95-891: Introduction to Artificial Intelligence

Session 4: Machine Learning: Unsupervised Methods

David Steier

steier@andrew.cmu.edu

September 4, 2025

Agenda

- Your project ideas
- Classification
  - Logistic regression
  - Decision trees
- Evaluating classified models
- Unsupervised learning
- *k*-means clustering
- Appendices:
  - A: Hierarchical clustering
  - B: DBSCAN

Classification vs. Regression/Prediction

Regression/Prediction is the task of predicting the value of a continuously varying variable (e.g. a price, a temperature, a conversion rate...) given some input variables

Classification is the task of predicting the value of a categorical variable (e.g. if an email is or is not a spam message) given some input variables

Logistic Regression as Classification

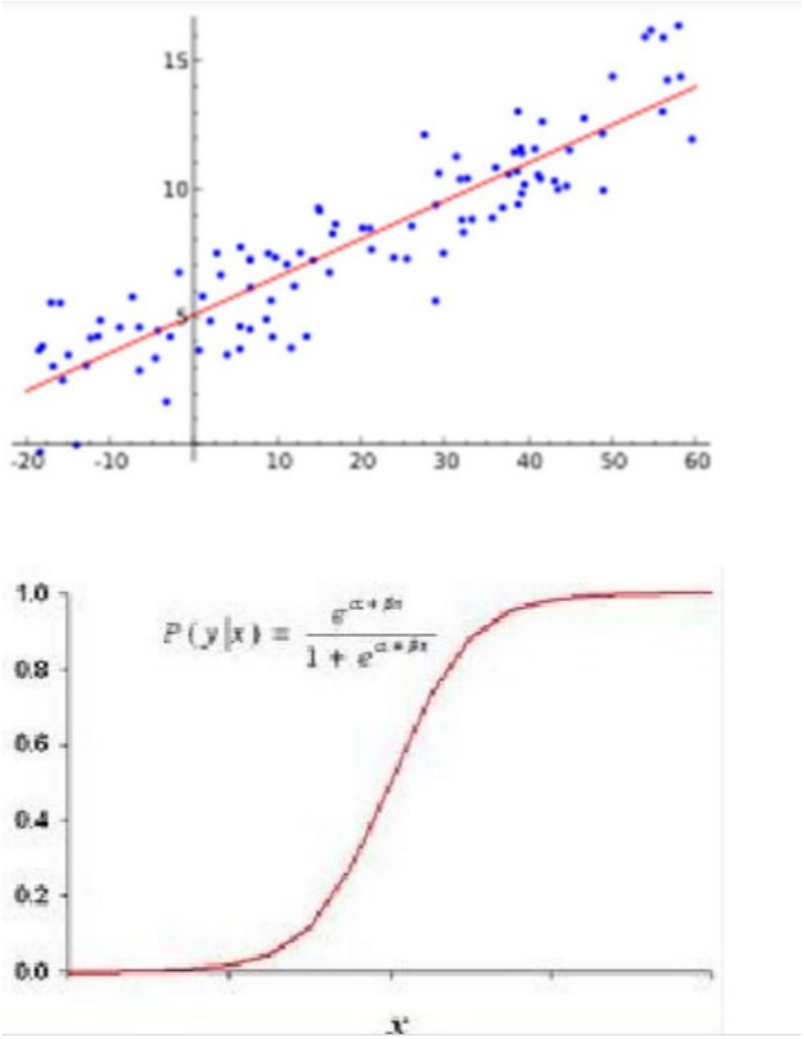
- Classification: Which of a set of categories (sub-populations) does an observation belong to?
  - A given email into "spam" or "non-spam" classes.
  - A customer will “purchase” or “not purchase”
- Binary Logistic Regression, or **logit** regression is used to predict a binary response based on one or more predictor (or independent) variables (features), The logit model solves these problems:

$$\ln[p/(1-p)] = \beta_0 + \beta_1 X + e$$

- Where
  - p is the probability that the event Y occurs, p(Y=1)
  - p/(1-p) is the "odds ratio", and
  - ln[p/(1-p)] is the log odds ratio, or logit

Linear vs. Logistic Regression

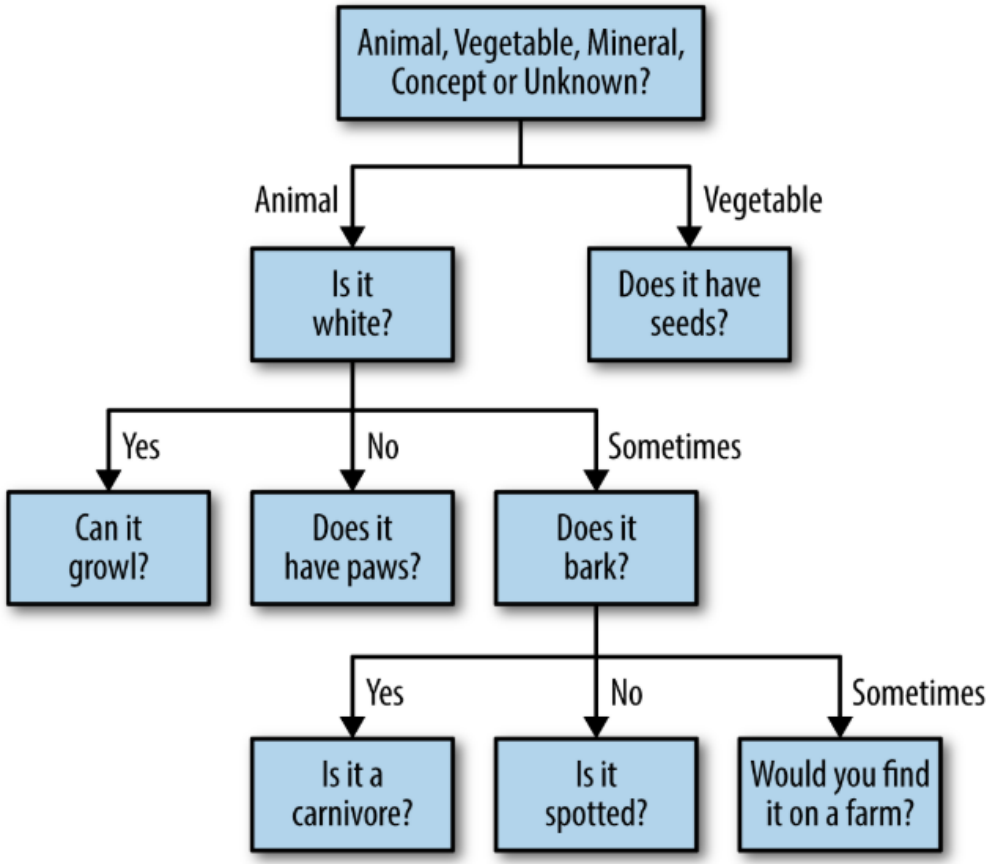
- In linear regression, we are predicting a continuous quantity Y, which varies in proportion to X
- In logistic regression, we are predicting the likelihood that Y=1 (rather than 0) given certain values of Xs.
- If X and Y have a positive linear relationship it indicates the probability that Y=1 will increase as value of X increases



P. Wu, et. al Comparison of linear and logistic regression for segmentation , 2014
https://www.casact.org/sites/default/files/presentation/rpm_2014_handouts_paper_2693_handout_1986_0.pdf

Decision Trees

- A **decision tree** uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes.
- Decision tree as a predictive model maps observations about an item to conclusions about the item's target value.



Hebron, P. Machine Learning for Designers, https://www.oreilly.com/library/view/machine-learning-for/9781491971444/#toc-start

Building Decision Trees

- Tree is grown from the root node, with each iteration splitting the data into two further sub-segments
- The algorithm stops when threshold conditions are met (e.g., exposure within the node becomes too small or maximum depth of tree has been reached)
- Pruning removes branches that make use of features having low importance (as judged by impact on accuracy)

Decision Tree in Python using Scikit-learn (1/3)

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample Data (replace with your actual data)
data = {
    'Feature1': [10, 20, 15, 25, 30, 12, 22, 18],
    'Feature2': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'B'],
    'Target': [0, 1, 0, 1, 1, 0, 1, 0]
}
df = pd.DataFrame(data)
# Encode categorical features
df['Feature2'] = df['Feature2'].astype('category').cat.codes
X = df[['Feature1', 'Feature2']]
y = df['Target']
```

Decision Tree in Python using Scikit-learn (2/3)

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the classifier
dt_classifier = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_classifier.fit(X_train, y_train)

# Make predictions
y_pred = dt_classifier.predict(X_test)

# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

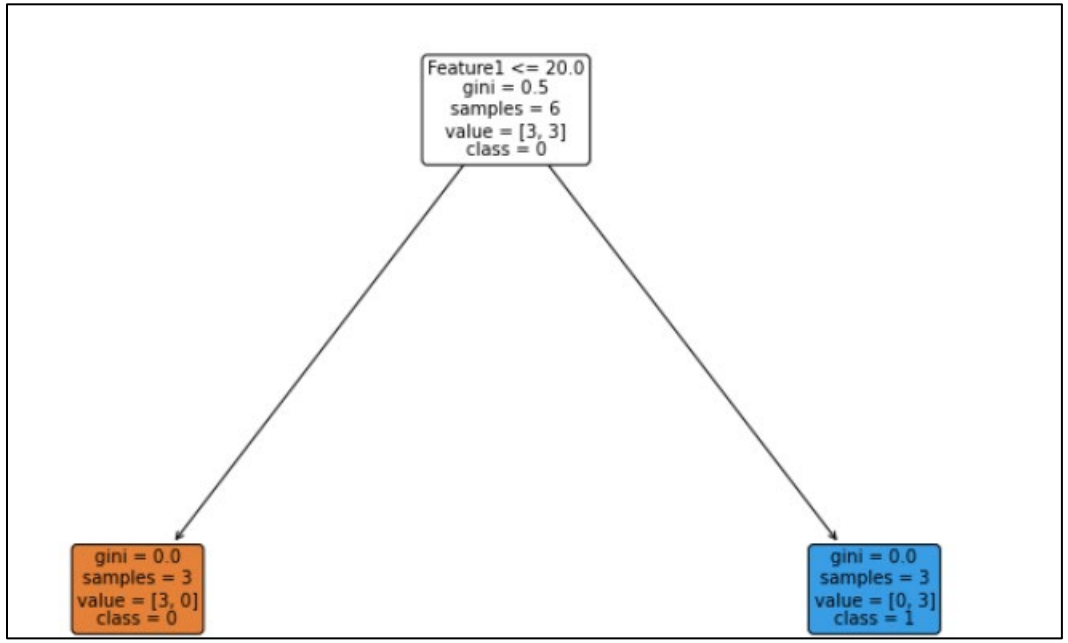
Accuracy: 0.5

Decision Tree in Python using Scikit-learn (3/3)

```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
feature_names = ['Feature1', 'Feature2_encoded']
class_names = [str(name) for name in dt_classifier.classes_]
# Create a figure and plot the tree
plt.figure(figsize=(15, 10))
plot_tree(
    dt_classifier,
    feature_names=feature_names,
    class_names=class_names,
    filled=True,
    rounded=True,
    fontsize=10
)
plt.show()
```

If Feature1 <= 20 → predict class 0
If Feature1 > 20 → predict class 1

Rule
Gini impurity
Samples
Value
Class



Gini Impurity Measures How Mixed Classes Are

For a node with K classes, let p_i be the proportion of samples in the node that belong to class i.

$$\text{Gini impurity} = 1 - \sum_{i=1}^{K} p_i^2$$

- If **all samples belong to one class** → $p_i = 1$ for some i, others 0 → Gini = 0 (pure node).
- If **samples are evenly split between classes** → e.g. binary case $p_0 = 0.5, p_1 = 0.5$:

$$1 - (0.5^2 + 0.5^2) = 1 - 0.5 = 0.5$$

That's the *maximum impurity* for 2 classes.

- If **3 classes evenly split** → $p_i = 1/3$:

$$1 - 3 \times (1/3)^2 = 1 - 1/3 = 0.667$$

Choosing the Best Split Using Information Gain

- Choose the attribute that yields the highest **information gain** in dividing the remaining observations
  - Yielding subsets of observations separated by outcome variable value that are least likely to be derived by chance
- Information gain defined in terms of **entropy** (uncertainty of a random variable measured in bits). For *n* outcomes, each with likelihood p_i

$$Entropy = -\sum_{i=1}^{n} p_i \log_2 p_i$$

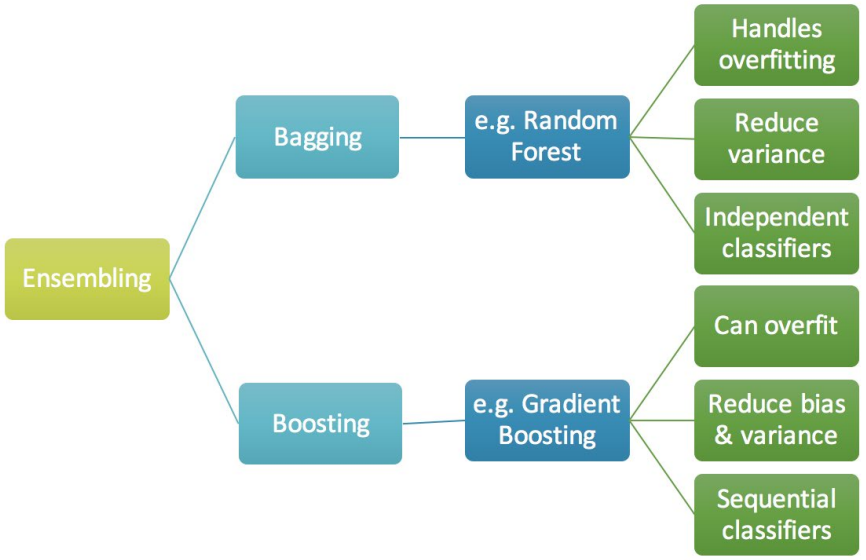
- Entropy is
  - 1 bit for a coin flip (two equally likely outcomes)
  - 2 bits for four-sided die (four equally likely outcomes)
  - .08 bits for a loaded coin that yields heads 99% of the time
- Split the decision using the attribute that yields the lowest weighted average entropy of the child nodes

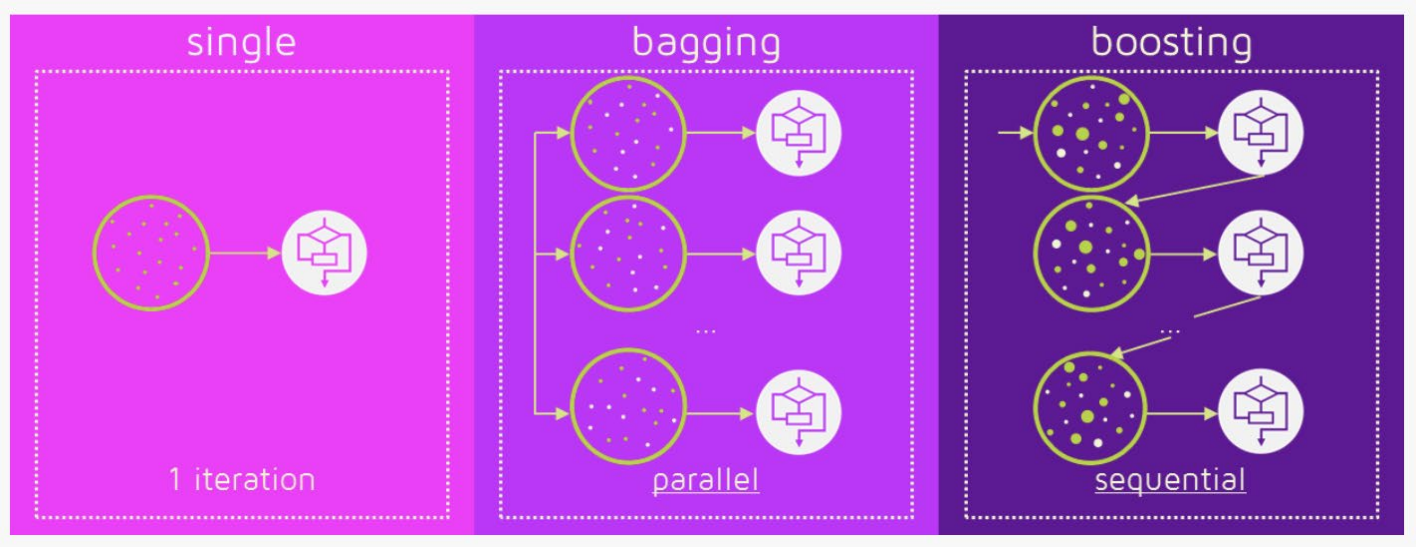
Decision Trees: When to Use Them?

- When many types of input variables are present: Binary, categorical, ordinal, continuous
- When large data sets are available to cover the space
- Where relationships are not linear

Ensemble Methods

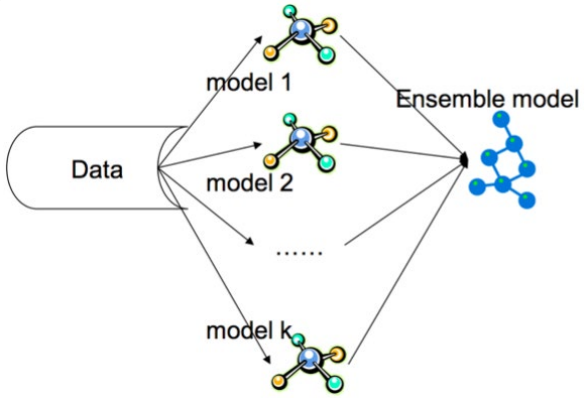
- Uses multiple models at a time to combine strengths of each
- Can reduce bias and overfitting, but may reduce interpretability
- Two ways of “ensembling”: Bagging and Boosting





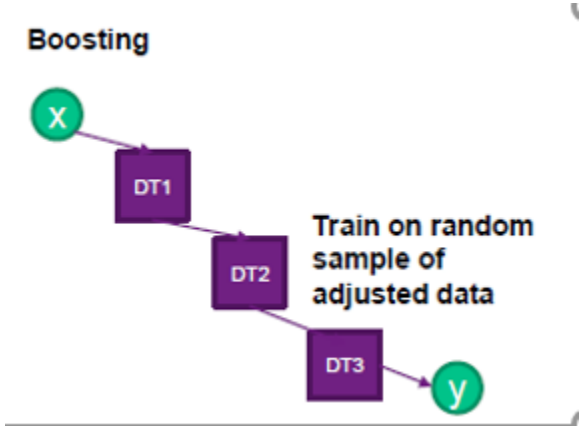
Bagged Decision Trees and Random Forests

- Decision trees are sensitive to order of training examples and irrelevant factors
- Multiple decision trees can be combined in an **ensemble** using **bagging** (short for bootstrap aggregation)
  - Prediction from each tree averaged together, weighted by tree accuracy
- **Random forests** are bagged decision trees trained by
  - Random subsets of data, to reduce variance
  - Random subsets of features, to prune irrelevant factors



Other Supervised Learning Models

- **Support vector machines**: Find hyperplane separated points in high-dimensional space
- **Boosting**:
  - Each model trained on the results of previous model
  - More weight is given in subsequent models to observations that were not classified correctly previously
  - **Gradient boosting** adds estimator that fits to the residual error of previous model
  - Example: **Gradient-tree boosting** works on decision trees.



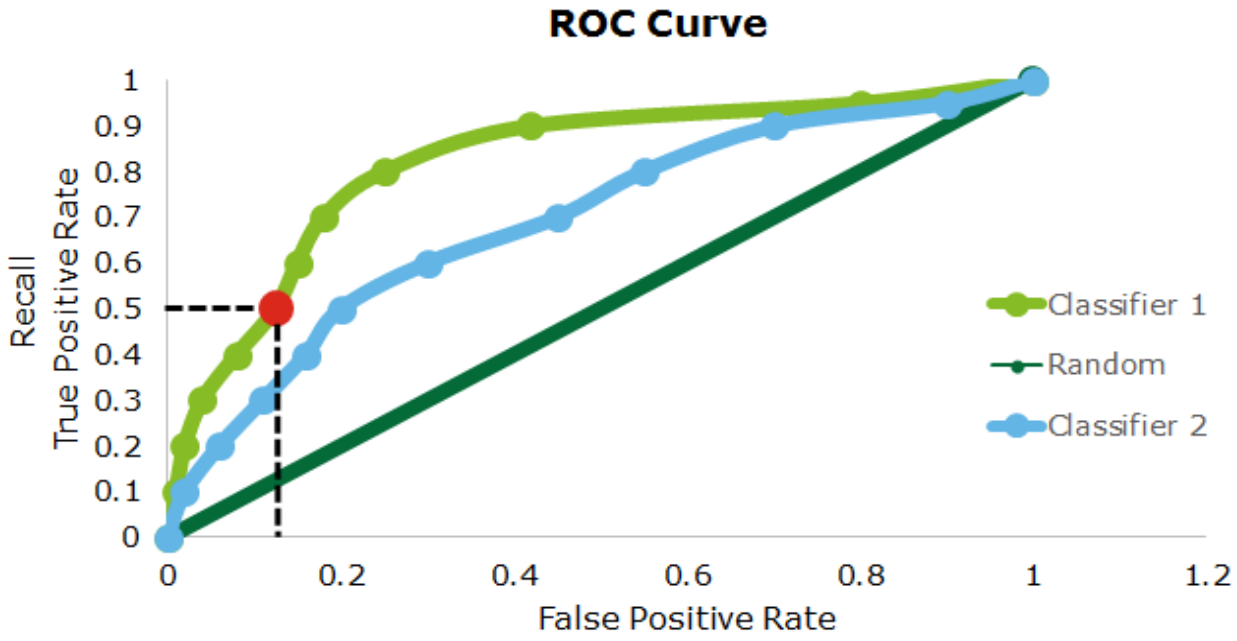
Evaluating Classification Performance

Confusion Matrix	Prediction = NEGATIVE	Prediction = POSITIVE
Ground Truth = NEGATIVE	True Negatives	False Positives
Ground Truth = POSITIVE	False Negatives	True Positives

- **Accuracy** = (TP+TN) / (TP+FP+TN+FN)
  - Fraction of the time the classifier is correct
- **Precision** = TP / (TP+FP)
  - Fraction of the time positive indications are correct
- **Recall** = **Sensitivity** = TP / (TP+FN)
  - Fraction of the time items in the class are detected
- **Specificity** = TN / (TN+FP)
  - Fraction of the time non-class items are correct

Receiver Operating Characteristic (ROC) Curves

- ROC curves evaluates model performance across the entire range of cutoffs
  - Axes are how
    - X: How often doe the model generate a false positive?
    - Y: What fraction of the true positives are recalled?
  - A confusion matrix is a single point on the curve, with a specific cutoff value
  - Each point on the ROC curve represents the model's performance at one cutoff value
- Compare models by comparing the area under the curve (AUC, higher is better)

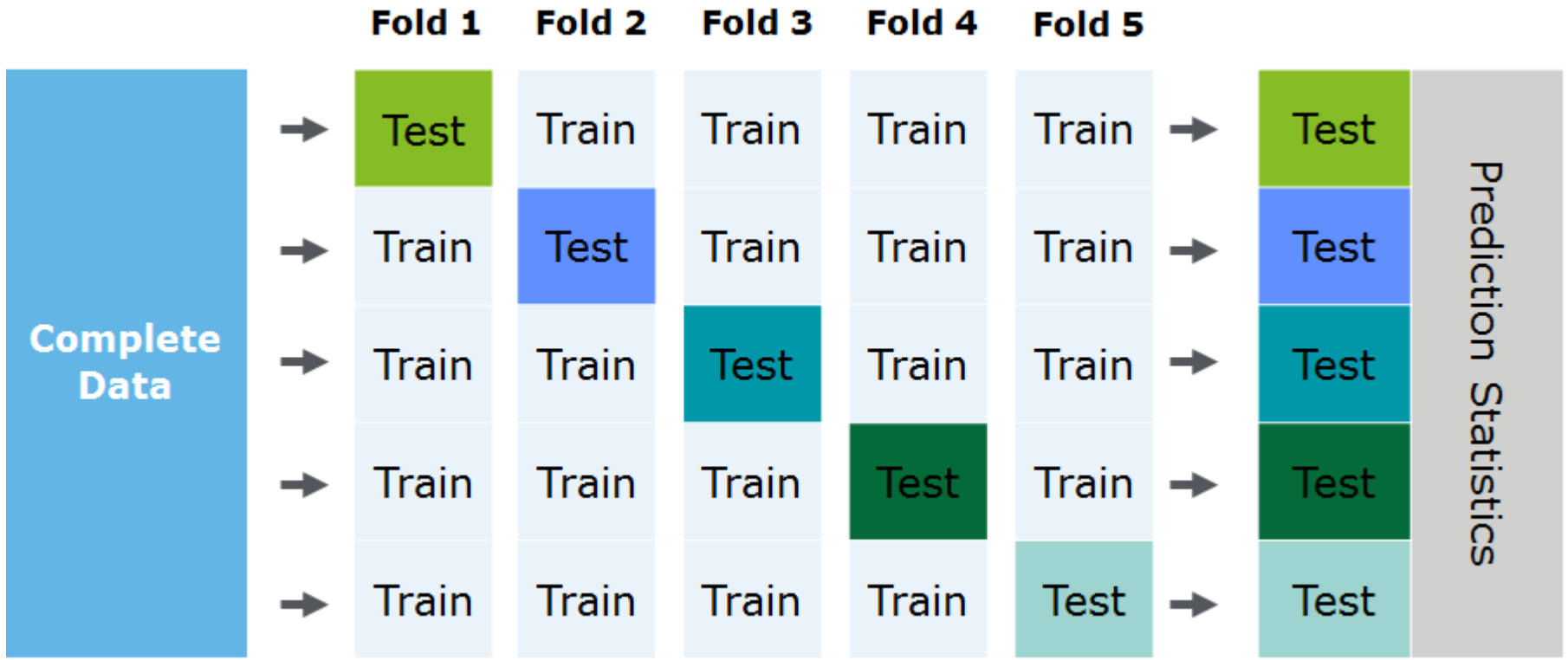


Cross-Validation for Evaluating Accuracy

- Evaluating a model based on just the data it was trained on (the *in-sample* estimate) will overestimate the goodness of fit.
- We need to evaluate models using validation data that was not used for training (to get an *out-of-sample* estimate of fit).
- Could just *hold out* an arbitrary sample for validation, but the resulting out-of-sample estimate may not be representative
- *Cross-validation* repeatedly splits the input data into training and test sets and trains and tests multiple models to get a better estimate of fit

K-Fold Cross-Validation

- Cross validation uses all the data to ensure that model measurements were not biased by a particularly lucky sample
- In 5-fold cross validation, one-fifth of the data is used for a testing set in each fold.



Supervised vs. Unsupervised Learning

Supervised Learning

- Modelling the dependency of an output (dependent or *target*) variable based on various independent input variables
- **Examples**
  - Logistic regression
  - Decision trees
  - Support Vector Machines

Unsupervised Learning

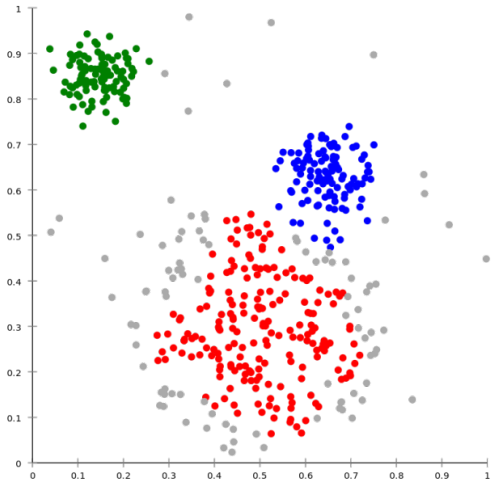
- Identification of patterns and regularities in the data such as groups of entities with similar characteristics or typical correlations without a target variable
- **Examples**
  - Clustering
  - Association rules
  - Dimensionality reduction

Clustering: What Is It?

Clustering: the process of finding a "natural" partition of a dataset

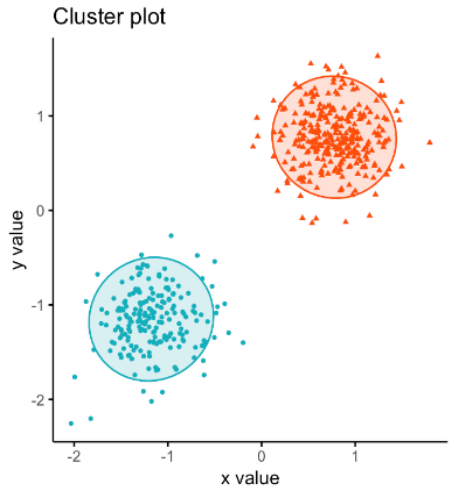
- A commonly-used form of *unsupervised* learning
- There is no single notion of a cluster - there are many clustering algorithms
- Clustering is usually an iterative, judgment-intensive activity

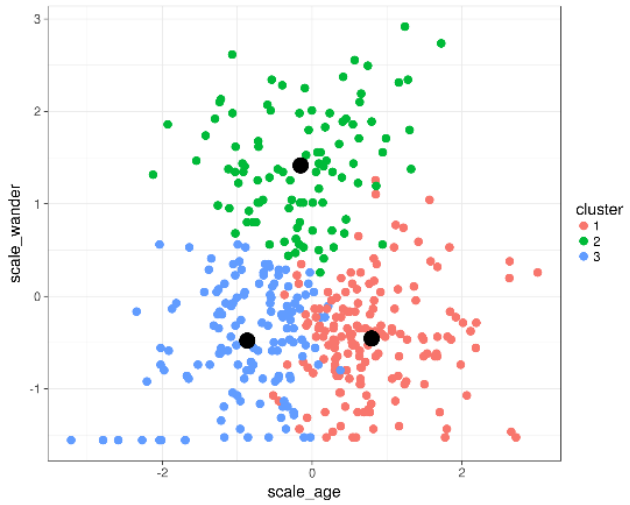
  - A clustering algorithm partitions a dataset into "natural" groups based on a set of variables presented for consideration.
  - Each variable corresponds to a dimension of the data-space that will be partitioned
  - "Distance" in this data space is interpreted as "similarity"
  - Clusters are derived in such a way that items in one cluster are similar to one another; dissimilar from items in other clusters.

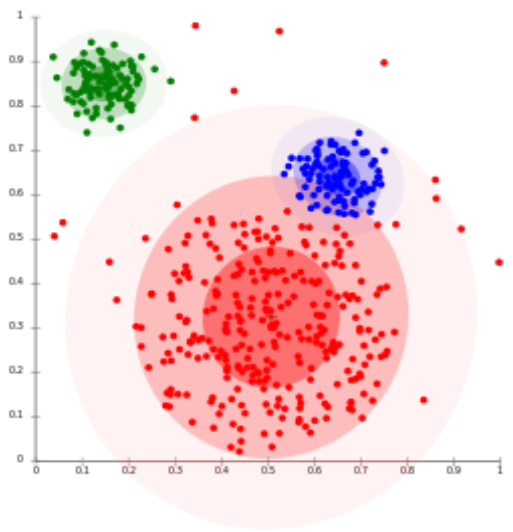


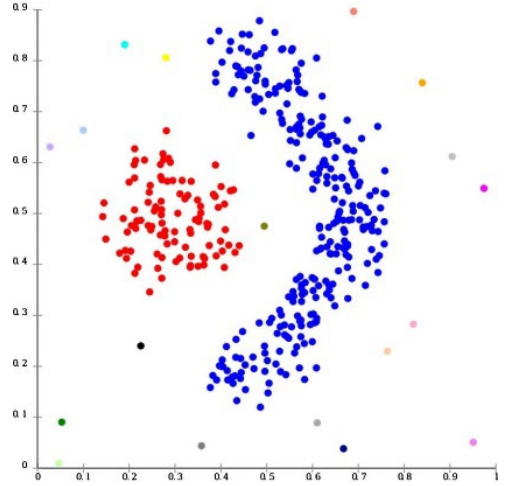
Cluster Properties

- Clusters may have different sizes, shapes, densities
- Clusters may form a hierarchy
- Clusters may be overlapping or disjoint









Clustering Applications

Methods

- K-means
- Hierarchical
- DBScan

Examples

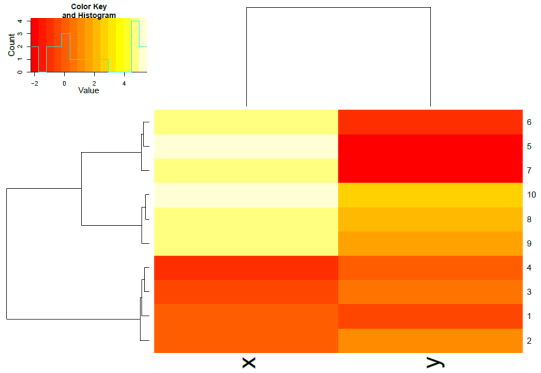
- Customer segmentation
- Molecule search
- Anomaly detection

- Find “natural” clusters and desc
  - Data understanding
- Find useful and suitable groups
  - Data Class Identification
- Find representatives for homogenous groups
  - Data Reduction
- Find unusual data objects
  - Outlier Detection
- Find random perturbations of the data
  - Noise Detection

Types of Clustering Approaches

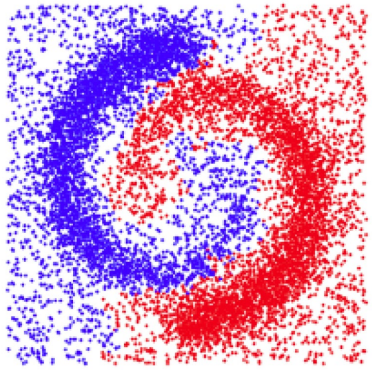
Linkage Based

e.g. Hierarchical Clustering



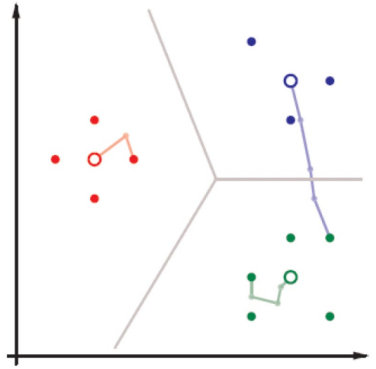
Density based Clustering

e.g. DBSCAN



Clustering by Partitioning

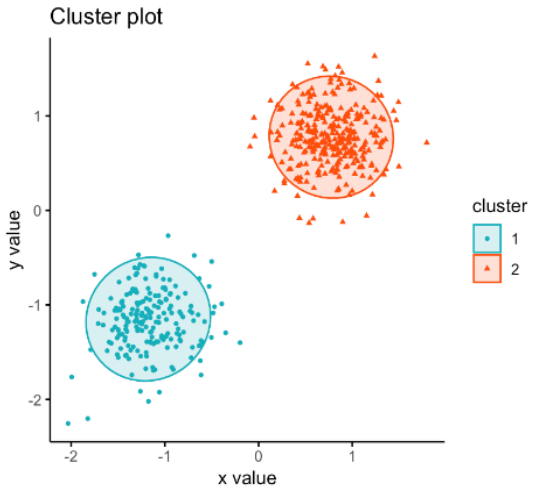
e.g. k-Means



Clustering by Partitioning

Goal:

A (disjoint) partitioning into k clusters with minimal costs



- Local optimization method:
  - choose k initial cluster representatives
  - optimize these representatives iteratively
  - assign each object to its most similar cluster representative
- Types of cluster representatives:
  - Mean of a cluster (*construction of central points*)
  - Median of a cluster (*selection of representative points*)
  - Probability density function of a cluster (*expectation maximization*)

k-Means Clustering

Partition *n* observations into *k* clusters.
3 steps:

1. **Initialization** – *k* initial "means" (centroids) are generated at random
2. **Assignment** – *k* clusters are created by associating each observation with the nearest centroid
3. **Update** – The centroid of each clusters becomes its new mean

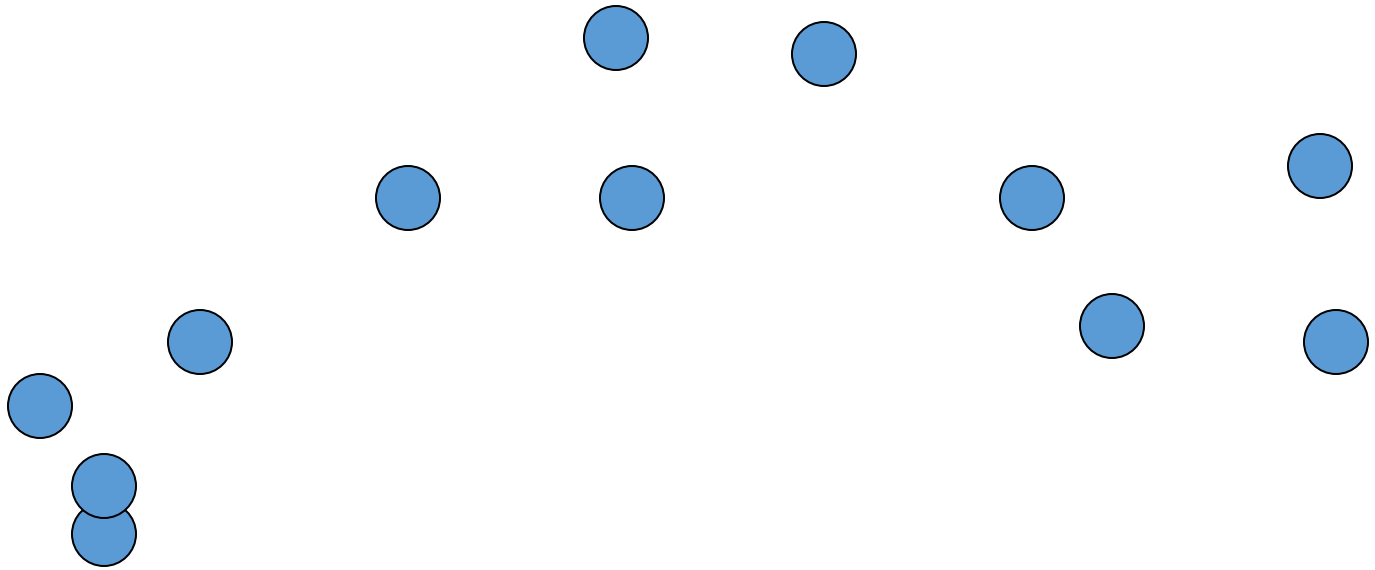
Assignment and Update are repeated iteratively until convergence

The end result is that the sum of squared errors is minimized between points and their respective centroids

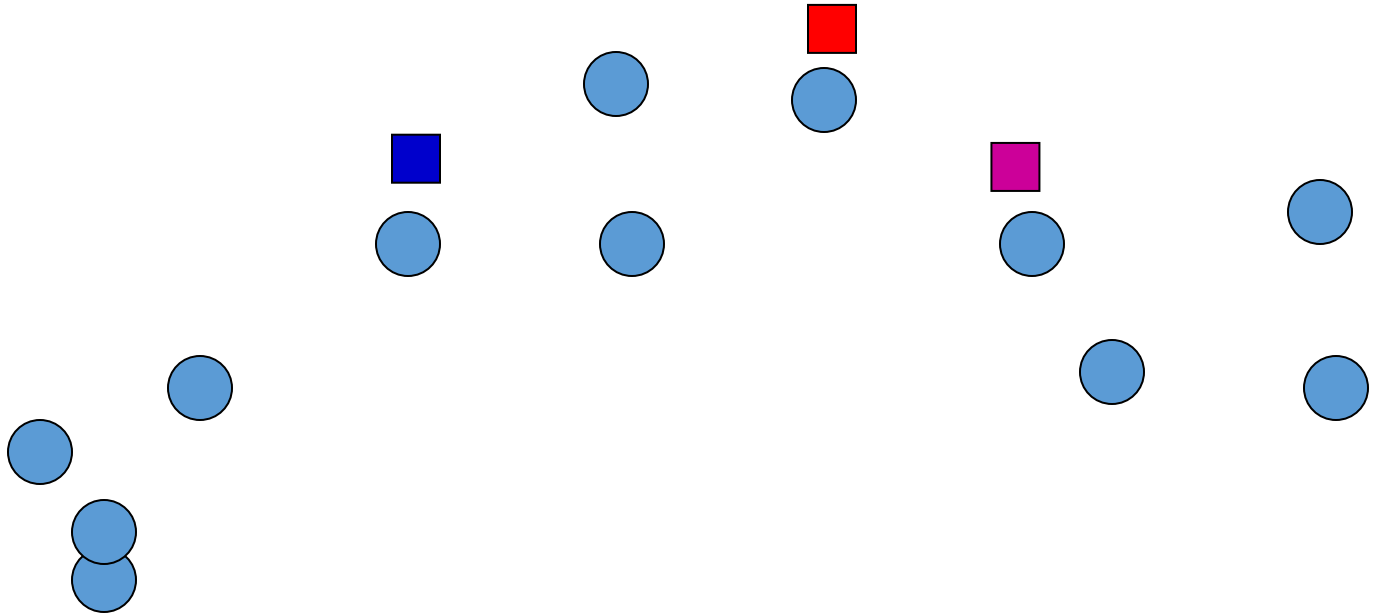
http://benalexkeen.com/k-means-clustering-in-python/

k-Means: An Example

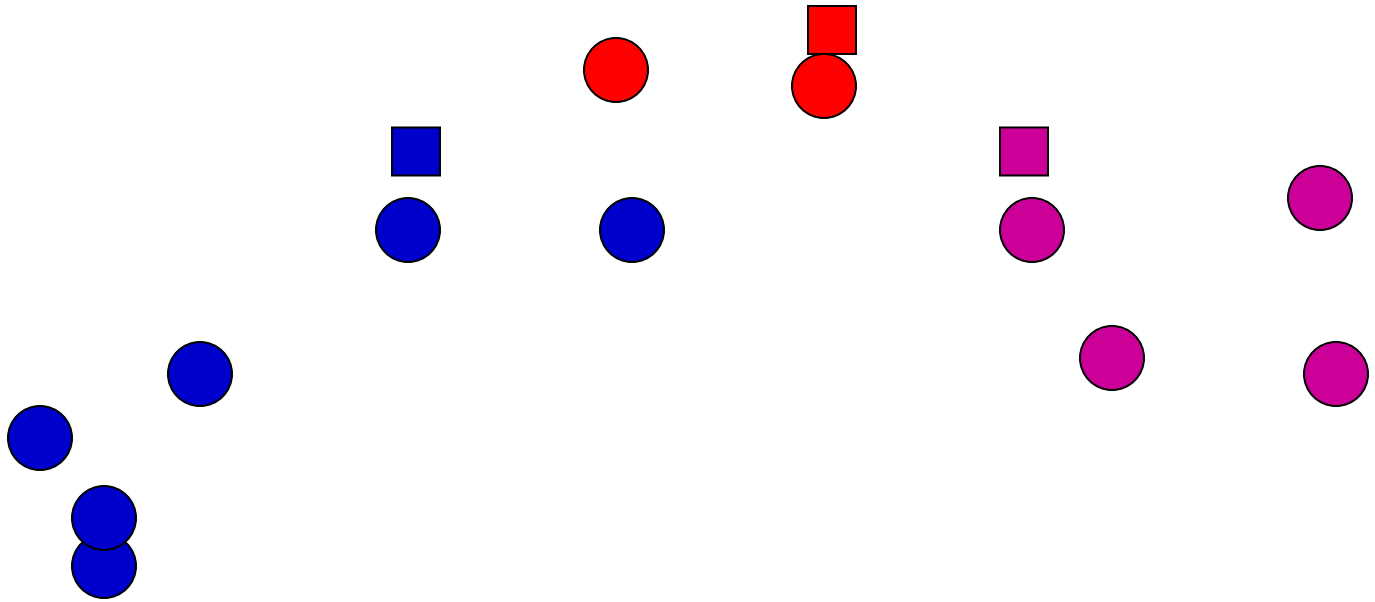
From www.cs.pomona.edu/~dkauchak/classes/f13/cs451-f13/.../lecture31-kmeans.pptx



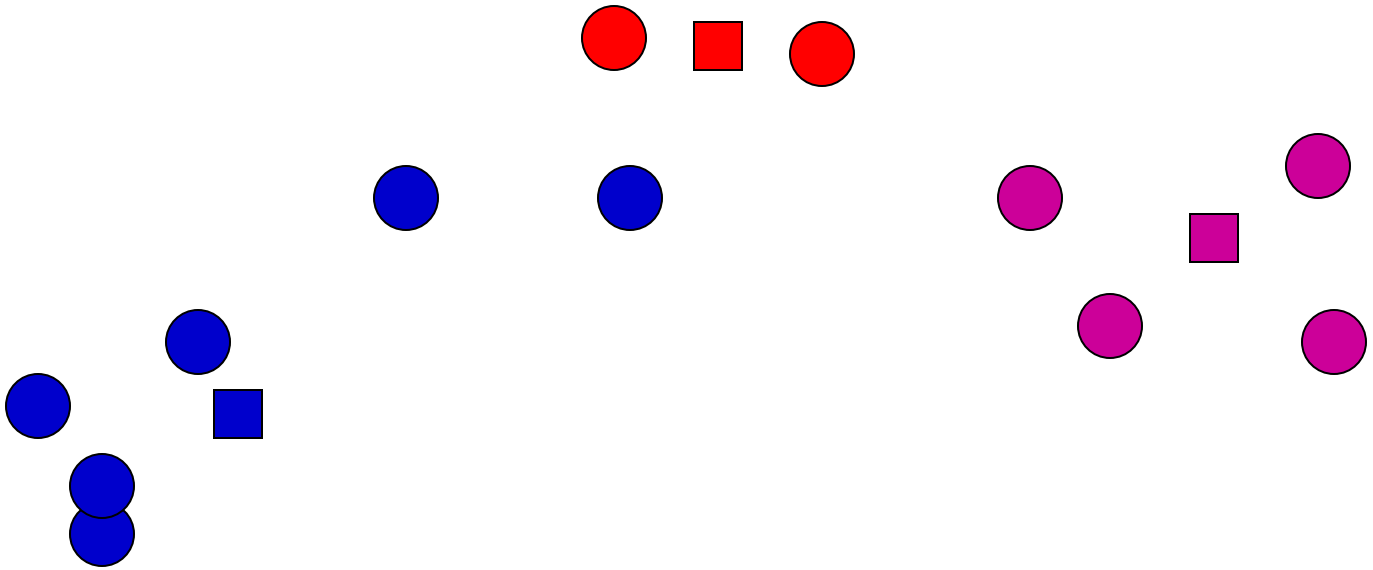
k-Means: Initialize Centers Randomly



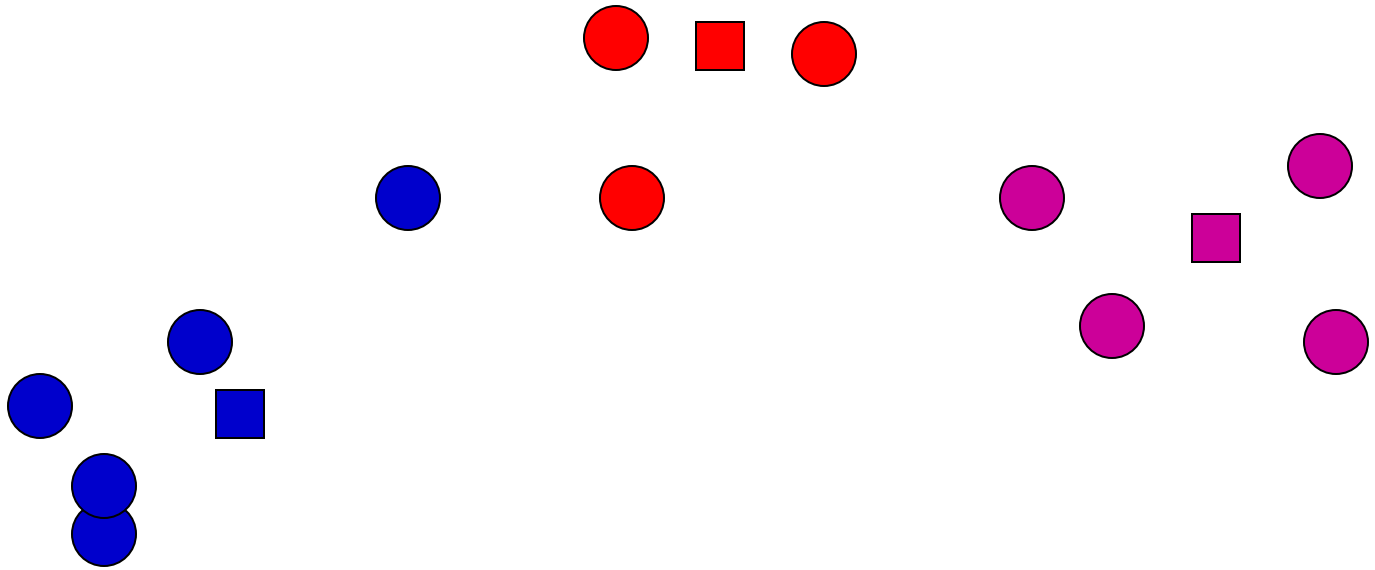
k-Means: Assign Points to Nearest Center



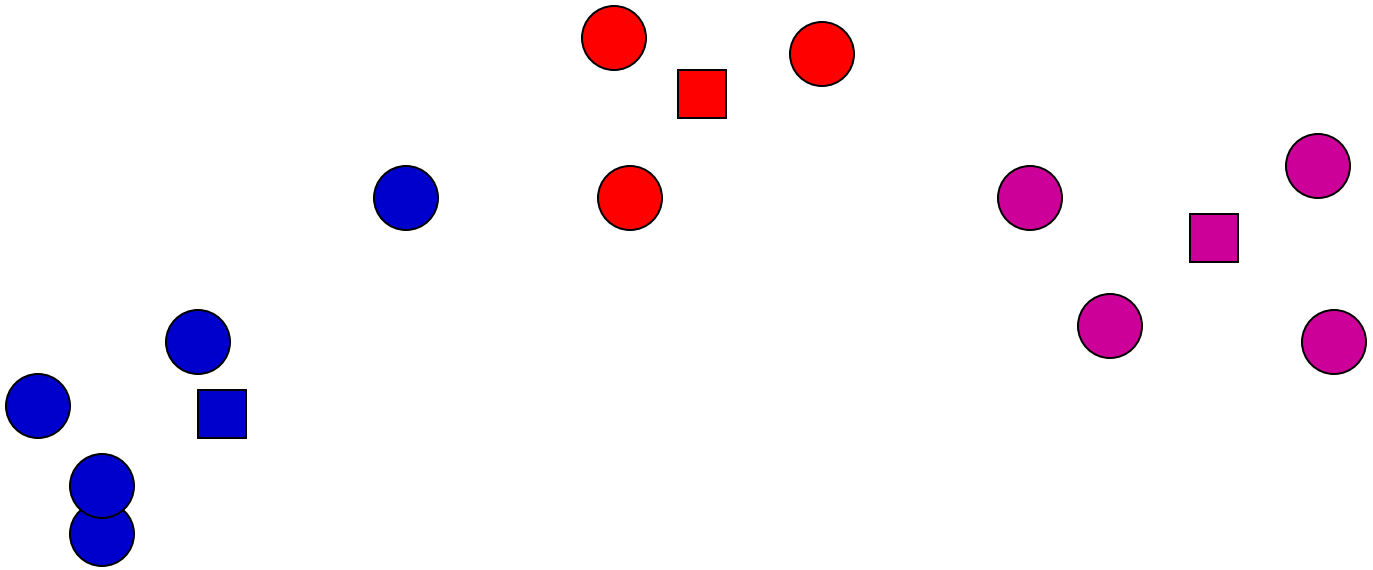
k-Means: Update Centers



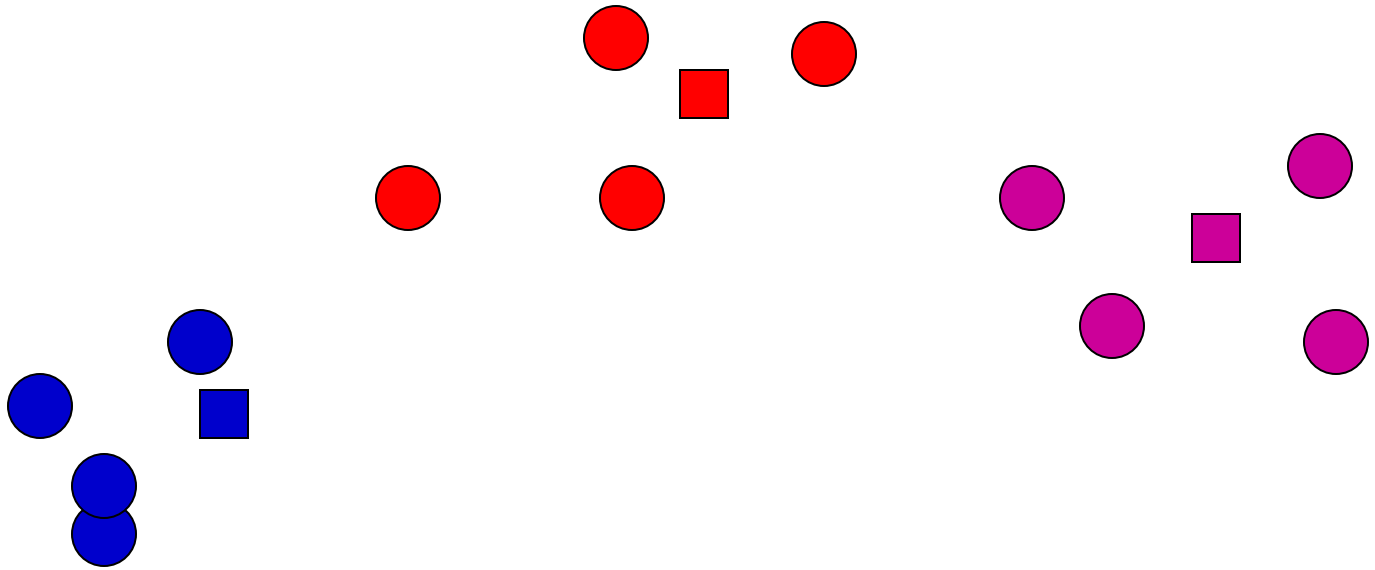
k-Means: Assign Points to Nearest Center



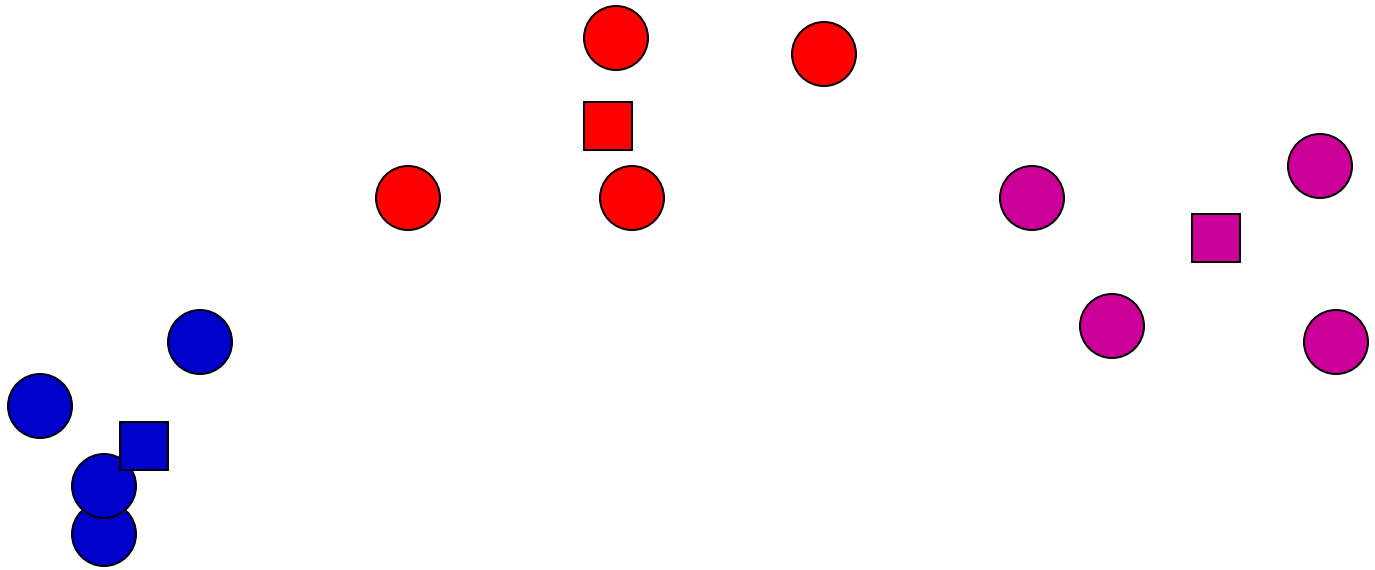
k-Means: Update Centers



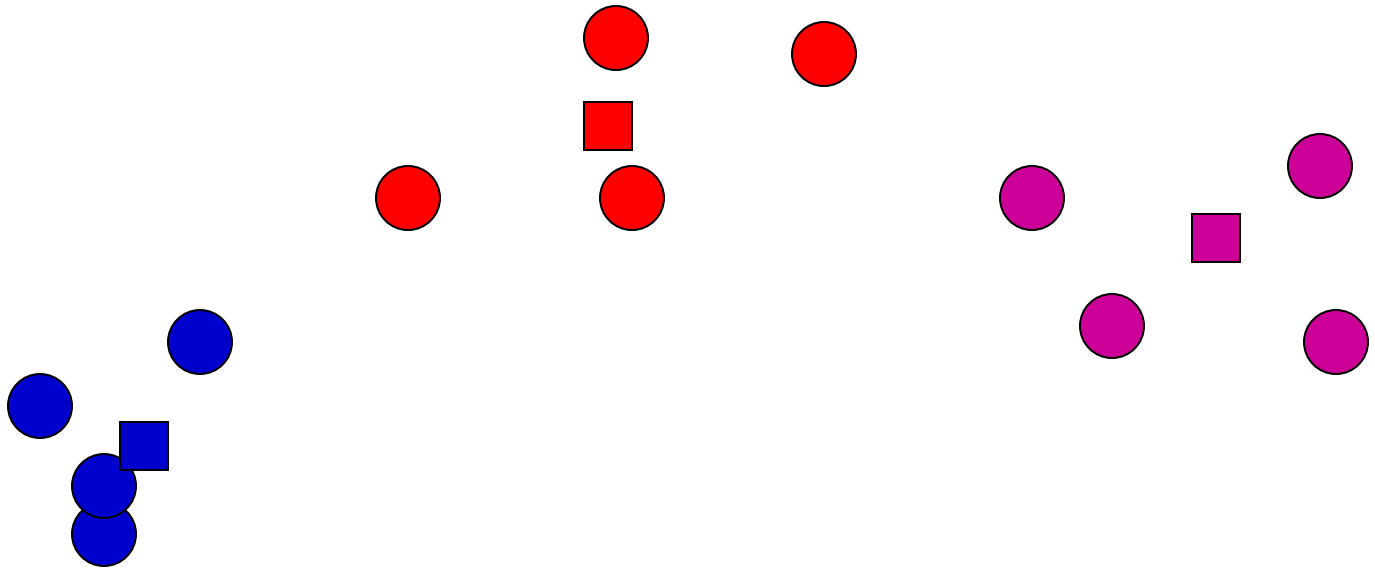
k-Means: Assign Points to Nearest Center



k-Means: Update Centers



k-Means: Assign Points to Nearest Center



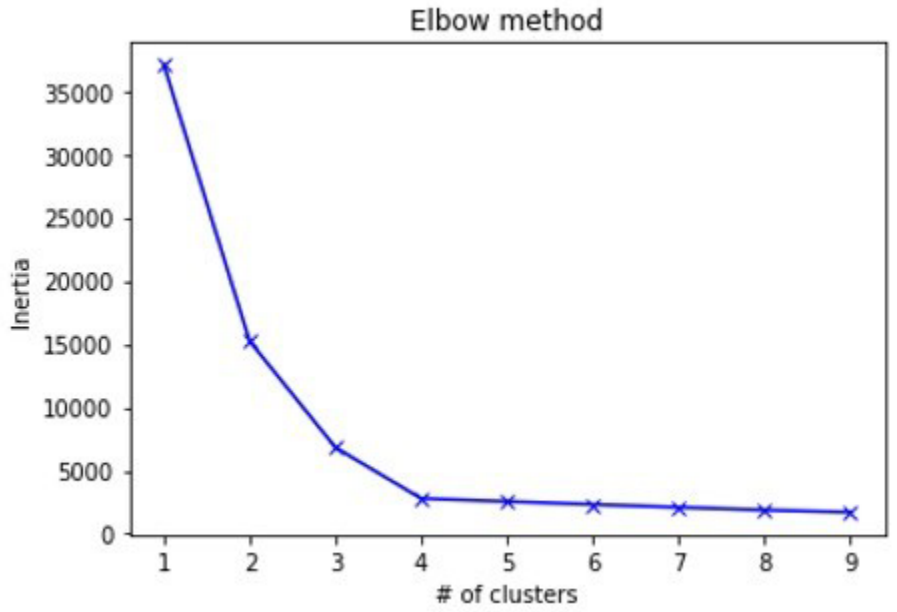
No changes: Done

k-Means Summary

- **Advantages:**
  - Relatively efficient
  - Simple implementation
- **Weaknesses:**
  - Often terminates at a local optimum
  - Applicable only when mean is defined (what about categorical data?)
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

Elbow Method for Choosing *k*

- Inertia is the average distance between the points in the cluster and a centroid

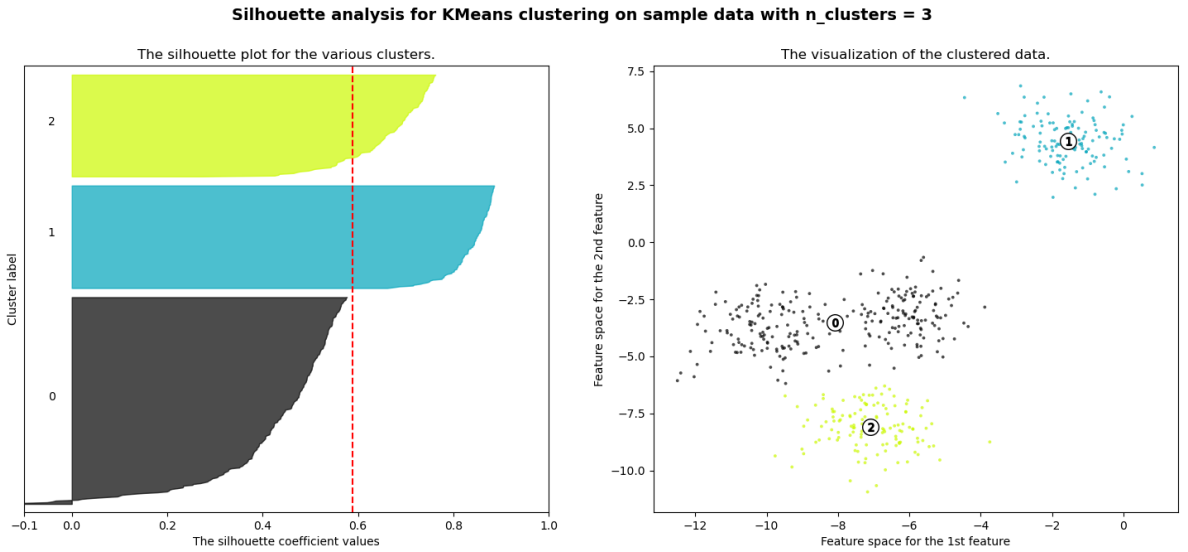


M. Eslamijam, "Customer segmentation: How machine learning makes marketing smart," December 28, 2020, https://bdtechtalks.com/2020/12/28/machine-learning-customer-segmentation/

Silhouette Method for Choosing k (k=3)

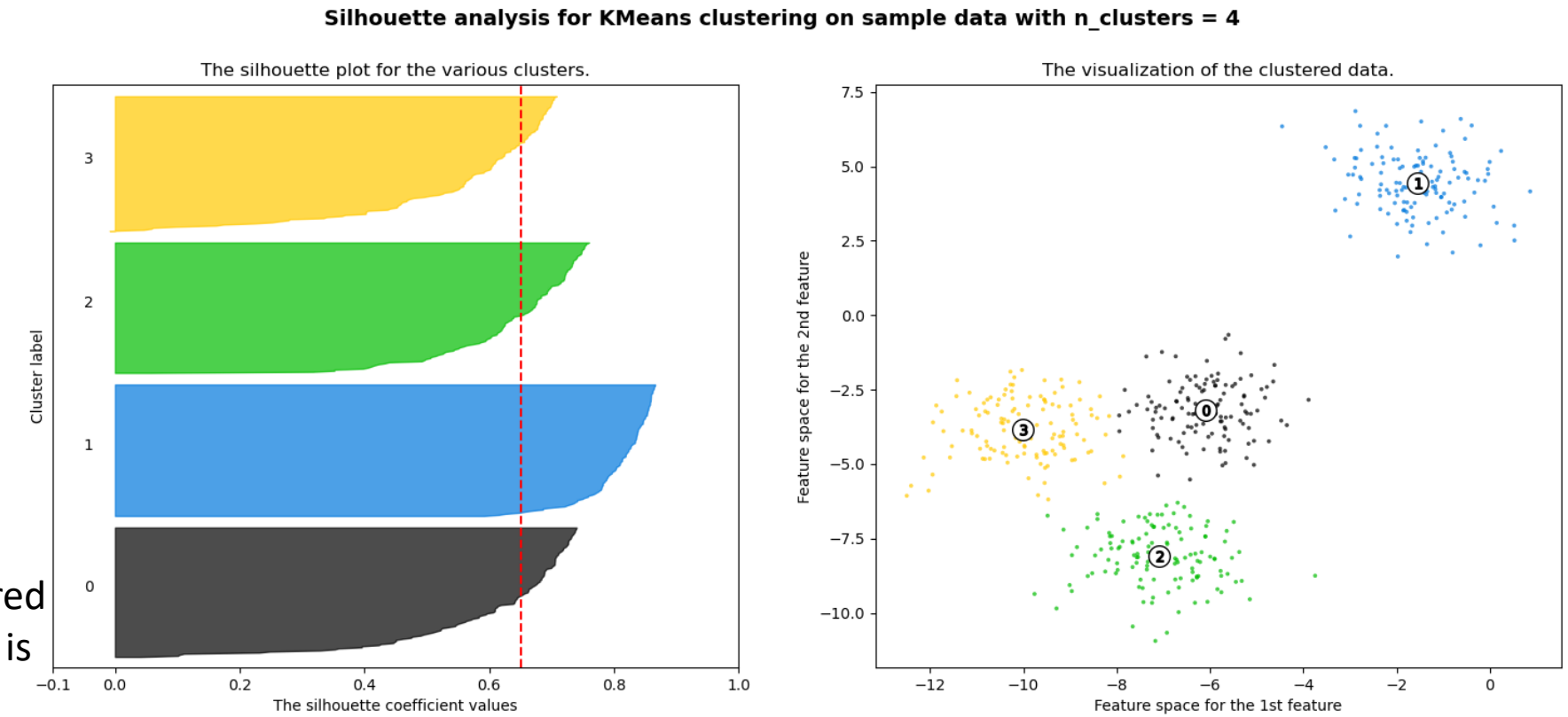
- Silhouette coefficients measure separation of points from other clusters +1 far away, 0 on the decision boundary, -1 might be in wrong cluster

For k=3, avg silhouette coefficient (red dashed line) is **.588**



Sci-kit learn documentation, "Selecting the number of clusters with silhouette analysis on KMeans clustering", https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Silhouette Method for Choosing k (k=4)

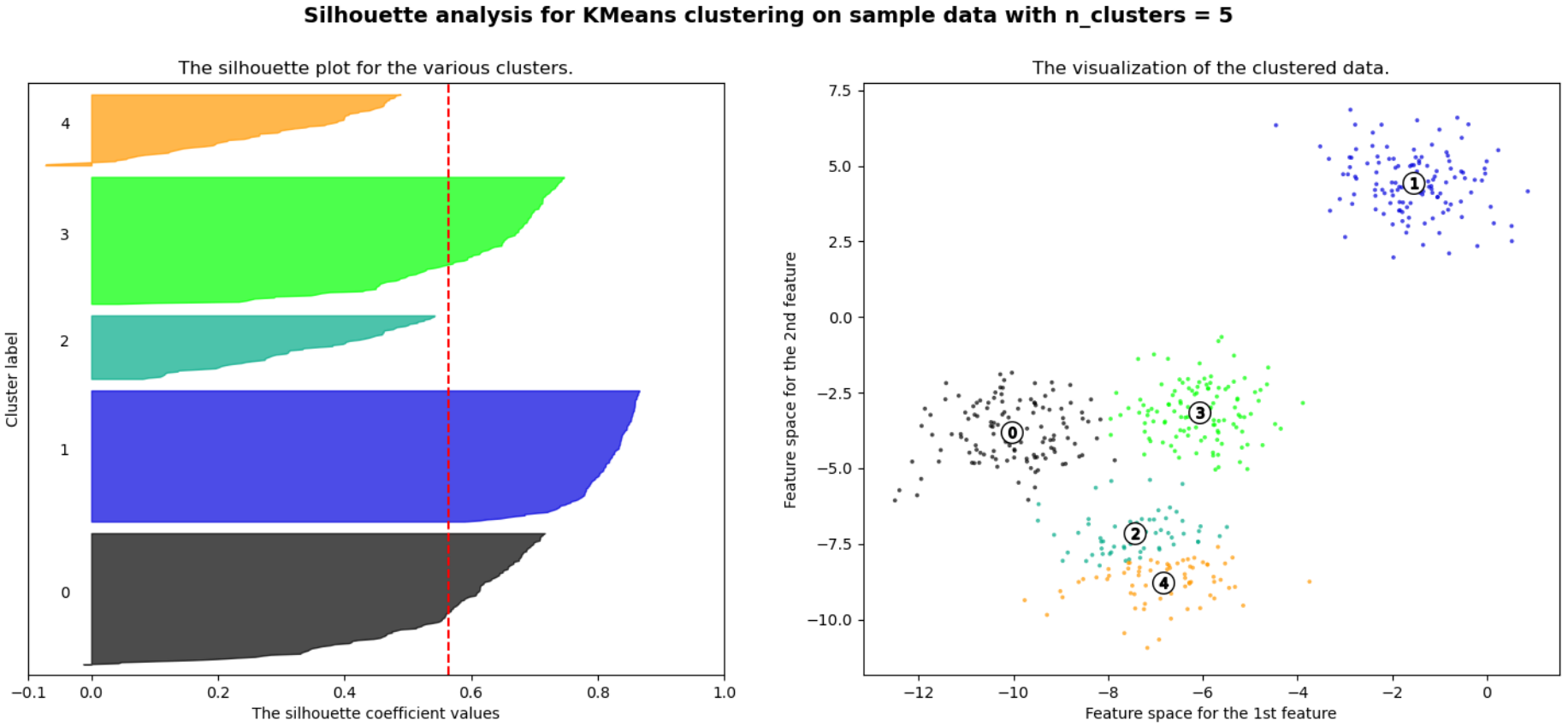


For k=4, avg silhouette coefficient (red dashed line) is **.651**

Sci-kit learn documentation, “Selecting the number of clusters with silhouette analysis on KMeans clustering”, https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Silhouette Method for Choosing k (k=5)

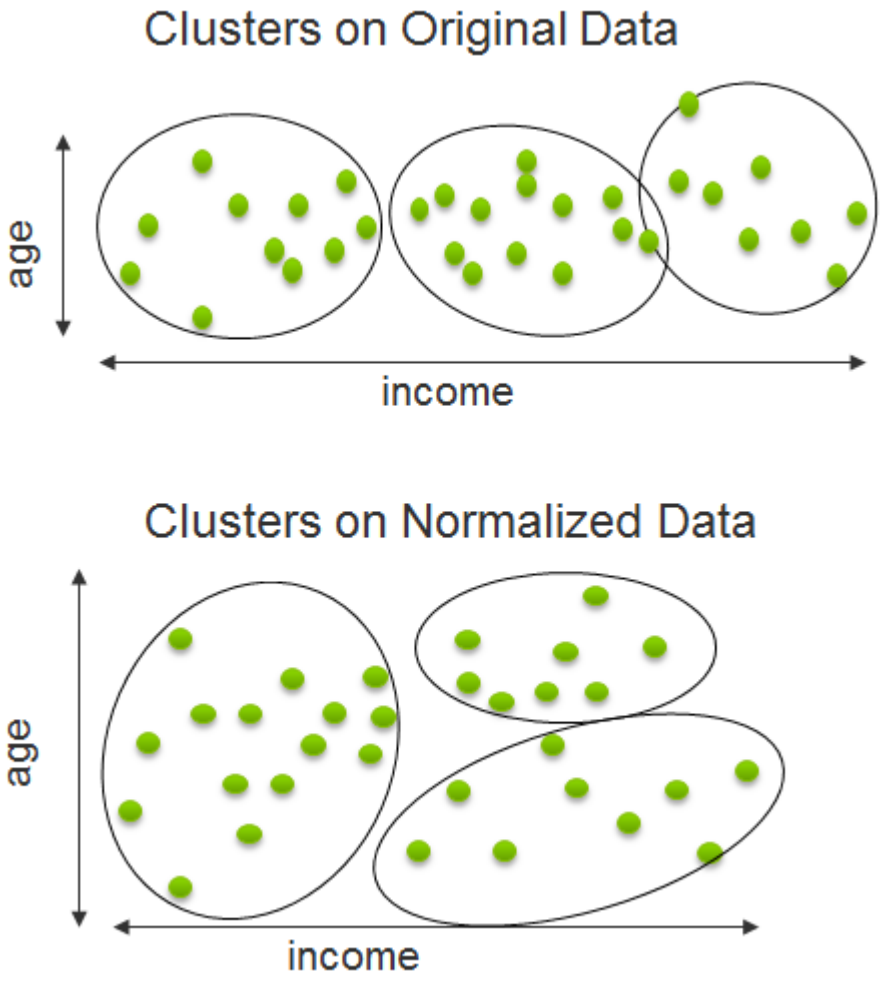
For k=5, avg silhouette coefficient (red dashed line) is **.566**



Sci-kit learn documentation, “Selecting the number of clusters with silhouette analysis on KMeans clustering”, https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Importance of Normalization in Clustering

- Clustering works best on normalized data, as two fields with drastically different value spreads can produce unintended results.
- For example, if you want to cluster on age and income, the algorithm may assign more importance to income as it has a broader range.
- A popular normalization method is to obtain the standard deviation of values for each field and use that statistic to calculate a Z-score.



Conclusion

- Classification techniques include
  - Logistic regression
  - Decision trees
  - Others: Random forests, Gradient-boosted trees, Support vector machines
- Use techniques such as ROC and cross-validation to evaluate classification model
- Techniques for unsupervised learning without a target variable includes *k*-means clustering
  - Elbow and silhouette methods can be used to determine value of *k*
- **Next class (Sep 9):** PCA, semi-supervised learning and expectation maximization
  - Quiz #1 (Sep 11)

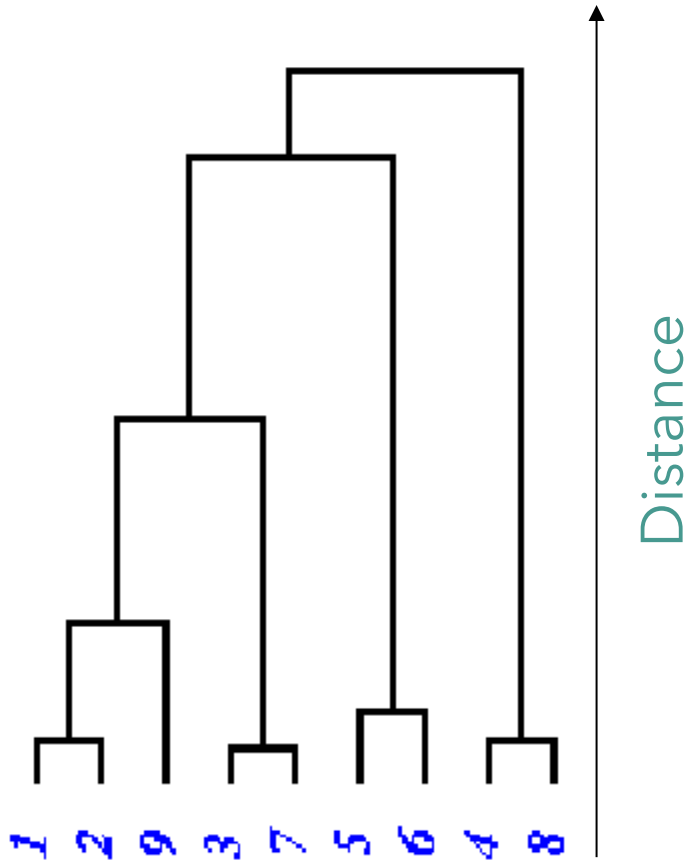
Appendix A: Hierarchical clustering

Goal

- Construction of a hierarchy of clusters (*dendrogram*) by merging/separating clusters with minimum/maxim

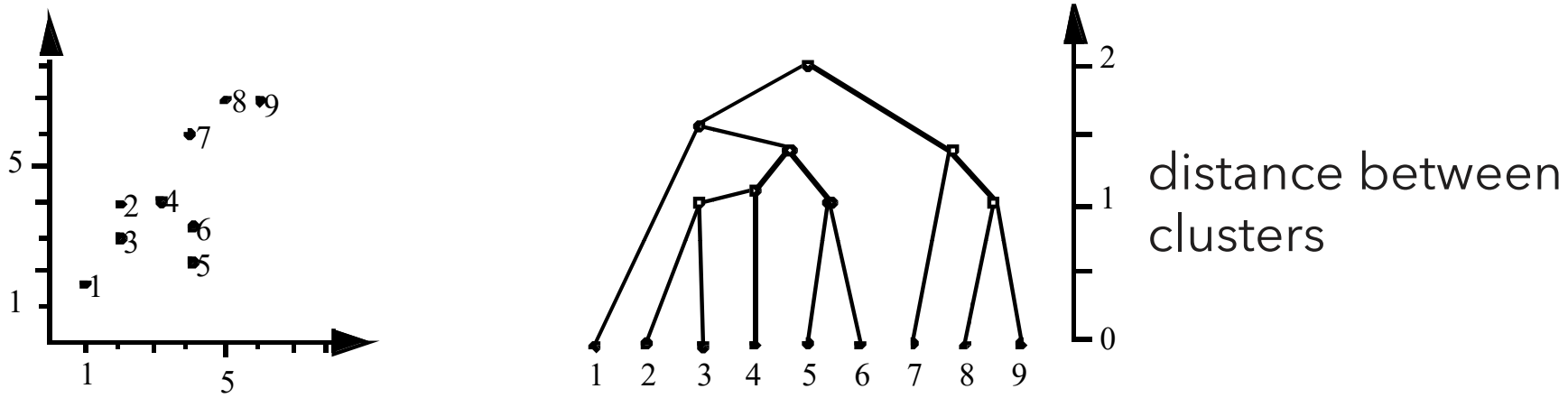
Dendrogram:

- A tree representing hierarchy of clusters, with the following properties:
  - Root: single cluster with the whole data set.
  - Leaves: clusters containing a single object.
  - Branches: merges / separations between larger clusters and smaller clusters / objects



Linkage Hierarchies: Basics

- Example dendrogram

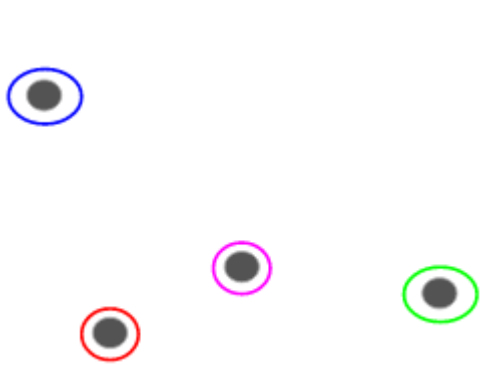


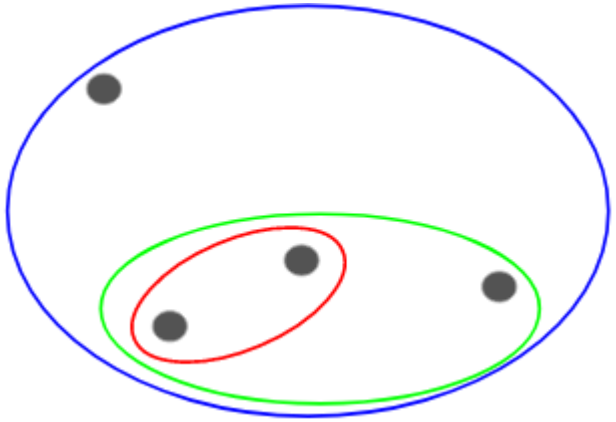
- Types of hierarchical methods
  - Bottom-up construction of dendrogram (*agglomerative*)
  - Top-down construction of dendrogram (*divisive*)

Base Algorithm

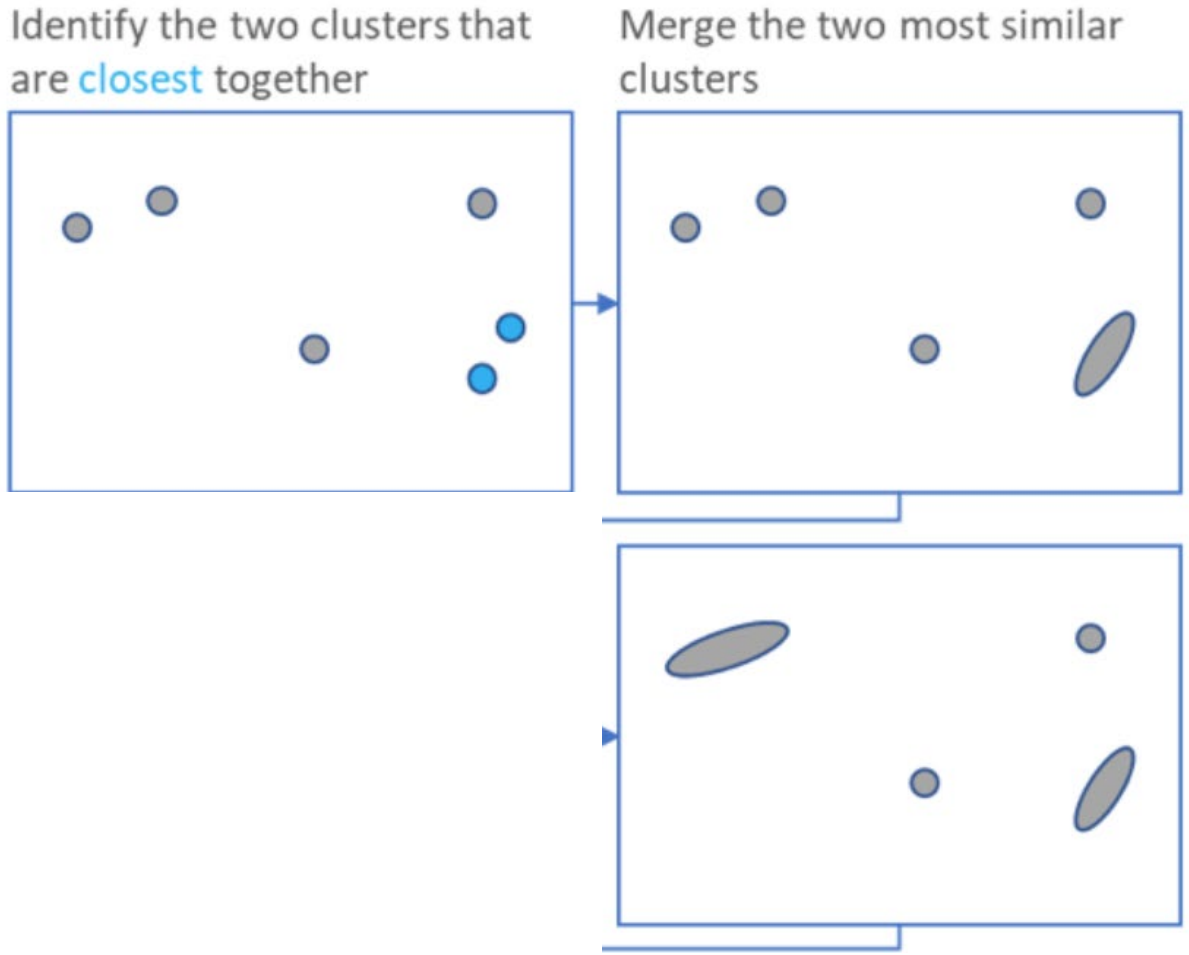
1. Form initial clusters consisting of a single object and compute the distance between each pair of clusters.
2. Merge the two clusters having minimum distance.
3. Calculate the distance between the new cluster and all other clusters.
4. If there is only one cluster containing all objects:
   Stop, otherwise go to step 2.

(Agglomerative) Hierarchical Clustering



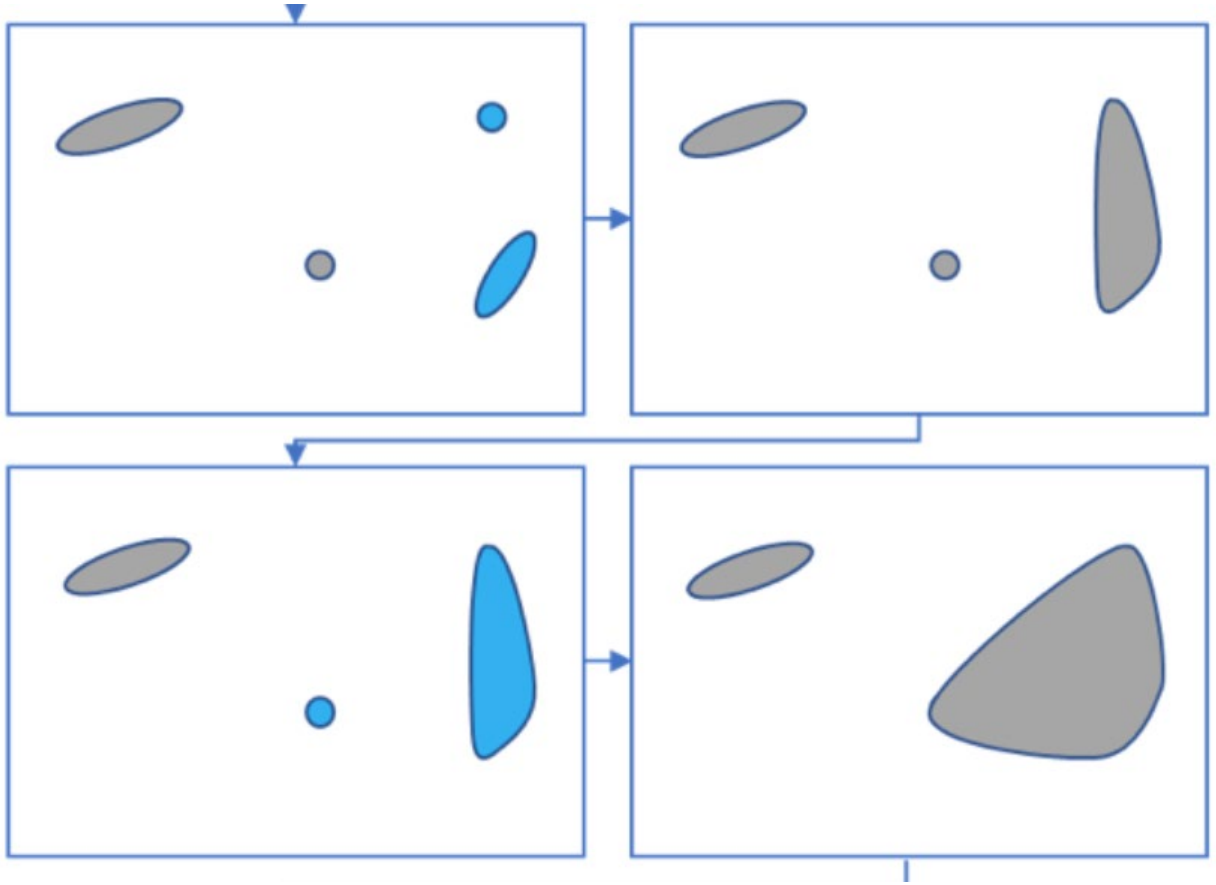


Hierarchical Clustering Example (1/3)



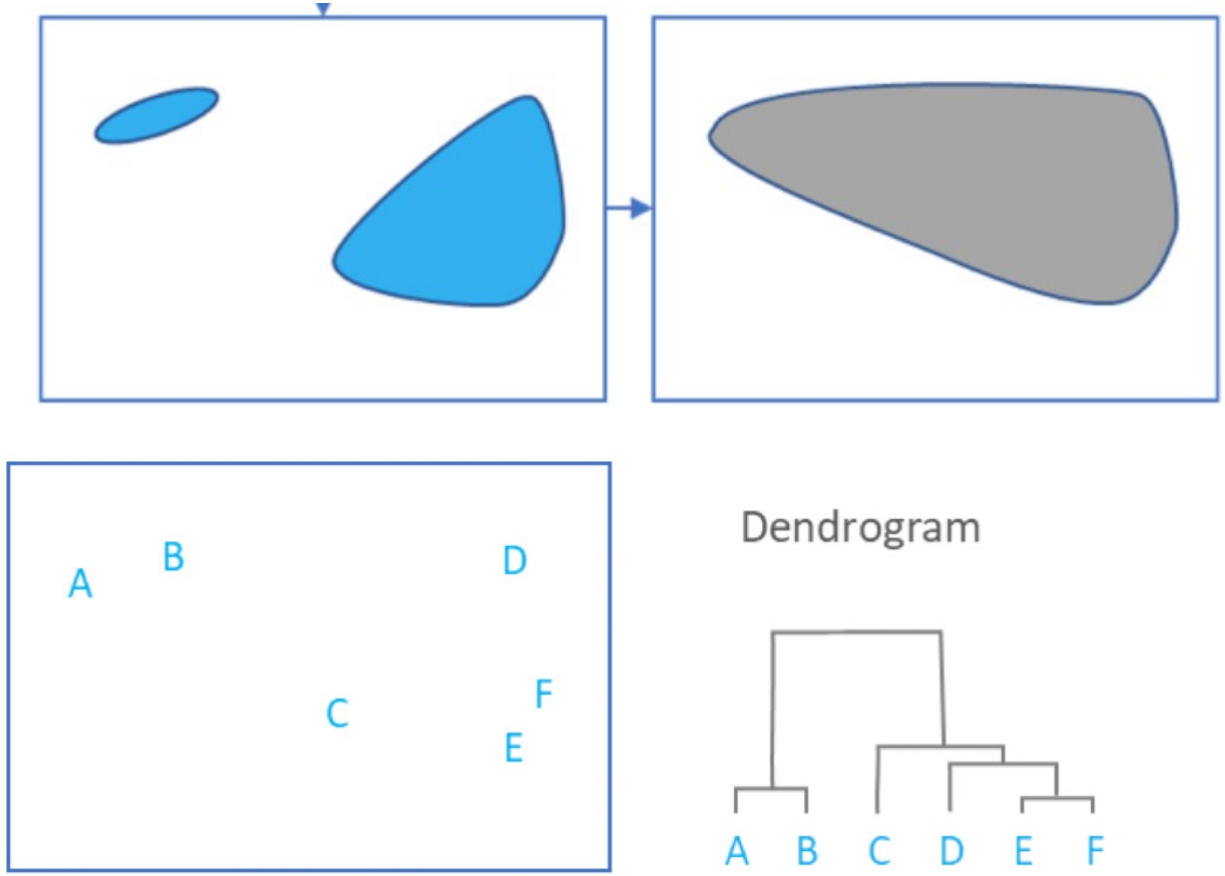
Bock, T "What is Hierarchical Clustering, " https://www.displayr.com/what-is-hierarchical-clustering/

Hierarchical Clustering Example (2/3)



Bock, T "What is Hierarchical Clustering, " https://www.displayr.com/what-is-hierarchical-clustering/

Hierarchical Clustering Example (3/3)



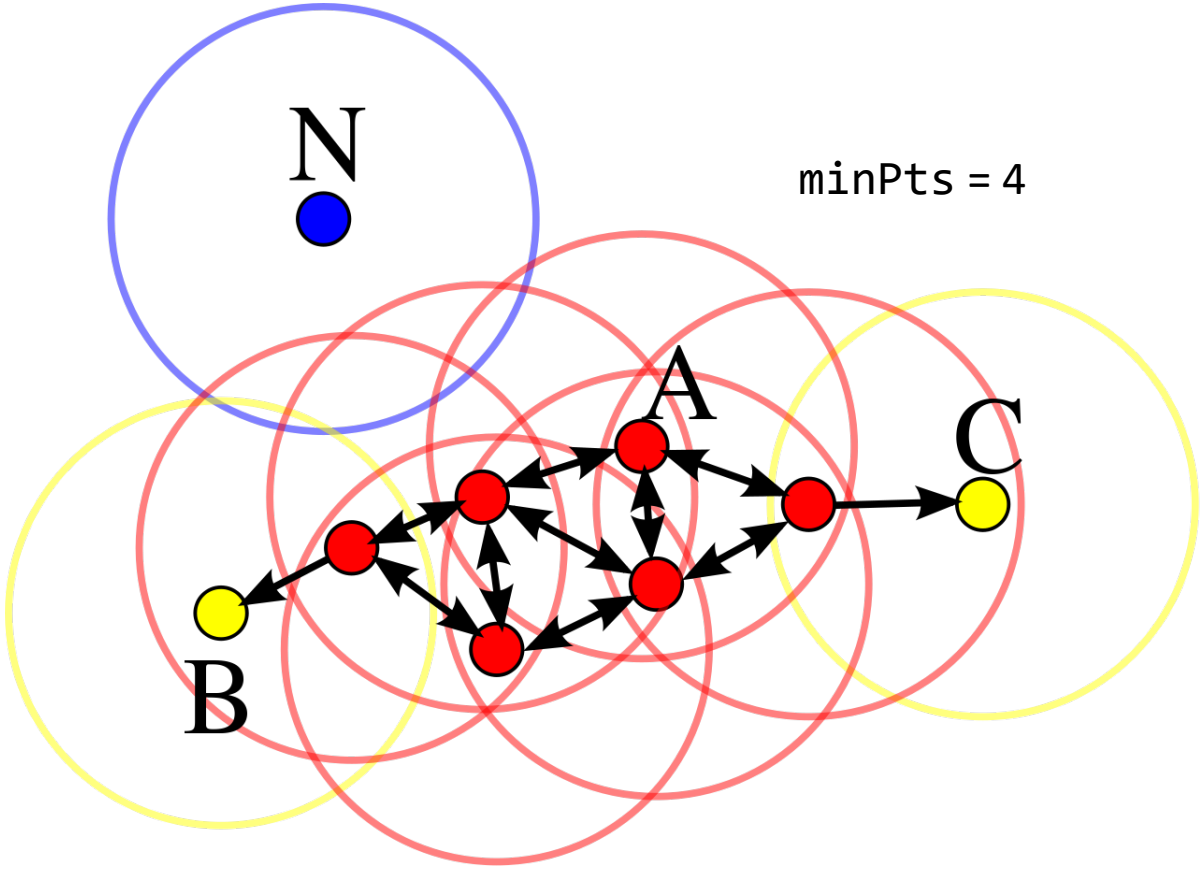
Bock, T "What is Hierarchical Clustering, " https://www.displayr.com/what-is-hierarchical-clustering/

K-means vs. Hierarchical Clustering

- *k*-means: When you know the number of clusters and can avoid placing cluster centers in low-density regions
- Hierarchical clustering: When clusters can be nested. More computationally expensive than k-means.

Appendix B: DBSCAN (Density-based Clustering)

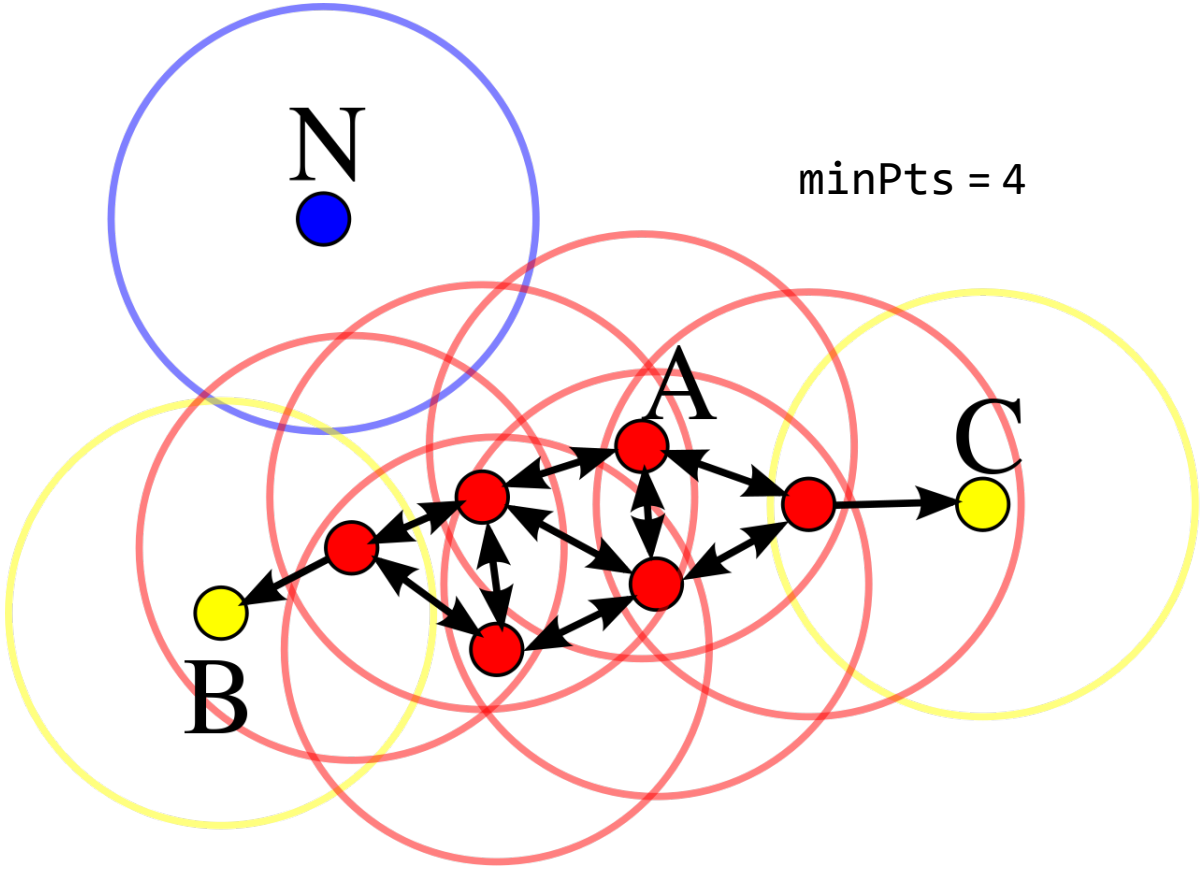
- DBSCAN groups together points that are closely packed together marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).
- **ε** (epsilon) defines the maximum distance a point can be from another point to be a neighbor
- A point p is a *core point* if at least **minPts** points are within distance ε of it (including p).



Algorithm due to Ester, et. al 1996, described in https://en.wikipedia.org/wiki/DBSCAN

DBSCAN (Density-based Clustering)

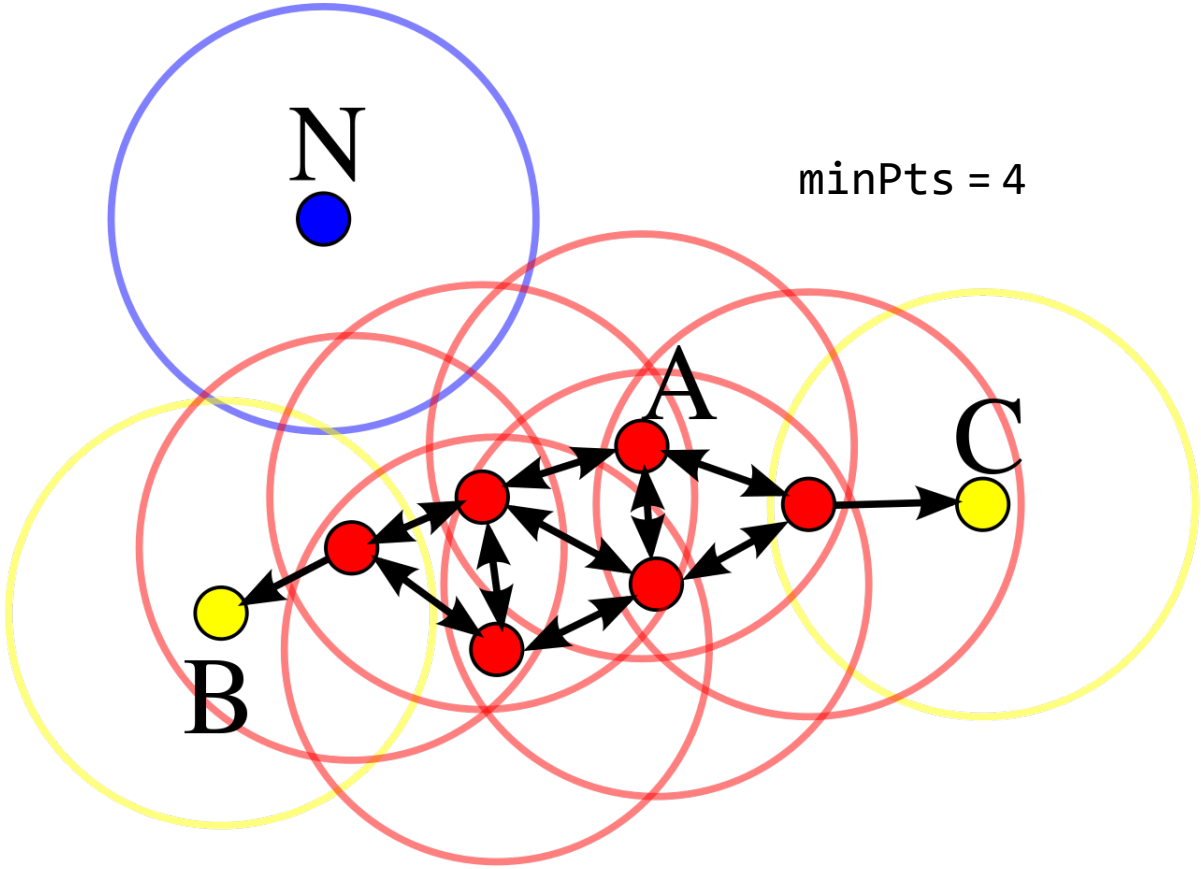
- Find the points in the ε (epsilon) neighborhood of every point, and identify the core points with more than `minPts` neighbors.
- Find the connected components of *core* points on the neighbor graph, ignoring all non-core points.
- Assign each non-core point to a nearby cluster if the cluster is an ε neighbor, otherwise assign it to noise.



Algorithm due to Ester, et. al 1996, described in https://en.wikipedia.org/wiki/DBSCAN

Example of DBSCAN

- minPts = 4
- Each circle has radius ε
- Point A and the other red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster.
- Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well.
- Point N is a noise point that is neither a core point nor directly-reachable.



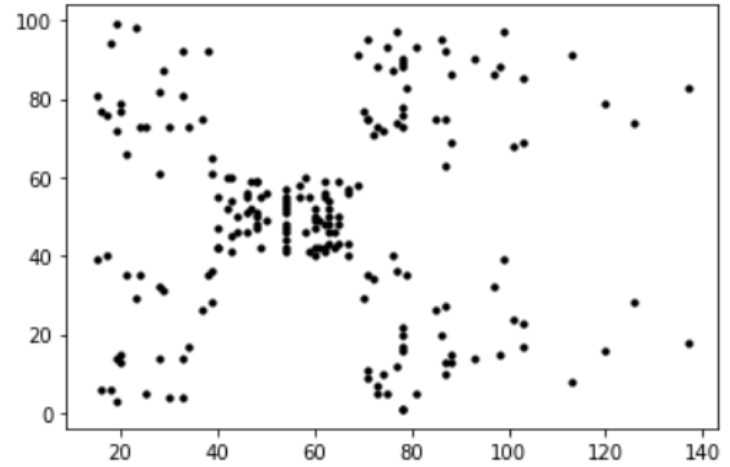
Algorithm due to Ester, et. al 1996, described in https://en.wikipedia.org/wiki/DBSCAN

DBSCAN for Customer Segmentation

```
# DBSCAN Clustering
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd# Importing the dataset
dataset = pd.read_csv(Mall_customers.csv')
data = dataset.iloc[:, [3, 4]].values
dataset.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
<matplotlib.collections.PathCollection at 0x1de01121988>
```

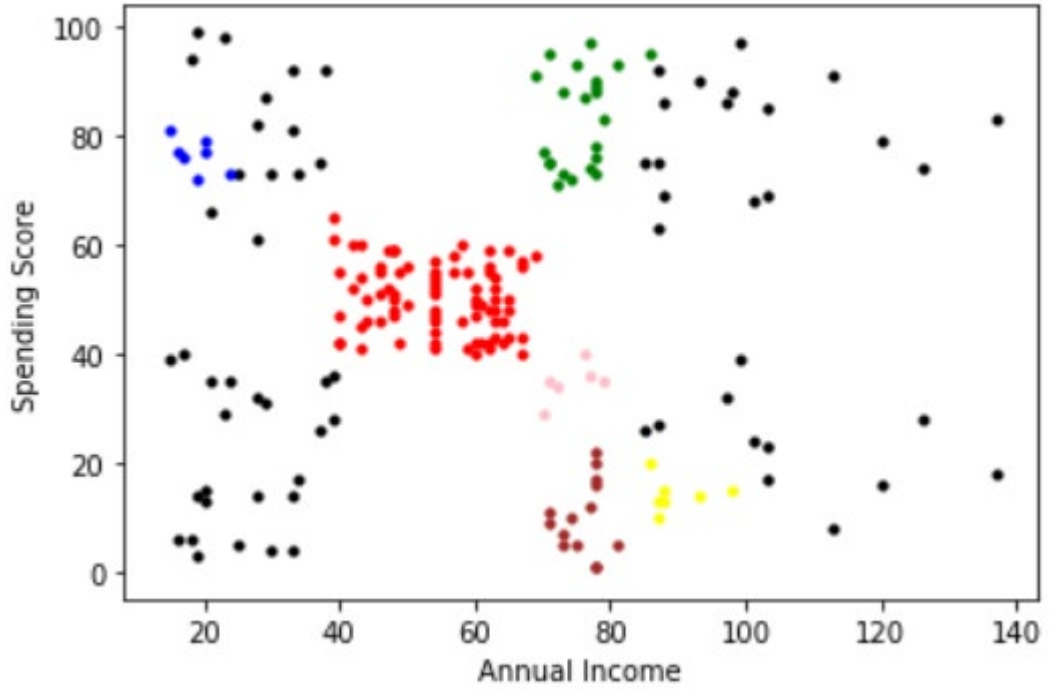


Annual Income vs. Spending Score

T. Akhtar, "Customer Clustering Using DBSCAN, September 18, 2020, https://towardsdatascience.com/customer-clustering-using-dbscan-4672f51fe5aa

Applying DBSCAN

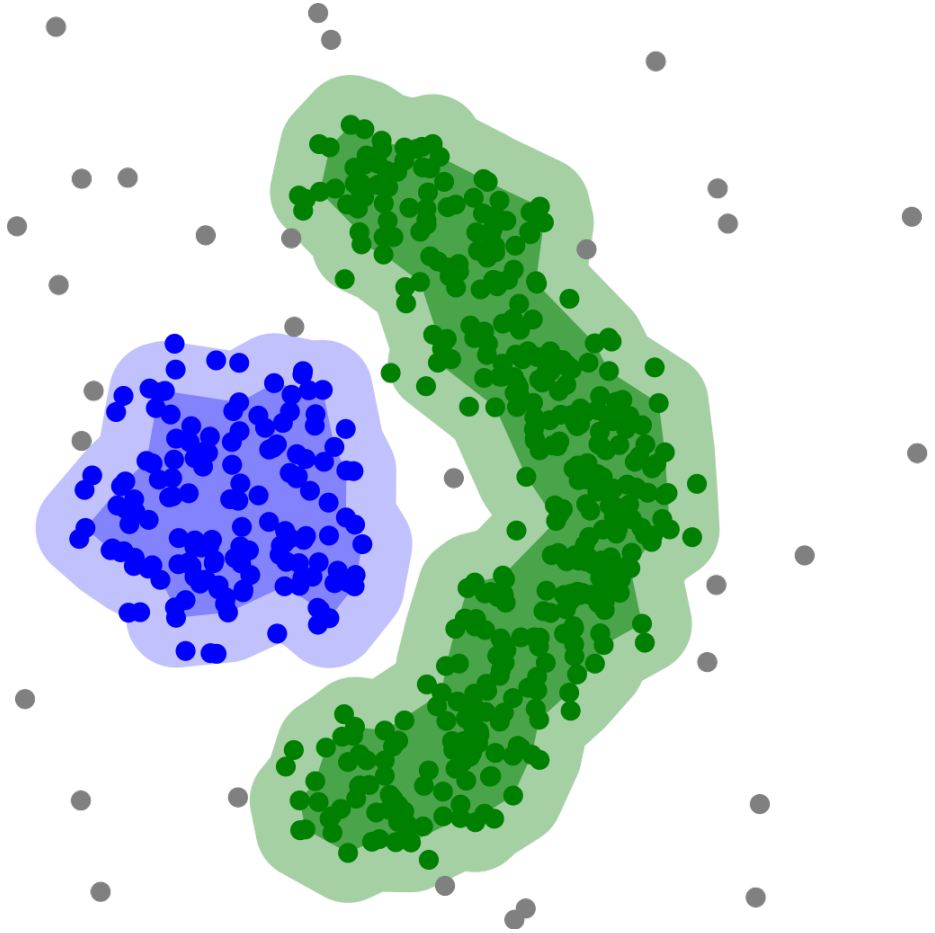
```
# Fitting DBSCAN to the dataset and predict the Cluster label
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=5.5, min_samples=4)
labels = dbscan.fit_predict(data)
np.unique(labels)
```



T. Akhtar, "Customer Clustering Using DBSCAN, September 18, 2020, https://towardsdatascience.com/customer-clustering-using-dbscan-4672f51fe5aa

Pros and Cons of DBSCAN

- Doesn’t require specifying the number of clusters in the data a priori, as opposed to k-means.
- DBSCAN can find arbitrarily-shaped clusters. DBSCAN has a notion of noise, and is robust to outliers.
- The parameters minPts and ε can be set by a domain expert, if the data is well understood.
- Still depends on Euclidean distance which is hard to compute with many dimensions
- Not good for data sets with large differences in density



https://en.wikipedia.org/wiki/DBSCAN