TU 257 – Fundamentals of Data Science

Data Analytics

L10 – Clustering Data

Brendan Tierney

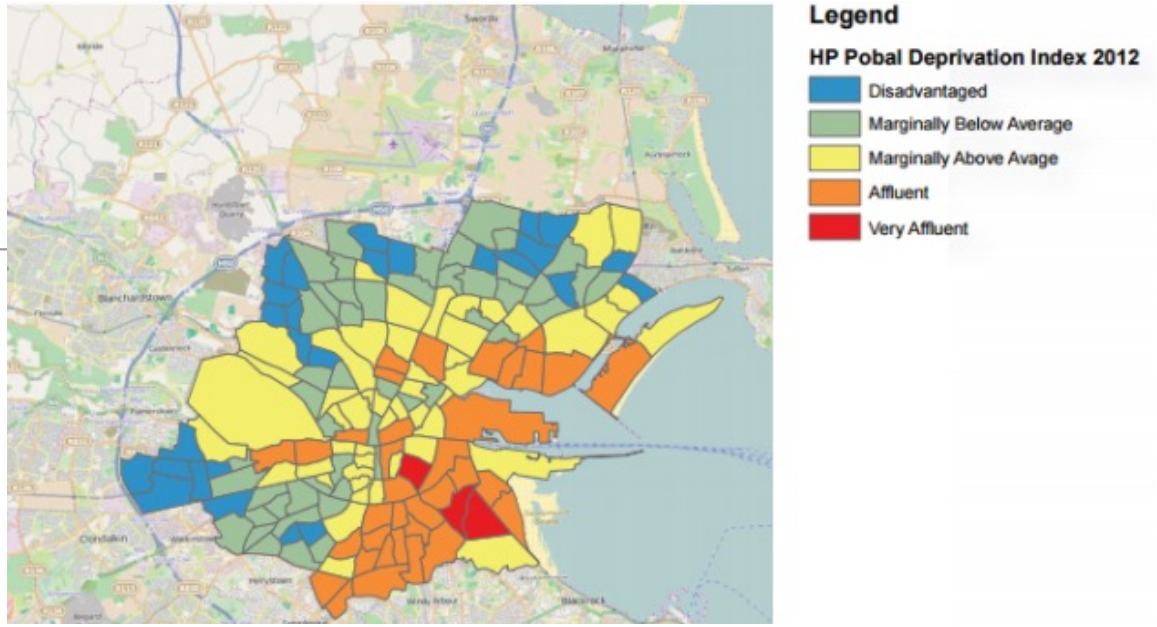
Agenda

- Segmentation
- What is Clustering
- k-Means
- Measuring/Evaluating Clusters
- Demo of k-Means
- Density Based Clustering (DBC)
- Demo of (DBC)

Segmentation

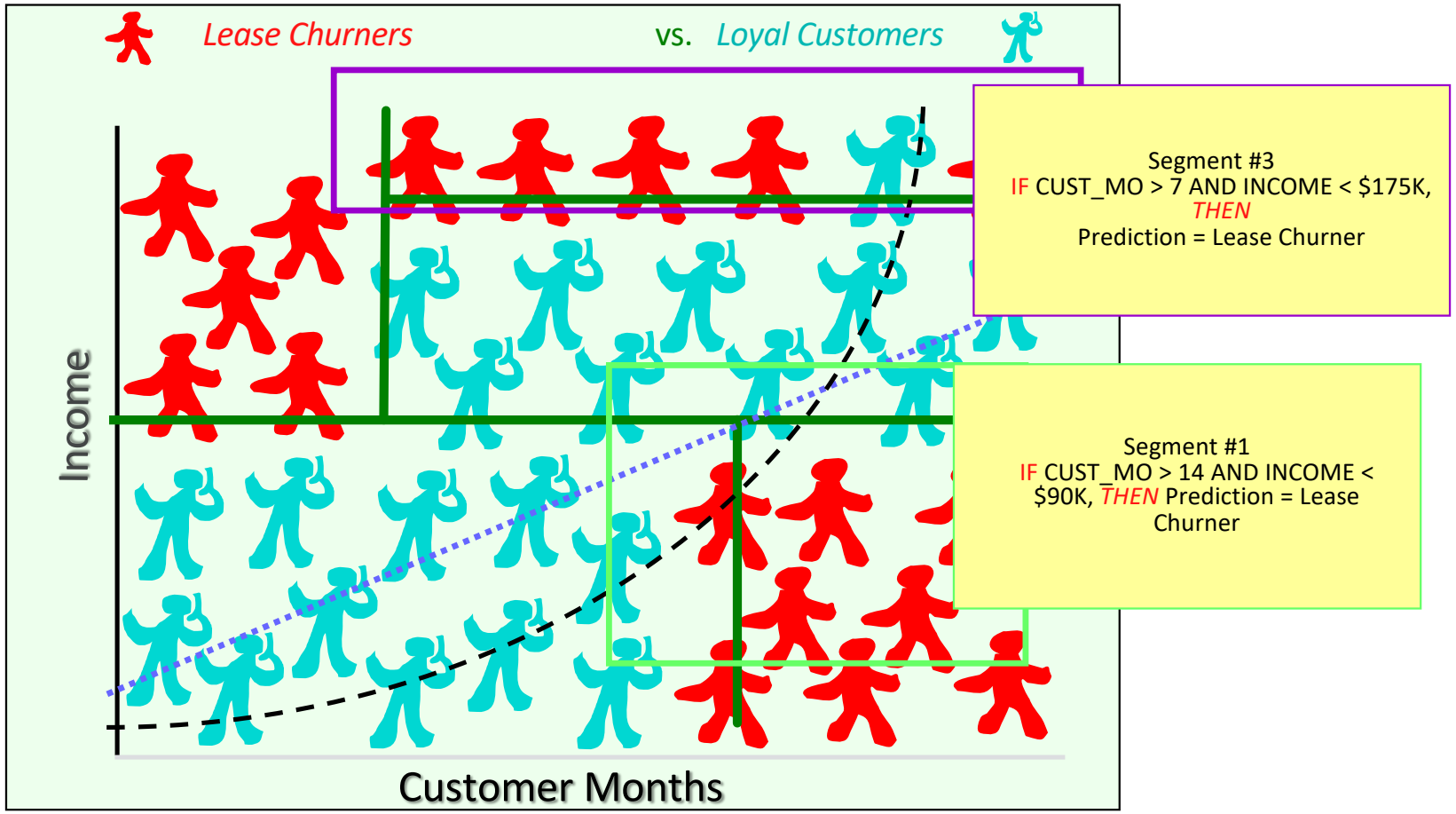
- Very often we explore data to find different groups of records/customers/events/etc
- Can be done using a Question-Answer type of exploring
- Always start with a Question
- Can you answer the Question by writing a query to find the sub-set of data to give the answer
  - Keep repeating and refining the Question-Answer until you have the answers you are looking for
  - You can discover a lot of useful information with each iteration of Question-Answer

Segmentation



- Who are our “Wealthy” customer and where do they live?
  - Who lives on the Northside vs Southside
  - Who lives in “Wealthier” areas very “Poorer” areas
  - How do you define ”Wealthy” vs “Poor”
  - Would you look at crime rate?
    - Does a high crime rate indicate ”Wealthy” or “Poor”

Segmentation



Segmentation

- Assumes prior knowledge
- Uses this to drive the analysis and creating different segments
- Different people could end up with different outputs
  - They have different prior knowledge
  - Different Domain knowledge
  - Difference in ability to analyse and Interpret the results

- Can give “Simple” results
- Based on human skill
- Humans can process X number of variables before it becomes a limitation

- Want to explore “groupings of data” across many different variables -> more complex

Segmentation

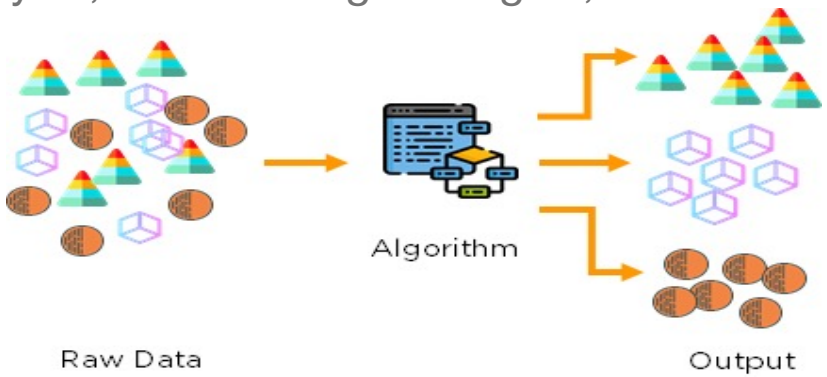
- Always start with a Question
- Segmentation will typically happen as part of Data Exploration
- Many answers will come from Data Exploration stage
- Only progress onto more advanced techniques and algorithms if additional answers are needed
- For example, Cluster Analysis

Unsupervised Machine Learning

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets.

- These algorithms discover hidden patterns or data groupings without the need for human intervention.

- Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, image recognition, etc



Unsupervised Learning

Data Mining/Machine Learning

Supervised ML

UnSupervised ML

Classification

Regression

Clustering

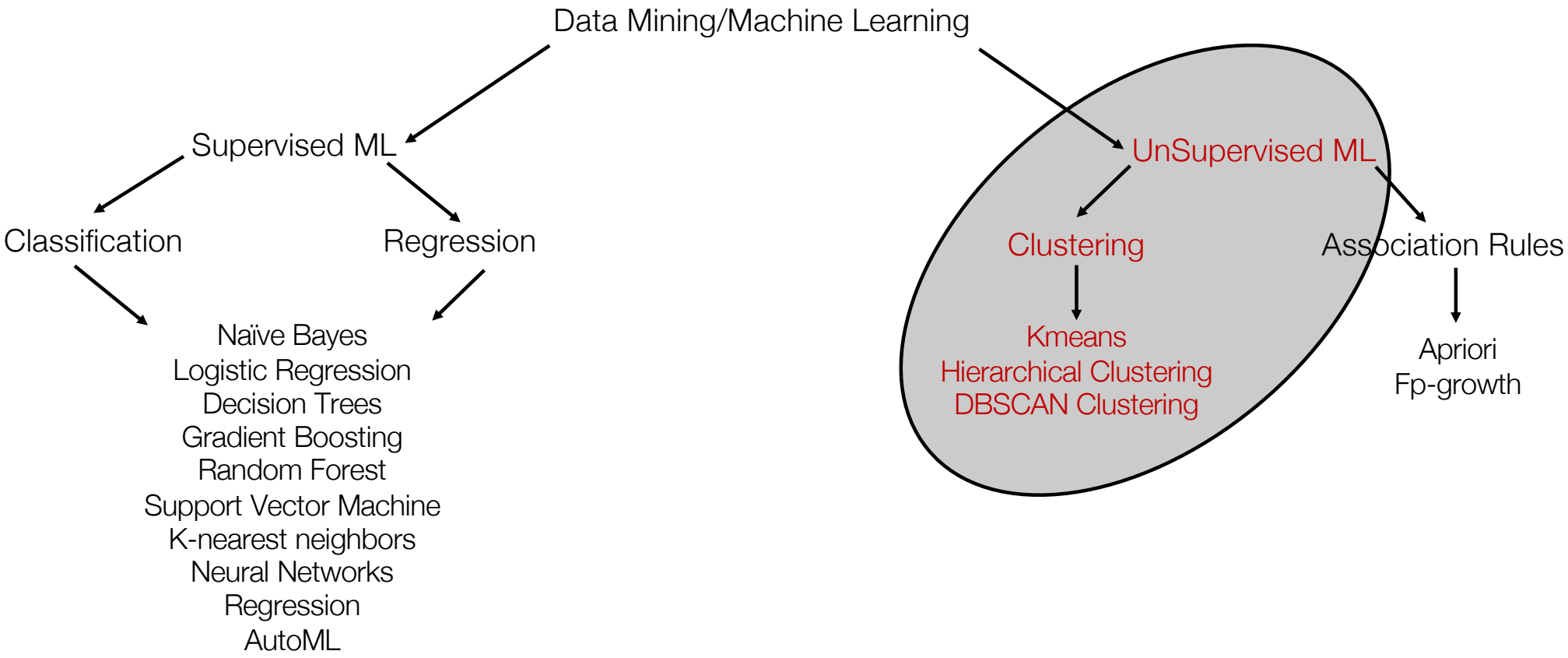
Association Rules

Naïve Bayes
Logistic Regression
Decision Trees
Gradient Boosting
Random Forest
Support Vector Machine
K-nearest neighbors
Neural Networks
Regression
AutoML

Kmeans
Hierarchical Clustering
DBSCAN Clustering

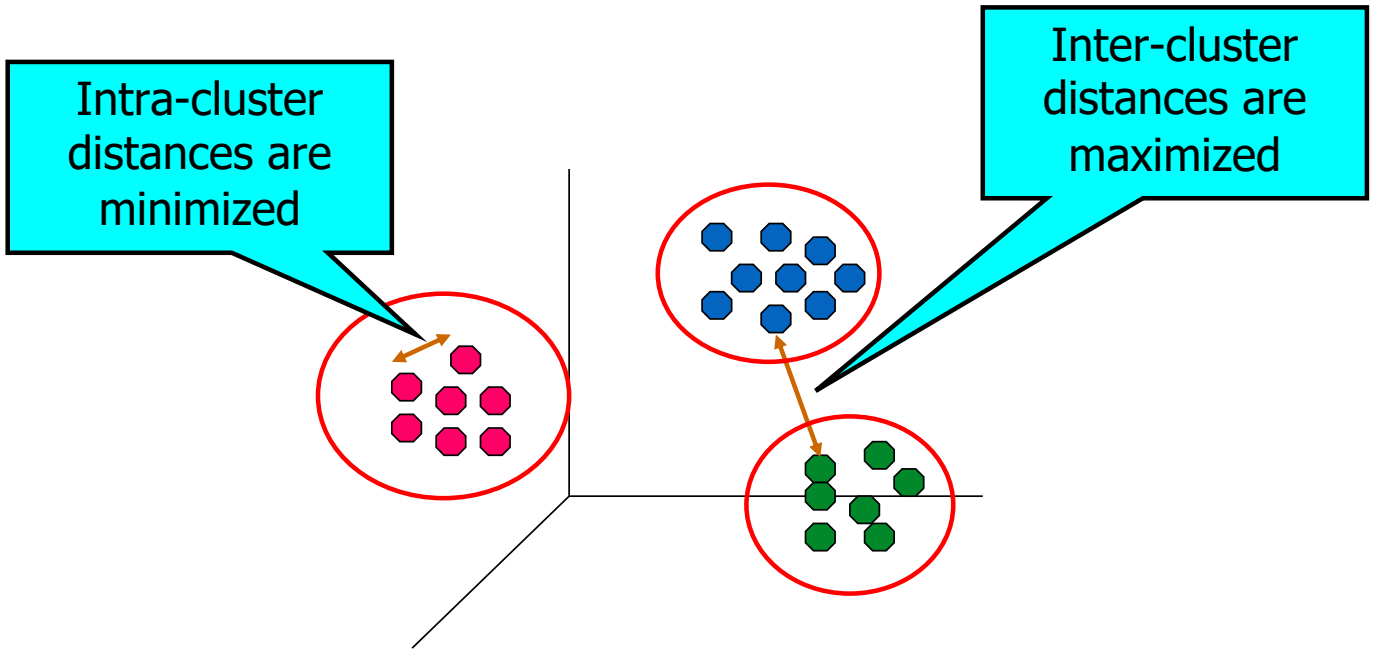
Apriori
Fp-growth

Unsupervised Learning



What is Cluster Analysis?

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



What is Clustering

- Clustering is a process of partitioning a set of data (or objects) into a set of "meaningful" (?) sub-classes, called clusters.
  - Help users understand the natural grouping or structure in a data set.
- Clustering: unsupervised classification: no predefined classes.
- Used either as a stand-alone tool to get insight into data distribution or as a pre-processing step for other algorithms.
  - data compression, outliers detection, understanding of human concept formation.

Notion of a Cluster can be Ambiguous

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Characteristics of the Input Data Are Important

- Type of proximity or density measure
  - This is a derived measure, but central to clustering
- Sparseness
  - Dictates type of similarity
  - Adds to efficiency
- Attribute type
  - Dictates type of similarity
- Type of Data
  - Dictates type of similarity
  - Other characteristics, e.g., autocorrelation
- Dimensionality
- Noise and Outliers
- Type of Distribution

Clustering Algorithms

- K-means and its variants

- Density-based clustering

K-means Clustering

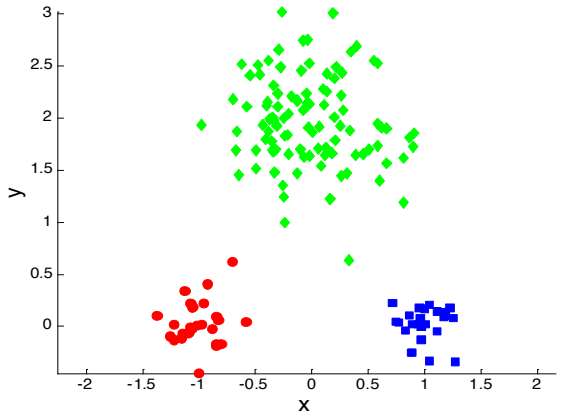
- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

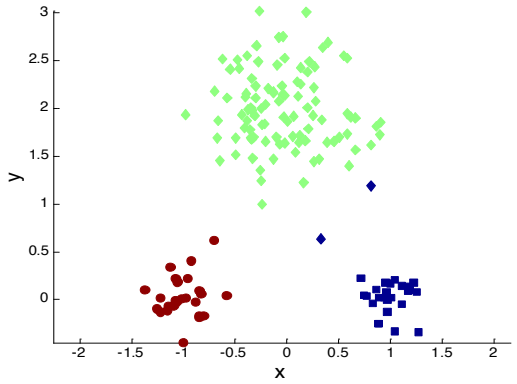
K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

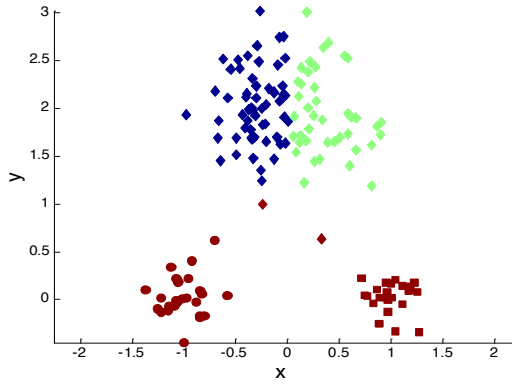
Two different K-means Clusterings



Original Points

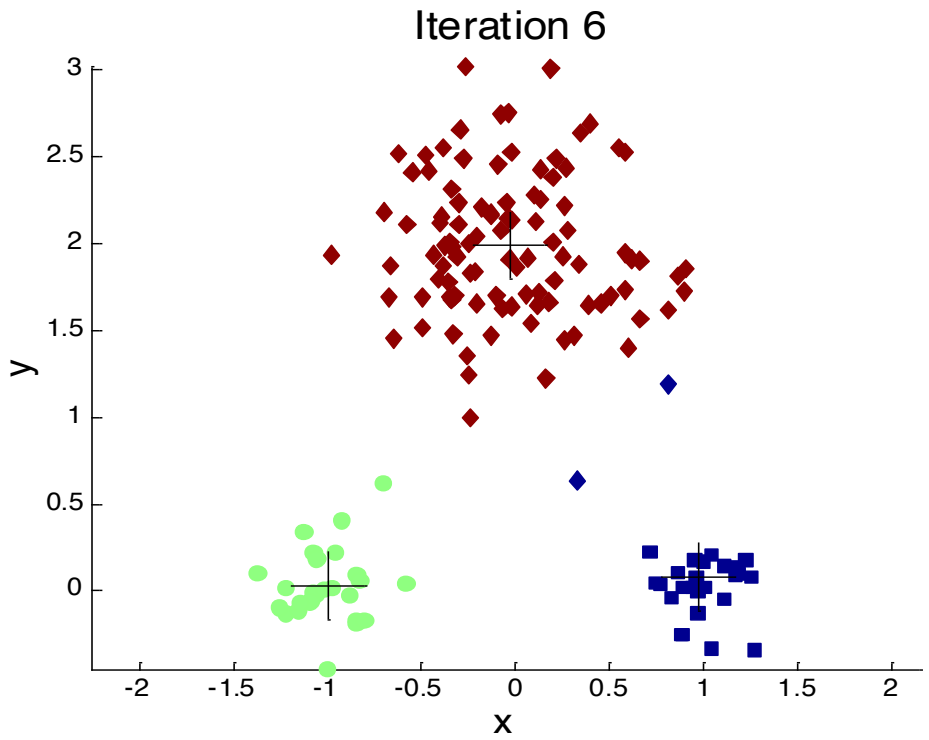


Optimal Clustering

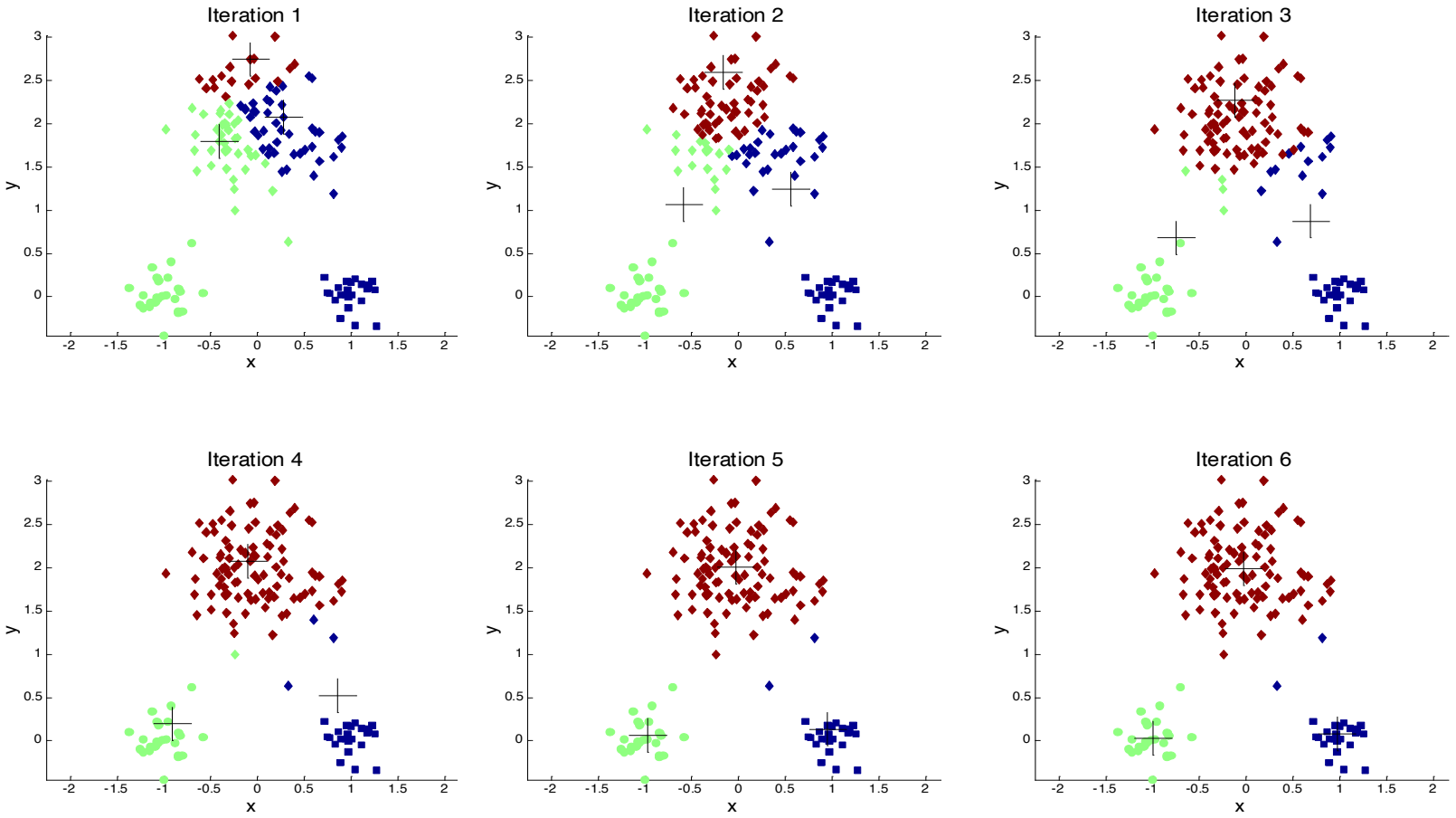


Sub-optimal Clustering

Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids



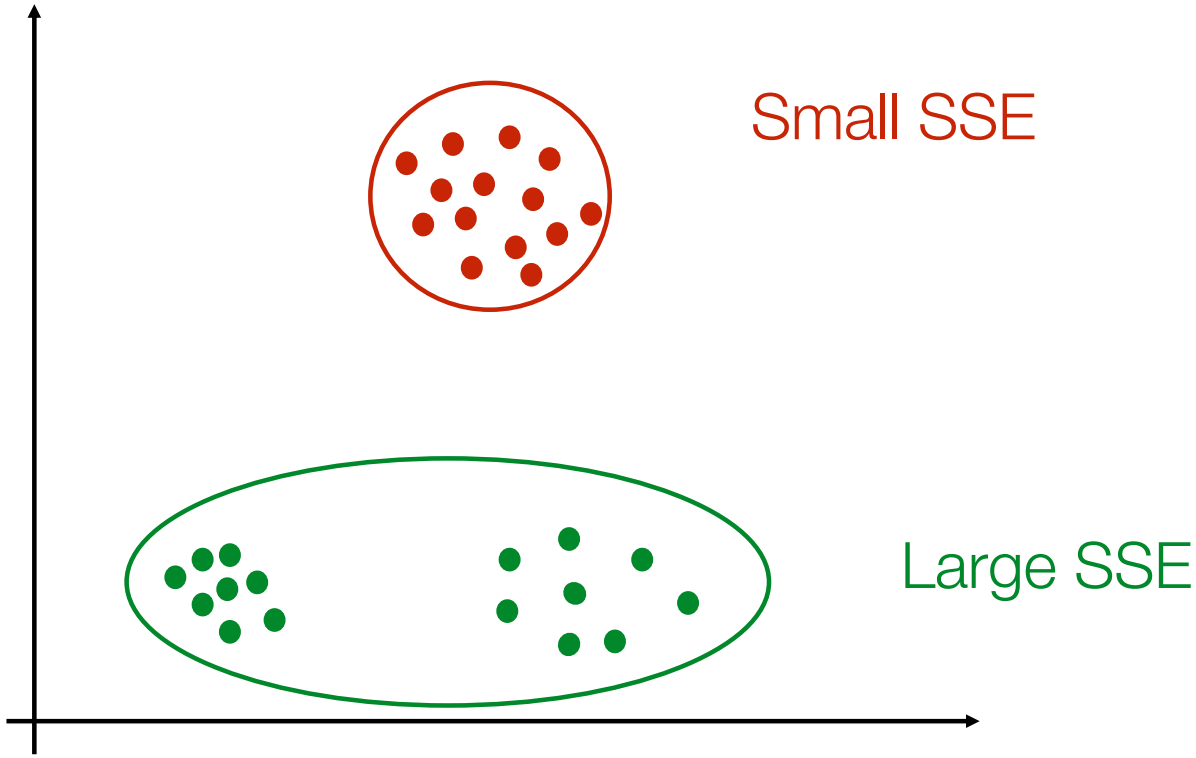
Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

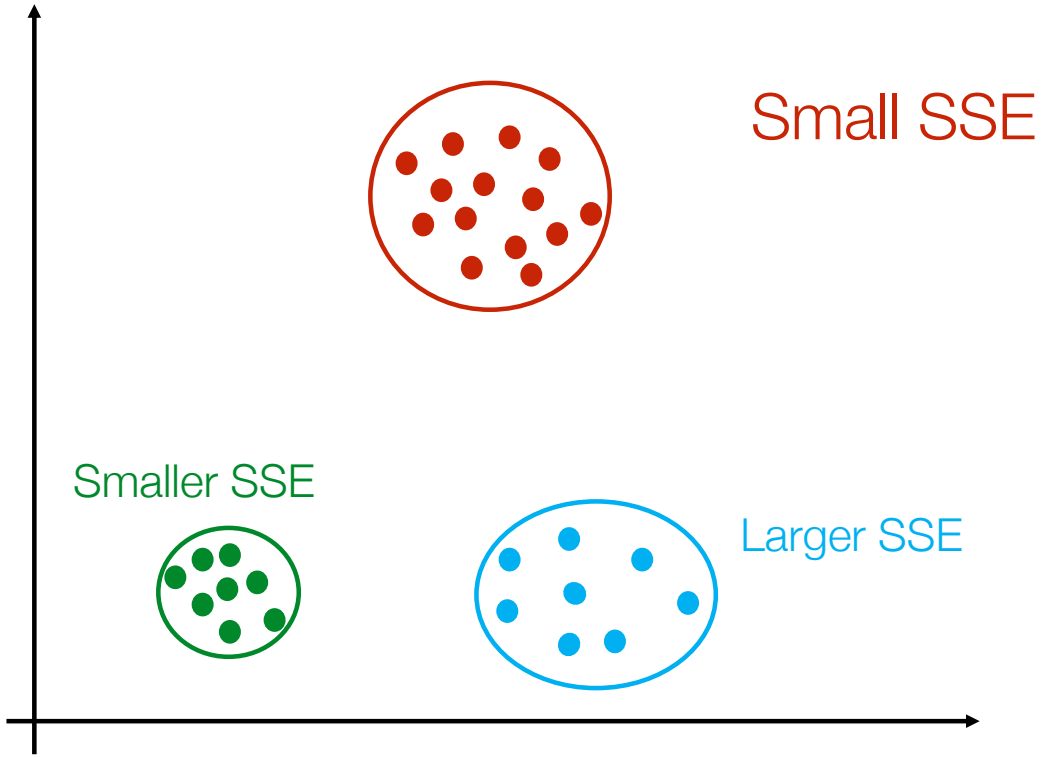
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

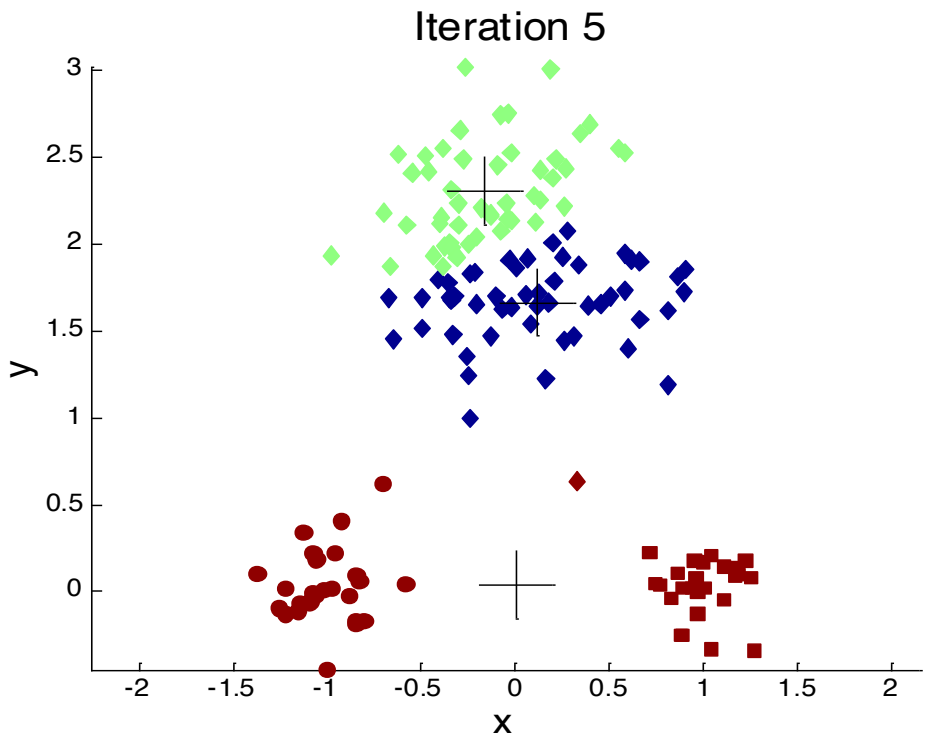
Evaluating K-means Clusters



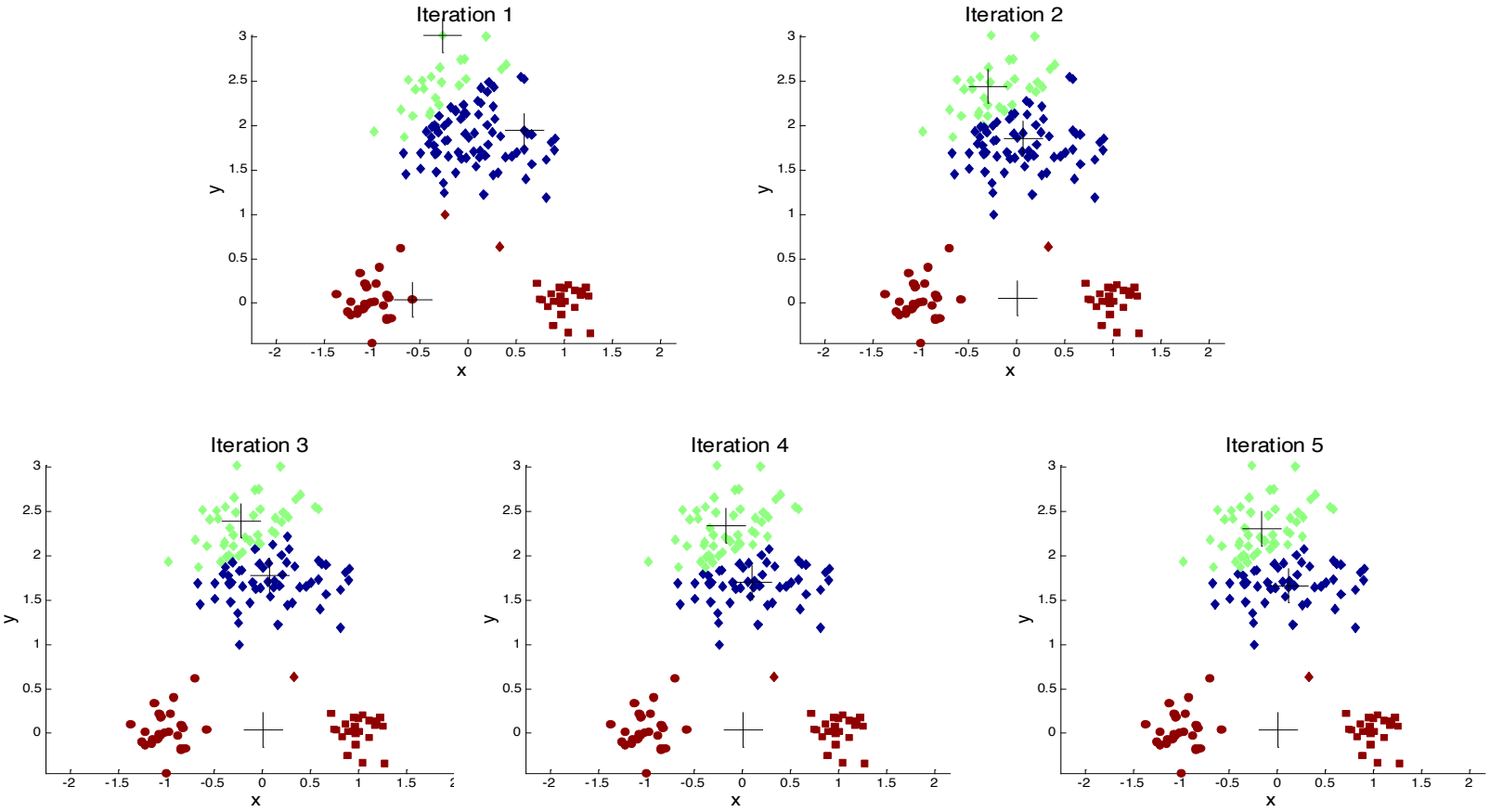
Evaluating K-means Clusters



Importance of Choosing Initial Centroids …



Importance of Choosing Initial Centroids …



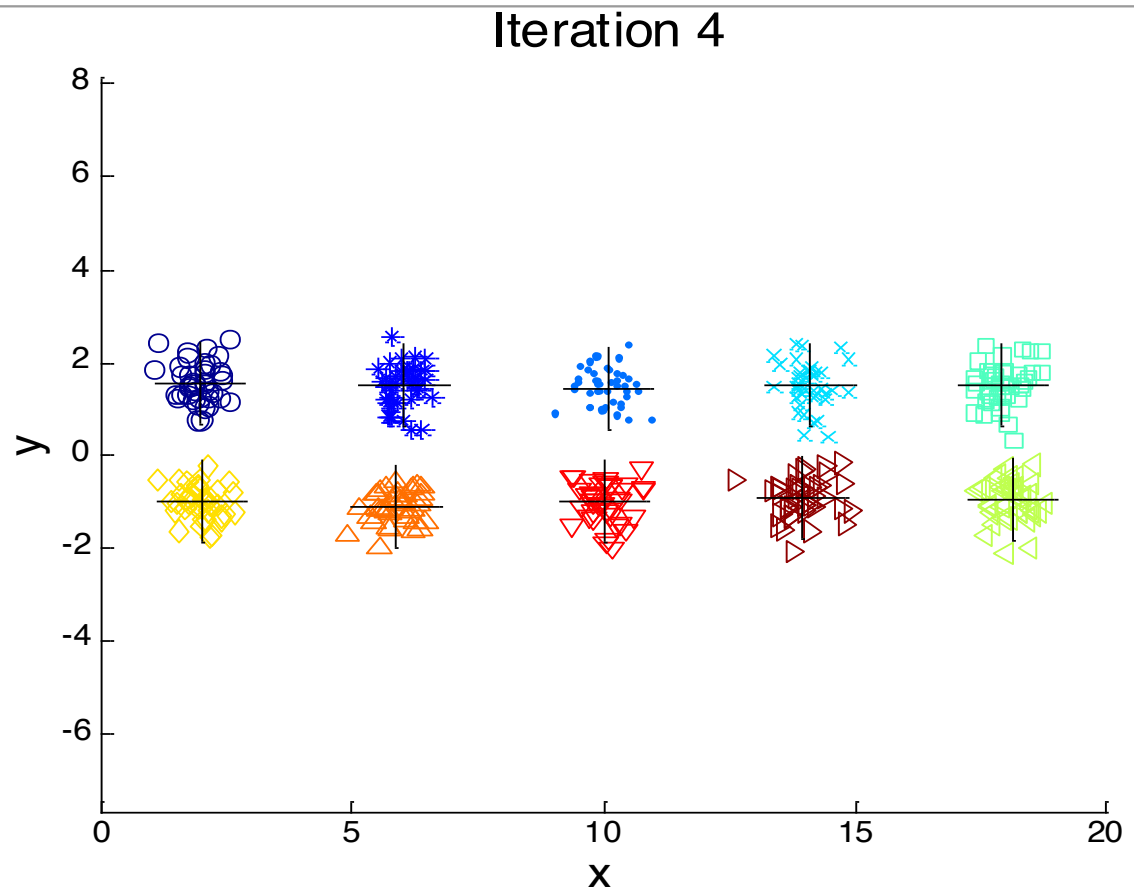
Problems with Selecting Initial Points

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

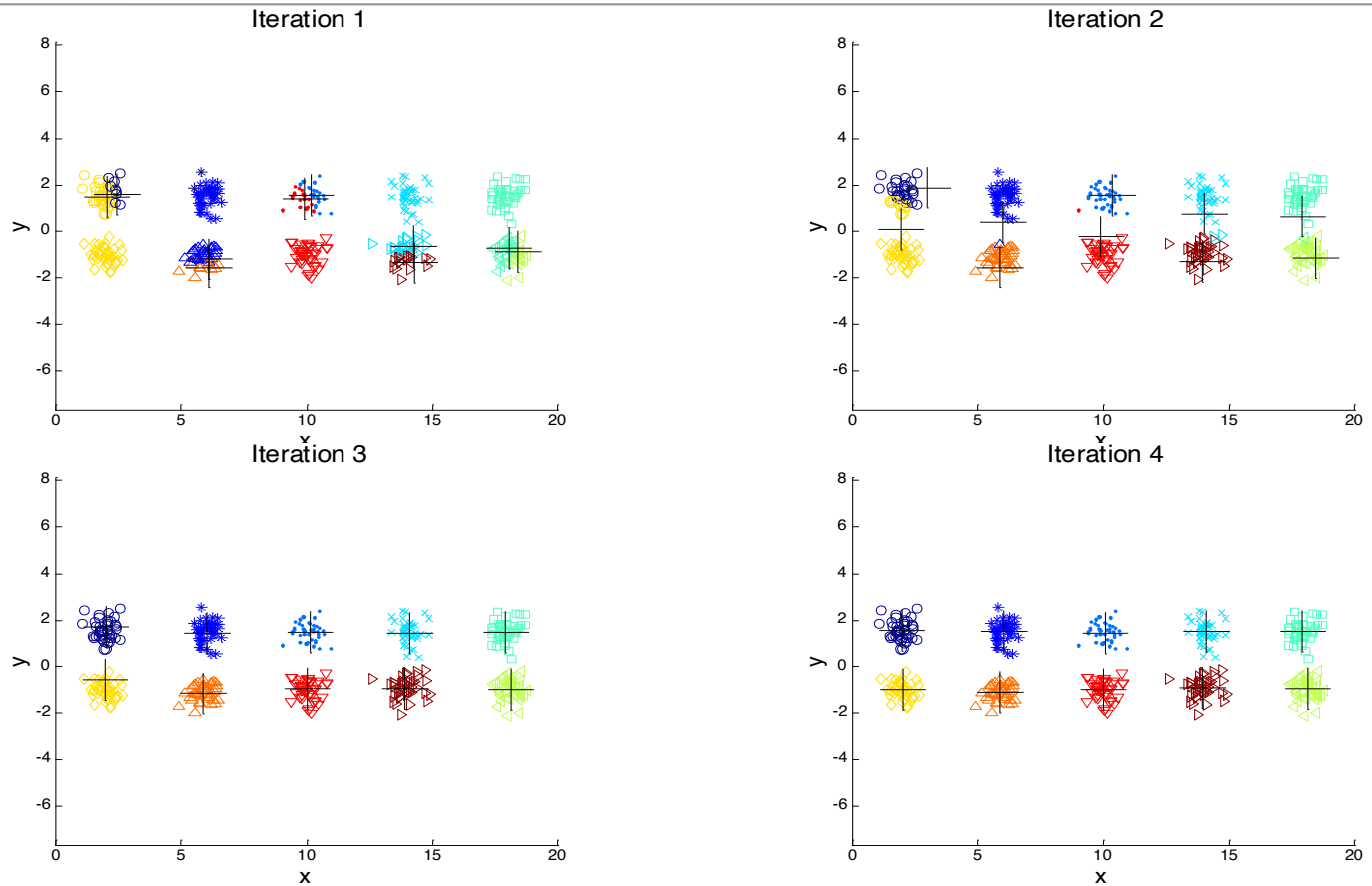
  - For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

10 Clusters Example



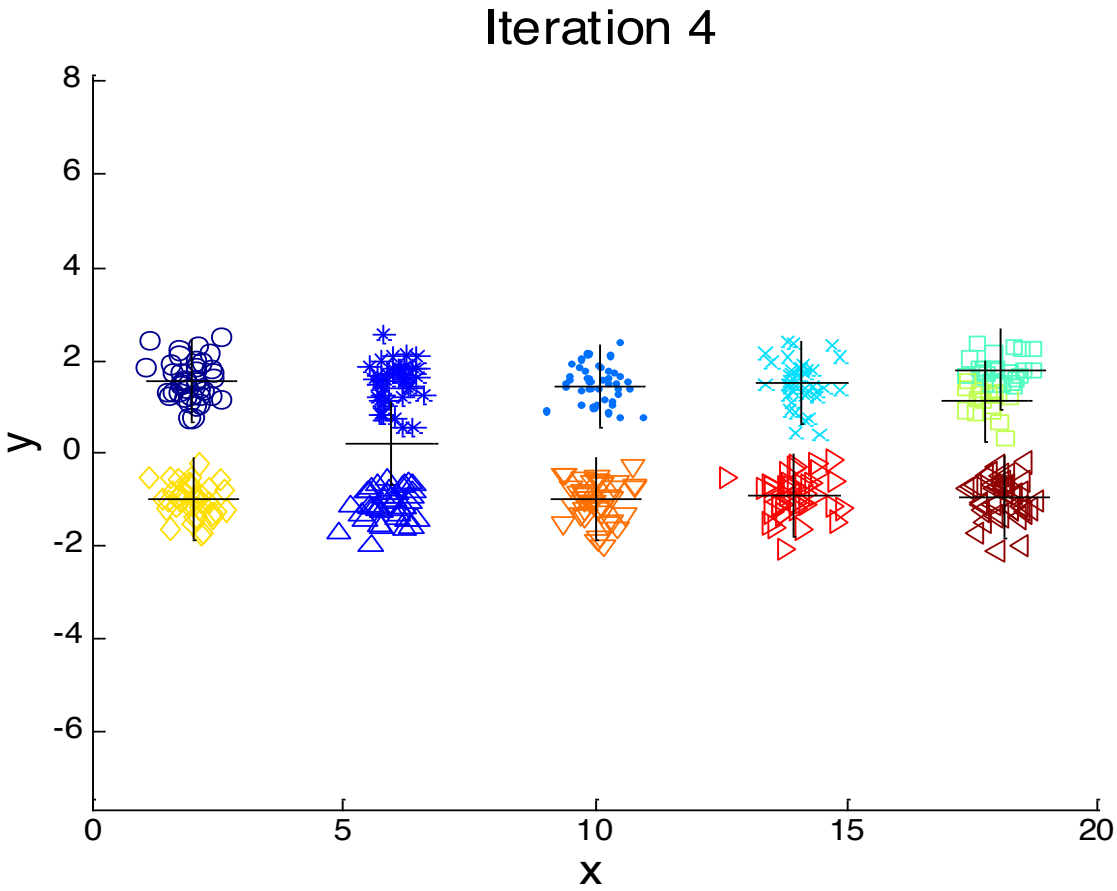
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



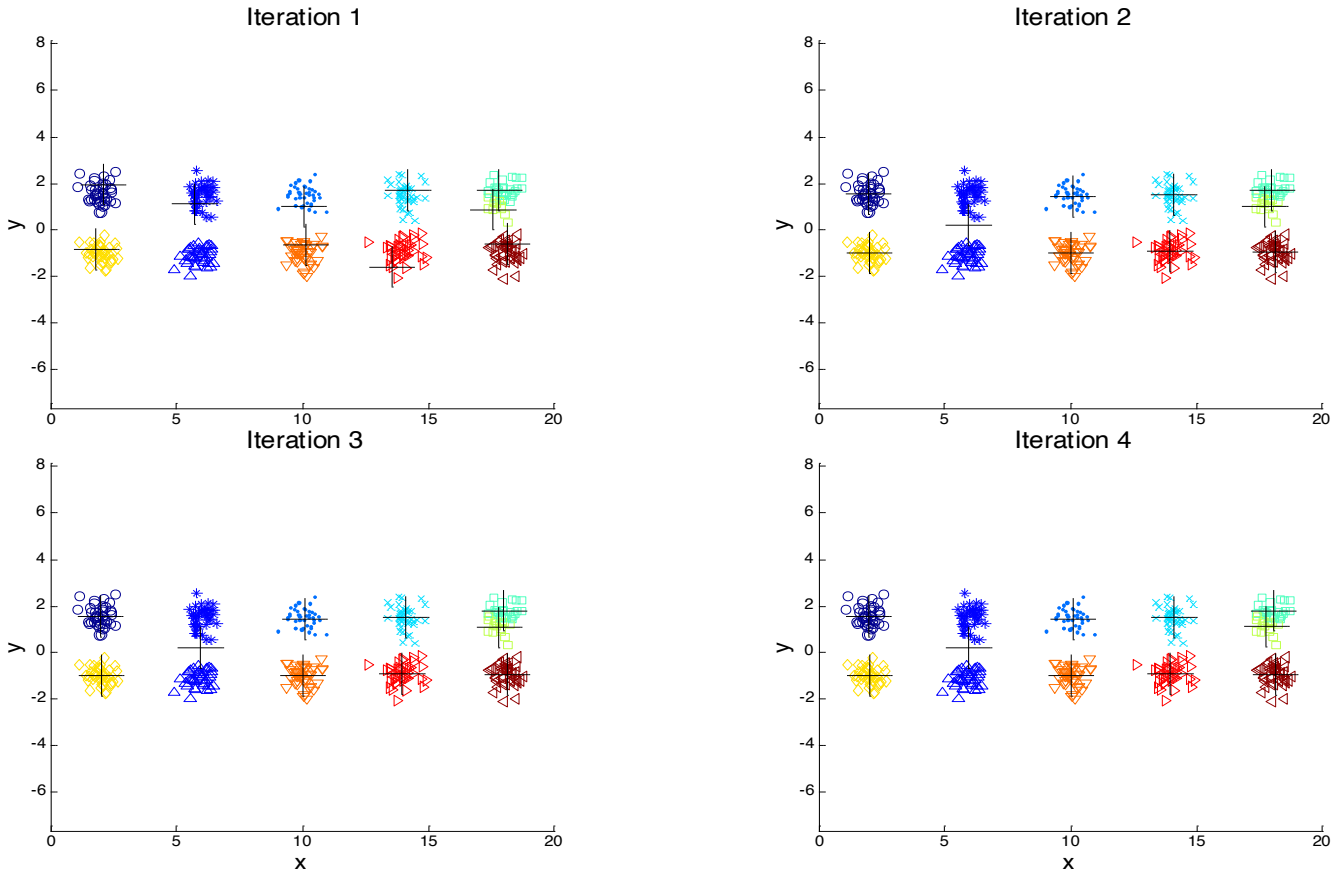
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

Solutions to Initial Centroids Problem

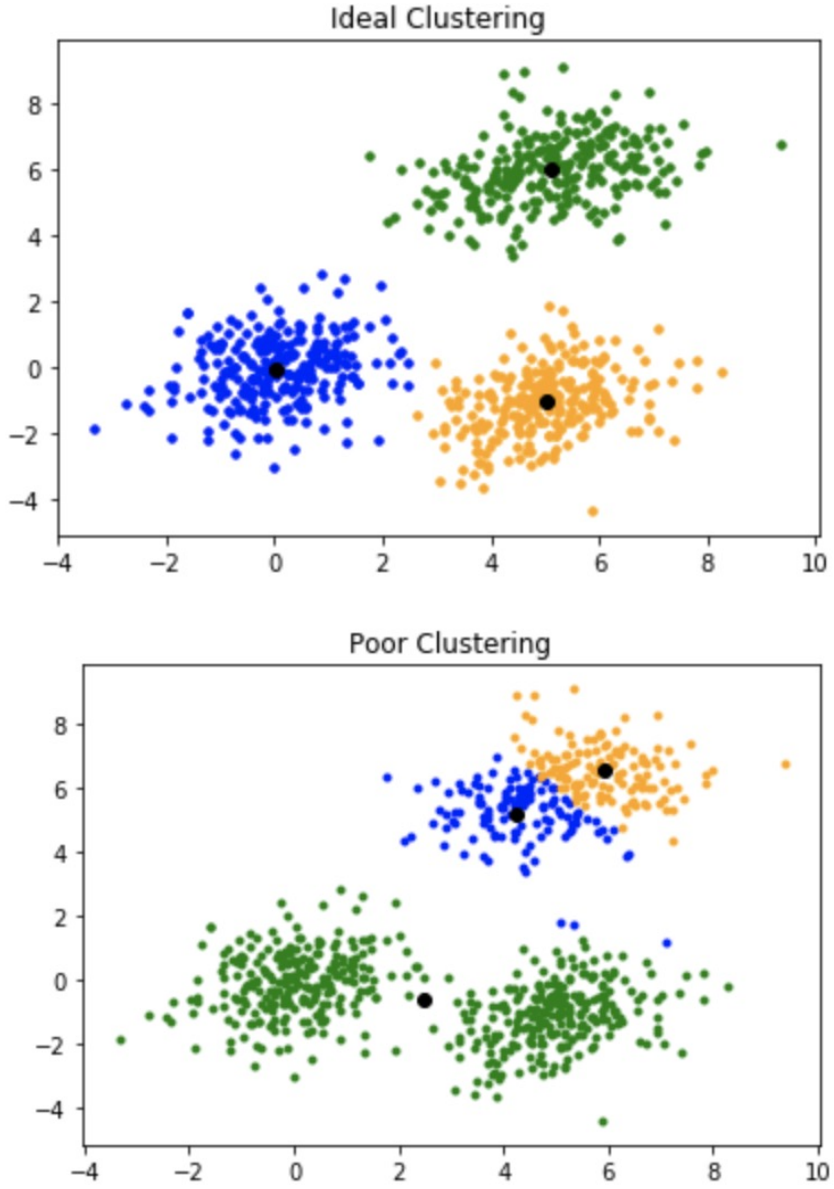
- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing
- Careful sampling

Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters
  - Or near empty clusters
- Several strategies
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE
  - If there are several empty clusters, the above can be repeated several times.
- Review Data set & Sampling
- Use clusters to eliminate or identify outliers

K-means vs K-means++

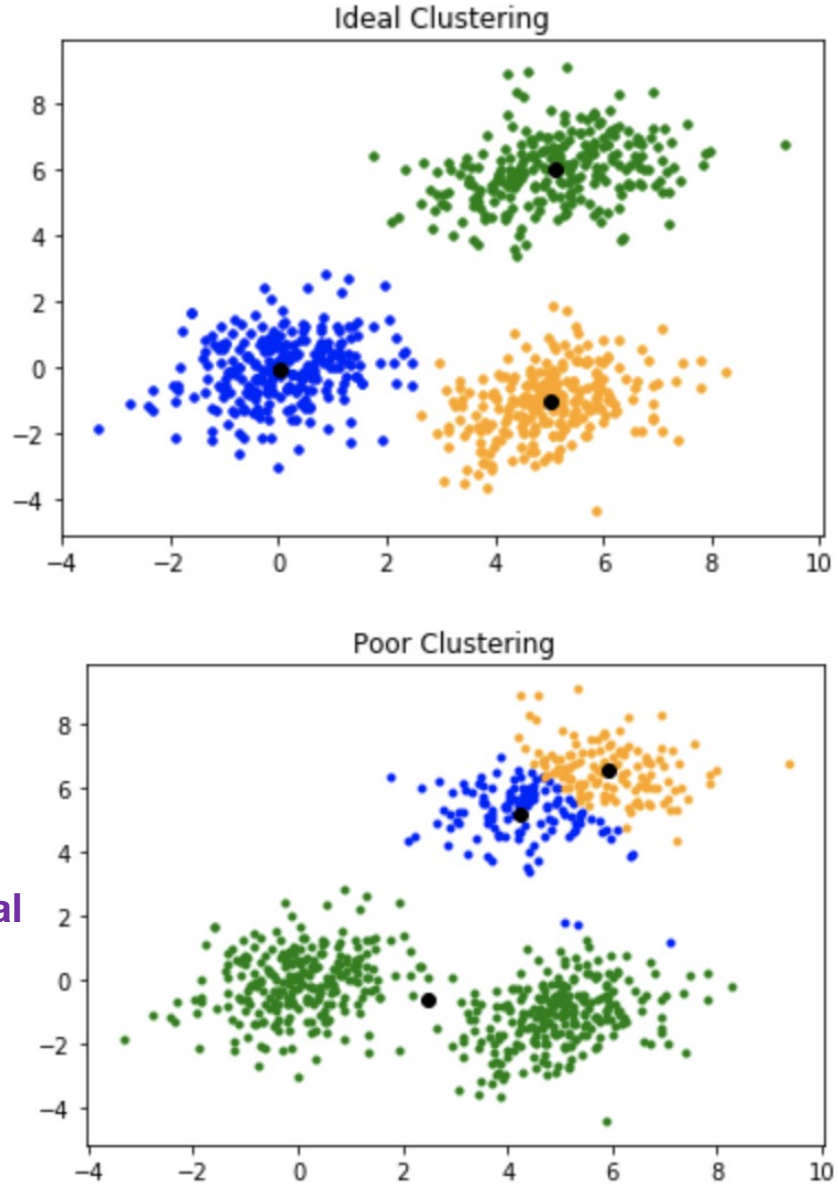
- Limitations of Kmeans
  - It is dependent on the initialization of the centroids or the mean points.
  - You can get a ”good” set of centriods, or a sub-optimal set of centroids
  - A centroid can be a "far away" point, it may very well wind up without any data point related with it
  - Similarly, multiple centroids can be created close to each other.



K-means vs K-means++

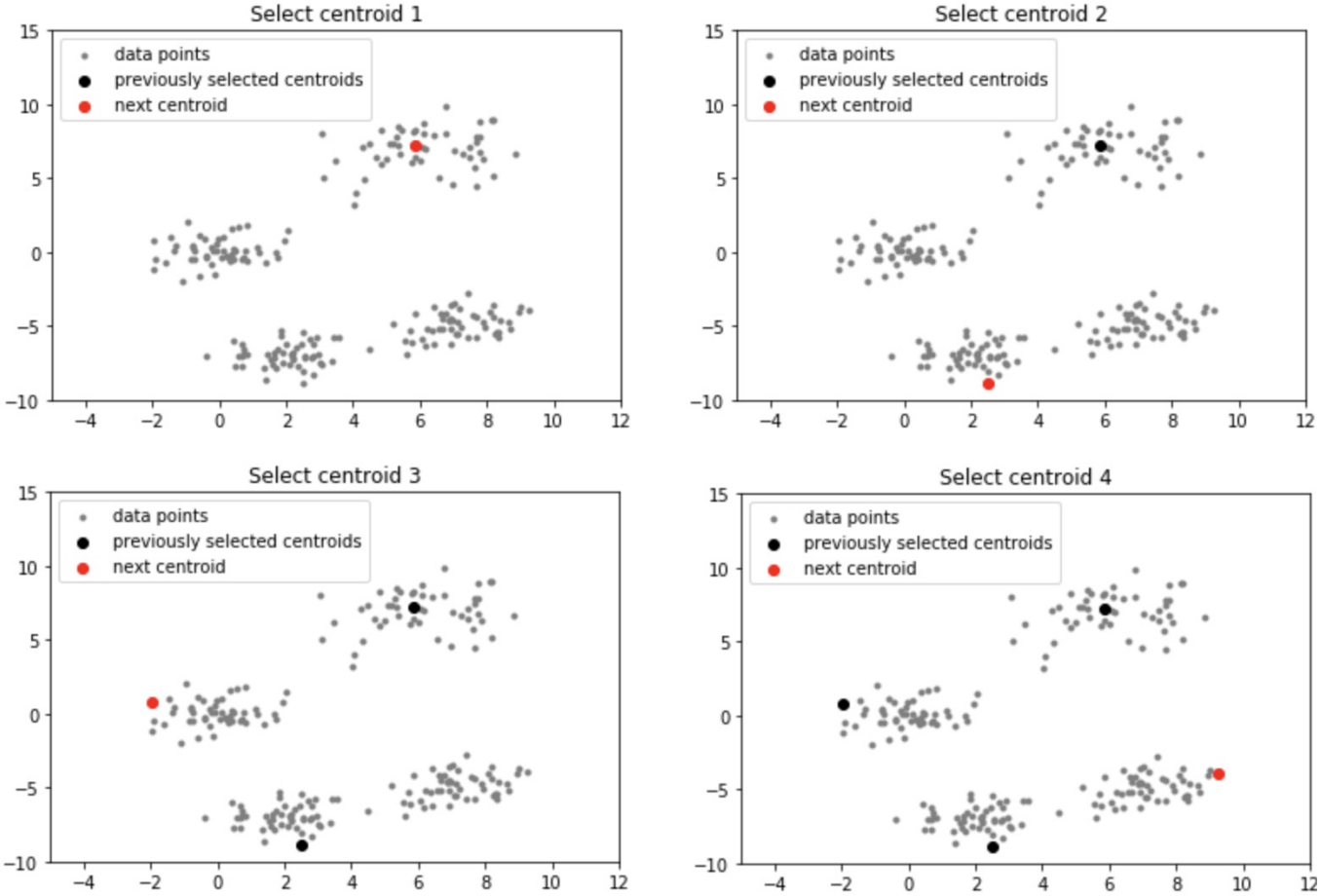
- Kmeans++
  - Operates in a similar manner to Kmeans
  - *Except for how the initial Centriods are selected*
  - Smarter initialization of the centroids and improves the quality of the clustering

1. Randomly select the first centroid from the data points.
2. For each data point compute its distance from the nearest, previously chosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as centroid is **directly proportional to its distance from the nearest, previously chosen centroid**. (i.e. the point having maximum distance from the nearest centroid is most likely to be selected next as a centroid)
4. Repeat steps 2 and 3 until k centroids have been sampled



K-means vs K-means++

- Although the initialization in K-means++ is computationally more expensive than the standard K-means algorithm
- The run-time for convergence to optimum is drastically reduced for K-means++.
- This is because the centroids that are initially chosen are likely to lie in different clusters already.



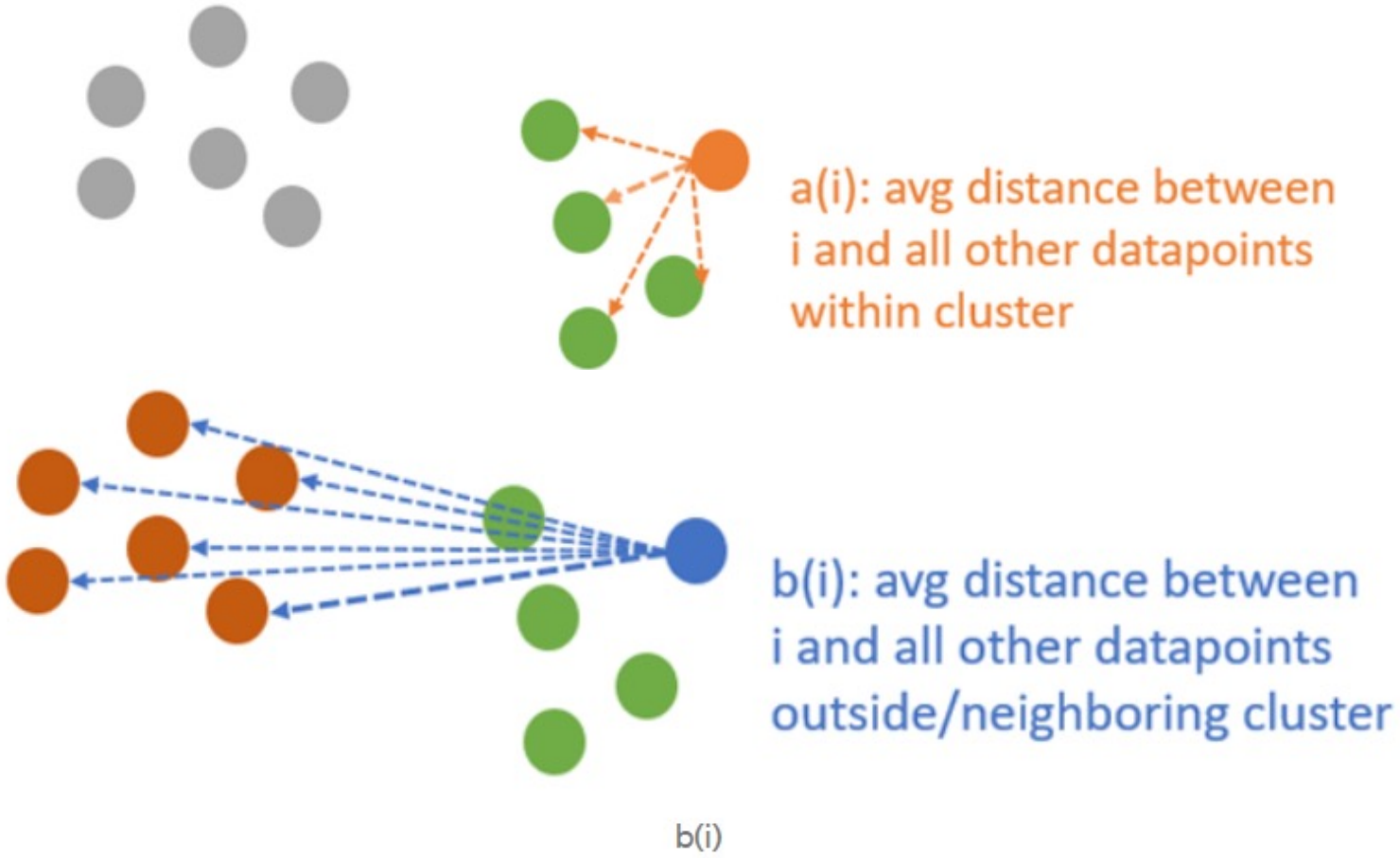
Silhouette – Analysing clusters

- The silhouette coefficient is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation).

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

- S(i) is the silhouette coefficient of the data point i.
- a(i) is the average distance between i and all the other data points in the cluster to which i belongs (same cluster)
- b(i) is the average distance from i to all clusters to which i does not belong (all other clusters)

Silhouette – Analysing clusters



Silhouette – Analysing clusters

- The silhouette coefficient is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation).

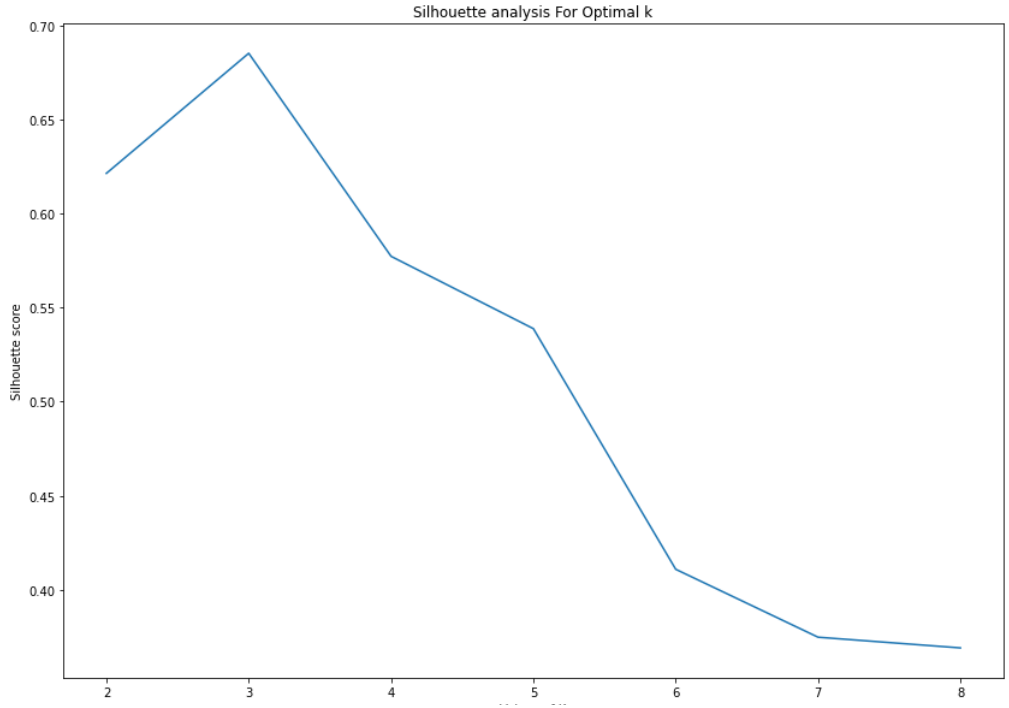
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

- Output value => `AverageSilhouette = mean{S(i)}`
  - The value of the silhouette coefficient is between [-1, 1].
  - A score of 1 denotes the best meaning that the data point i is very compact within the cluster to which it belongs and far away from the other clusters.
  - The worst value is -1, can indicate that the samples might have got assigned to the wrong clusters
  - Values near 0 denote overlapping clusters. A value near 0 represents overlapping clusters with samples very close to the decision boundary of the neighbouring clusters

Silhouette – Analysing clusters – Iris dataset

```
from sklearn.metrics import silhouette_samples, silhouette_score


range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
silhouette_avg = []

for num_clusters in range_n_clusters:
  # initialise kmeans
  kmeans = KMeans(n_clusters=num_clusters)
  kmeans.fit(df2)
  cluster_labels = kmeans.labels_

  print("For n_clusters =", num_clusters, "The average
silhouette_score is :", silhouette_score(df2, cluster_labels))

  # silhouette score
  silhouette_avg.append(silhouette_score(df2, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg)
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
```

```
For n_clusters = 2 The average silhouette_score is : 0.6213468887749368
For n_clusters = 3 The average silhouette_score is : 0.6851559988597147
For n_clusters = 4 The average silhouette_score is : 0.5772635813809022
For n_clusters = 5 The average silhouette_score is : 0.5388083715021846
For n_clusters = 6 The average silhouette_score is : 0.41093583693136543
For n_clusters = 7 The average silhouette_score is : 0.3748392514136022
For n_clusters = 8 The average silhouette_score is : 0.3691562978922058
```



Pre-processing and Post-processing

- Pre-processing
  - Normalize the data
  - Eliminate outliers

- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split ‘loose’ clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are ‘close’ and that have relatively low SSE
  - Can use these steps during the clustering process

Biggest Challenge

- The Biggest Challenge with Clusters is ?

- What do the Clusters actually mean.

- Can you apply a business meaning to the clusters.

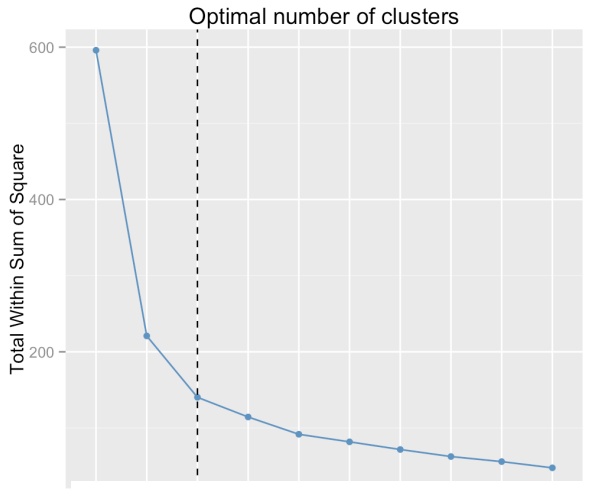
How Many Clusters ?

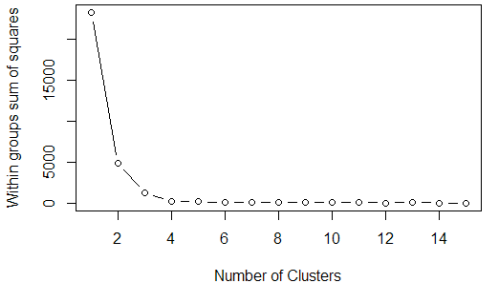
- Difficult to determine.
- It all depends on the data
- Need to try various numbers of K
- See/observe what gives optimal number
- Maybe based on SSE

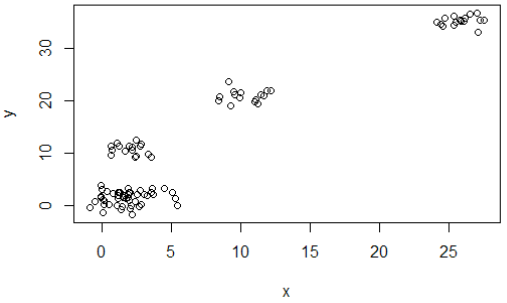
What number of Clusters ?

- Sum-of-the-squares can be used to measure/determine the number of clusters to use.

```
#read the Iris dataset into DF
iris_ds = datasets.load_iris()
df=pd.DataFrame(iris_ds['data'])
#display the data
df.head(10)

#Apply Elbow method and calculate
wcss = []
K = range(1,11)
for i in K:
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

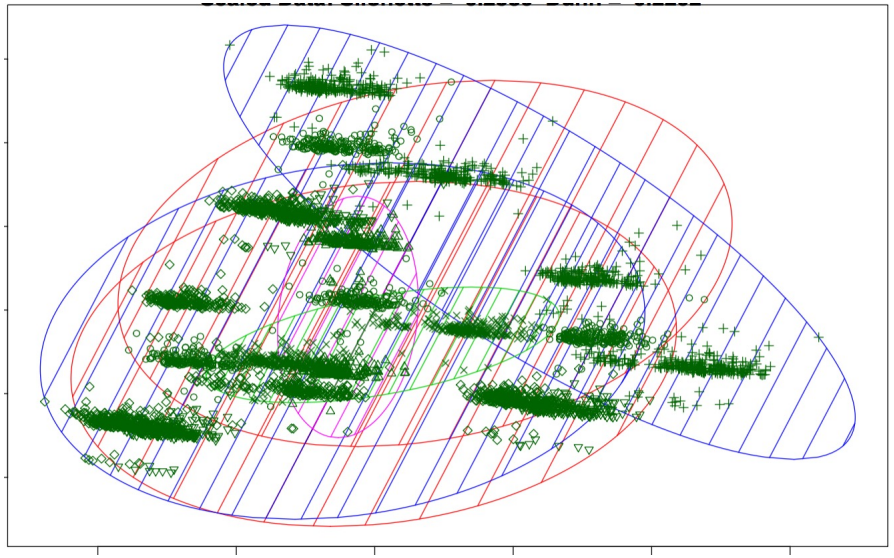






Limitations of K-means

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes



- K-means has problems when the data contains outliers.

Demo

Clustering - 2

Density Based Clustering using DBScan

Cluster Analysis

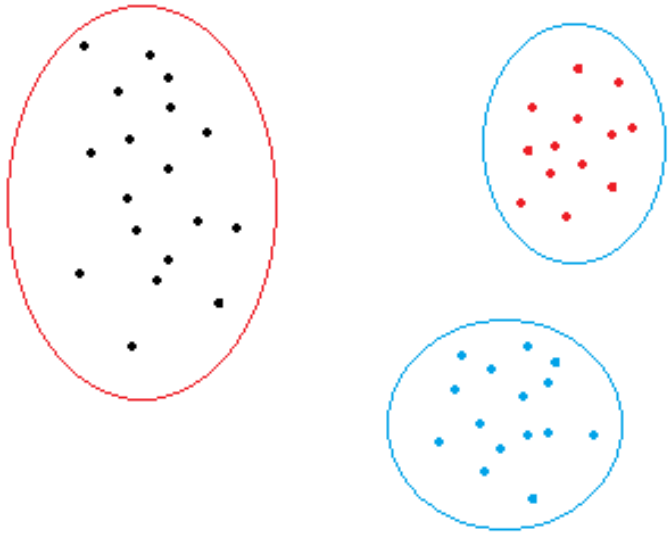
Clustering analysis is an unsupervised learning method that separates the data points into several specific bunches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense.

There are different approaches and algorithms to perform Clustering tasks which can be divided into three sub-categories:

- Partition-based clustering: E.g. k-means, k-median
- Hierarchical clustering: E.g. Agglomerative, Divisive
- Density-based clustering: E.g. DBSCAN

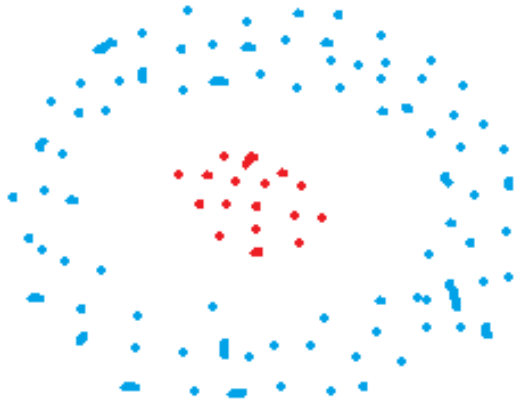
Re-cap

- Partition-based and hierarchical clustering techniques are highly efficient with normal shaped clusters.
- For example, the dataset in the figure can easily be divided into three clusters using k-means algorithm.
- However, when it comes to arbitrary shaped clusters or detecting outliers, density-based techniques are more efficient.

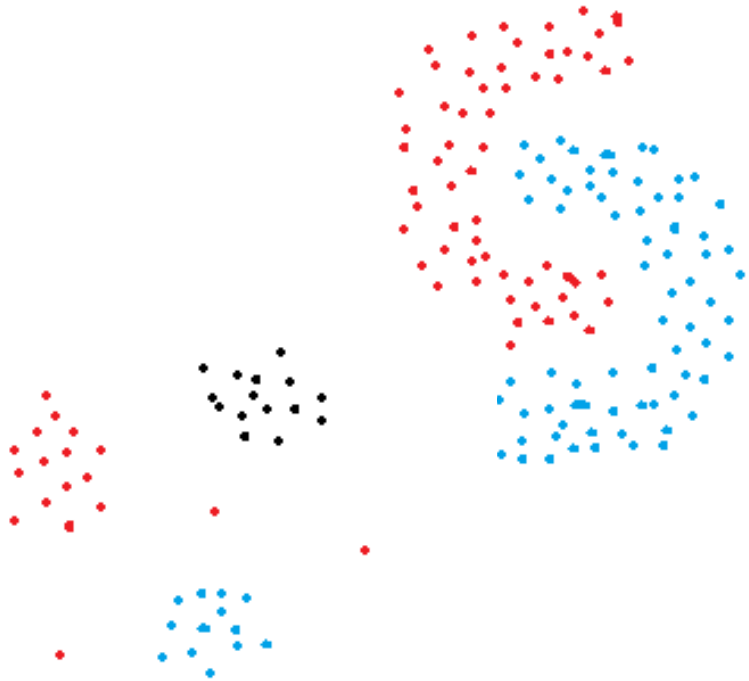


But

- What about these figures ?



- The data points in these figures are grouped in arbitrary shapes or include outliers.



- Density-based clustering algorithms are very efficient at finding high—density regions and outliers

- It is important to detect outliers for some tasks - anomaly detection

DBSCAN Algorithm

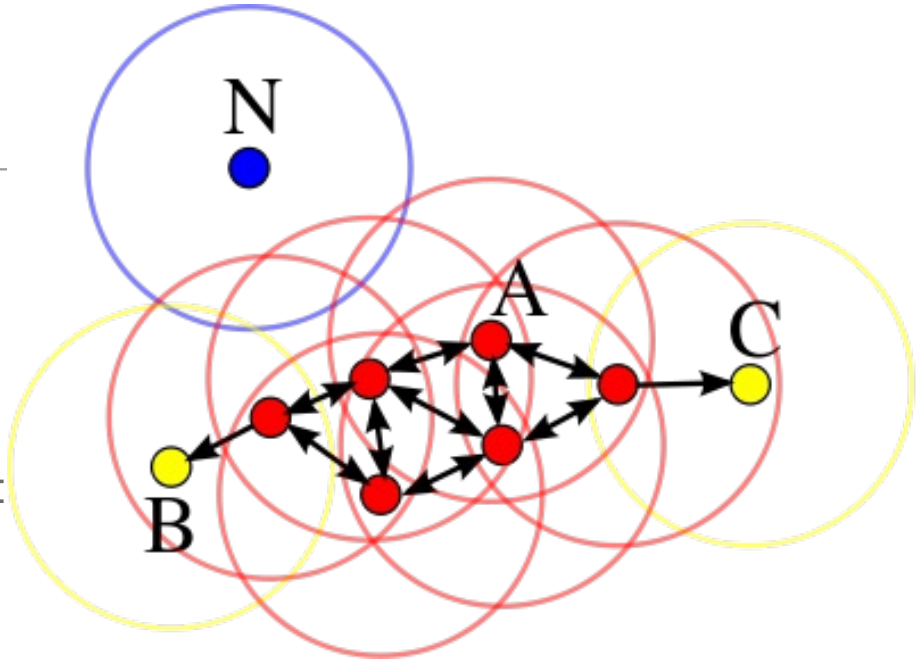
- DBSCAN stands for **d**ensity-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise. It is able to find arbitrary shaped clusters and clusters with noise (i.e. outliers)

- The main idea behind DBSCAN is that a point belongs to a cluster if it is close to many points from that cluster.

- There are two key parameters of DBSCAN:
  - **eps**: The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to eps.
  - **minPts:** Minimum number of data points to define a cluster.

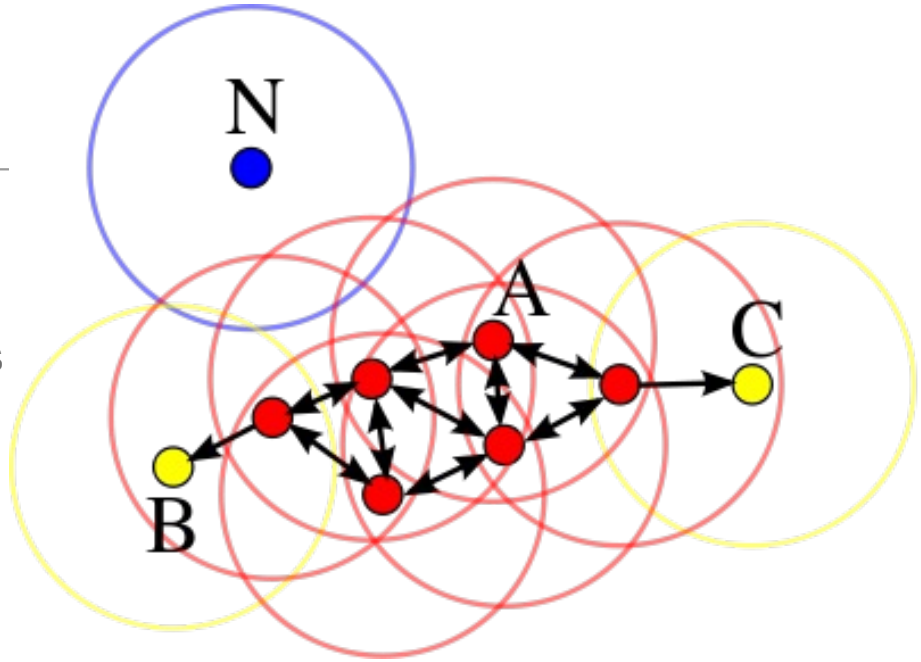
DBSCAN Algorithm

- Based on these two parameters, points are classified as core point, border point, or outlier:
  - **Core point:** A point is a core point if there are at least minPts number of points (including the point itself) in its surrounding area with radius eps.
  - **Border point:** A point is a border point if it is reachable from a core point and there are less than minPts number of points within its surrounding area.
  - **Outlier:** A point is an outlier if it is not a core point and not reachable from any core points.
- These points are explained in this diagram.



DBSCAN Algorithm

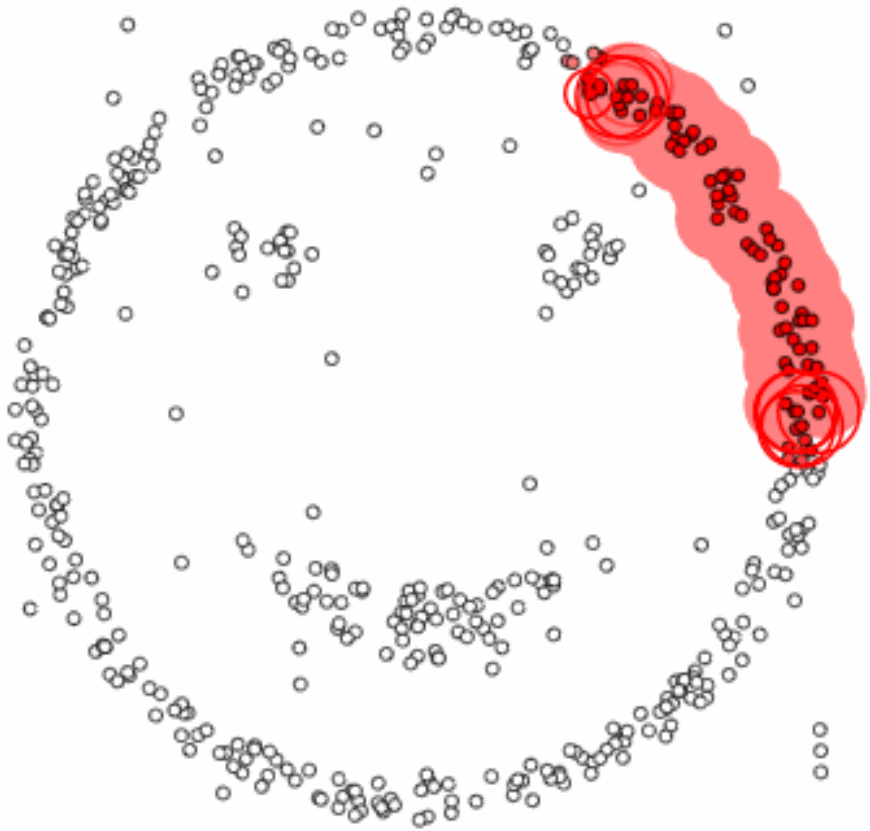
- In this examples, minPts is 4.
- The Red points are core points because there are **at least** 4 points within their surrounding area with radius eps. This area is shown with the circles in the figure.
- The yellow points are border points because they are reachable from a core point and have less than 4 points within their neighborhood.
- Reachable means being in the surrounding area of a core point.
- The points B and C have two points (including the point itself) within their neigborhood (i.e. the surrounding area with a radius of eps).
- Finally N is an outlier because it is not a core point and cannot be reached from a core point



DBSCAN Algorithm

- How DBSCAN Algorithm works
  - minPts and eps are determined.
  - A starting point is selected at random at it's neighborhood area is determined using radius eps. If there are at least minPts number of points in the neighborhood, the point is marked as core point and a cluster formation starts. If not, the point is marked as noise. Once a cluster formation starts (let's say cluster A), all the points within the neighborhood of initial point become a part of cluster A. If these new points are also core points, the points that are in the neighborhood of them are also added to cluster A.
  - Next step is to randomly choose another point among the points that have not been visited in the previous steps. Then same procedure applies.
  - This process is finished when all points are visited.

The distance between points is determined using a distance measurement method as in k-means algorithm. The most commonly used method is euclidean distance



epsilon = 1.00
minPoints = 4

Restart

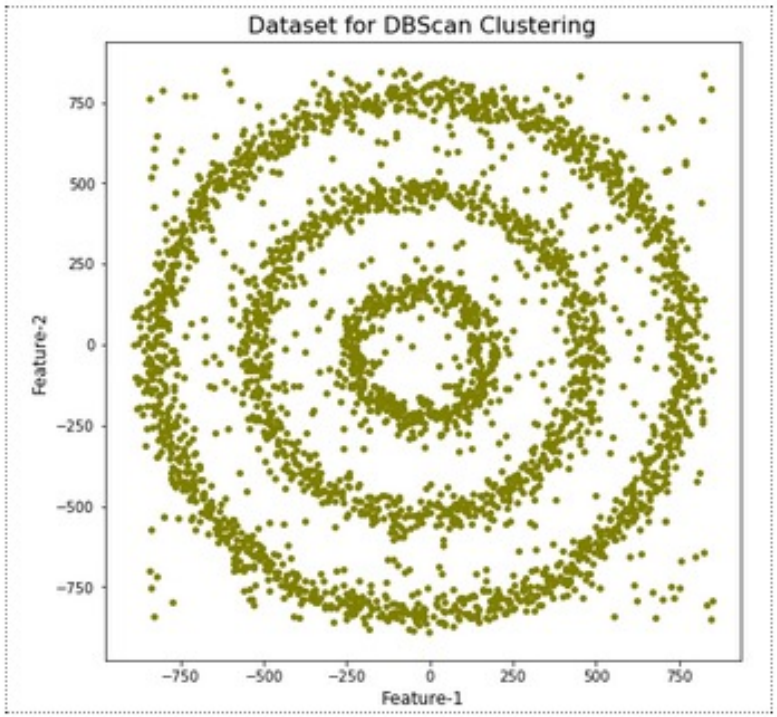
Pause

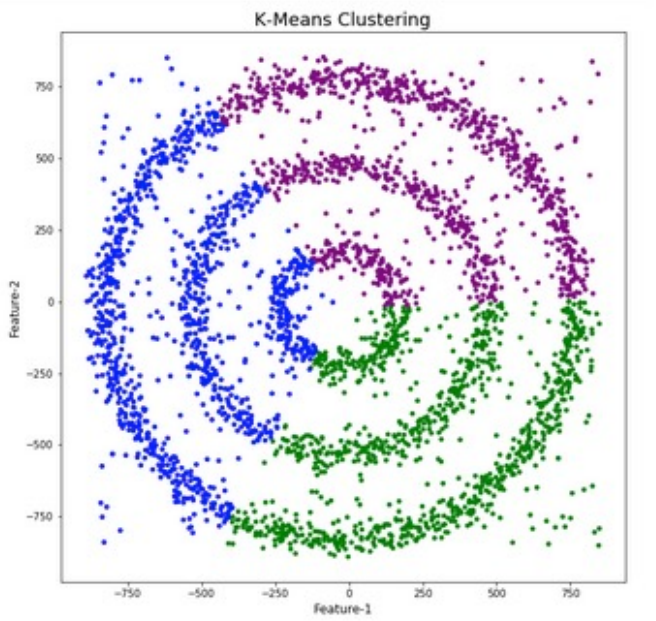
DBSCAN

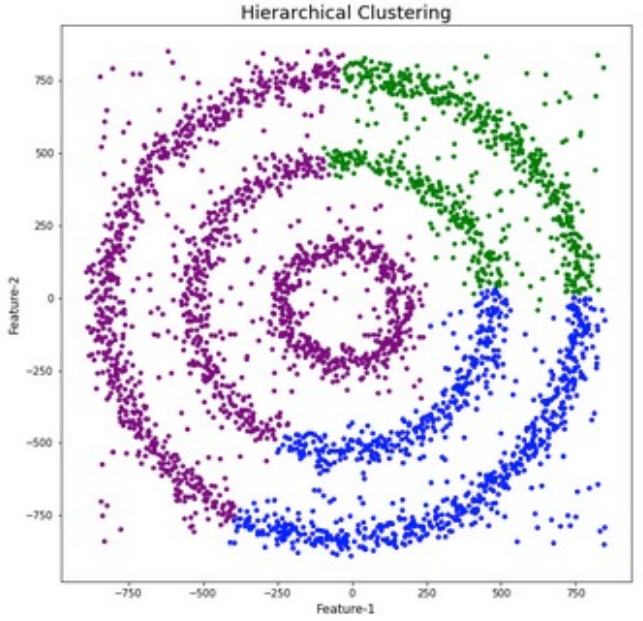
k-means

Example

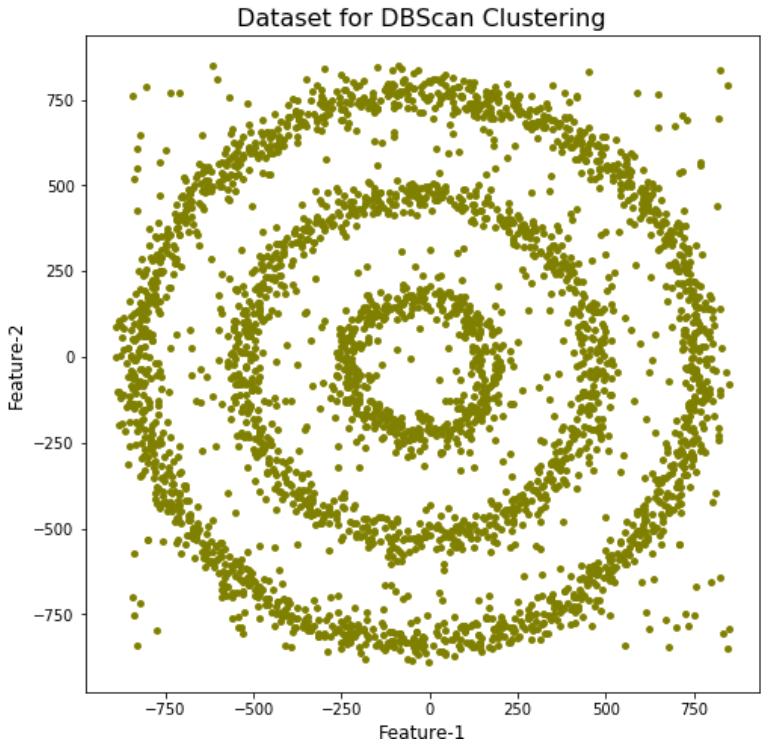
See blog posts : DBScan Clustering in Python
Comparing Cluster Algorithms on Density Data





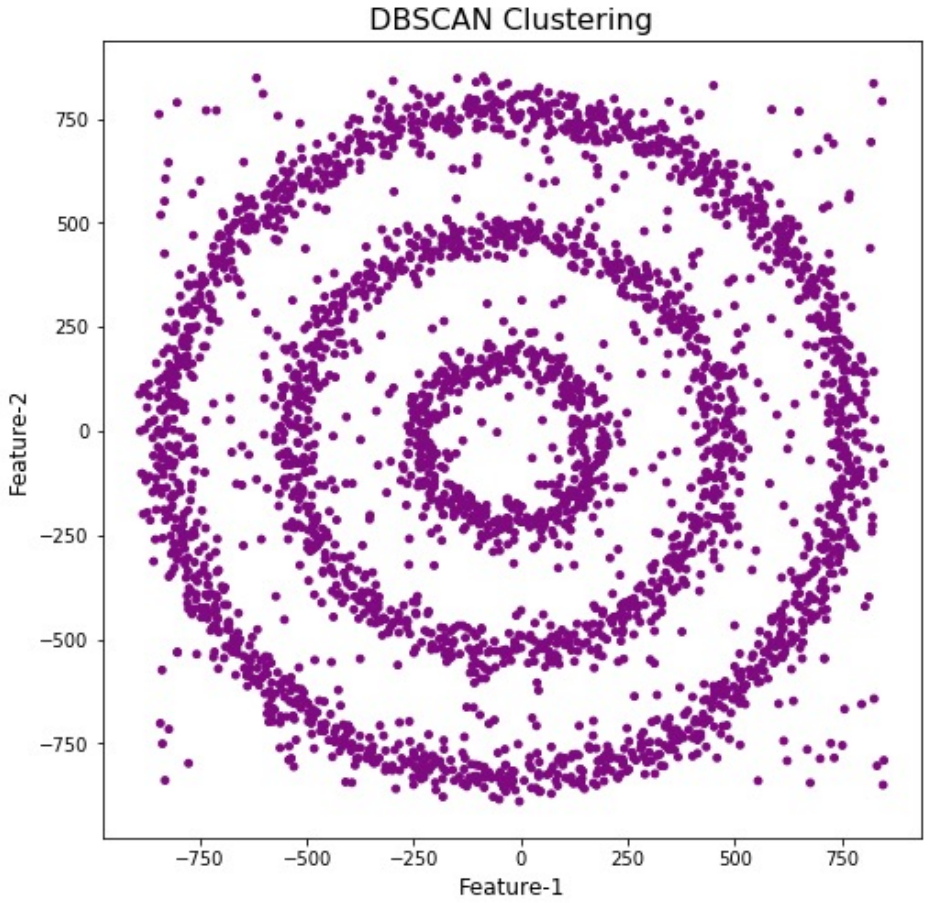


```
from sklearn.cluster import DBSCAN
```

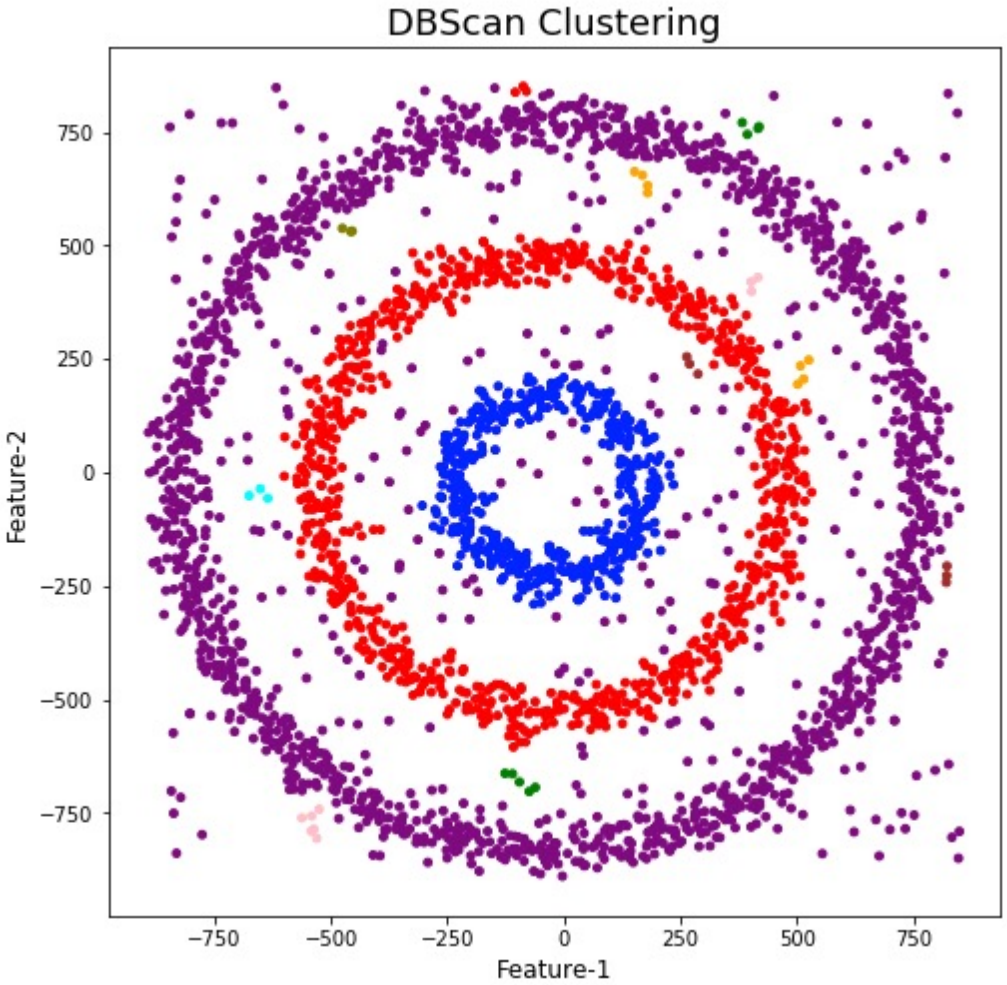


DBSCAN with default settings
All points are in the one/same Cluster

eps is too small



Adjusted values for eps

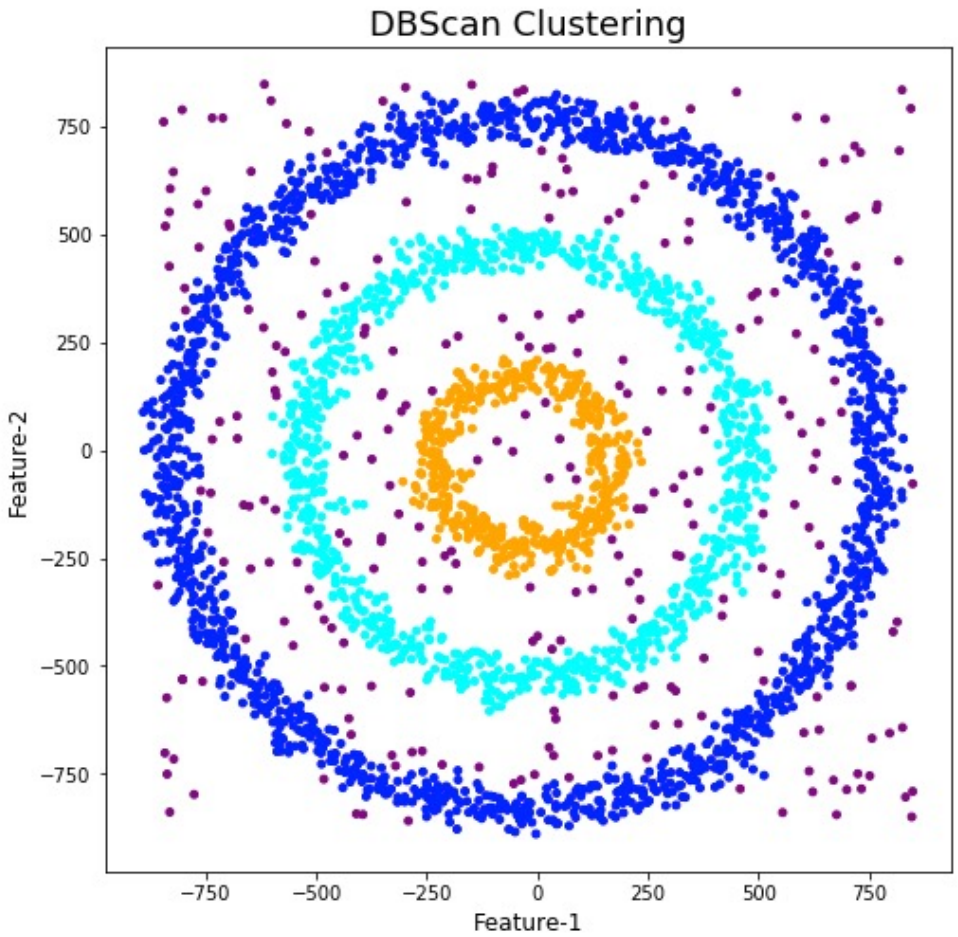


We do get Outliers, these are indicated by Cluster -1
But others are also Outliers

```
df['DBSCAN_opt_labels']=dbscan_opt.labels_
df['DBSCAN_opt_labels'].value_counts()

 0     1559
 2      898
 3      470
-1      282
 8        6
 5        5
 4        4
 10       4
 11       4
 6        3
 12       3
 1        3
 7        3
 9        3
 13       3
Name: DBSCAN_opt_labels, dtype: int64
```

Remove some of these Outliers
[might be better to recast to -1]

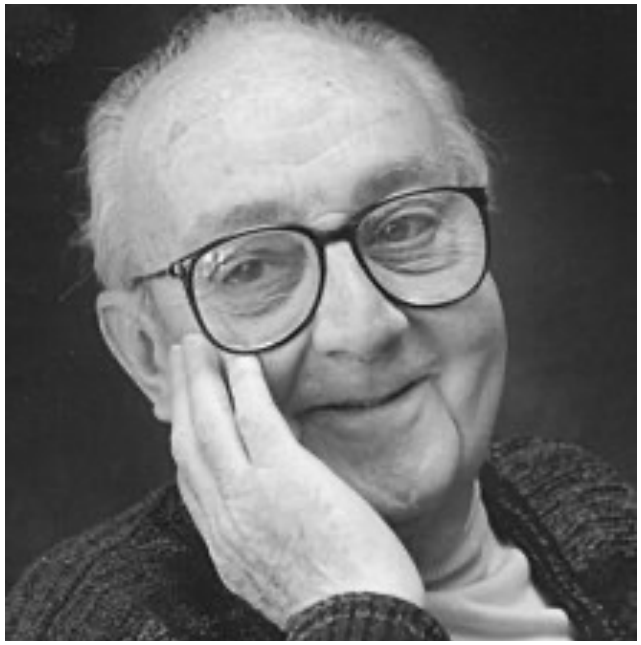
```
df2 = df[df['DBSCAN_opt_labels'].isin([-1,0,2,3])]
df2['DBSCAN_opt_labels'].value_counts()
 0    1559
 2     898
 3     470
-1     282
Name: DBSCAN_opt_labels, dtype: int64
```

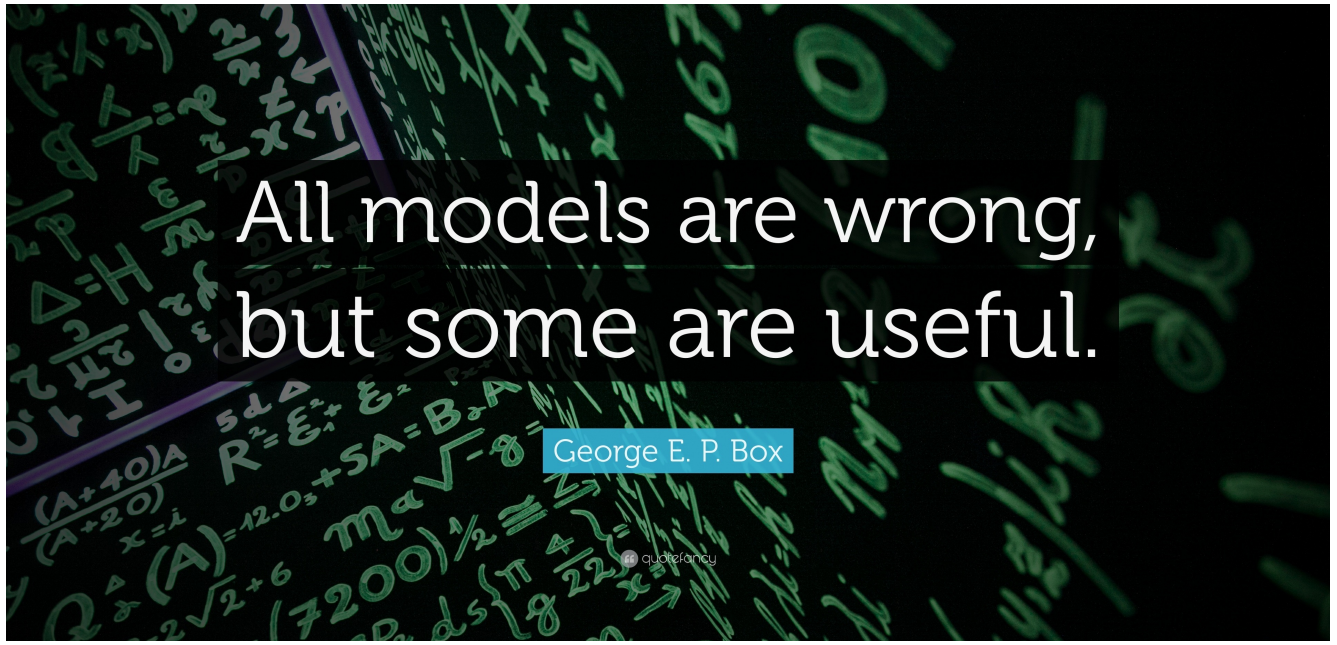


Try it yourself

Code and demo can be found at

See blog posts : DBScan Clustering in Python
Comparing Cluster Algorithms on Density Data



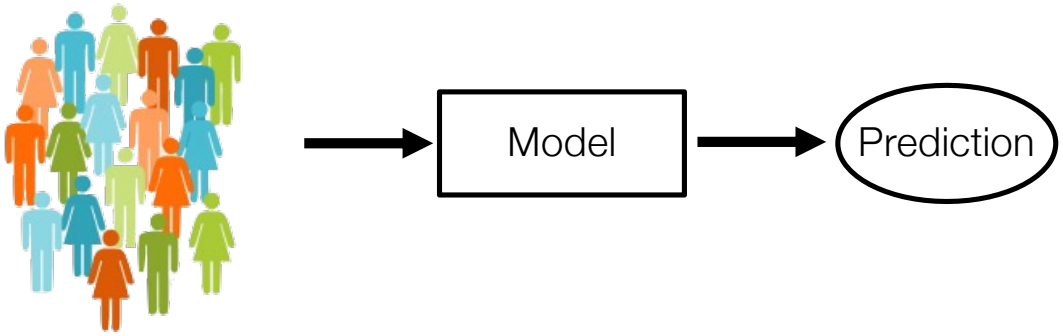


His paper was published in the Journal of the American Statistical Association, 1976
Book Empirical Model-Building and Response Surfaces, 1987


One more thing...

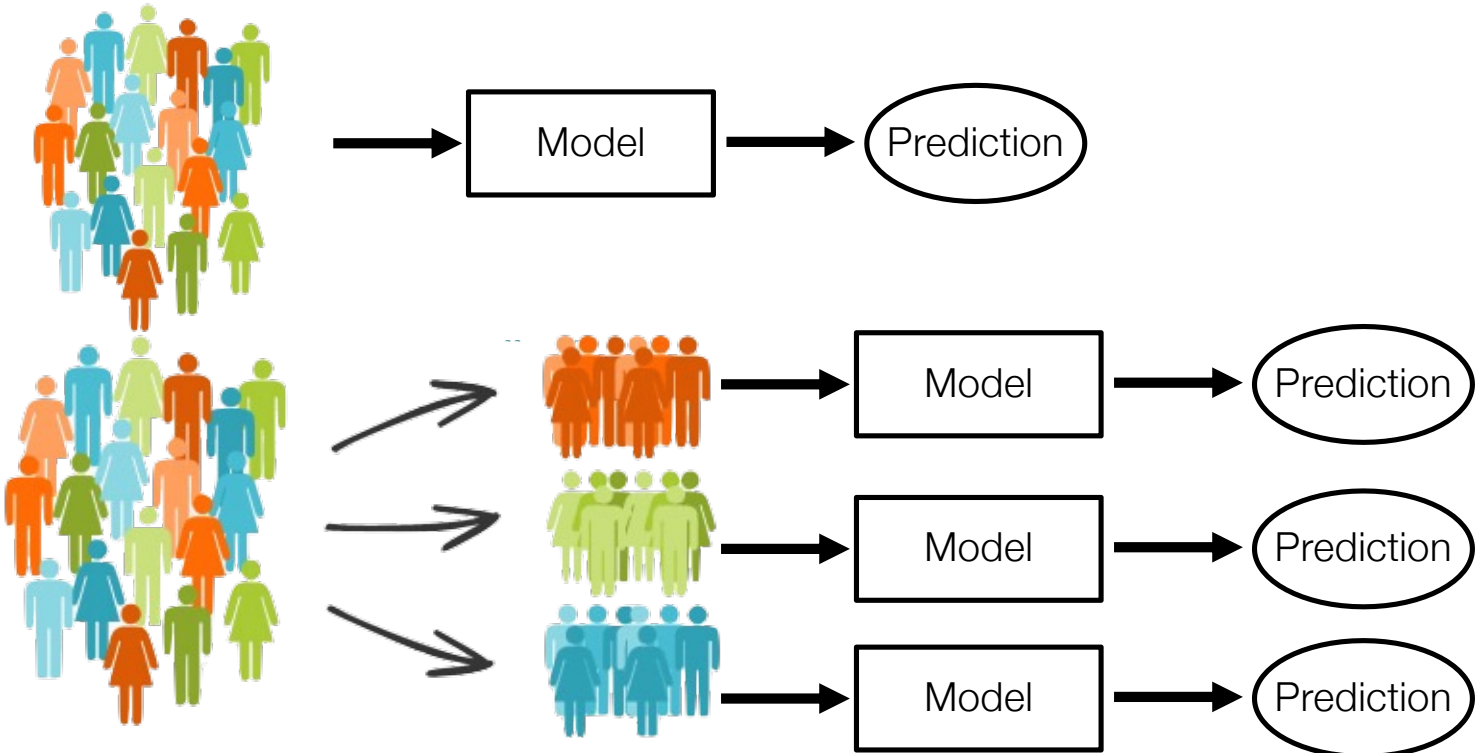
Better Data Preparation

- Using Segmentation / Clustering for Data Preparation
  - Improve accuracy
  - Models are focused on each group



Better Data Preparation

- Using Segmentation / Clustering for Data Preparation
  - Improve accuracy
  - Models are focused on each group



Demo

Any Questions ?

What Now/Next ?