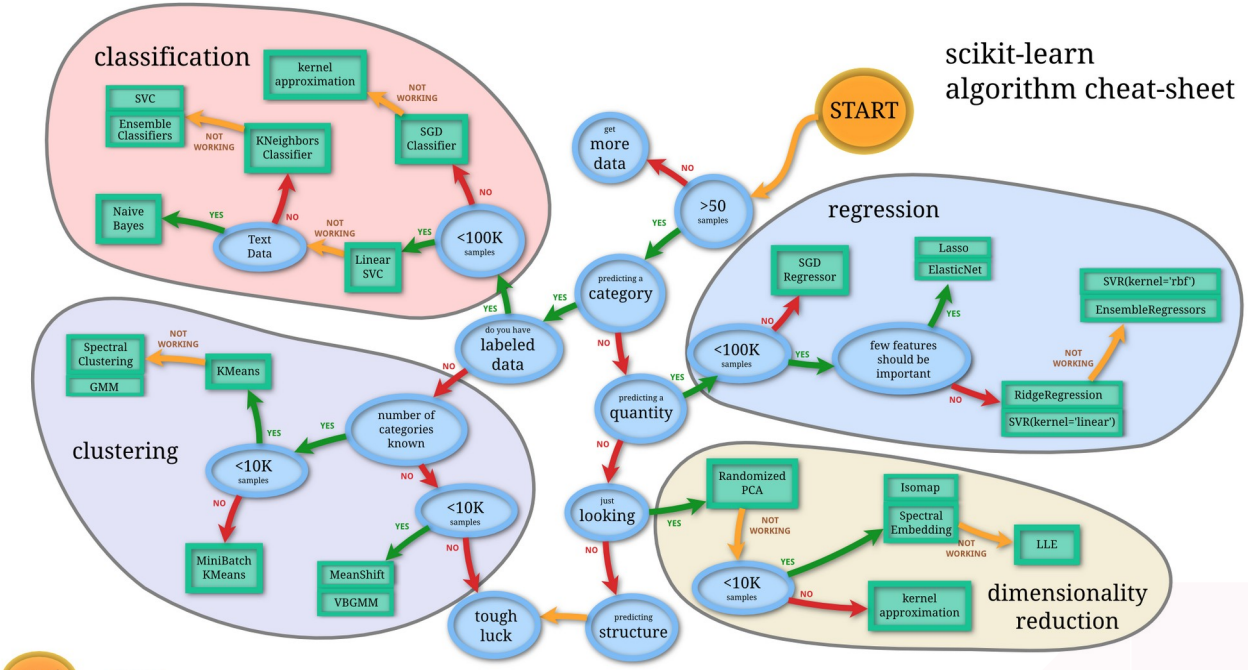
Galvanize Data Science
Clustering

Land Belenky

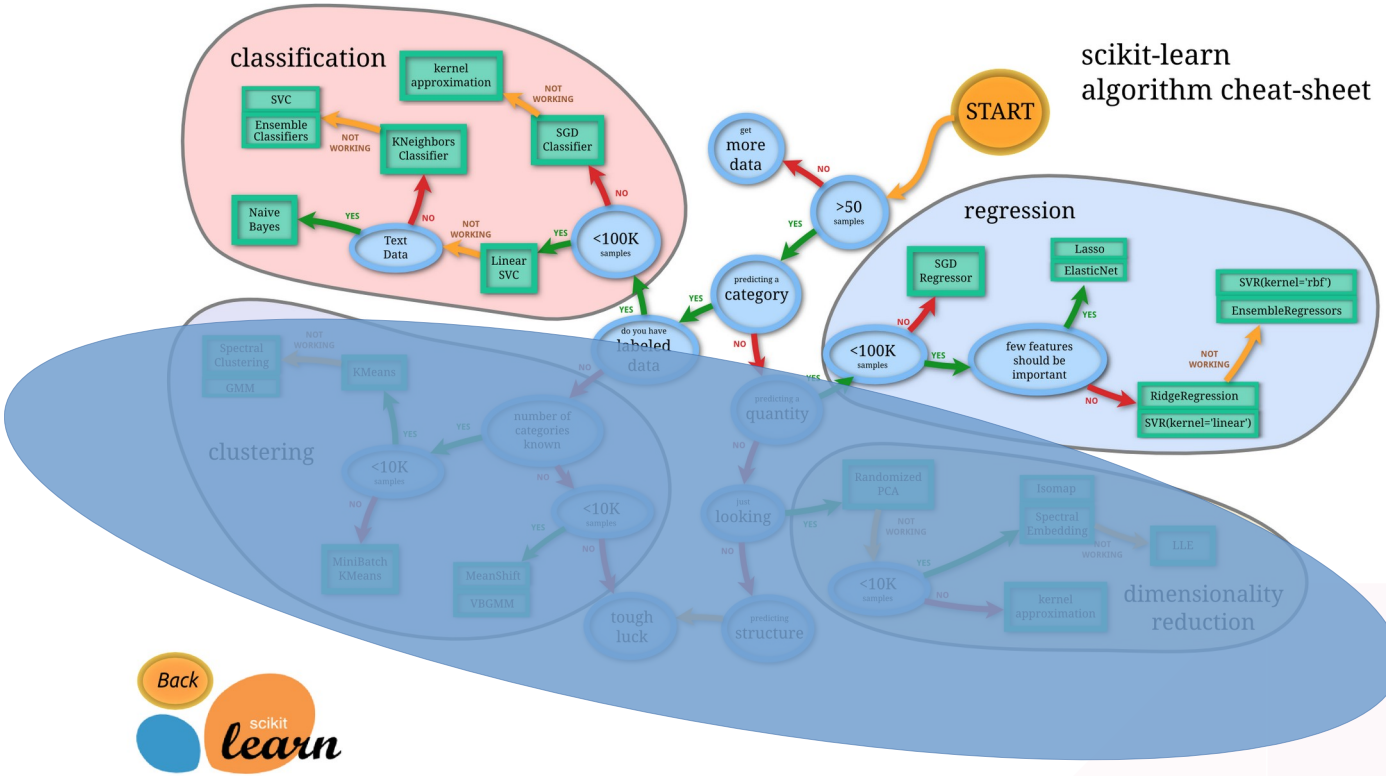
Overview

- **Supervised vs. Unsupervised**
- **Two Types of Clustering:**
  - **K-Means**
  - **Hierarchical**
- **Methods:**
  - **Distances**
  - **Scaling**
  - **Initialization**
  - **Stopping**
- **Measurements**
- **Selecting the Right K**
- **Combining Clustering and Regression**

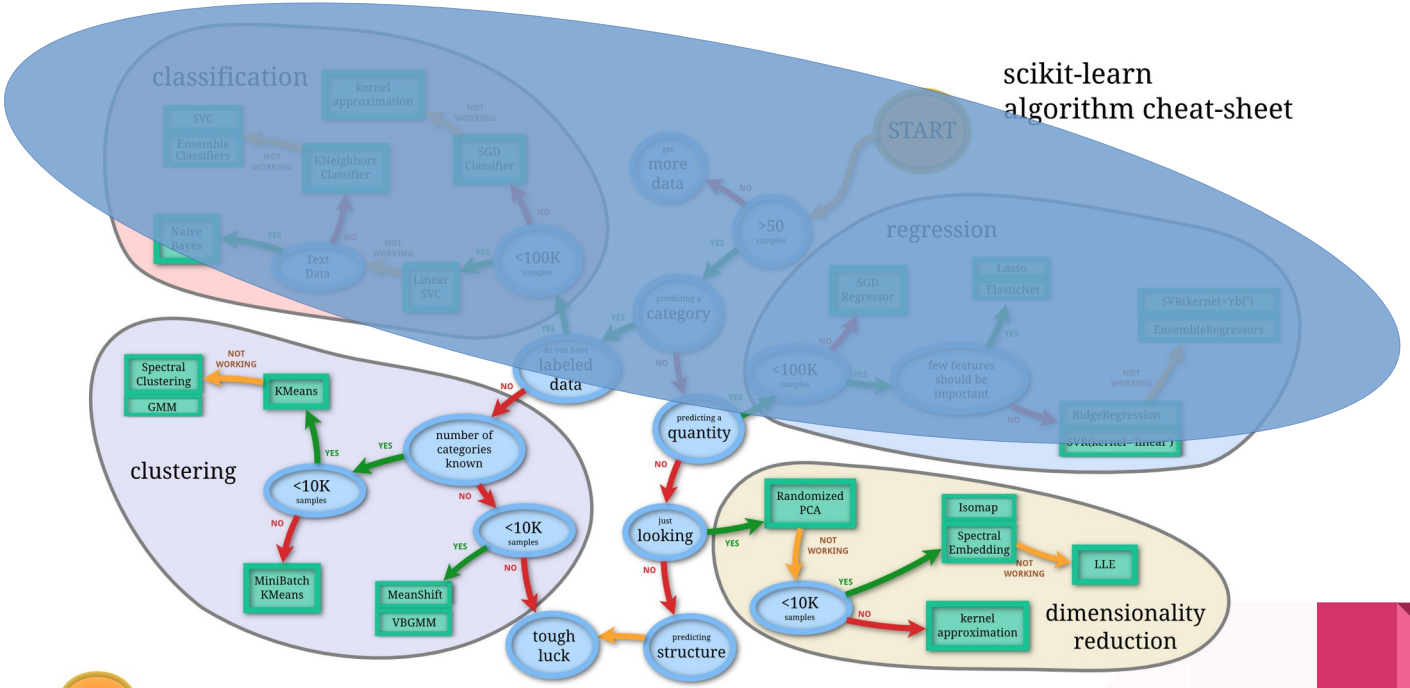
Supervised vs. Unsupervised



Supervised vs. Unsupervised



Supervised vs. Unsupervised



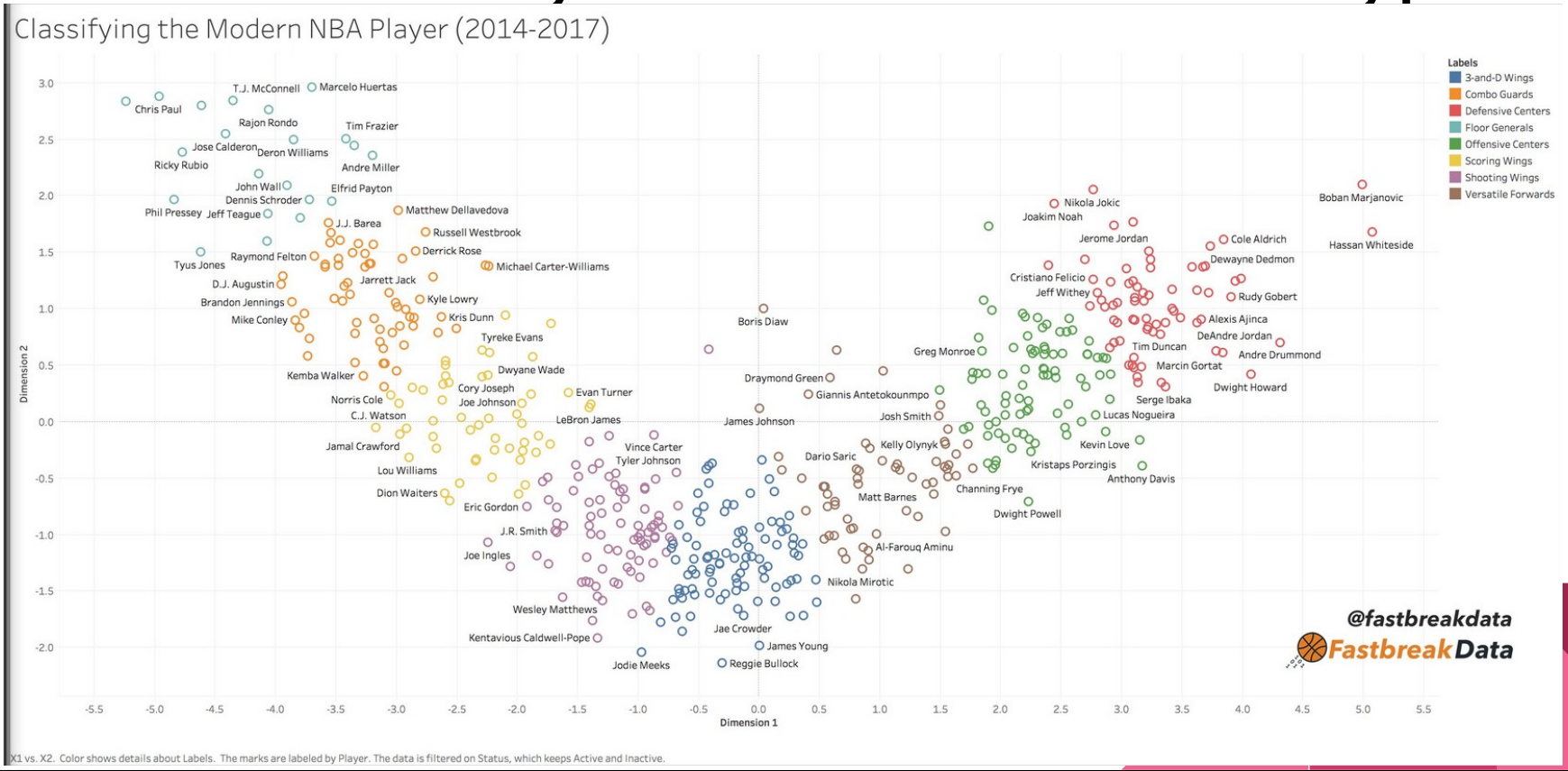
Real World Use Cases

- Customer Segmentation
- Product Segmentation
- Image Segmentation
- Anomaly Detection
- Social Network Analysis
- As a complement to supervised learning

Classifying the Modern NBA Player

- All players exist on a spectrum (offense, defense, passing, assists, rebounding, blocking, etc)
- Historically, 3 positions
  - Guard, forward, center
- In the 1980s, the game evolved and 5 positions were identified
  - Point guard, shooting guard, small forward, power forward, center
- Some Non-standard and hybrid positions:
  - Point forward, swingman, big, stretch four
- Some systems:
  - Running guard, combo guard, stationary guard
- Understanding positions helps with coaching, in-game decisions, player development, recruiting and trading strategy

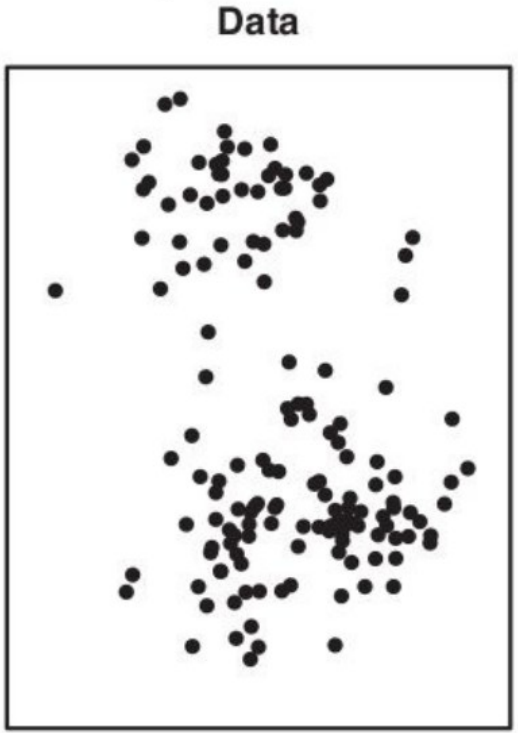
Modern Players: 8 semi-distinct types



X1 vs. X2. Color shows details about Labels. The marks are labeled by Player. The data is filtered on Status, which keeps Active and Inactive.

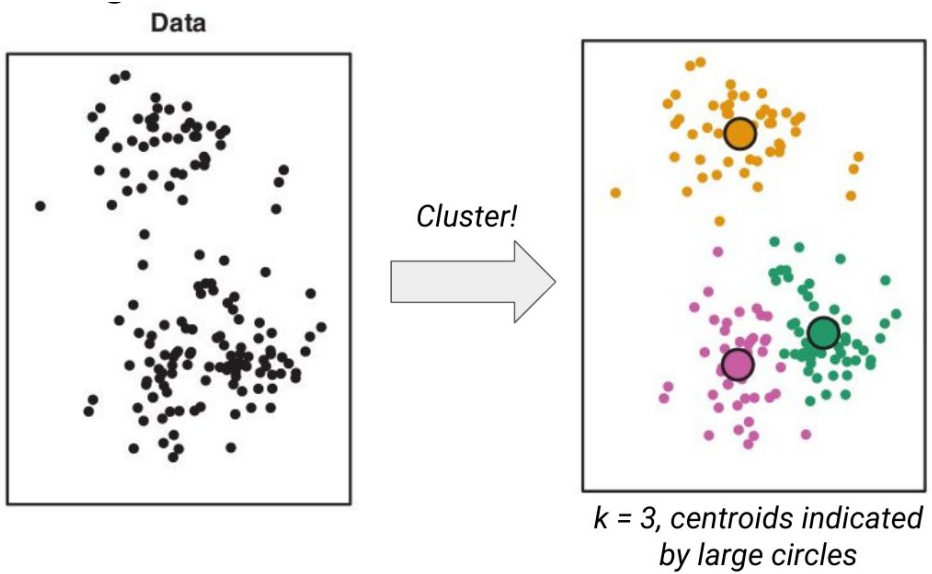
Clustering

- Data Exists
- Distributed Across Multiple Dimensions
- Reason to believe (or suspect) that difference exist in the data that have not been *explicitly* labeled
- Reason to believe (or suspect) that knowing these differences helps make decisions, or improve outcomes



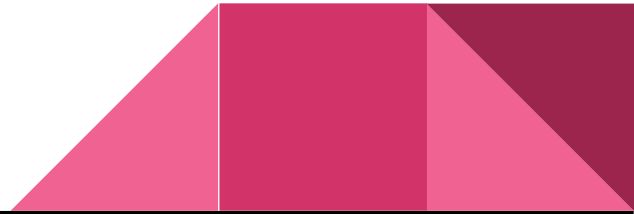
Hypothesis:

There is an unseen label that helps to explain the distribution of data.
This label can be deduced from the distributions of features



Objectives

- Given *unlabeled* data, how can we infer labels?
- How can we measure the integrity of these clusters?
- How can we determine the appropriate number of clusters?
- How can we use this new information to drive decisions?



Measurement

- A cluster is *good* when the points in the cluster are near to each other and far away from points outside of the cluster
- Minimize “Within Cluster Variation”: WCV

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

$|C_k|$ is number of observations in k-th cluster

Centroid of cluster C_k

WCV

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

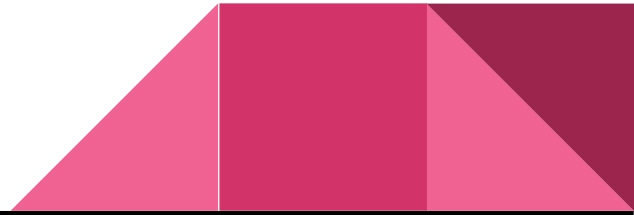
$|C_k|$ is number of observations in k-th cluster

Centroid of cluster C_k

- The average distance from each point to the center of the cluster that contains that point
- Extreme Case:
  - If there is one point in every cluster, each point *is* the center of its own cluster. Therefore all distances are zero and WCV is minimized

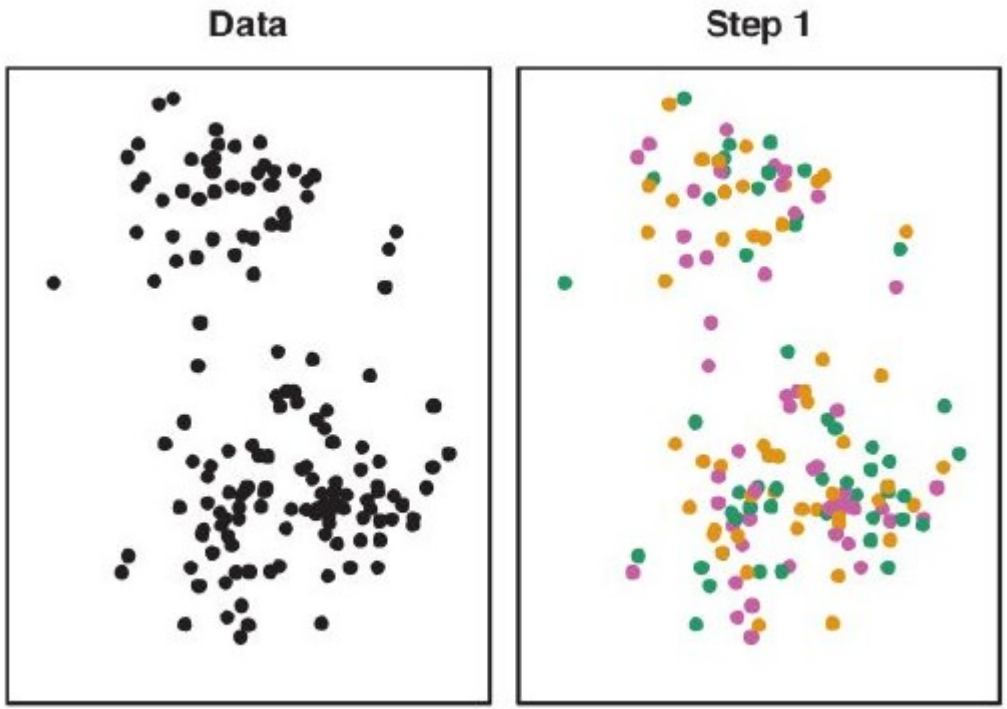
K-Means Clustering
(Not to be confused with K-Nearest Neighbors)

- Pick a value for *k* (number of clusters)
- Randomly initialize *k* points (centroids)
- Tentatively assign each *observation point* to the *nearest* centroid
- Iterate:
  - Move centroid points to the average position of all points that have been assigned to that cluster
  - Re-assign observations to nearest centroid
- Repeat until convergence



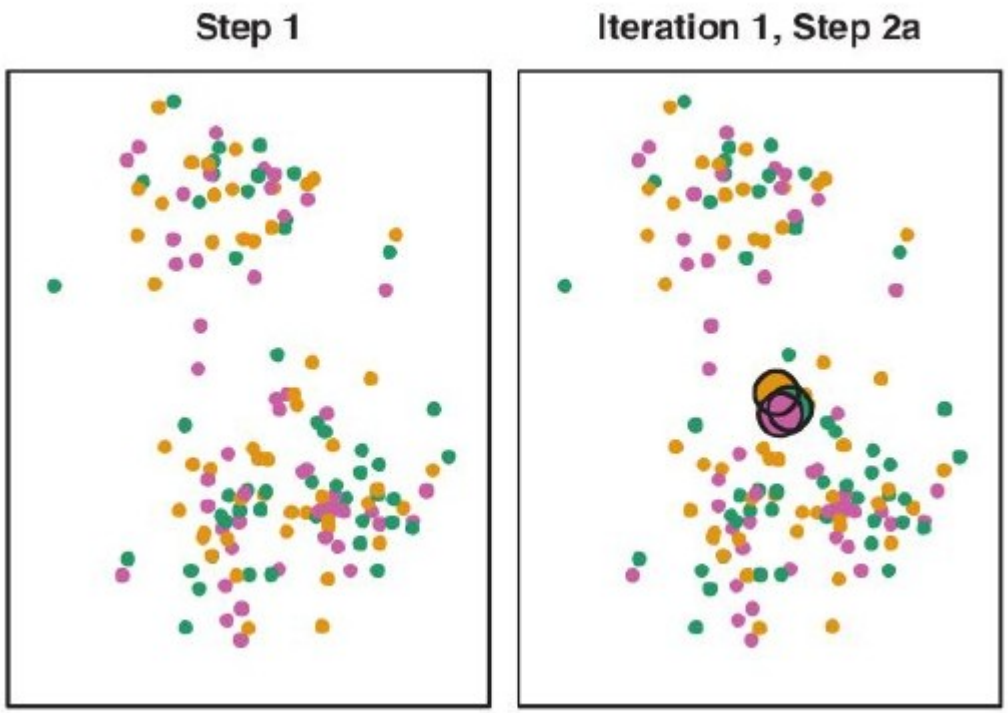
Step 1: Random Initialization

- $k = 3$
  - Green
  - Orange
  - Pink
- All points *randomly* assigned to clusters
- This ensure that centroids are within the range of data (but results in broad, overlapping clusters)



Step 2: Create Centroids

- Compute the centroid of each cluster as the average position of all points of that cluster
- Due to random initialization, each centroid will be near to the center of all data

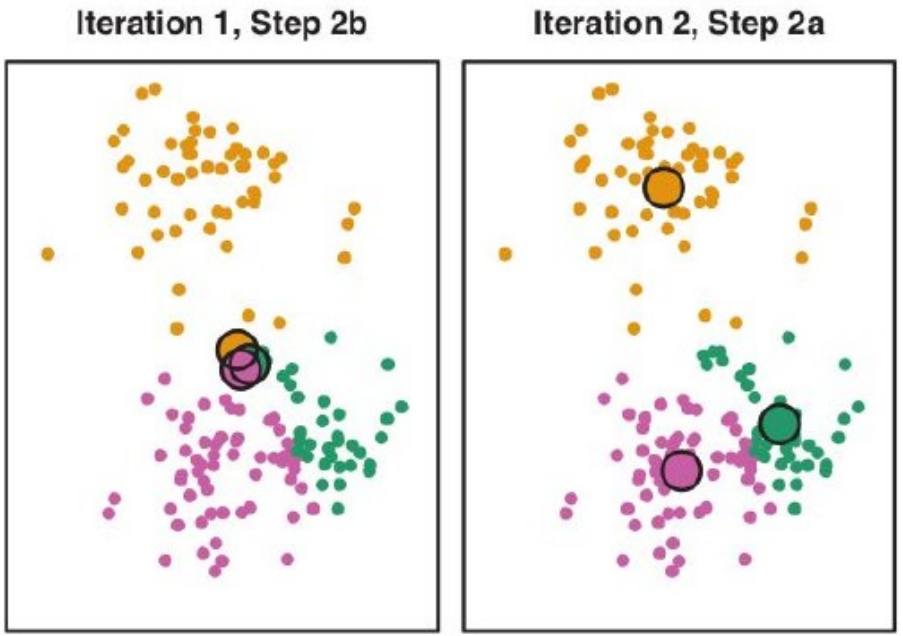


Step 3: Reassign Points according to nearest centroid

- Discard information about how observation points were originally assigned to clusters
- For each observation:
  - Measure distance to each centroid
  - Determine closest centroid
  - (tentatively) Re-assign point to cluster with closest centroid

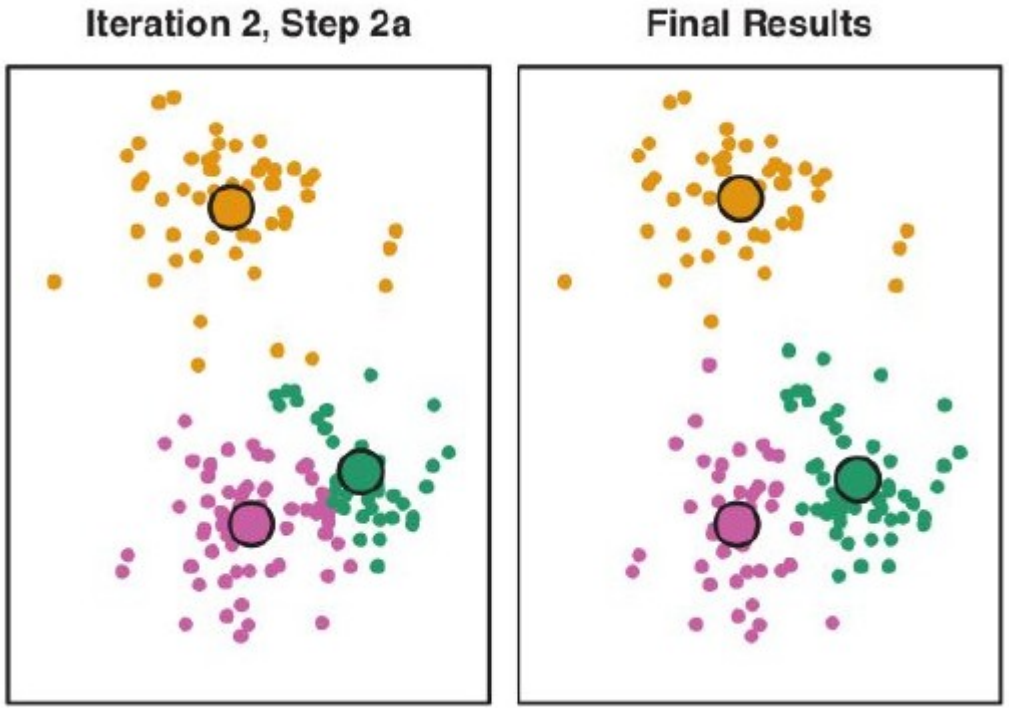
Step 4: Relocate Centroids

- For each cluster:
  - Identify the average position of all points in that cluster
  - Discard previous information about centroid of that cluster
  - Create new centroid at average position



Iterate

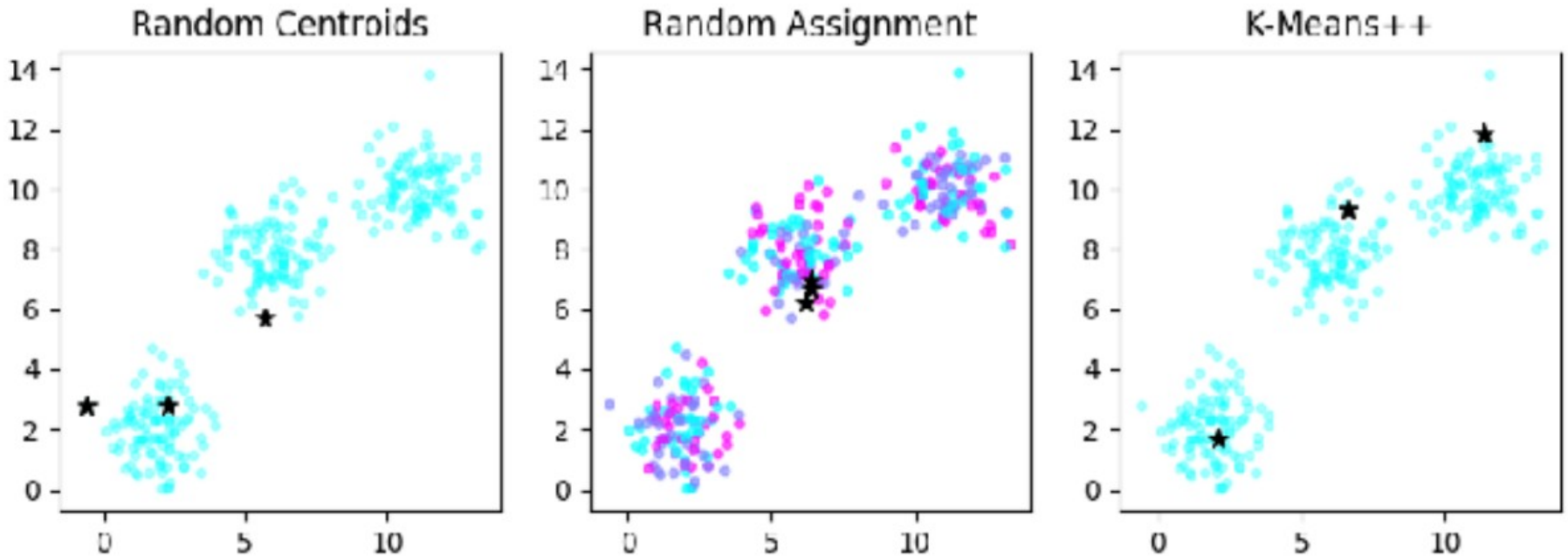
- Since centroids have just moved, some points may now lay closer to other centroids than their own.
- Iterate:
  - Reassign points to cluster with nearest centroid
  - Relocate centroid to center of observations



Different Approaches to Initialization

- Simplest: randomly choose *k* points from observations to serve as *initial* centroids
- Randomly assign points to clusters (as shown above)
- **k-means++**:
  - Chooses well-spread initial centroids.
  - First centroid chosen randomly.
  - Subsequent centroids chosen with probability proportional to the squared distance to the closes existing centroid. (i.e, “fill in the gaps”)
  - Default initialization in sklearn

Initialization



Stopping Criteria

- Repeat iteration process until:
  - Specified number of iterations
    - sklearn default: max_iter = 1000
  - Centroids don’t change at all / No points change clusters
  - Until centroids don’t move by very much
    - Sklearn default: tol = 0.0001

Non-Deterministic

- Due to random initialization, it is not assured that points will always be assigned to the same cluster
- Especially if clusters are not well separated or wrong value of *k* is chosen
- May be necessary to repeat clustering multiple times

Evaluating Clustering

Want to minimize Intra-Cluster Variance or Within Cluster Variance (WCV)

$$\min\left\{\sum_{k=1}^{K} WCV(C_k)\right\}$$

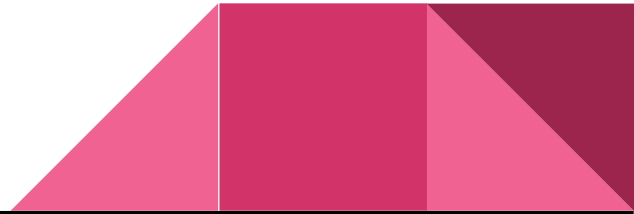
Where WCV for the kth cluster is the sum of all the pairwise Euclidean distances

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

Centroid of cluster C_k

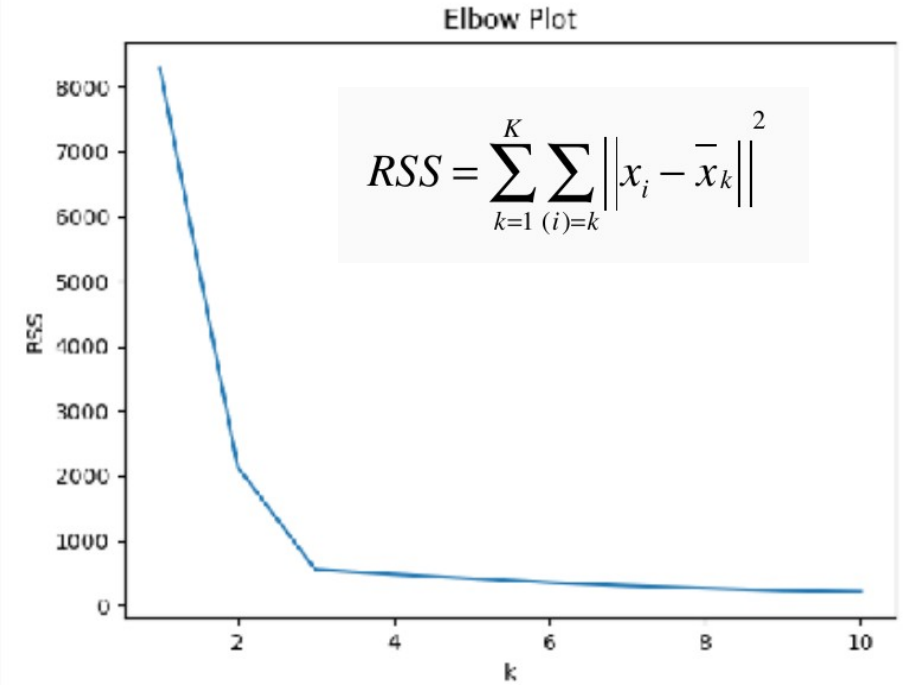
Picking the right value for k

- No precise method: requires judgment call
- Elbow Plot
- Silhouette Score
- GAP Statistic
- Domain Knowledge



Elbow Plot

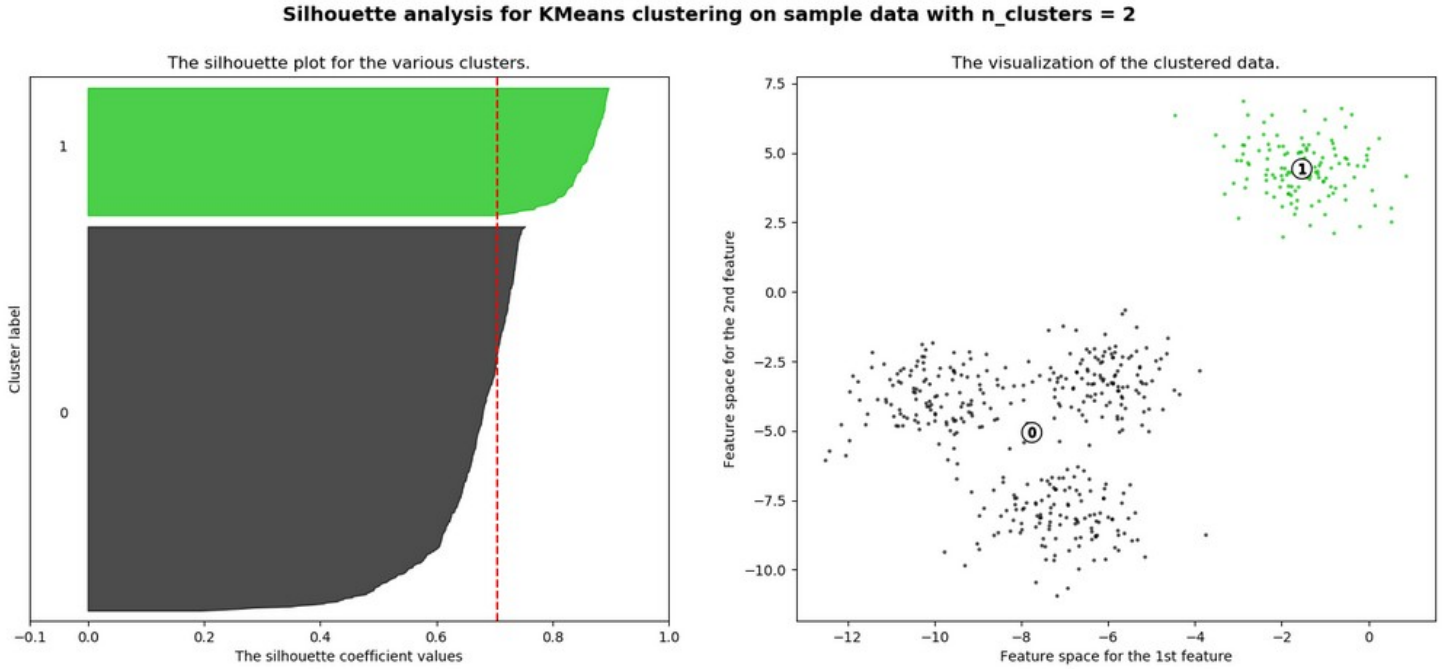
- Increasing *k* will always result in lower RSS, with diminishing returns
- Identify “elbow” where improvements become less significant
- Preference for simpler models (lower *k)*



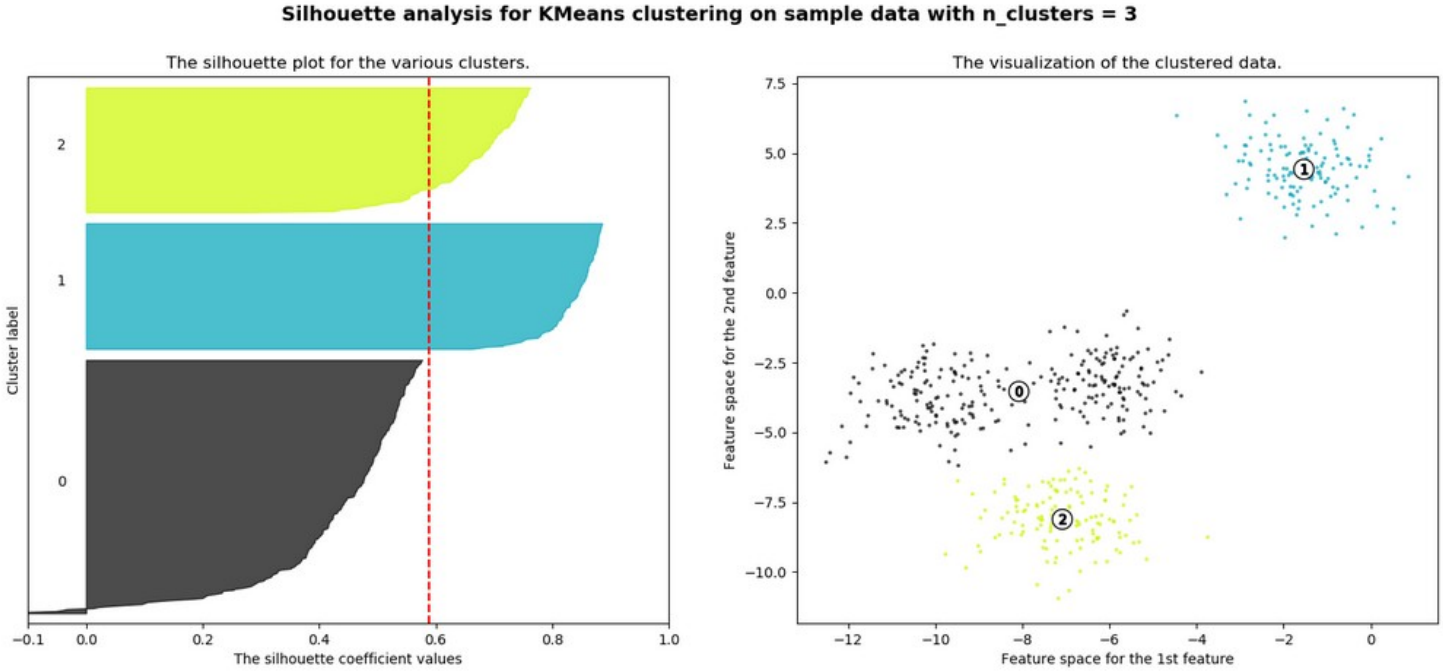
Silhouette Score

- Good clusters should have:
  - Low average intra-cluster distance (a)
  - High average nearest-cluster distance (b)
  - Silhouette = (b-a)/max(a,b)
  - Best: 1: (a=0, S = b/b)
  - Worst: -1: (b=0, S = -a/a)
  - S = 0: indicates b=a, or intra-cluster equal nearest=cluster distance

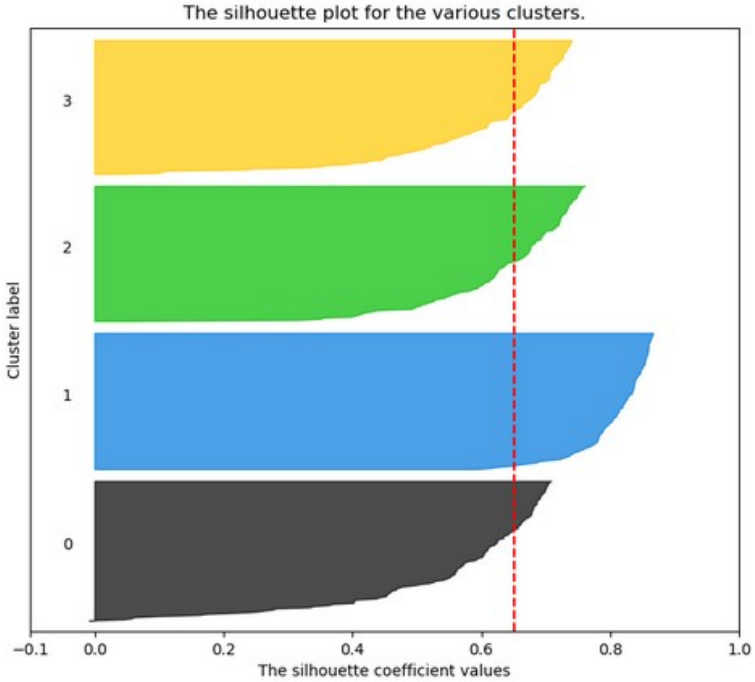
Silhouette Plots (k=2)

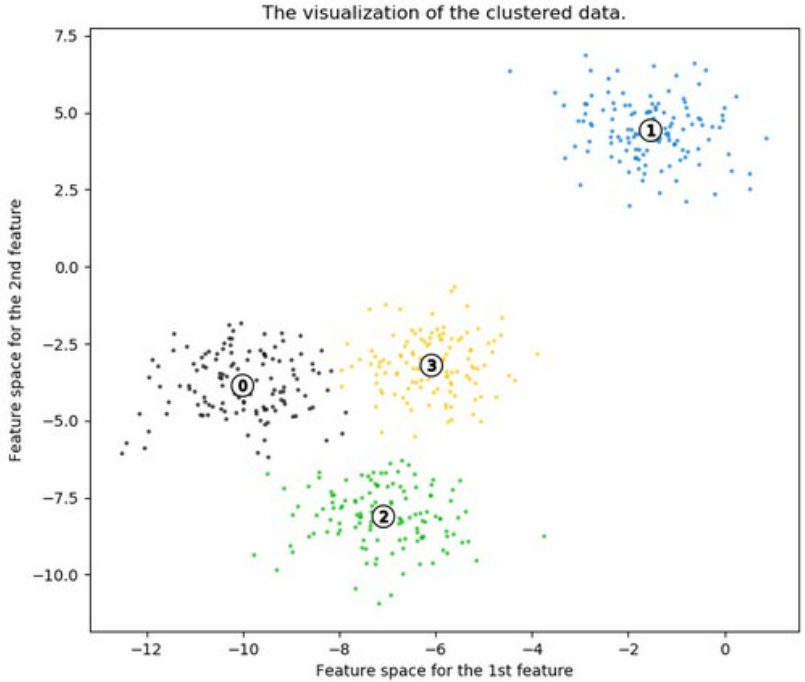


K = 3

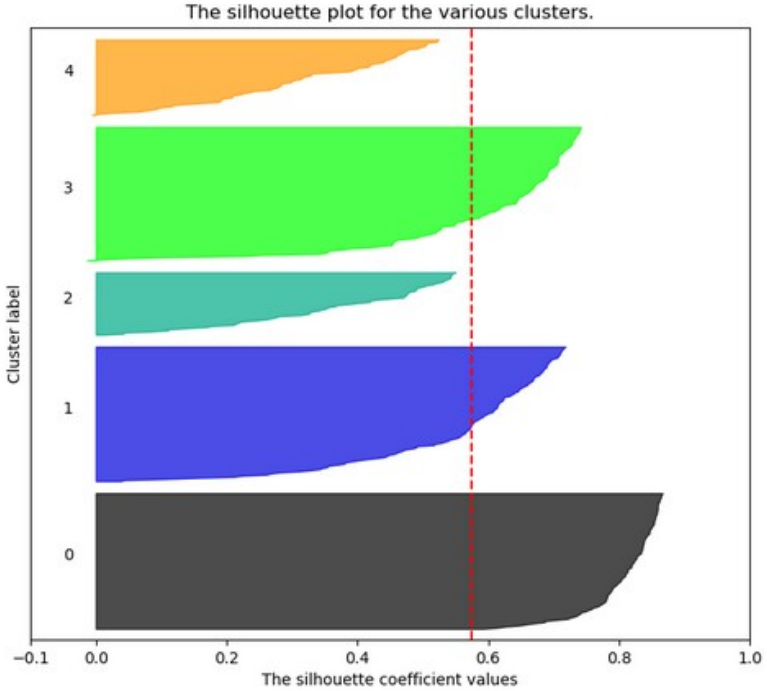

Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
Cluster label
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0
Feature space for the 1st feature

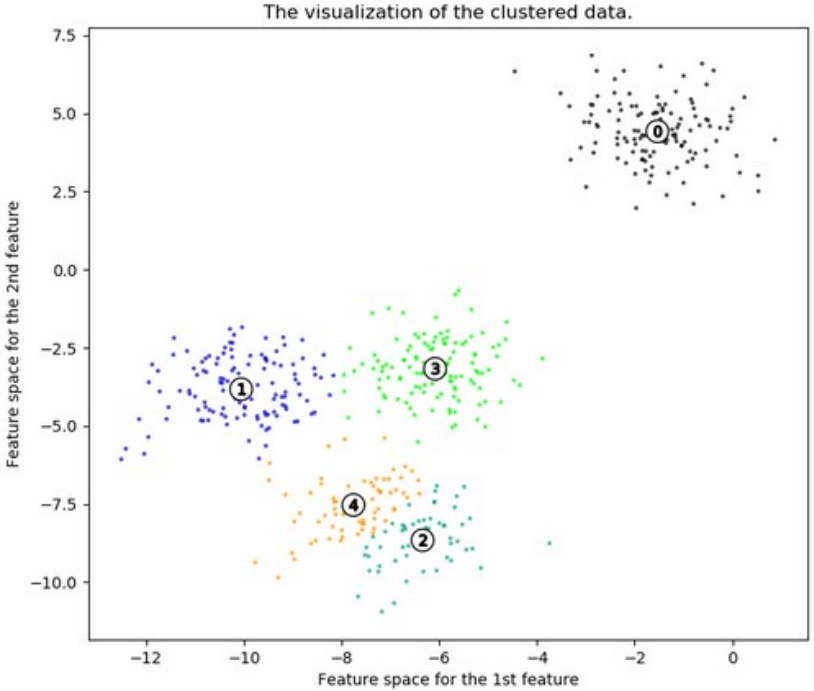
K = 4 (ideal)


The silhouette plot for the various clusters.
Cluster label
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values

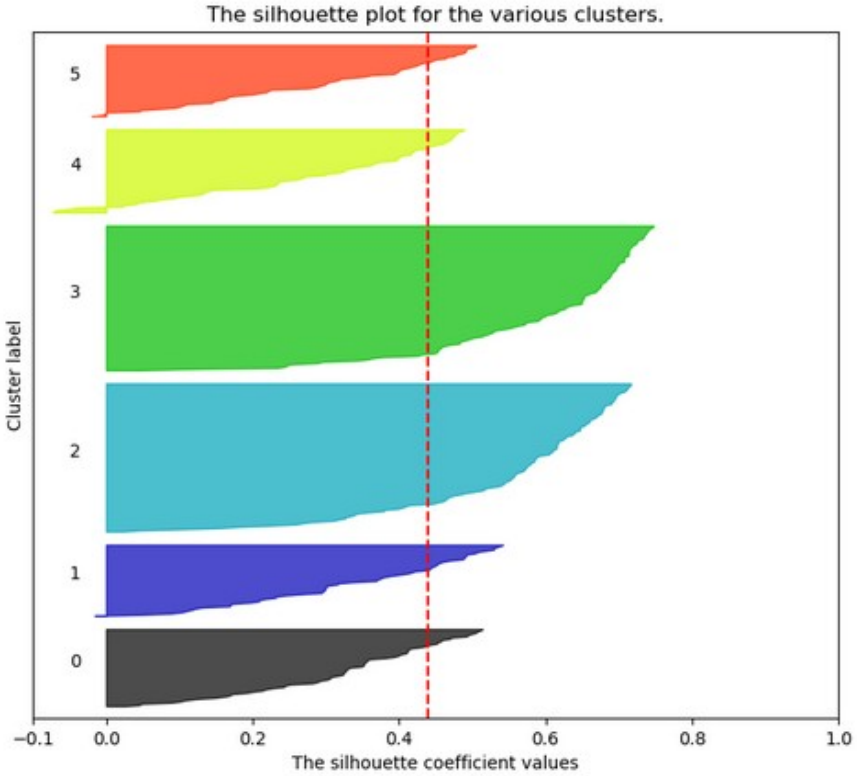

The visualization of the clustered data.
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0
Feature space for the 1st feature
0
1
2
3

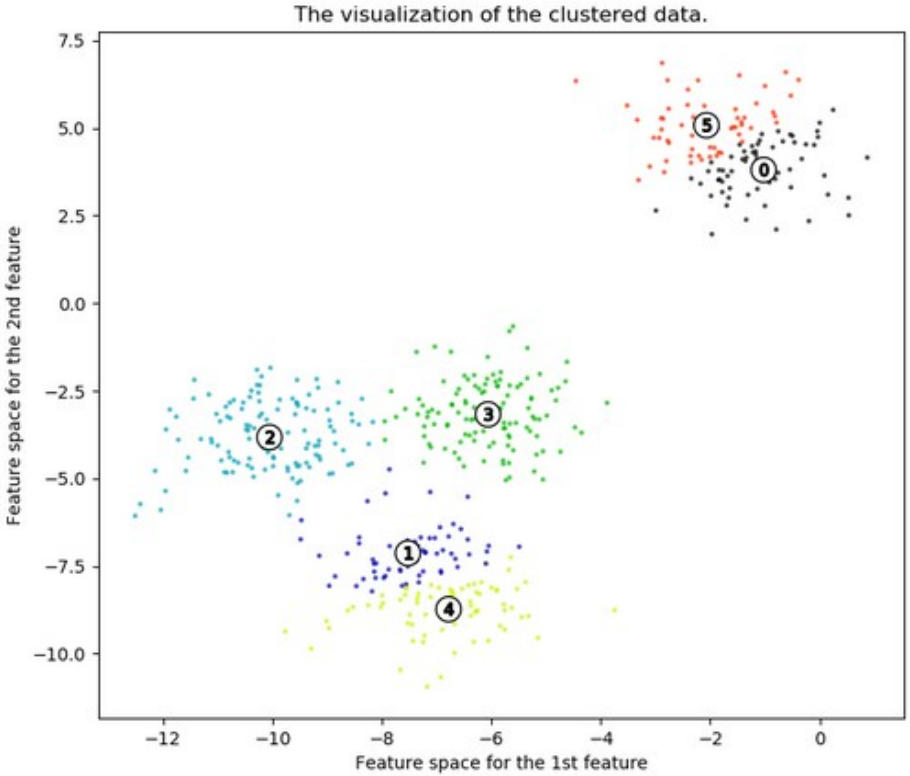
K = 5 (too many)


The silhouette plot for the various clusters.
Cluster label
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The visualization of the clustered data.
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0
Feature space for the 1st feature
0
1
2
3
4

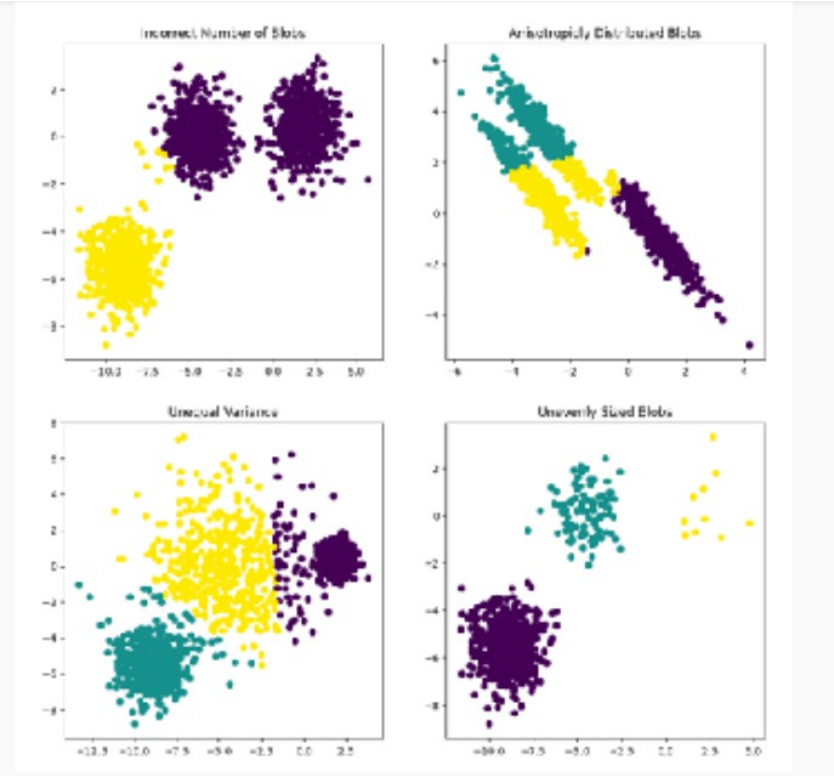
K = 6


The silhouette plot for the various clusters.
Cluster label
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The visualization of the clustered data.
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0
Feature space for the 1st feature
5
0
2
3
1
4

Assumptions

- Correct *k*
- Equal Variance
- Isotropic Shape
- Clusters do not need same number of points

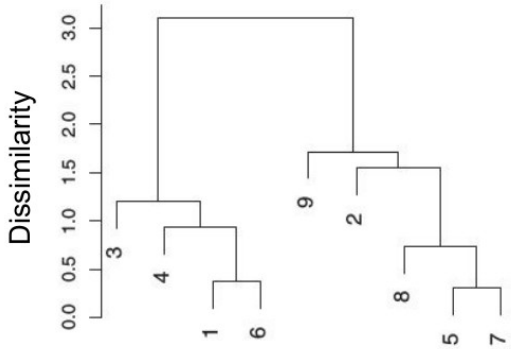


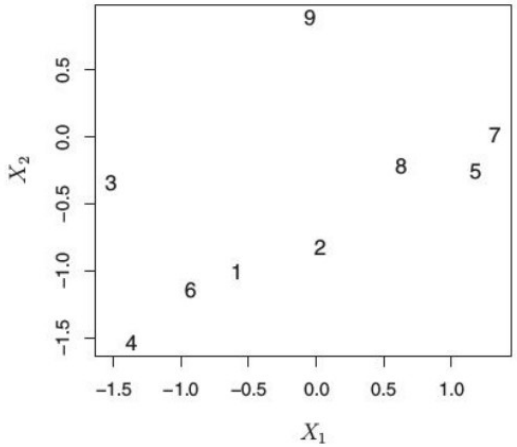
Hierarchical Clustering

- Do not pick *k* value in advance
- Effectively scan through a range of *k*
- Deterministic: no random initialization. Always same results
- Not limited to Euclidean distance

Dendrogram

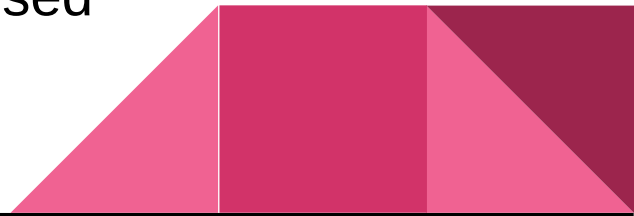
- X-axis arbitrary
- Y-axis: distance at which a connection can be made
- Points connected low are most similar
- Points connected high are most distant





Hierarchical Clustering Algorithm

- For each observation, measure distance to all other observations.
- Distance may be euclidean, cosine, etc
- Identify closest points
- Chart closest points on dendrogram at height = distance
- Fuse into a new cluster
- Recalculate distances between all points/clusters
- Continue until all points/clusters have been fused

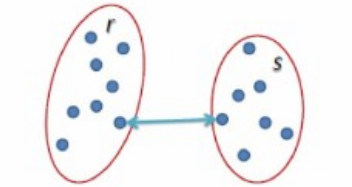


Fusing

Single Linkage In single linkage hierarchical clustering, the distance between two clusters is defined as the *shortest* distance between two points in each cluster.

"Nearest neighbor"

Drawback: Chaining - several clusters may be joined to just because of a few close cases

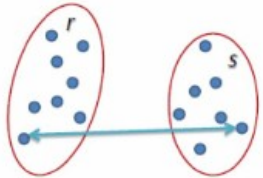


$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete Linkage In complete linkage hierarchical clustering, the distance between two clusters is defined as the *longest* distance between two points in each cluster.

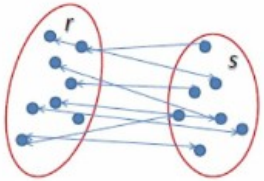
"Farthest neighbor"

Drawback: Cluster outliers prevent otherwise close clusters from merging.



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

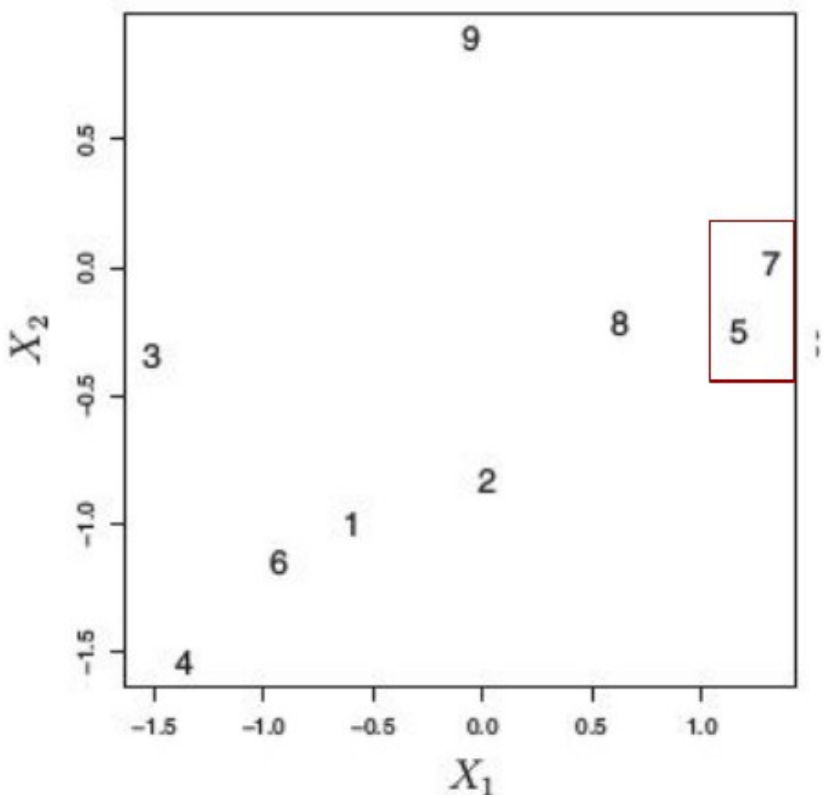
Average Linkage In average linkage hierarchical clustering, the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.



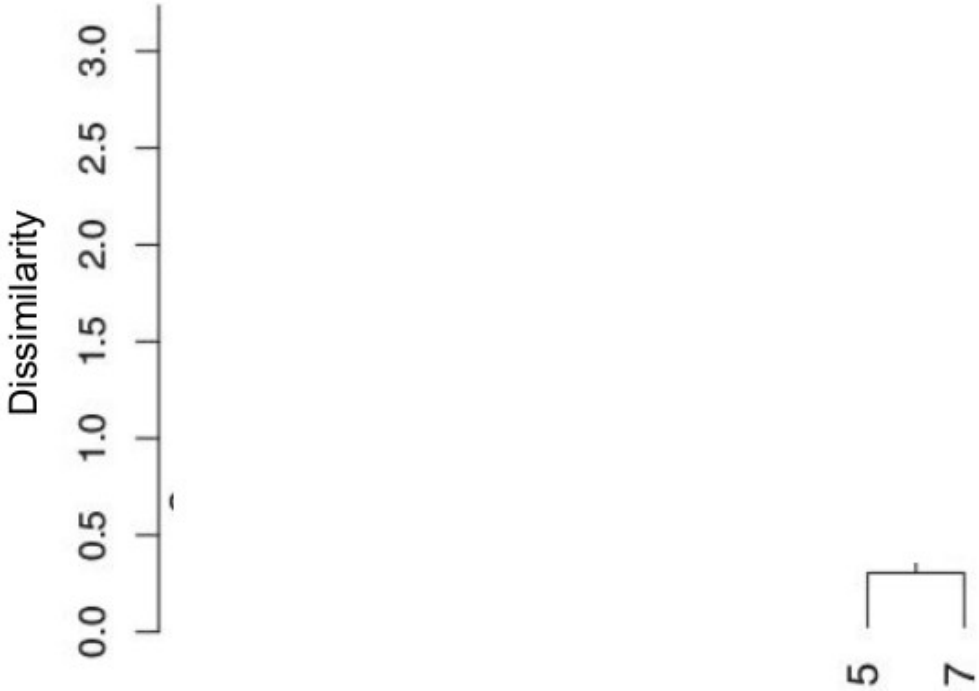
$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

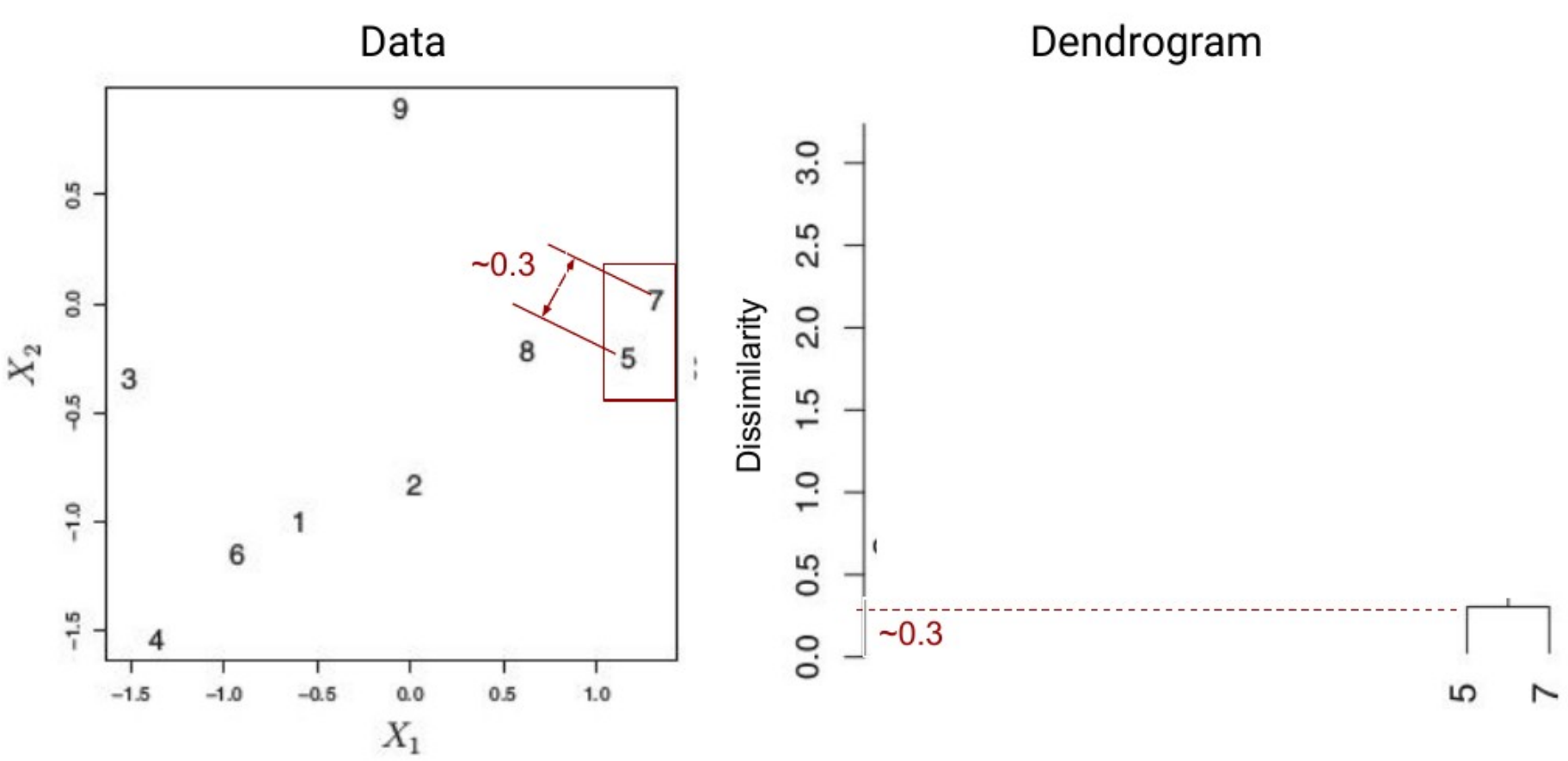
Average and complete are most common

Data



Dendrogram

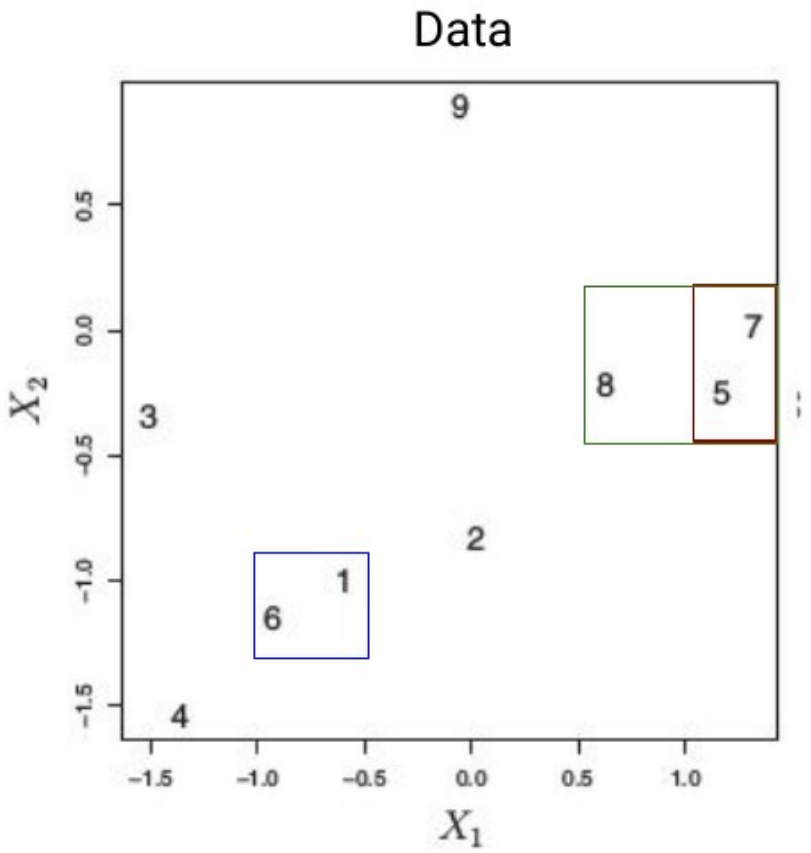


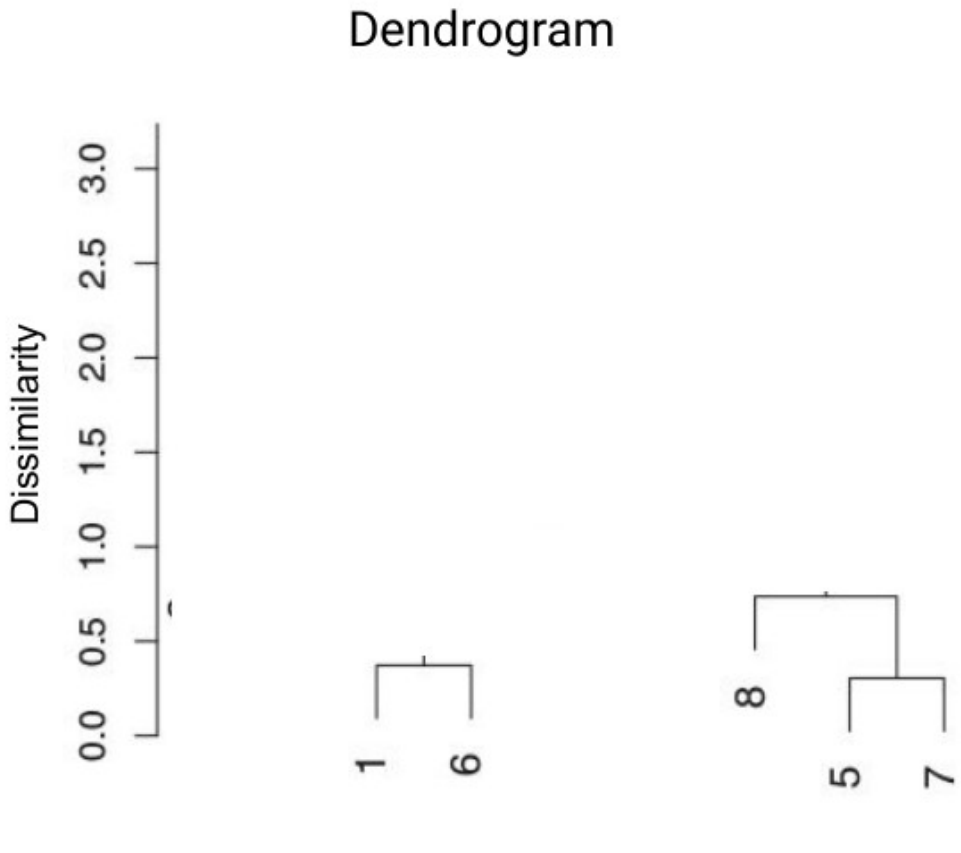

Data
Dendrogram
9
7
8
5
3
2
1
6
4
~0.3
~0.3
X_1
X_2
Dissimilarity
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
7

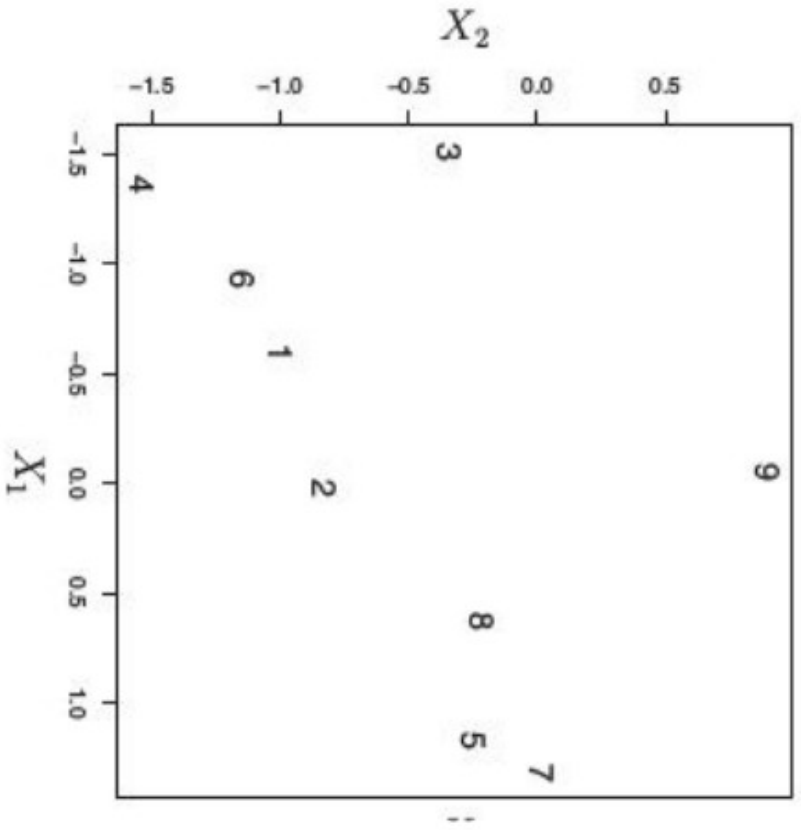
Data
9
7
8
5
3
2
1
6
4
X_2
X_1
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0

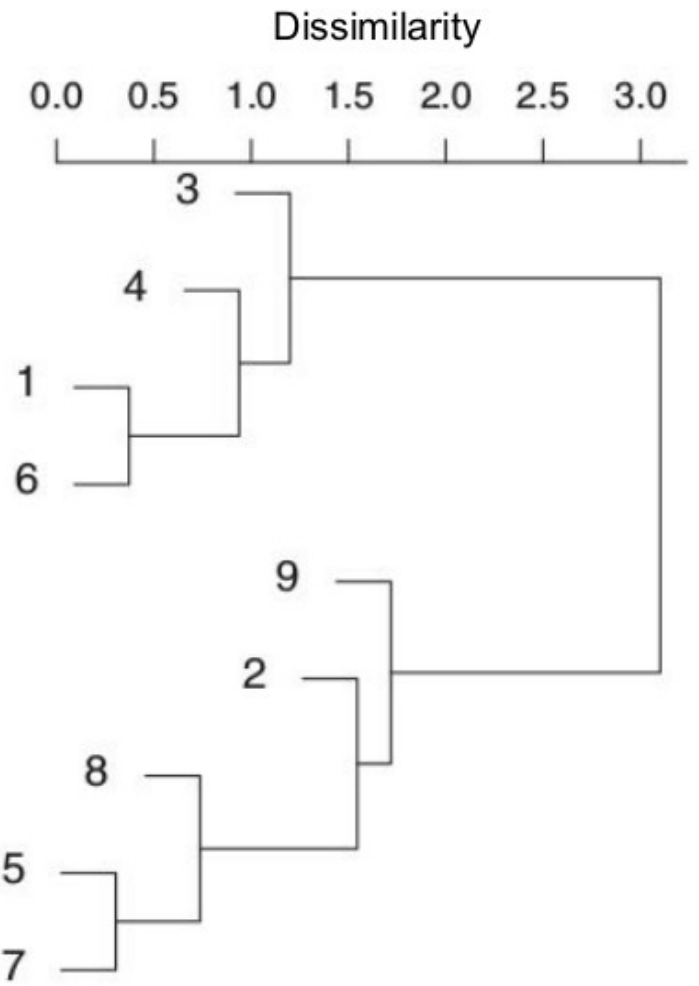
Dendrogram
Dissimilarity
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1
6
5
7

Hierarchical clustering algorithm (visualized)

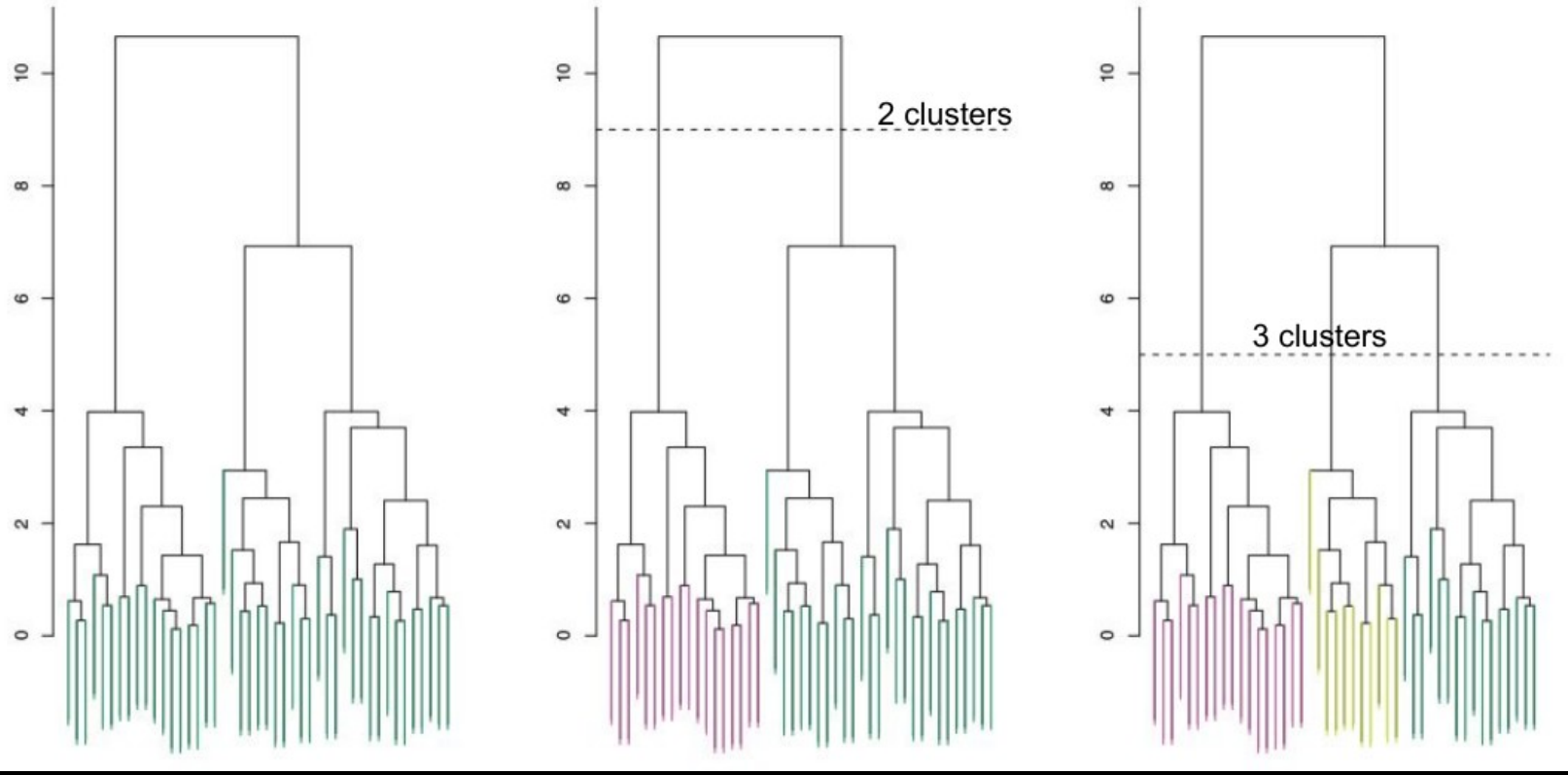





X_2
−1.5
−1.0
−0.5
0.0
0.5
X_1
−1.5
−1.0
−0.5
0.0
0.5
1.0
3
4
6
1
2
9
8
5
7


Dissimilarity
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3
4
1
6
9
2
8
5
7

Selecting *k*



Summary Questions

- When do you choose *k* in each method?
- What methods help pick *k?*
- What is a stopping criteria?
- What is the stopping criteria for hierarchical clustering?
- How do we initialize centroids in k-Means?
- What’s a good cluster?