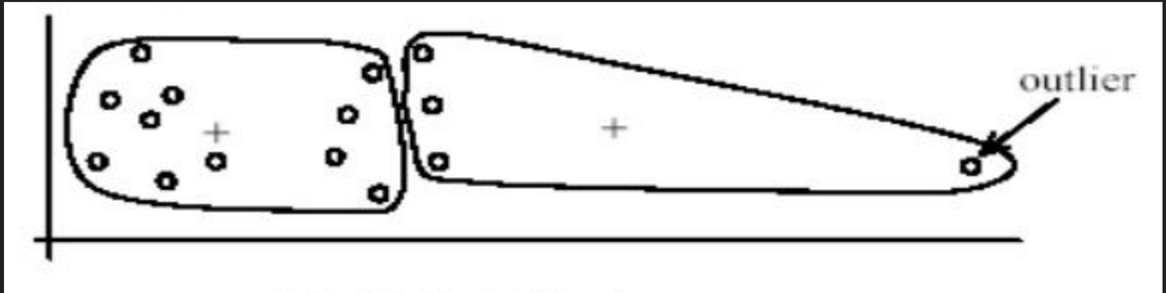
1. Pick a value for k (the number of clusters to create)
2. Initialize k 'centroids' (starting points) in your data
3. Create your clusters. Assign each point to the nearest centroid.
4. Make your clusters better. Move each centroid to the center of its cluster.
5. Repeat steps 3-4 until your centroids converge.

Unfortunately there is no formula to determine the absolute best number of k clusters. Unsupervised learning is inherently subjective! We can, however, choose the "best" k based on predetermined criteria.

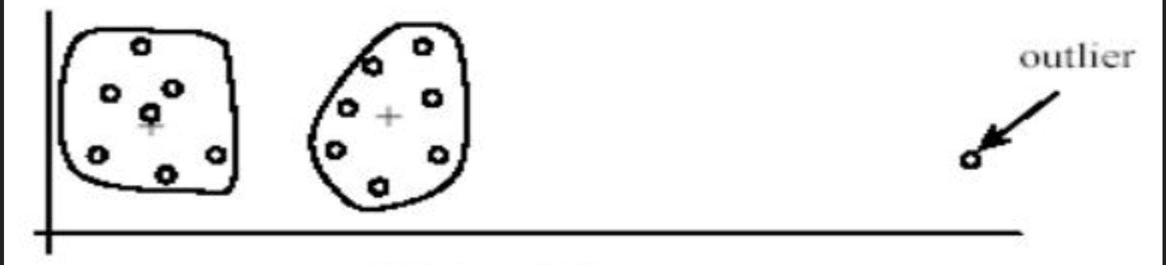

Iteration 1
Iteration 2
Iteration 3
Iteration 6
Iteration 9
Converged!
X
Y
0
5
10
15
20
25
80
90
100
110
120


c1
c2
c3

K-means is sensitive to outliers

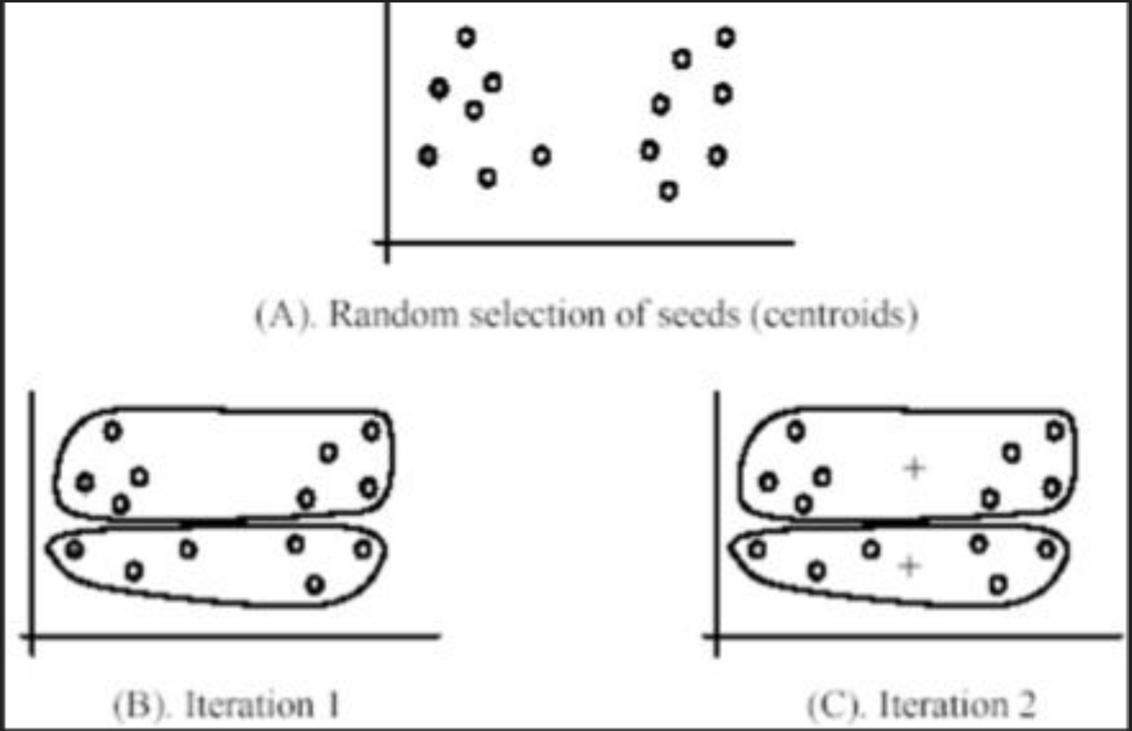


(A): Undesirable clusters



(B): Ideal clusters

K-means is sensitive to centroid initialization



Initializing Centroids

- Randomly
- Manually
- Special KMeans++ method in Sklearn (This initializes the centroids to be generally distant from each other)

Depending on your problem, you may find some of these are better than others.

Note: Manual is recommended if you know your data well enough to see the clusters without much help, but rarely used in practice.

K-means pros and cons

Advantages

K-Means is popular because it's simple and computationally efficient.

Easy to see results / intuitive.

Disadvantages

However, K-Means is highly scale dependent and isn't suitable for data of varying shapes and densities.

Evaluating results is more subjective, requiring much more human evaluation than trusted metrics.

Have to specify k.

K-means problems

Different solutions can converge with random initialization

Not everything can be reflected as a "glob" of points

Irregular shapes

Different densities

Inconsistent cluster spacing

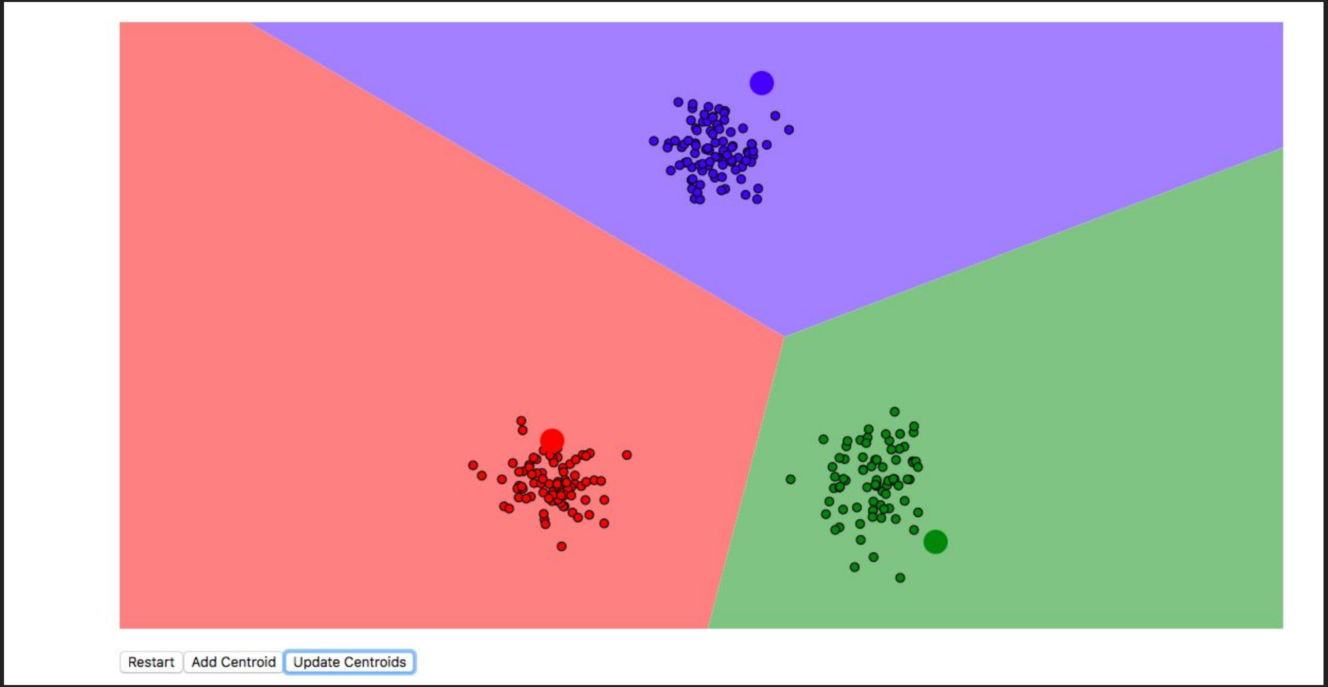
Interpretation of results difficult without subject matter expertise

Accuracy is highly subjective

No formula for the correct number of clusters

Visualizing k-means

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

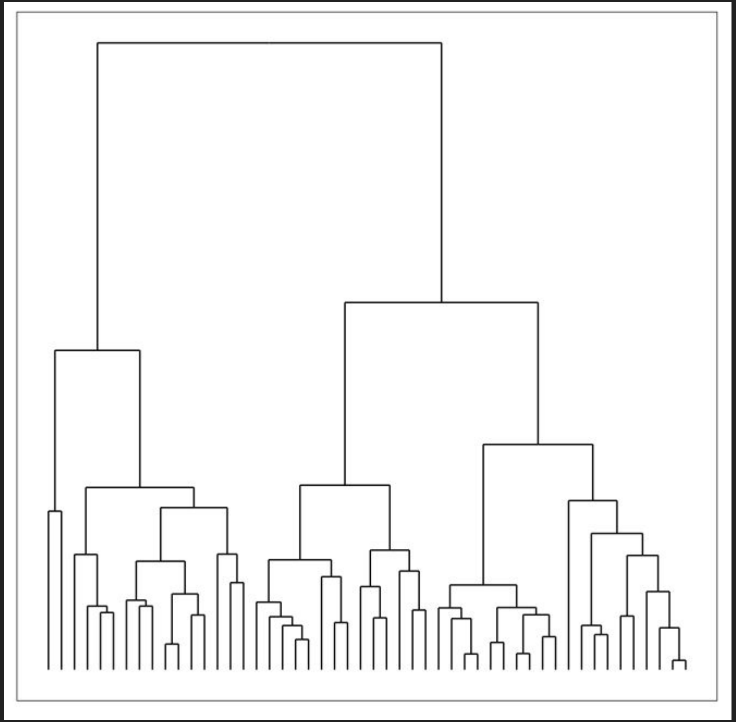


Hierarchical Clustering

Build hierarchies of clusters.

Connect the clusters in the hierarchy with links.

Once the links are determined, we can display them in what is called a dendrogram - a graph that displays all of these links in their hierarchical structure.



When does hierarchical clustering perform well?

Hierarchical clustering works well for non-spherical clusters

It also works well on smaller datasets - this algorithm has a longer computational time and doesn't work well for larger datasets

Hierarchical clustering works well for instances where we are working with frequencies.

Hierarchical Clustering vs. k-means

Both are unsupervised learning algorithms to divide data into groups.

In k-Means you decide what k is, and the algorithm finds the clusters by moving the centroids until none of the points change class.

In hierarchical clustering, the algorithm builds classification trees using the data that merge groups of similar data points. You don’t have to define “k.”

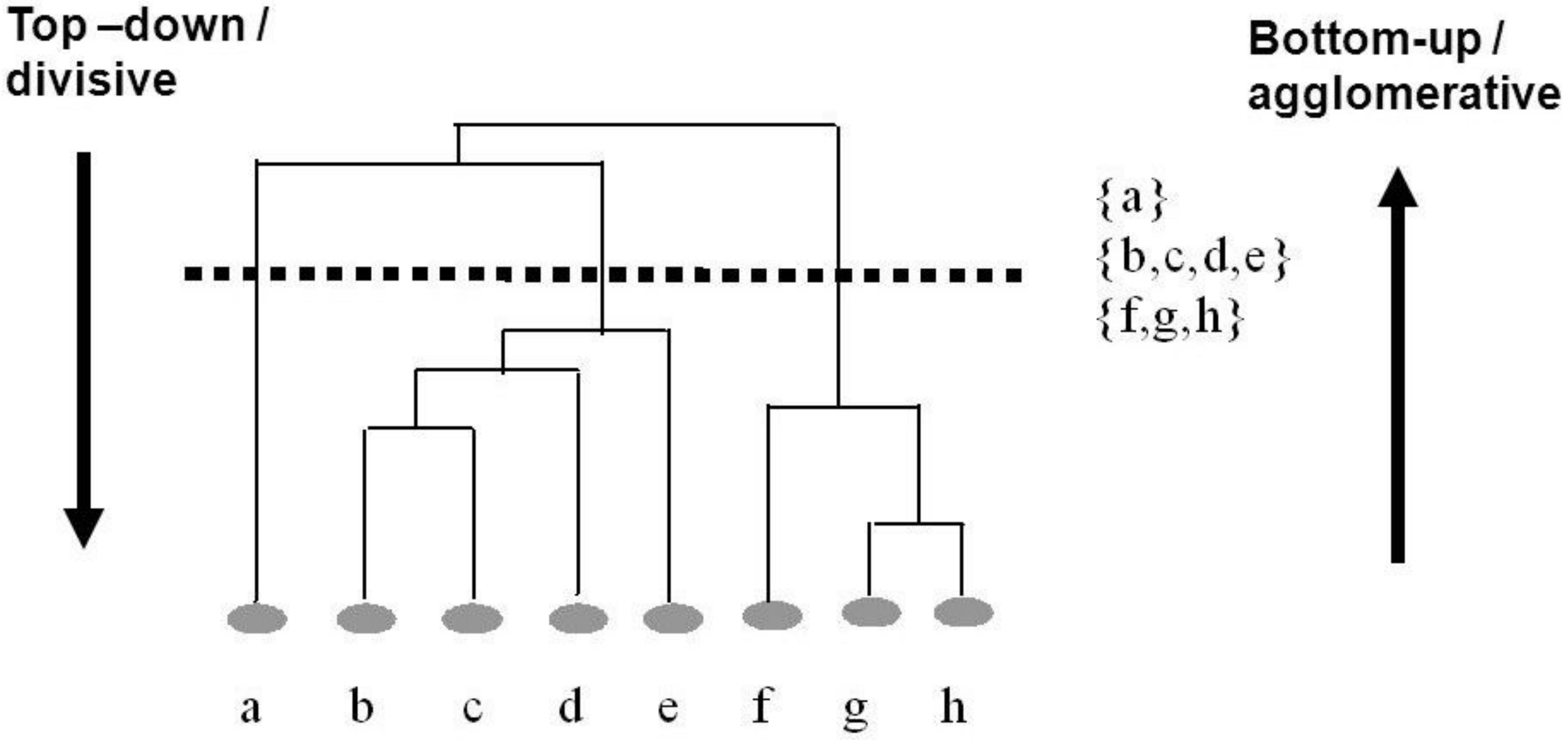
In k-Means, the boundaries between the various clusters are distinct and independent , whereas in hierarchical clustering there are shared similarities between those groups represented by the classification tree.

How Hierarchical Clustering Works

There are two forms of hierarchical clustering; agglomerative hierarchical clustering and divisive hierarchical clustering.

> Agglomerative goes from the bottom-up, starting with single data points and merging them into groups.
>
> Divisive goes from the top down, starting with all the data points and dividing them.

Whether starting from the top or bottom, at each step the algorithm makes the best choice it can to join or split the data based on the current sample.

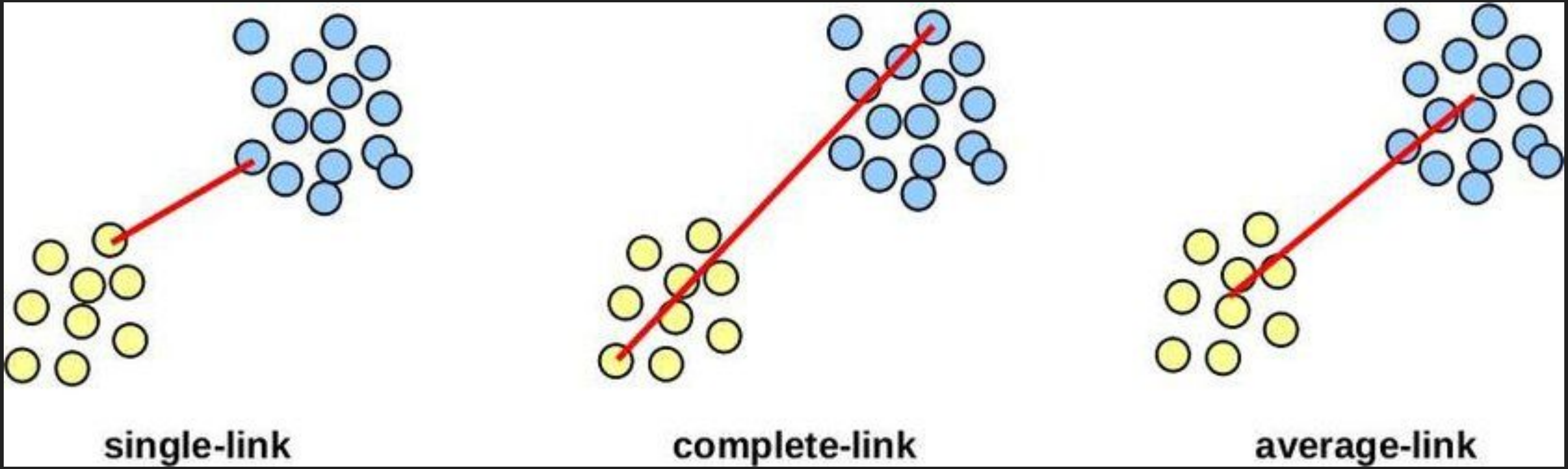

Top –down /
divisive
Bottom-up /
agglomerative
{a}
{b,c,d,e}
{f,g,h}
a
b
c
d
e
f
g
h

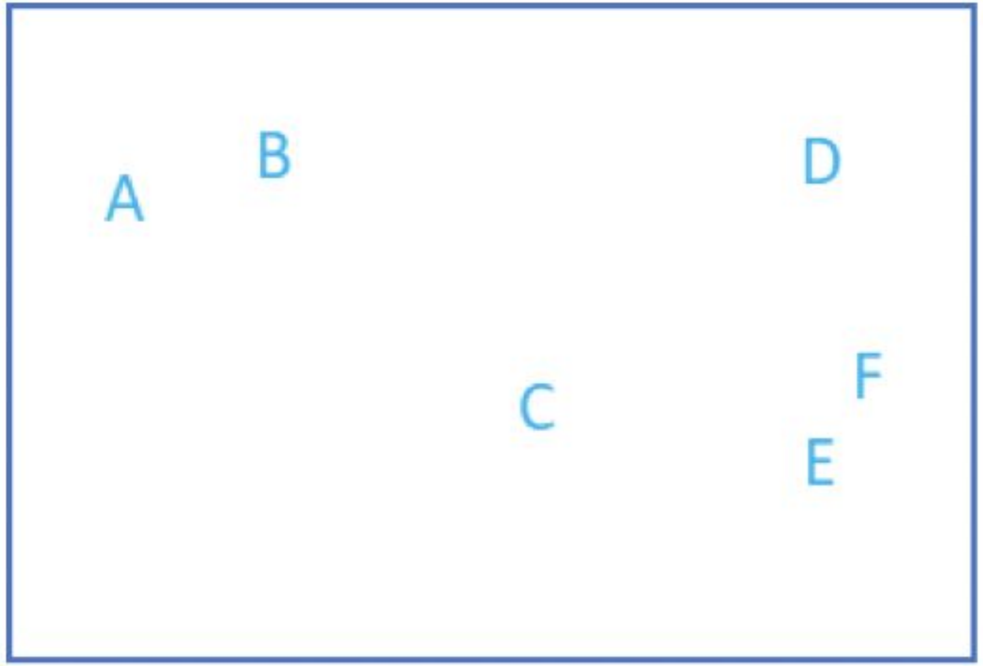
Algglomerative Clustering

Single linkage: compute the distances between the most similar members for each pair of clusters and merge the two clusters with the smallest distance

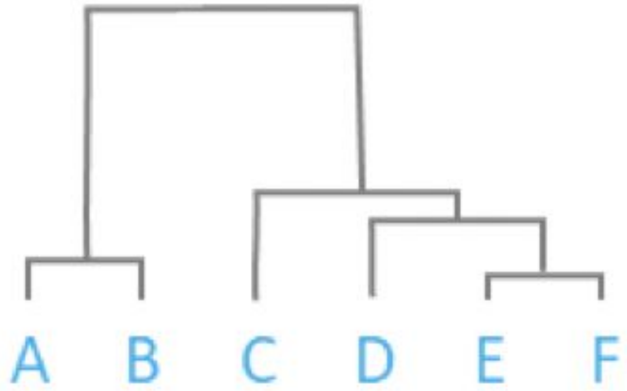
Complete linkage: same but compare most dissimilar members.

Ward: least increase in total variance of centroids of cluster (this is usually the one we pick)





Dendrogram



DBSCAN

Density-Based Spatial Clustering of Applications with Noise

Clusters of high density are separated by clusters of low density

The number of points that lie in a radius ε. You choose ε as well as the minimum number of points it takes to make a cluster.

RULES OF THUMB: choosing the minimum points - you should always aim to have the minimum number of points be greater or equal to the amount of dimensions in your data, plus one. This typically will give the algorithm a good estimation of how to evaluate the clusters. Calculating epsilon is a bit trickier and uses a method called the k-distance, which can help visualize the best epsilon.

DBSCAN Algorithm

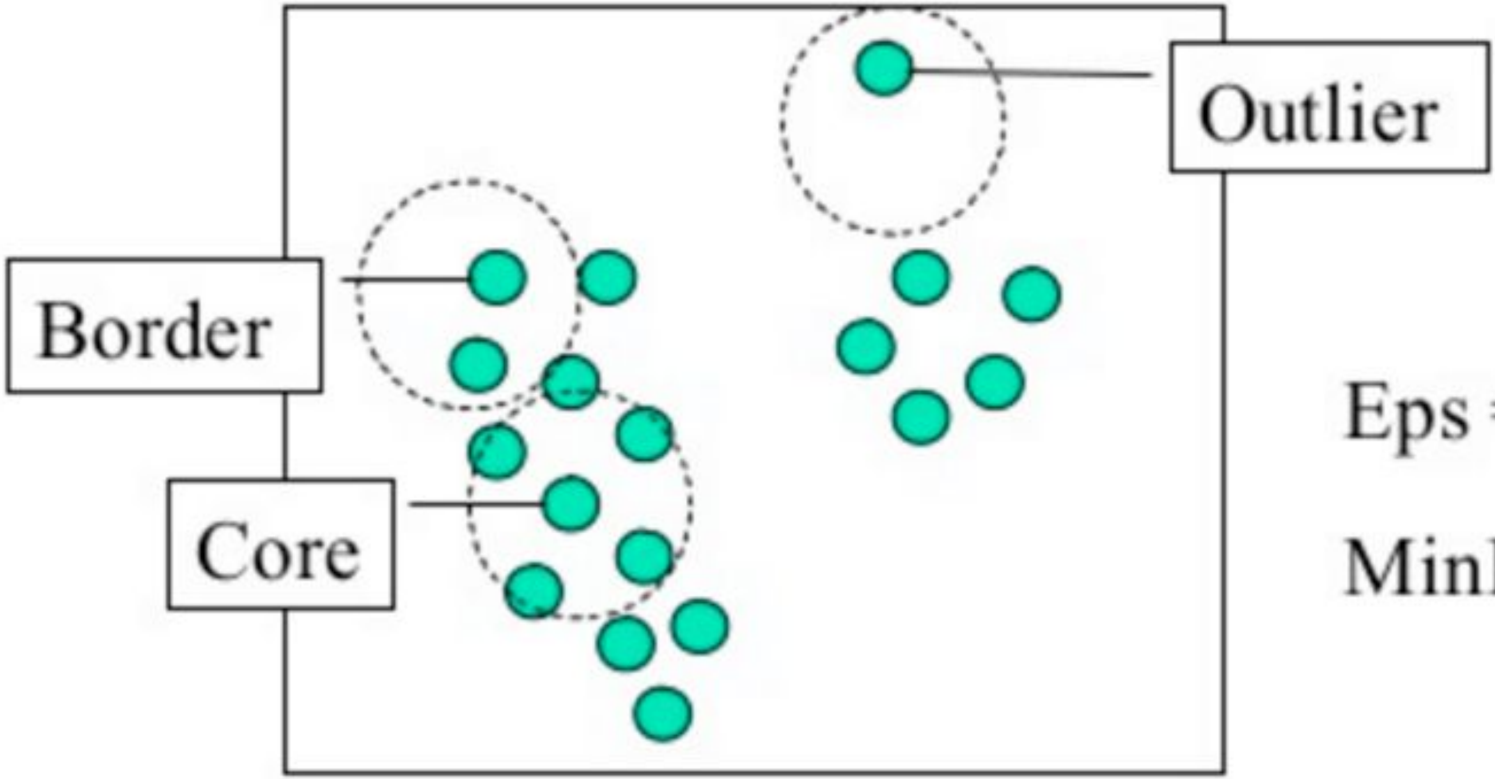
1. Choose an “epsilon” and “min_samples”

2. Pick an arbitrary point, and check if there are at least “min_samples” points within distance “epsilon”

    If yes, add those points to the cluster and check each of the new points

    If no, choose another arbitrary point to start a new cluster

3. Stop once all points have been checked



Eps = 1cm

MinPts = 5

Pros and Cons of DBSCAN

PRO: Doesn’t assume clusters have spherical shape and can handle noise

PRO: eps implicitly controls number of clusters

CON: Increasing number of features brings on curse of dimensionality

CON: Have to optimize two hyperparameters (can be difficult if density differences are large)

CON: If you don’t set the parameters well, everything gets labeled as noise (-1)

DBSCAN vs. k-means

k-means is a "general" clustering approach, DBSCAN performs especially well with unevenly distributed, non-linear clusters.

DBSCAN is density based, which means that it determines clusters based on the number of points in a certain area

By choosing too few points for DBSCAN, i.e. less than two, we'll effectively get a straight line if we connect the points, just like linkage clustering.

DBSCAN vs. hierarchical clustering

DBSCAN can be useful to us when we have a lot of dense data. IADVANTAGES –

Clusters can be any size or shape

No need to choose number of clusters

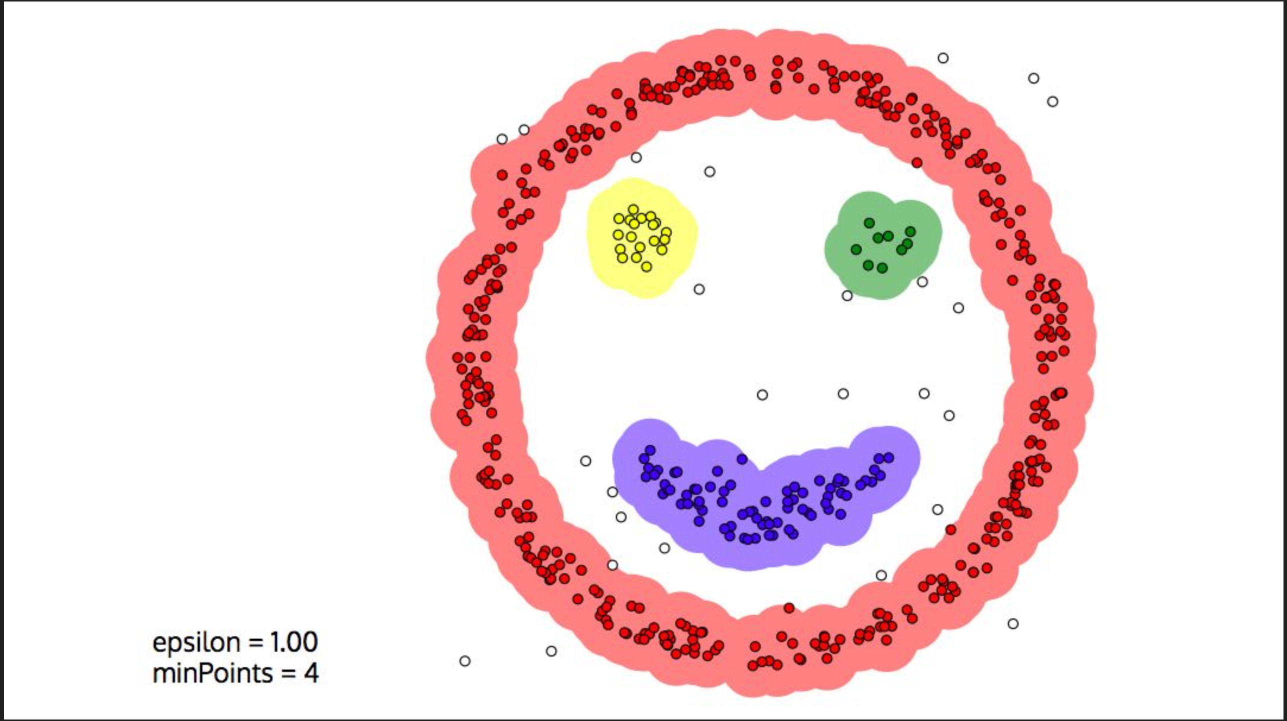
BUT

- More parameters to tune
- Doesn’t work with clusters of varying density

NOTE: Not every point is assigned to a cluster!

Visualizing DBSCAN

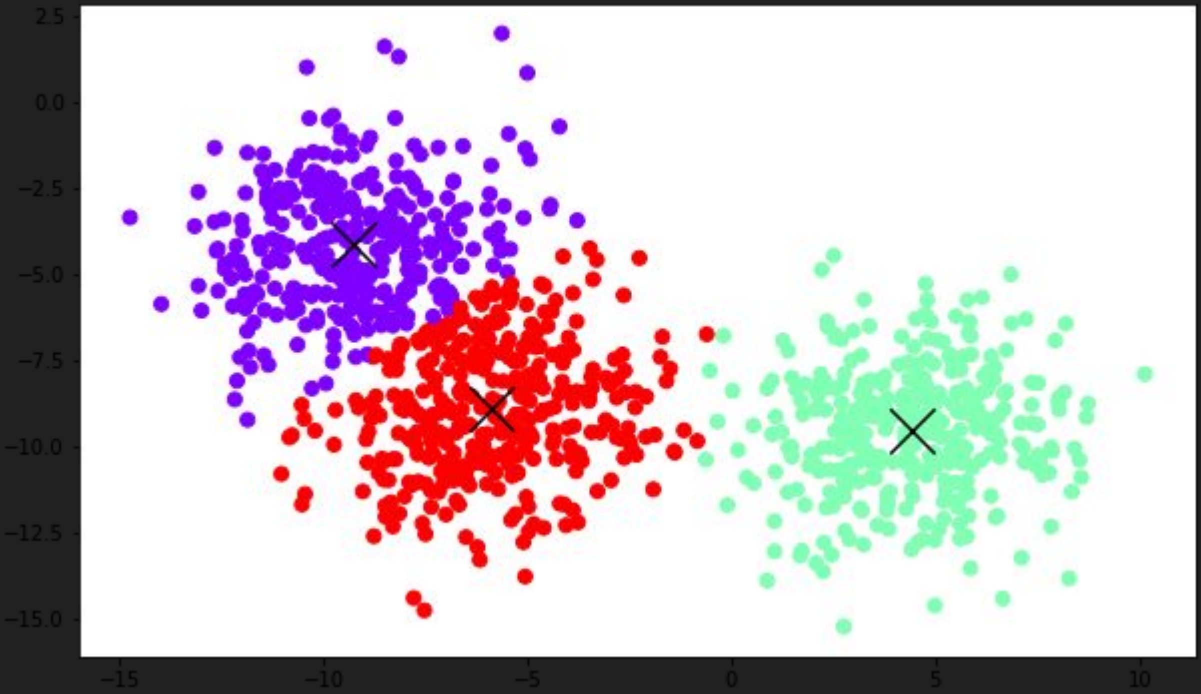
https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/



Best way to evaluate clusters?

Visually examine them!

(If you can)



Evaluating Clusters

Less automatic than supervised learning because we don’t have ground truth

Multiple clusterings may be possible

Task-based evaluation → performance of tasks that has an objective basis for comparison (e.g. gene families)

effectiveness of clustering-based features for a supervised learning tasks--overall accuracy gain from adding a feature

Some evaluation metrics exist, but they can be unreliable

How do we interpret the meanings of label clusters? Need human expertise

Metrics

Inertia -- sum of squared errors for each cluster

Low inertia = dense cluster

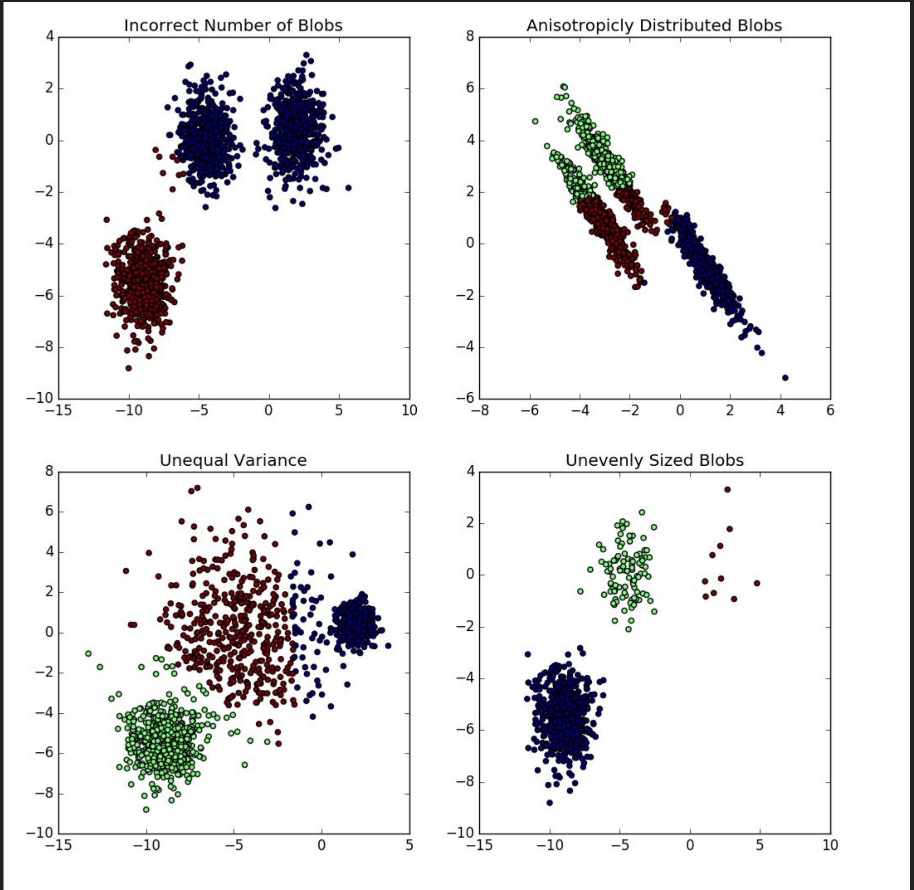
Internal coherence

Silhouette Coefficient -- measure of how far apart clusters are

High Silhouette Score = clusters are well separated

External coherence

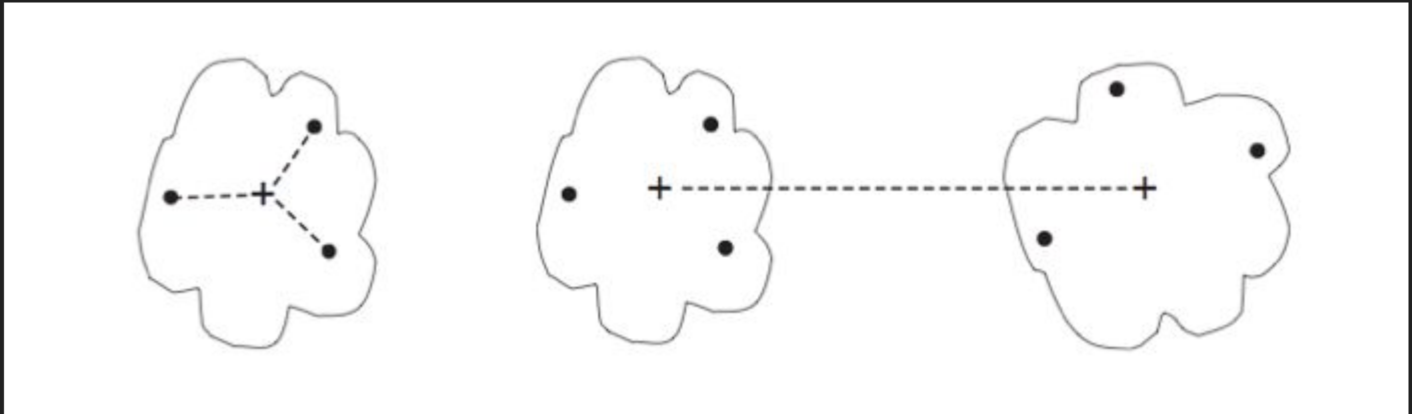
Problems with Inertia



Silhouette Coefficient

Cohesion: measures clustering effectiveness within a cluster.

Separation: measures clustering effectiveness between clusters.


(a) Cohesion. (b) Separation.

Silhouette Coefficient

The silhouette coefficient combines the cohesion and the separation into a single metric. The silhouette coefficient for a single observation is calculated:

$$SC_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

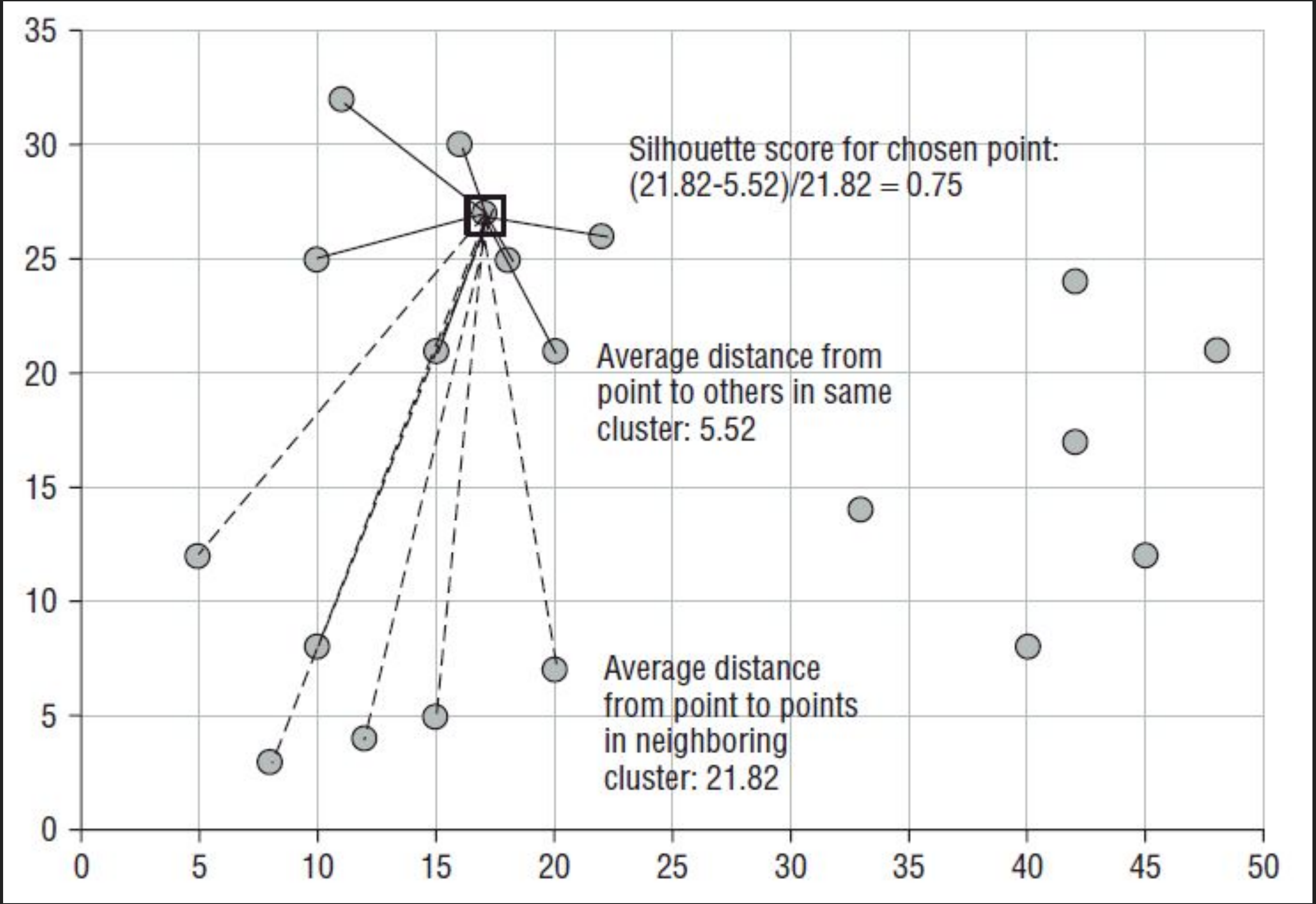
a_i= mean intra-cluster distance of the sample to others within the cluster (cohesion)

b_i = the mean nearest-cluster distance from the sample to those within the nearest non-assigned cluster (separation)

The coefficient ranges from -1 to 1:

A score of 1 indicates the maximum cohesion and separation of clusters.

A score of -1 indicates the minimum cohesion and separation of clusters.


Silhouette score for chosen point:
(21.82-5.52)/21.82 = 0.75
Average distance from point to others in same cluster: 5.52
Average distance from point to points in neighboring cluster: 21.82
0
5
10
15
20
25
30
35
40
45
50

Interpreting the silhouette coefficient

When you calculate the silhouette score using `sklearn` you get out a single number. This is the average silhouette score for all of the individual observations.

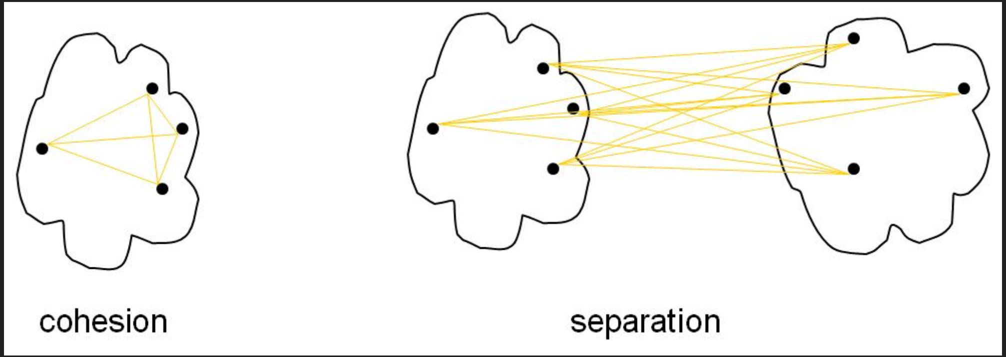
In general, we want separation to be high and cohesion to be low. This corresponds to a value of SC close to +1.

A negative silhouette coefficient means the cluster radius is larger than the space between clusters, and thus clusters overlap. Another way to think about this is that negative values indicate that non-assigned clusters are more similar than the assigned cluster.

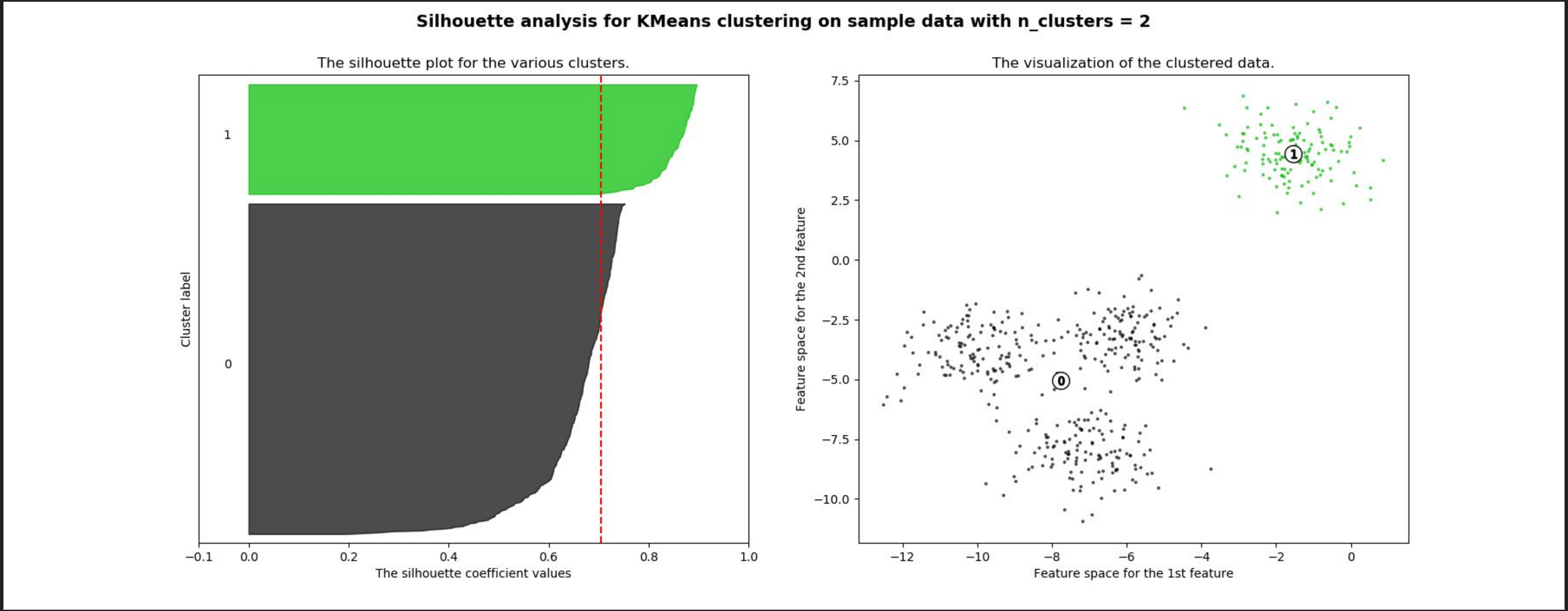
Silhouette plots

Evaluating how the silhouette changes different numbers of K can give you a rough sense of the quality for K clusters in terms in cohesion and separation.

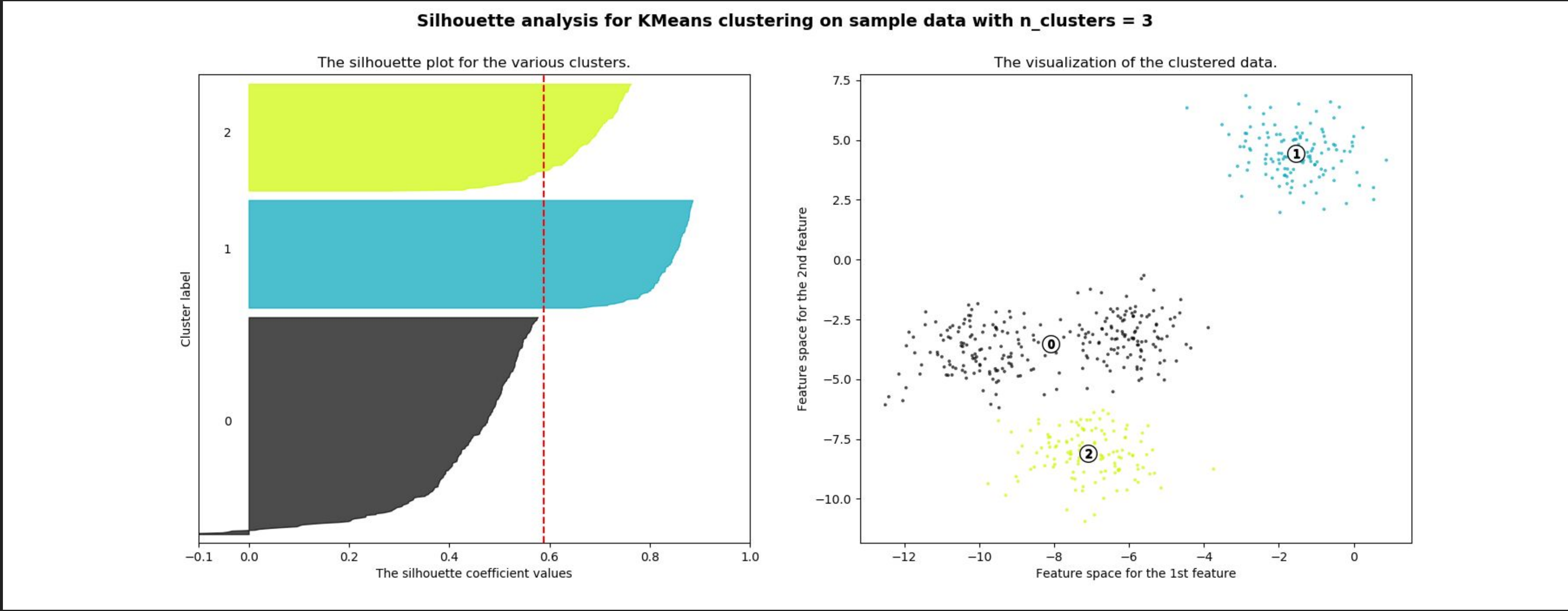
Remember that we have a silhouette of 0 if cohesion = separation. Ideally, we want separation >> cohesion because we want that sample to be distinct and different to samples in other groups but really similar to samples in its group.



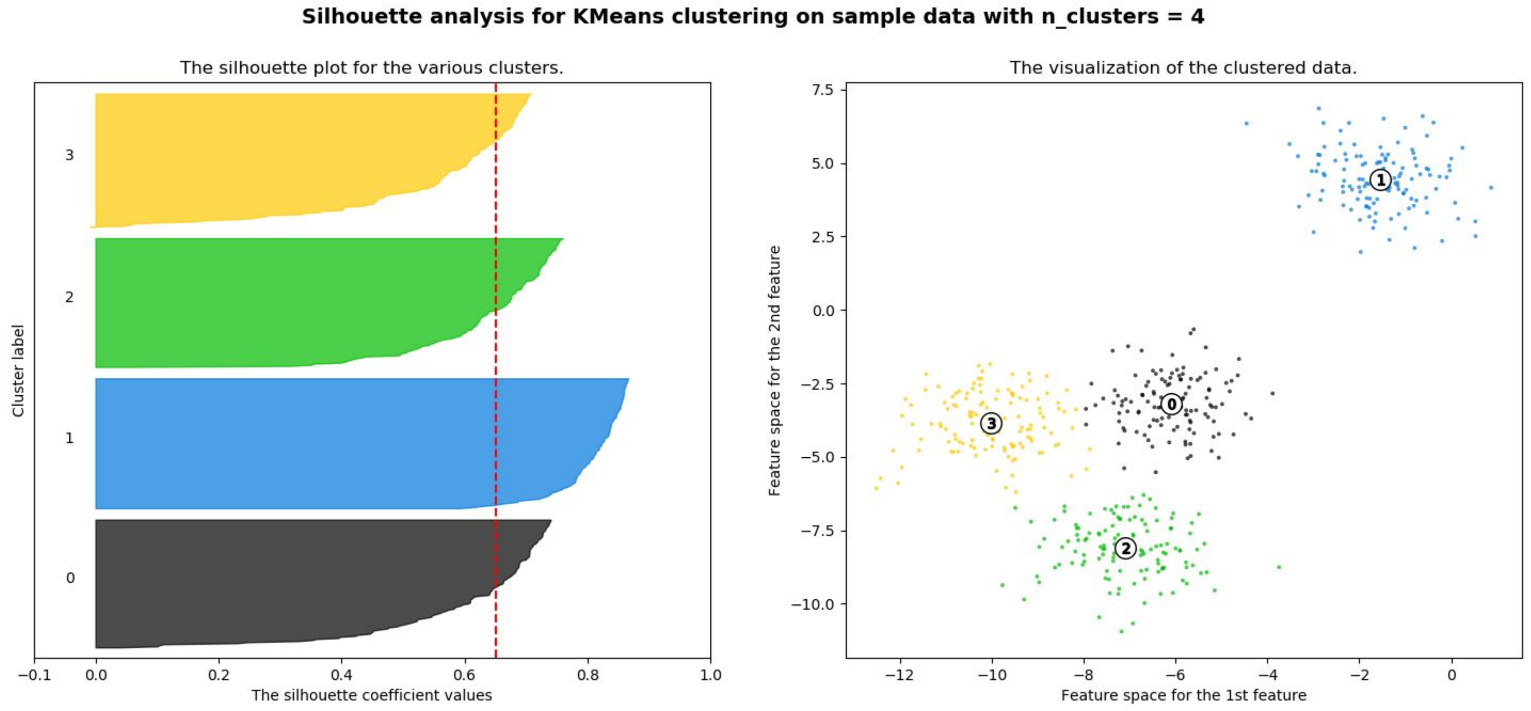
Interpreting Silhouette Plots (K = 2)



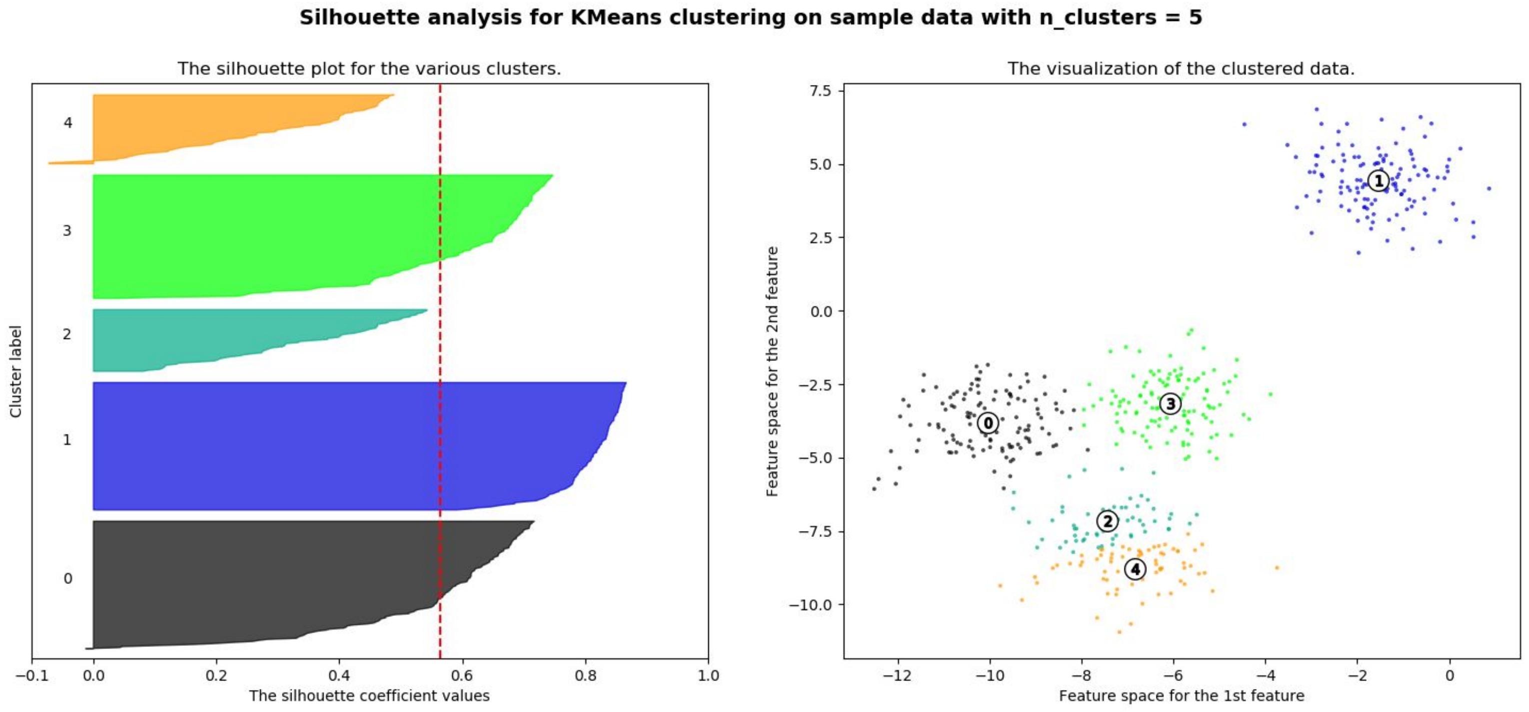
Interpreting Silhouette Plots (K=3)



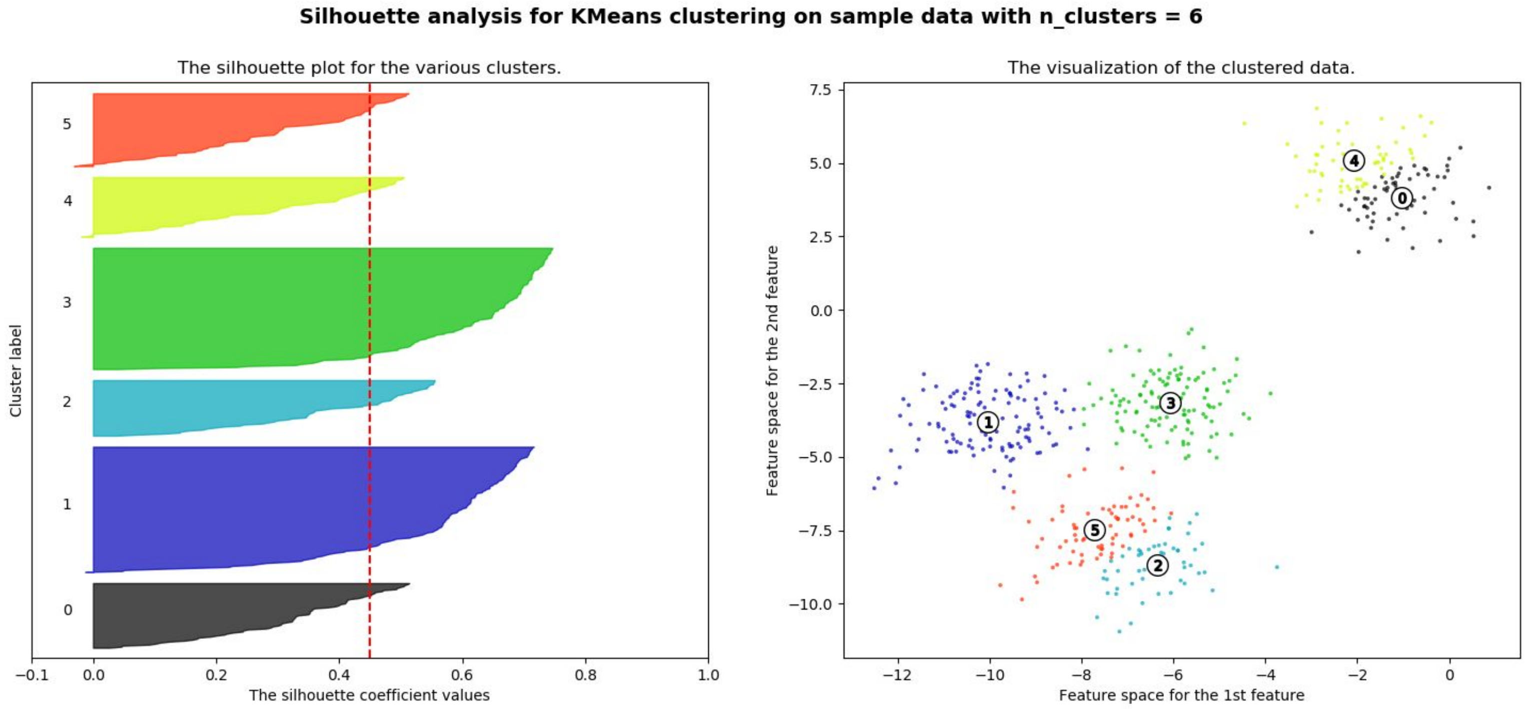
Interpreting Silhouette Plots (K=4)



Interpreting Silhouette Plots (K=5)



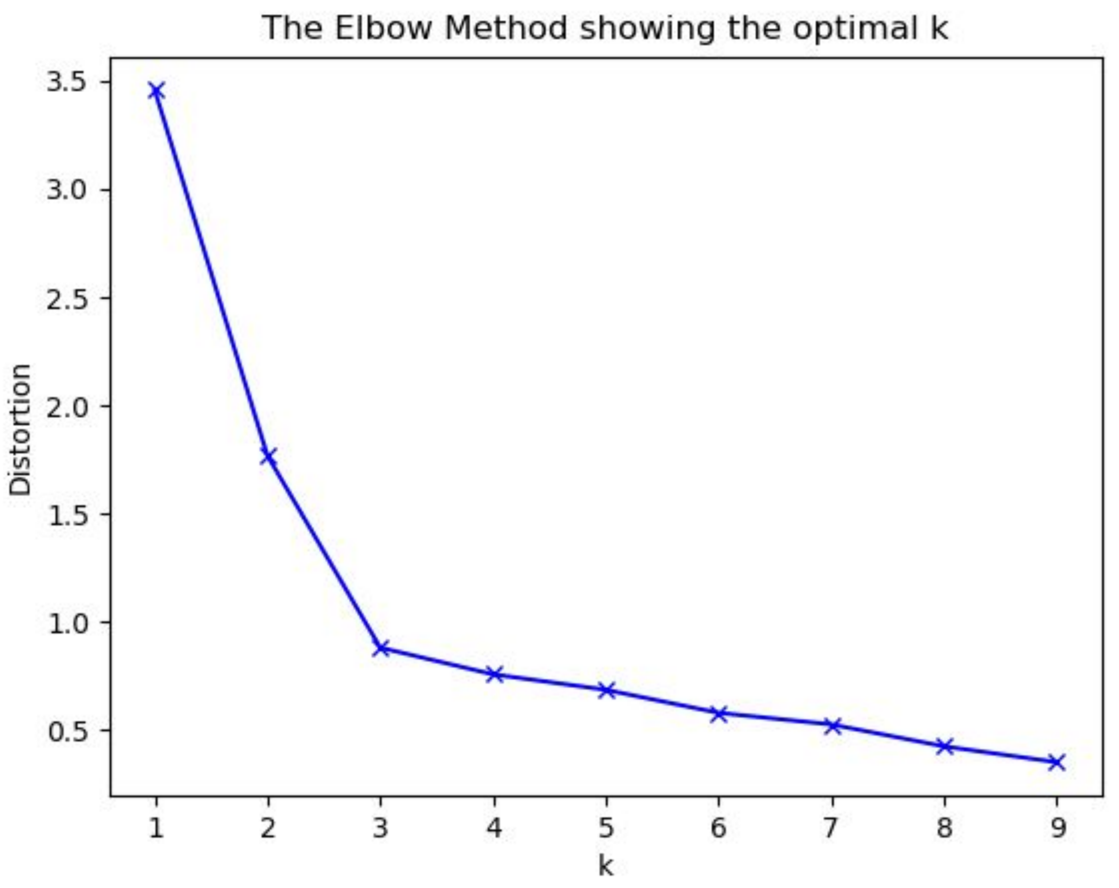
Interpreting Silhouette Plots (K=6)



Elbow Method (choosing K)

Plot inertia (aka distortion) vs. the K number of clusters to get an idea of what the optimal number of clusters would be for the dataset.

Elbow method: look for the K where the inertia has an "elbow": the point where decreases in inertia are considerably more marginal than for previous increases in K.


The Elbow Method showing the optimal k
Distortion
k
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
4
5
6
7
8
9