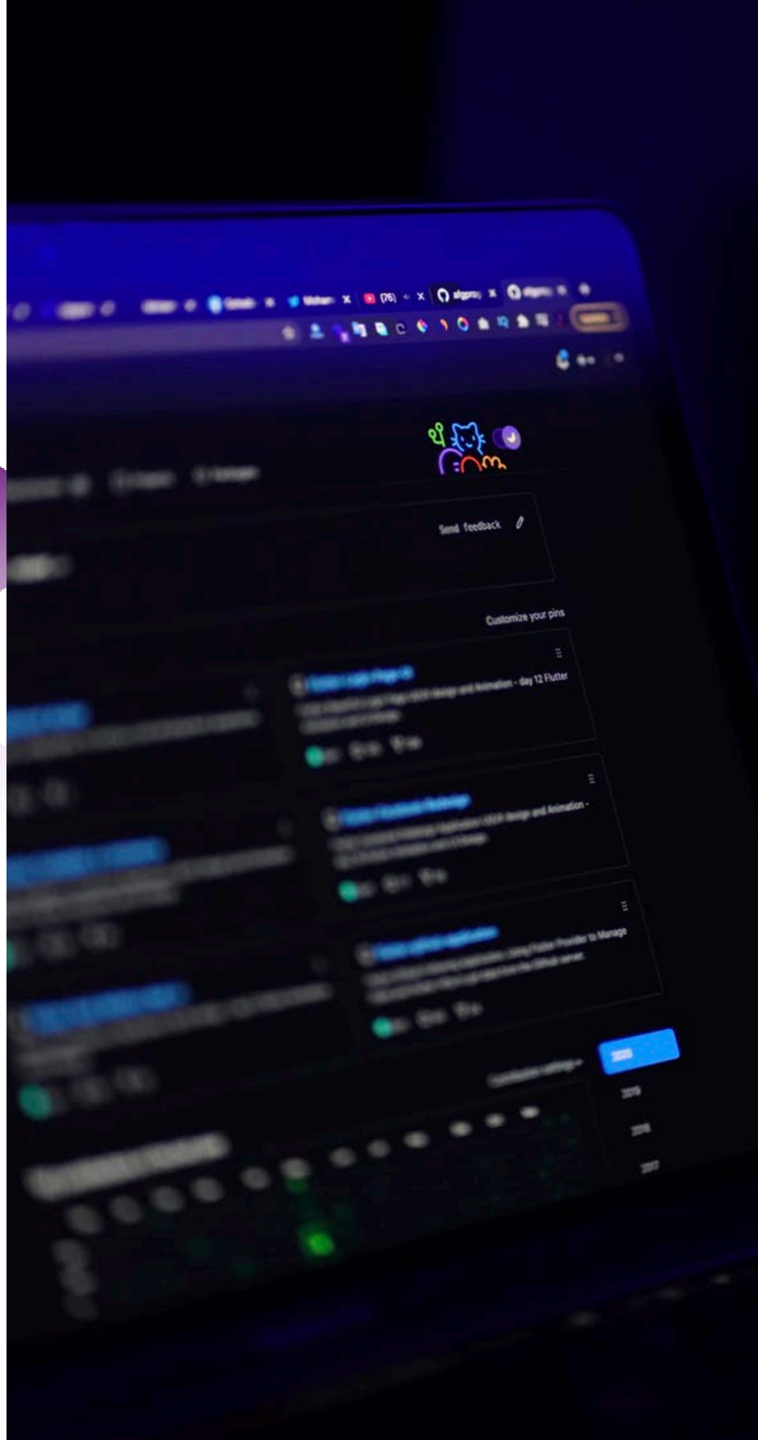
Machine Learning Part 2: Clustering

Louise Capener, Research Associate at the UK Data Service (Cathie Marsh Institute, UoM)

Economic and Social Research Council



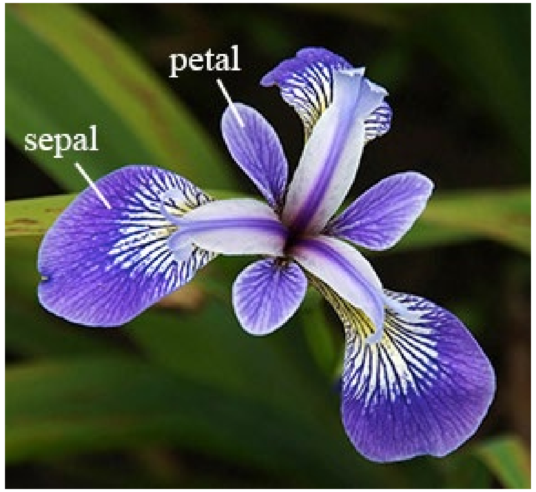
Outline

- Recap
- What is clustering?
- Why bother with it?
- Types of clustering algorithms
- K-Means
- Hierarchical clustering

Recap

Supervised learning	Unsupervised learning
Input data is labelled	Input data is unlabelled
Data is classified based on the training dataset	Assigns properties of given data to classify it
Divided into Regression and Classification	Divided into Clustering and Association
Used for prediction	Used for analysis
Algorithms include: decision trees, logistic regressions, support vector machine	Algorithms include: k-means clustering, hierarchical clustering, apriori algorithm
A known number of classes	An unknown number of classes

Recap (contd.)

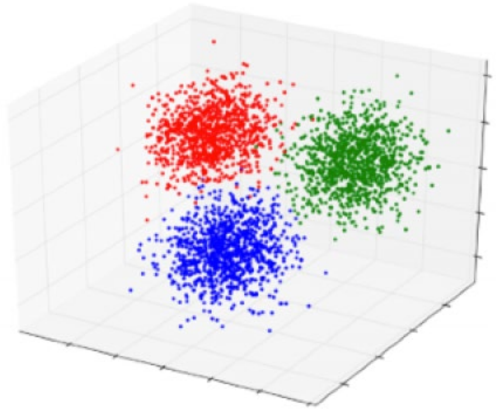


Supervised learning: used for prediction

Dps	Sepal length (cm)	Petal length (cm)	Petal width (cm)	Species
A	3.5	1.4	0.2	Iris-Versicolour
B	3.2	5.7	2.3	Iris-Setosa
C	3.2	5.9	2.3	Iris-Setosa
D	2.9	4.7	1.4	Iris-Virginica
E	3.7	1.5	0.4	Iris-Versicolour
F	3.1	5.5	2.2	?

Unsupervised learning: used for analysis

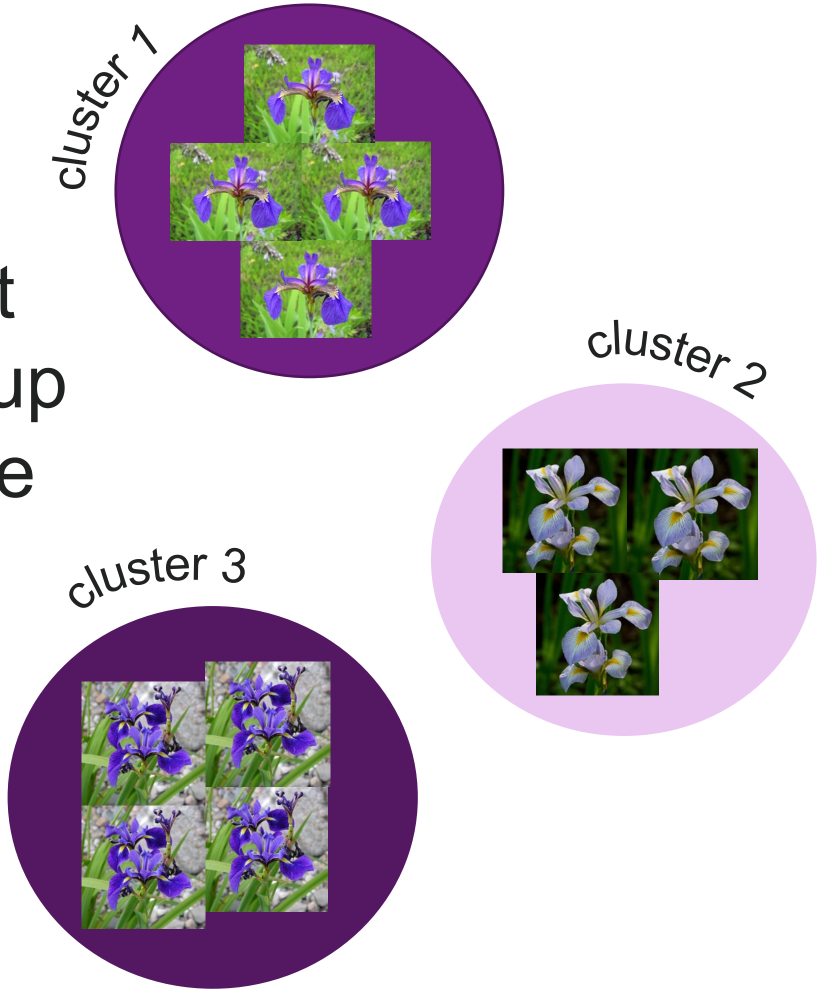
Dps	Sepal length (cm)	Petal length (cm)	Petal width (cm)
A	3.5	1.4	0.2
B	3.2	5.7	2.3
C	3.2	5.9	2.3
D	2.9	4.7	1.4
E	3.7	1.5	0.4



What is clustering?

“Clustering is the task of partitioning the dataset into groups, called clusters. The goal is to split up the data in such a way that points within a single cluster are very similar and points in different clusters are different.”

(Müller and Guido 2017)



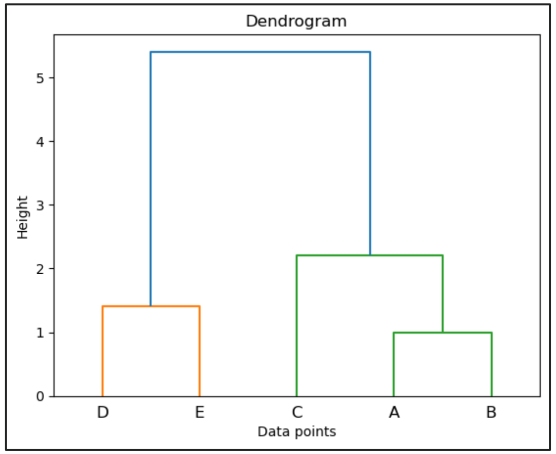
Dps	Sepal length (cm)	Petal length (cm)	Petal width (cm)	cluster
A	3.5	1.4	0.2	1
B	3.2	5.7	2.3	2
C	3.2	5.9	2.3	2

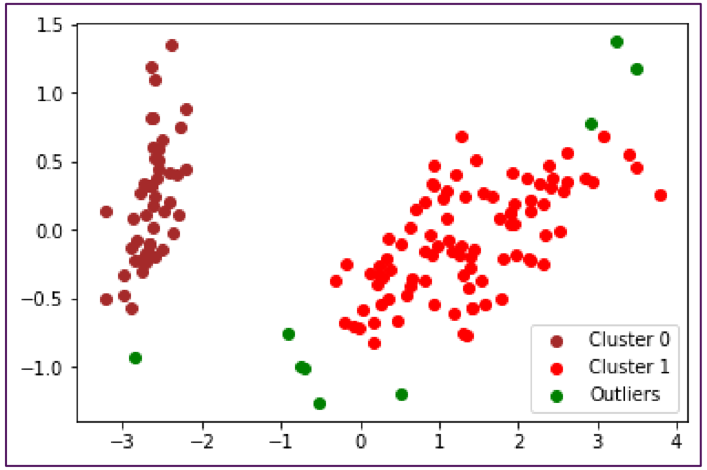
Why bother with it?

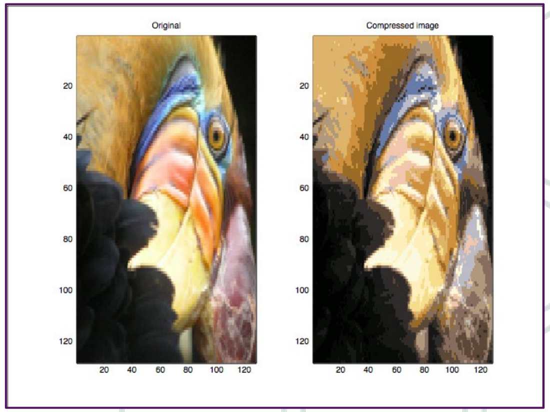
- It provides more information on the structure of the data → patterns

- It can help identify problems in the data, such as outliers

- It can be used to compress data

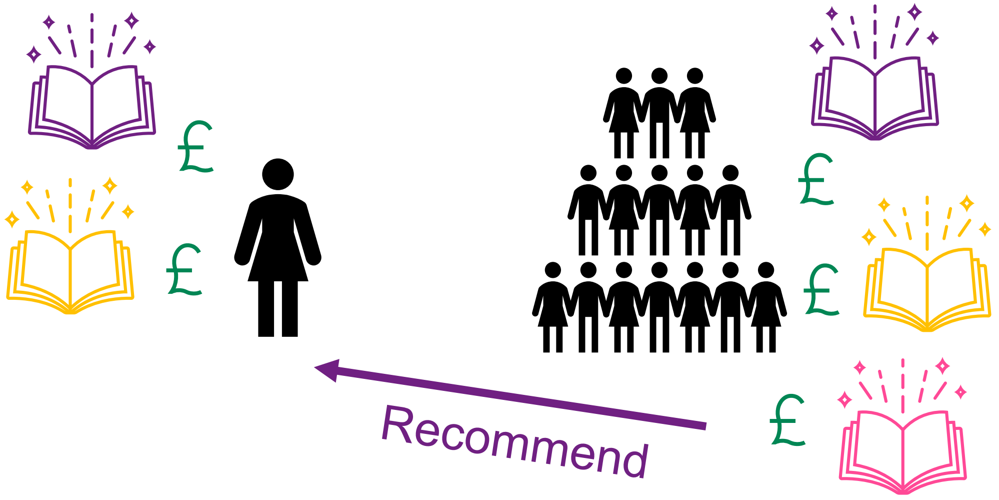


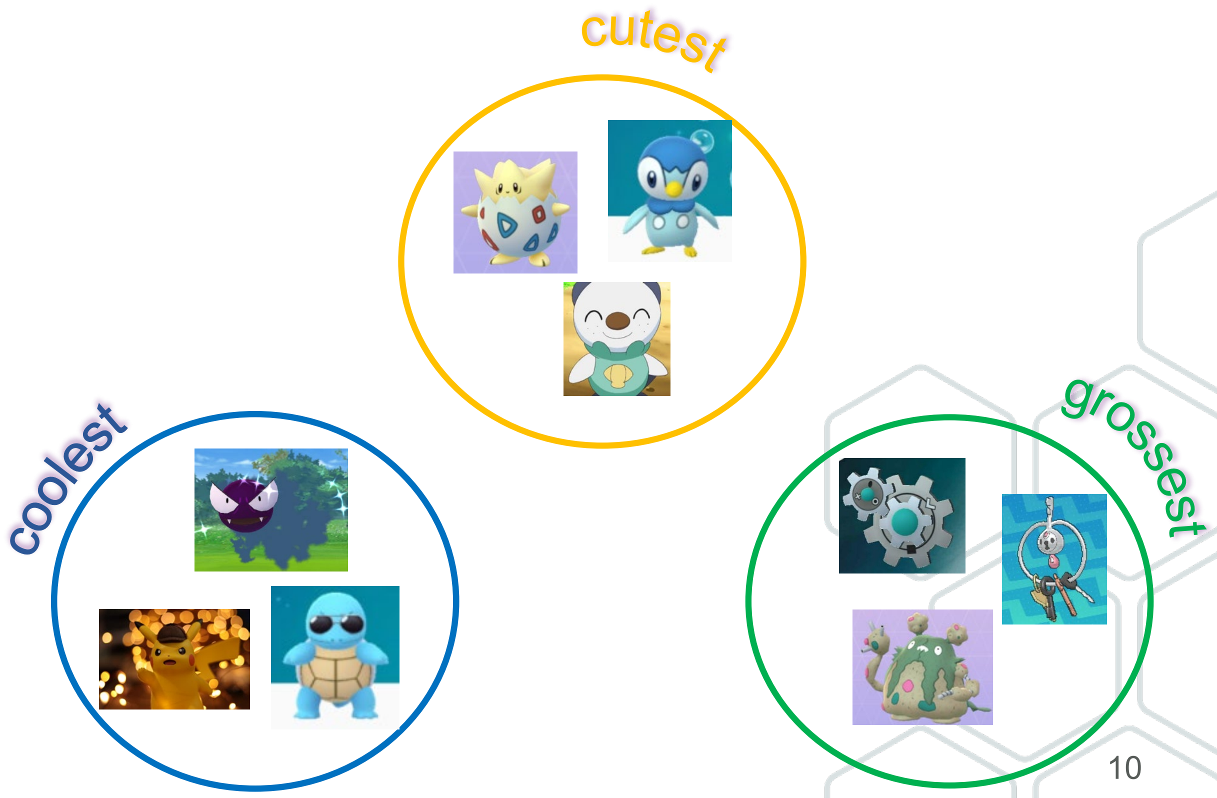




Other use cases

- Customer recommendation systems: "People who bought *Harry Potter and the Philosopher's Stone* also bought *The Hunger Games*…"
- Grouping DNA sequences of different strains of HIV into families of genetically similar viruses
- Identifying fake news by clustering the words used in articles. Certain words may appear more in sensationalized click-bait articles.
- And the more frivolous and fun side projects…





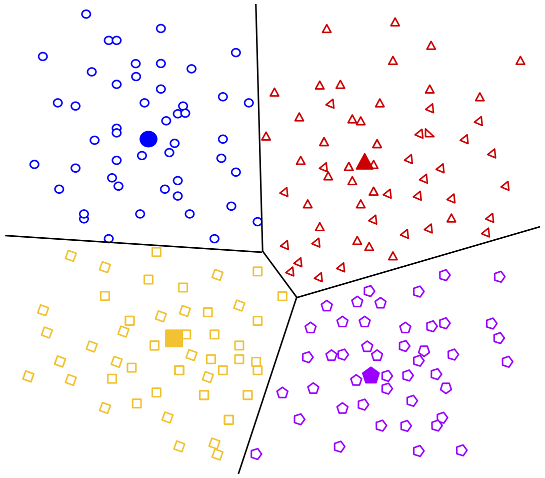
What is a cluster?

“There is no universal definition of what a cluster is: it really depends on the context, and different algorithms will capture different kinds of clusters.”

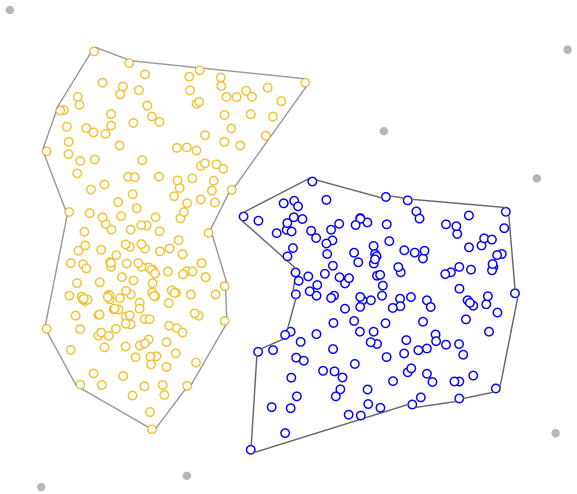
(Géron, 2019)

Types of clustering algorithms

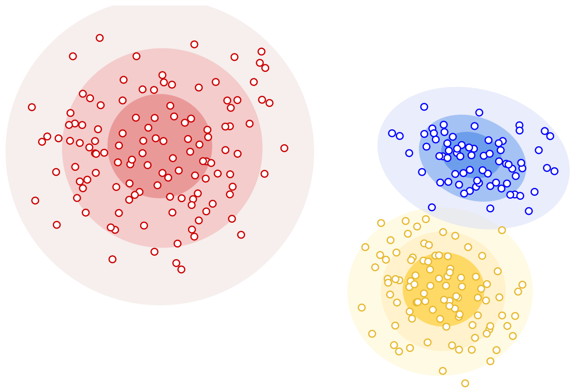
Centroid-based



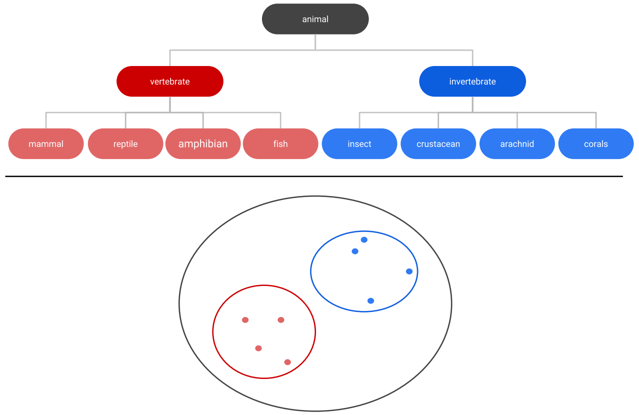
Density-based



Distribution-based



Hierarchical clustering

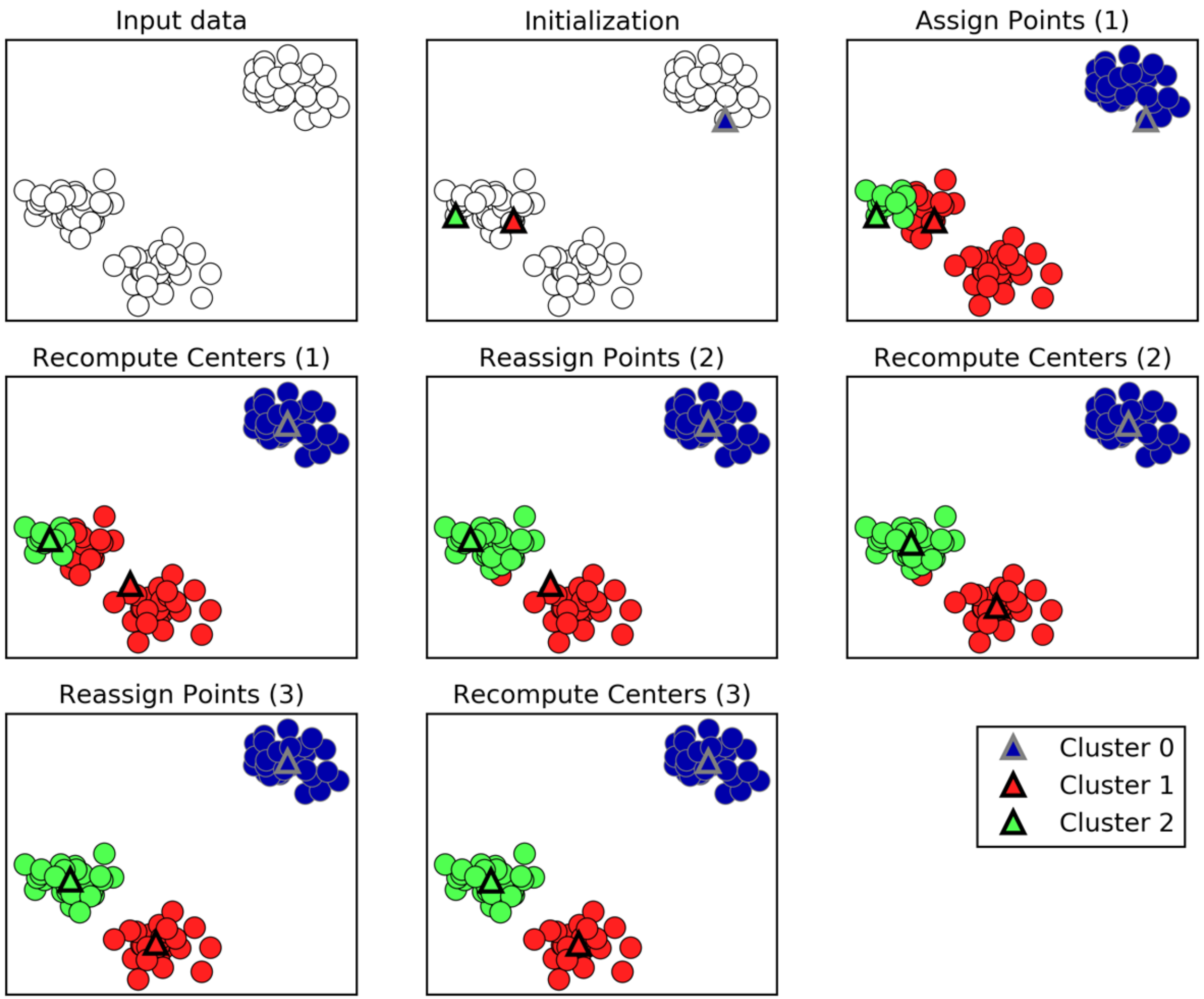


How do I know which type of algorithm is right for me?

EXPLORE YOUR DATA →

K-Means clustering

- We want to separate our data points into k clusters
- First, we initialize the algorithm with k random points (our centroids)
- Then, we assign each data point to its nearest initialisation point – using the Euclidean distance
- Once each data point is assigned, we relocate the initialisation point to the mean of the data points that were assigned to it
- Repeat the highlighted steps until the assignment of data points to centroids remains unchanged



Introducing pseudocode…

```
K-MEANS(P, k)
    Input: a dataset of points P = {p_1, ..., p_n}, a number of clusters k
    Output: centers {c_1, ..., c_k} implicitly dividing P into k clusters

    choose k initial centers C = {c_1, ..., c_k}
    while stopping criterion has not been met
        do ▷ assignment step:
           for i = 1, ..., N
               do find closest center c_k ∈ C to instance p_i
                  assign instance p_i to set C_k
           ▷ update step:
           for i = 1, ..., k
               do set c_i to be the center of mass of all points in C_i
```

Pseudo English

Aim: Separate data points into k clusters

1. Initialise the algorithm with k random points (centroids)

2. Assign each data point to its nearest initialisation point, using the Euclidean distance

3. Relocate the initialisation point to the centre of its cluster (find the mean of the data points)

4. Repeat steps 2 and 3 until the assignment of data points to centroids remains unchanged

Pseudocode

K-MEANS(P, k)

Input: a dataset of points $P = \{p_1, \ldots, p_n\}$, a number of clusters k
Output: centers $\{c_1, \ldots, c_k\}$ implicitly dividing P into k clusters

choose k initial centers $C = \{c_1, \ldots, c_k\}$
while stopping criterion has not been met
    do $\triangleright$ assignment step:
        for $i = 1, \ldots, N$
            do find closest center $c_k \in C$ to instance p_i
                assign instance p_i to set C_k
        $\triangleright$ update step:
        for $i = 1, \ldots, k$
            do set c_i to be the center of mass of all points in C_i

Code

```
import numpy as np
import matplotlib.pyplot as plt
import random

def load_data(fname):
    features = []

    with open(fname) as f:
        for line in f:
            p = line.strip().split(' ')
            features.append(np.array(p[1::], dtype=float))
    return np.array(features)

#This function returns the Euclidean distance squared between x and y
def distance(X, Y):
    return np.linalg.norm(X-Y)**2

# normalize a vector in L2 norm
def l2_normalize(data):
    return data / np.sqrt(np.sum(data ** 2))

"""
    loads all the required datasets
    Input:
        normalize - true/false - if normalization requires
    Outputs:
        dataset - concatenated datasets
        target - list of all the individual data( used for evaluating performance)
"""
def load_dataset(normalize = False):
    # load the datasets from files
    animals = load_data("animals")
    countries = load_data("countries")
    fruits = load_data("fruits")
    veggies = load_data("veggies")

    #normalize if required
    if normalize:
        animals = l2_normalize(animals)
        countries = l2_normalize(countries)
        fruits = l2_normalize(fruits)
        veggies = l2_normalize(veggies)

    # just a list of all datasets
    targets = [animals, countries, fruits, veggies]

    # concatenata all datasets into one
    dataset = np.concatenate((animals, countries, fruits, veggies))
    return dataset, targets

"""
    gets initial random centroids for the kmeans algo
    Input:
        k - num of centroids
        dataset - the entire dataset
    Output:
        list of k centroid ( each is 300 long vector )
"""
def get_random_centroids(k, dataset):
    # Get Random centroids for k
    cids = random.sample(range(0, len(dataset)), k)
    centroids = []

    # Then, the randomly chosen centroids are compiled into a list
    for i in cids:
        centroids.append(dataset[i])

    print("cids ", cids)
    # print("centroids ", centroids)
    return centroids

"""
    Assign data points to clusters with given centroids
```

```
"""
    Assign data points to clusters with given centroids
    Input:
        dataset - the entre dataset
        centroids - coordinates of k centroids
    Output:
    clusters - a list of cluster ids ( 0 to k-1 ) for all data points
"""
def get_clusters(dataset, centroids):
    clusters = []
    # for every data point
    for val in dataset:
        min_dist = np.Inf
        cluster_id = None
        # find the closest centroid by checking the distance
        # for every centroid
        for cid, centroid in enumerate(centroids):
            dist = distance(centroid, val)
            if dist < min_dist:
                min_dist = dist
                cluster_id = cid

        #record the closest centroid id as cluster id of this point
        clusters.append(cluster_id)

    #print("Clusters ", clusters)
    return clusters

"""
    Finds new k centroids
    Inputs:
        dataset
        clusters - current cluster assignments for all the data points
        mean_fun - mean or median - function to use for calculating center of a cluster
    Ouputs:
        list of modified k centroids
"""
def find_new_centroids(dataset, clusters, mean_fun = np.mean):
    unique_clusters = np.unique(clusters)

    # compute the new centroids
    new_centroids = []
    # go through all the clusters
    for cid in list(unique_clusters):
        cluster_members = []
        pids = []
        #go through all the data points and collect the point
        #belonging to the particular cluster
        for pid in range(len(dataset)):
            # for all the points in that claster
            if clusters[pid] == cid:
                cluster_members.append(dataset[pid])
                pids.append(pid)
        #make a 2d matrix
        cluster_members = np.stack(cluster_members, axis = 0)
        # find the coordinates of the center by applying "mean" function
        new_centroid = mean_fun(cluster_members, axis = 0)

        new_centroids.append(new_centroid)

    #print("New Centroids: ")
    #print(new_centroids)
    return new_centroids


def evaluate_clusters(clusters, targets):
    #print("Evaluating..")
    wrong = 0
    total = 0
    for i, d in enumerate(targets):
        start_pos = total
        end_pos = start_pos + len(d)
        total += len(d)
        cluster = clusters[start_pos:end_pos]
        # cid = clusters[start_pos]
        cid = max(set(cluster), key=cluster.count)
```

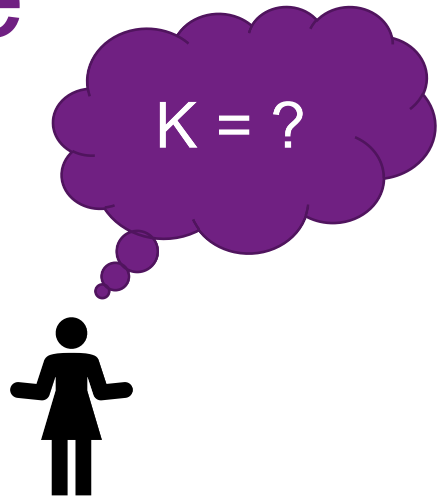
Initialisation – how do we select our centroids?

- Forgy's method: choose k random data points from the dataset

- Random Partition method: Randomly assign data points to a cluster. Then calculate the mean of each cluster to get the initial centroids.

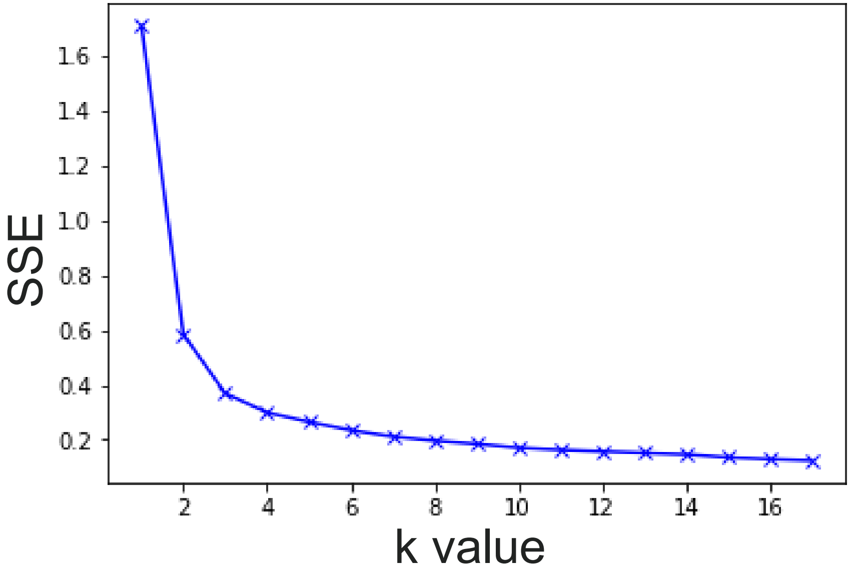
- K-means++: first centroid is a random datapoint, but remaining centroids are chosen based on the maximum squared distance → centroids are spread out evenly

Okay… but how do we determine the number of clusters we want?

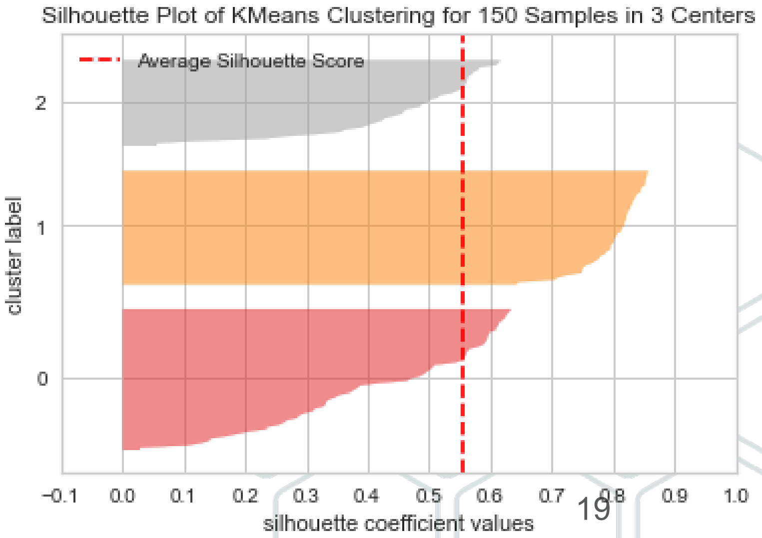


Sepal length (cm)	Petal length (cm)	Petal width (cm)
3.5	1.4	0.2
3.2	5.7	2.3
3.2	5.9	2.3
2.9	4.7	1.4
3.7	1.5	0.4

Elbow plot



- Each time we increase the number of clusters → the SSE decreases
- Goal: select a small value of k that still has a low SSE
- Elbow represents where we start to have diminishing returns by increasing k

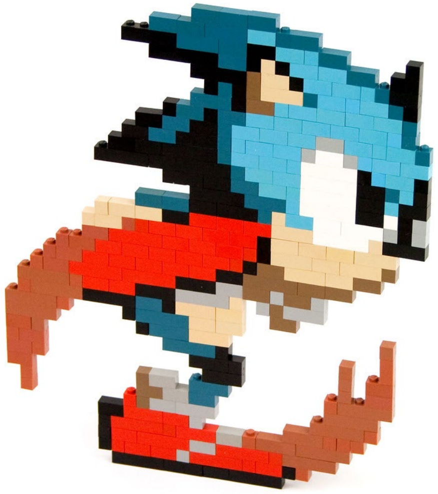


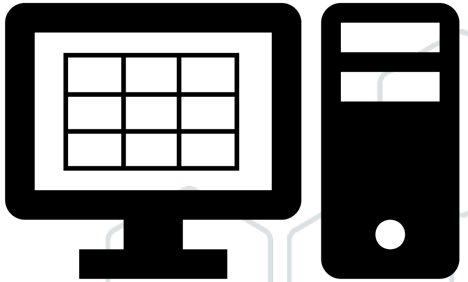
What are the strengths?

- Easy to understand and implement

- Fast

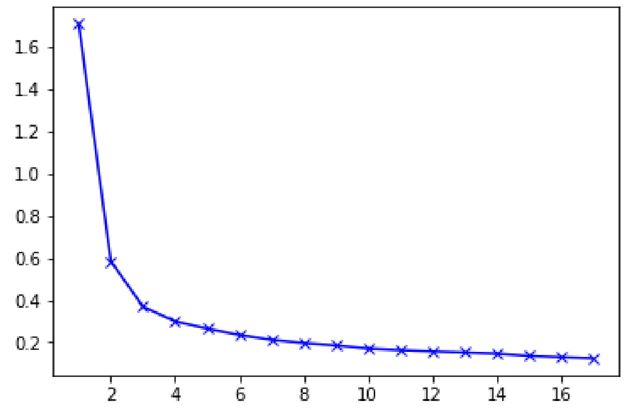
- Scalable





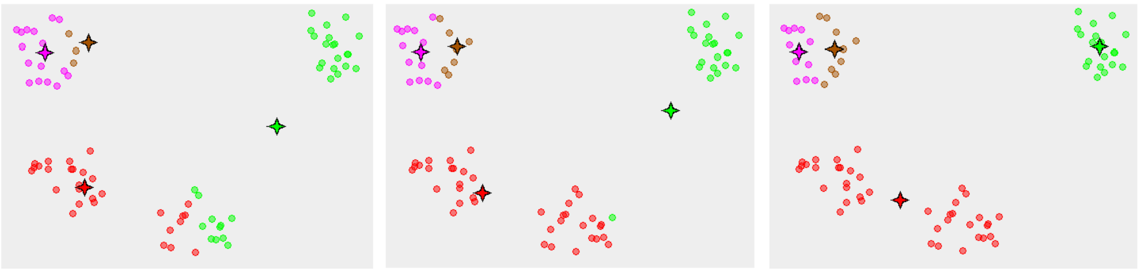
What are the limitations?

- Choosing k manually – it's a hassle!



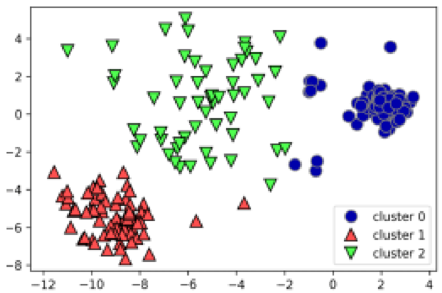
Elbow method

- It is dependent on initial values: necessary to run the algorithm several times to avoid suboptimal solutions – converges to a local minimum

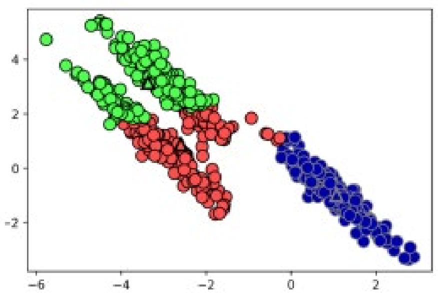


Bad centroid initialization → Suboptimal solution

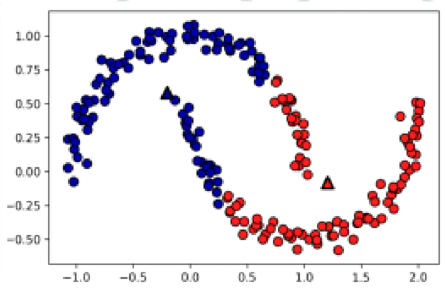
- Not good at clustering data of varying sizes, densities, or nonspherical shapes



density



direction

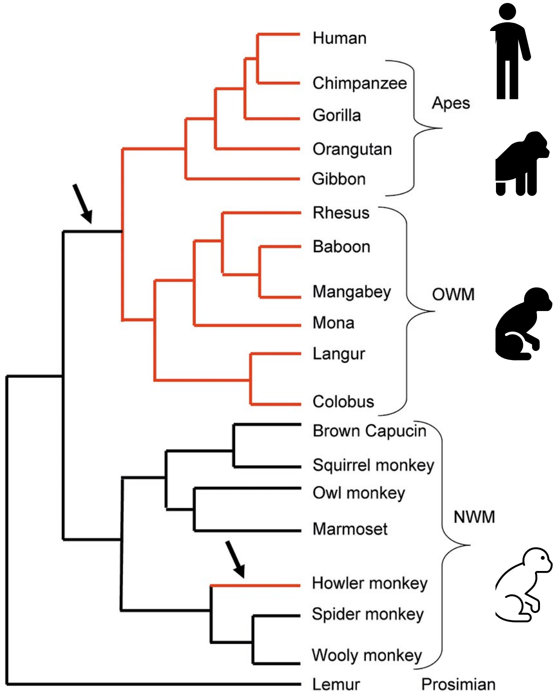


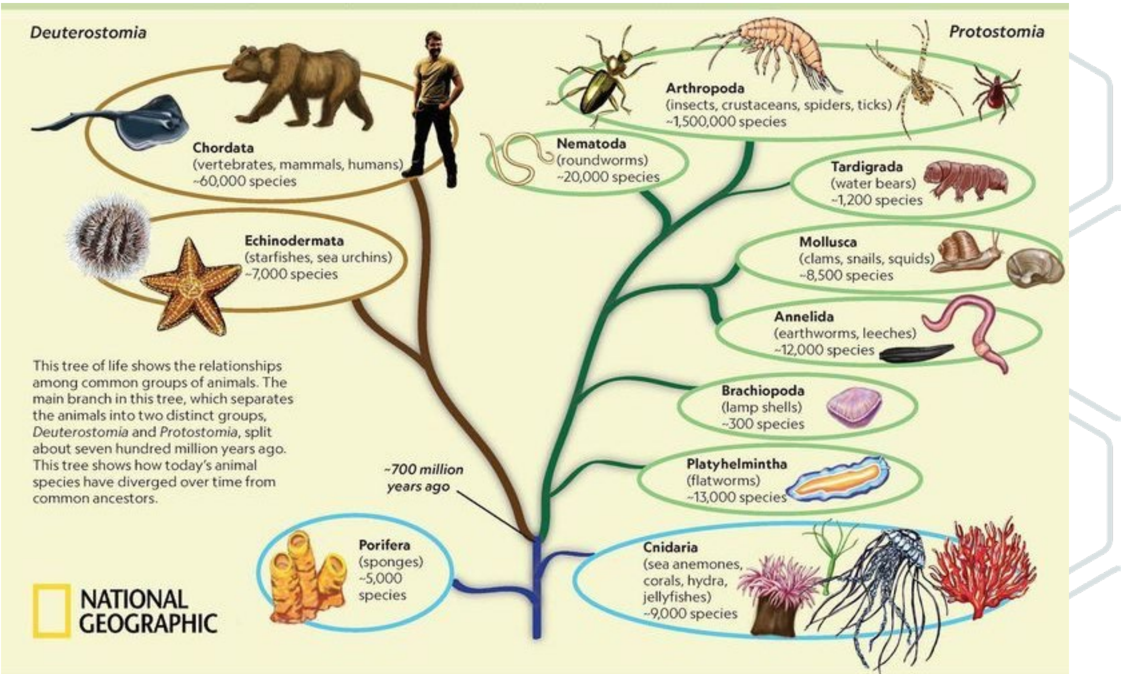
shape

Hierarchical clustering

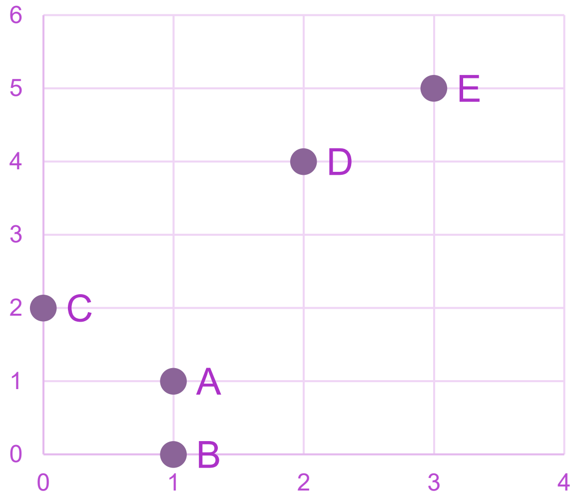
“Hierarchical clustering algorithms […] approach the problem of clustering by developing a binary tree-based data structure called the dendrogram. Once the dendrogram is constructed, one can automatically choose the right number of clusters by splitting the tree at different levels to obtain different clustering solutions for the same dataset without rerunning the clustering algorithm again.”

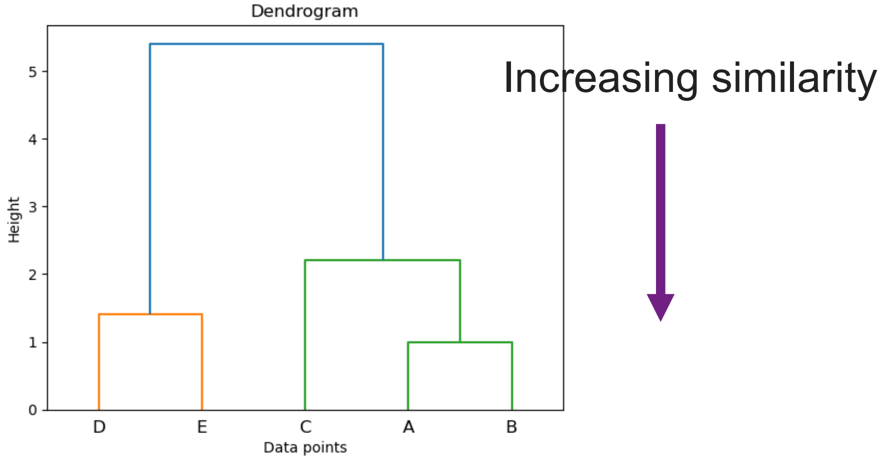
(Reddy and Vinzamuri, 2015)

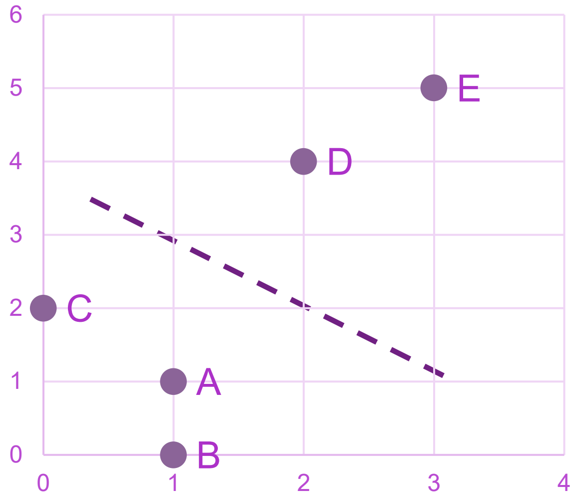


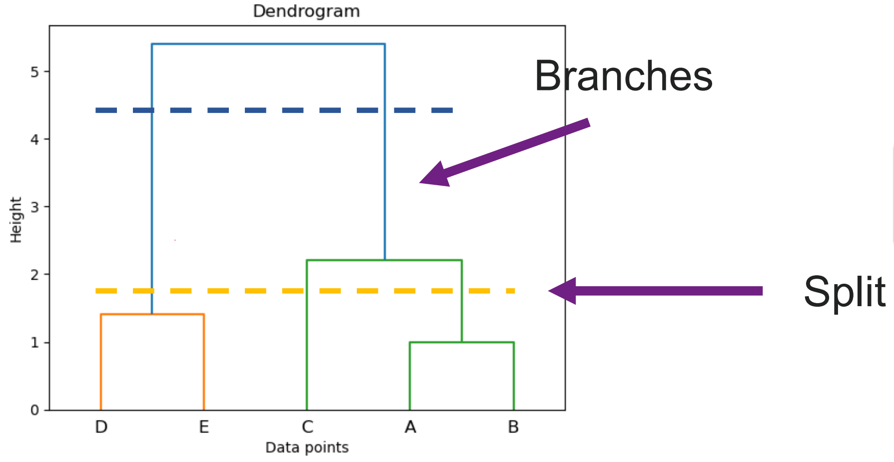


How do I read a dendrogram?



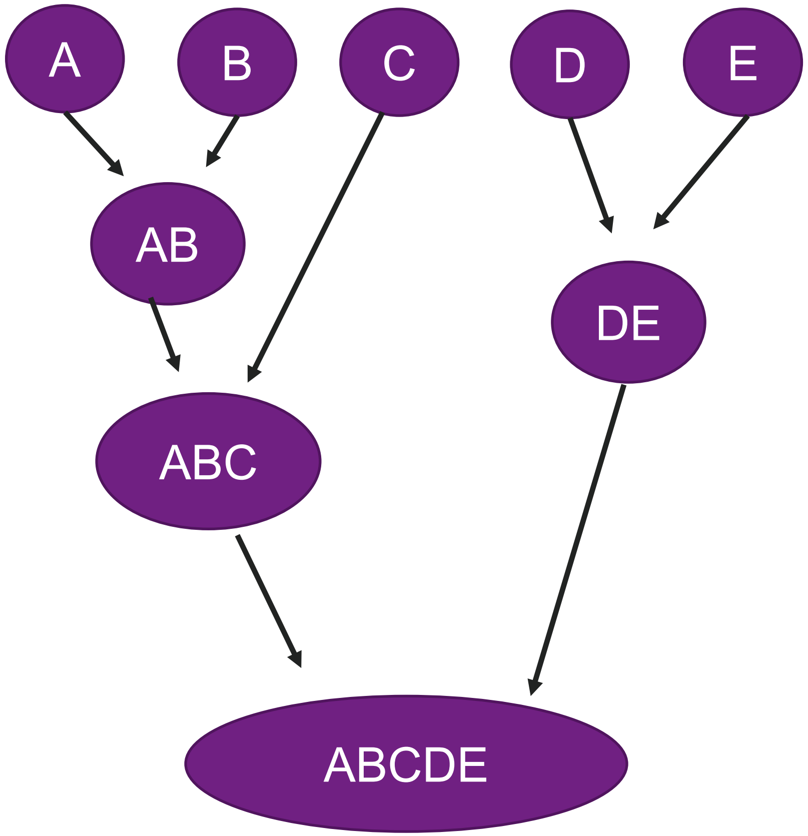


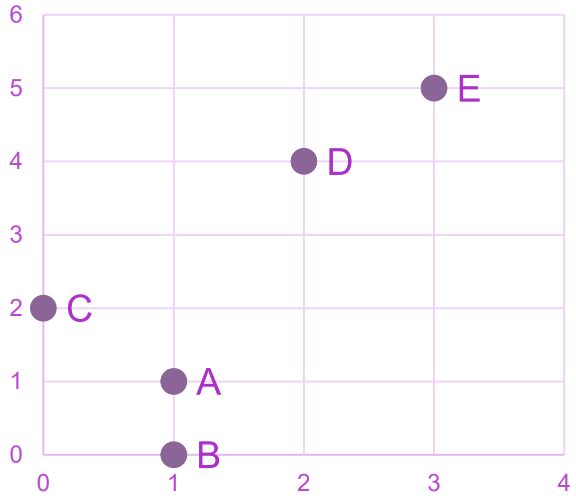


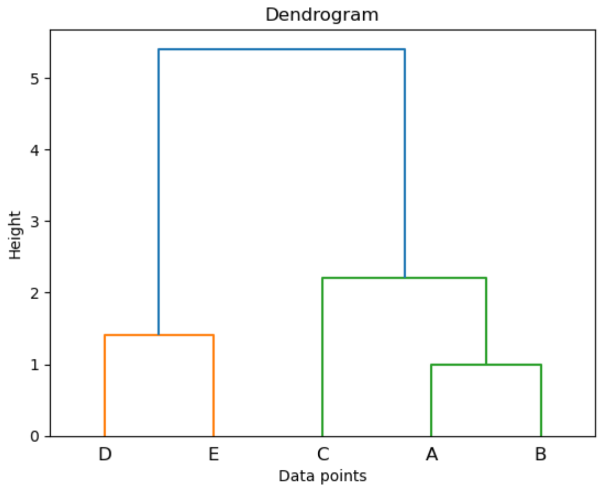


What are the 2 main approaches to hierarchical clustering?

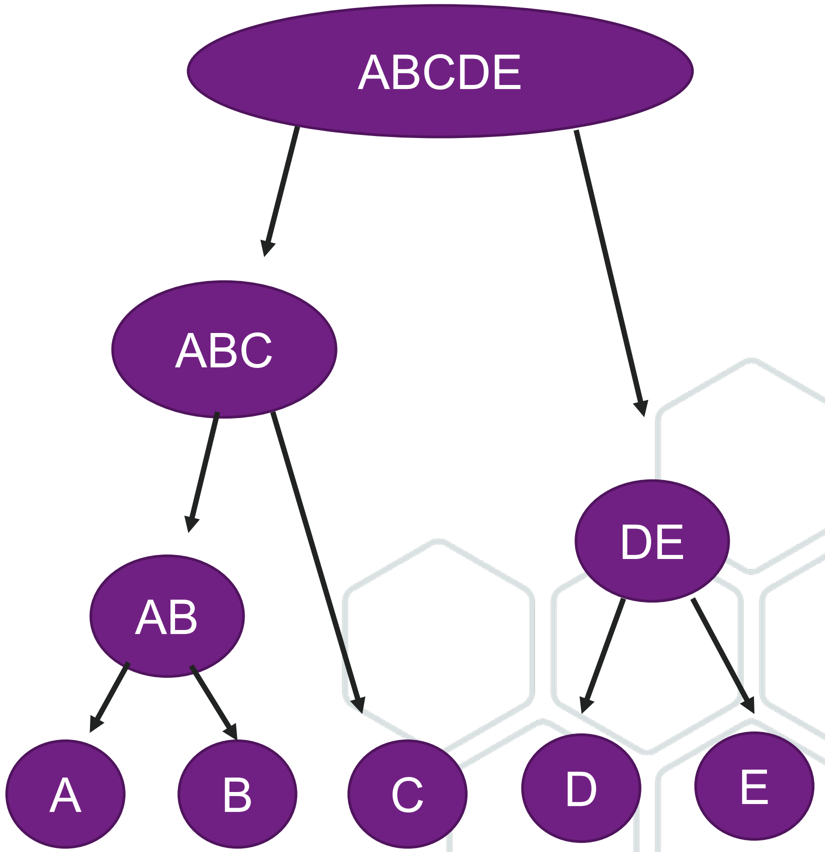
1) Agglomerative







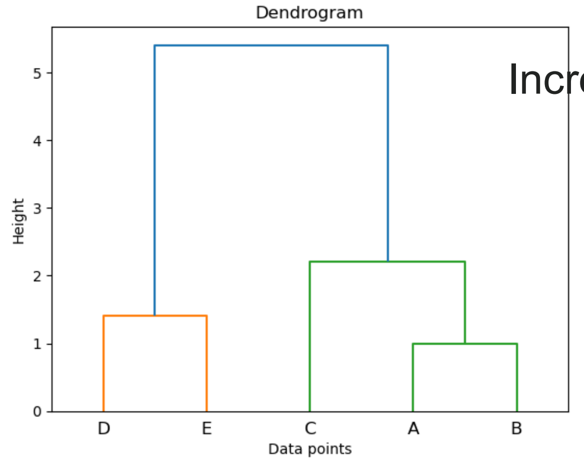
2) Divisive



…but how do we know which clusters should be combined, or split?

1) Measure of distance – some measure of similarity

- Hierarchical clustering is proximity-based
- Affects the shape of the clusters
- Used to build distance matrix
- Default is Euclidean distance, but other measures exist: correlation-based, Levenshtein distance etc.



Increasing similarity

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

p	q	ED
3	4	1.414214
2	1	

2) Linkage criterion – different ways to link clusters based on distance

- A means of determining whether certain clusters should be merged
- Default is complete-linkage
- Other commonly used linkage criteria: single-linkage, average-linkage
- Used to update the distance matrix and merge clusters

Agglomerative hierarchical clustering: Using complete-linkage

PSEUDO ENGLISH

1. Turn each data point into a singleton, i.e., into a cluster of a single element
2. For each pair of clusters, calculate their distance
3. Merge the pair of clusters that take the smallest distance
4. Continue with step 2 and 3 until the termination criterion is satisfied
5. The termination criterion most commonly used is a threshold of the distance value

Algorithm Agglomerative (D)

Input: Dataset (D)
Output: Dendrogram

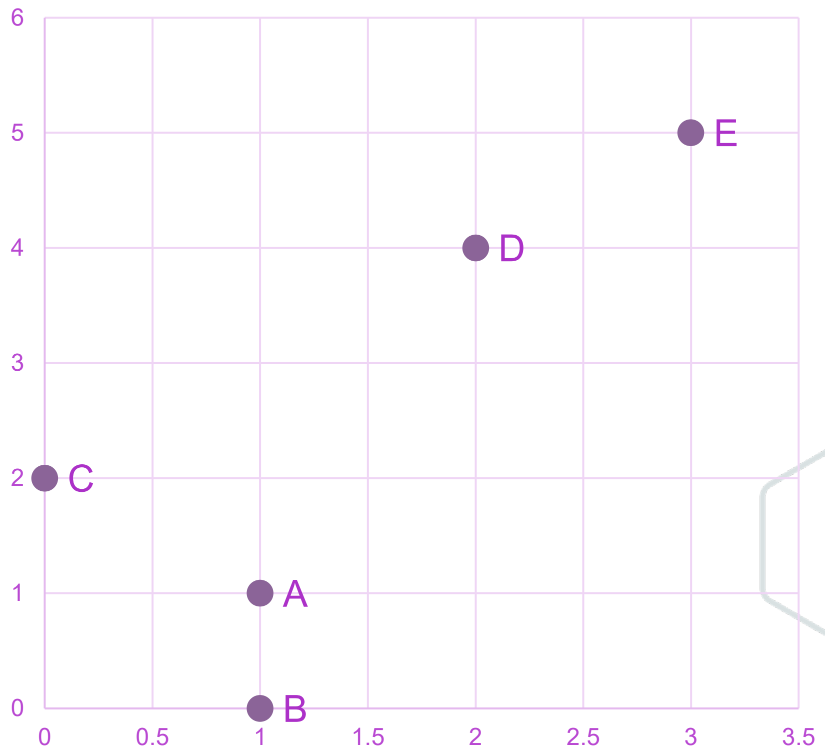
1. Make each data point in the dataset D a cluster,
2. Compute all pair-wise distances of $x1, x2, \ldots, xn \in$
3. **repeat**
4. find two clusters that are nearest to each other;
5. merge the two clusters to form a new cluster c
6. compute the distance from c to all other clusters;
7. **until** there is only one cluster left

PSEUDOCODE

Step by step…

1) Load in dataset

	x	y
Dps	**sepal length (cm)**	**Petal length (cm)**
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5



2) Build distance matrix and identify smallest distance

	A	B	C	D	E
A	0	1	1.4	3.2	4.5
B	1	0	2.2	4.1	5.4
C	1.4	2.2	0	2.8	4.2
D	3.2	4.1	2.8	0	1.4
E	4.5	5.4	4.2	1.4	0

3) Perform merge and update distance matrix

Updated distance matrix:

	AB	C	D	E
AB	0			
C	2.2	0		
D	4.1	2.8	0	
E	5.4	4.2	1.4	0

d[(A,B),C] = max {d(A,C), d(B,C)}
= max {1.4, 2.2}

d[(A,B),D] = max {d(A,D), d(B,D)}
= max {3.2, 4.1}

d[(A,B),E] = max {d(A,E), d(B,E)}
= max {4.5, 5.4}

Continue merging and updating the distance matrix…

	AB	DE	C
AB	0		
DE	5.4	0	
C	2.2	4.2	0

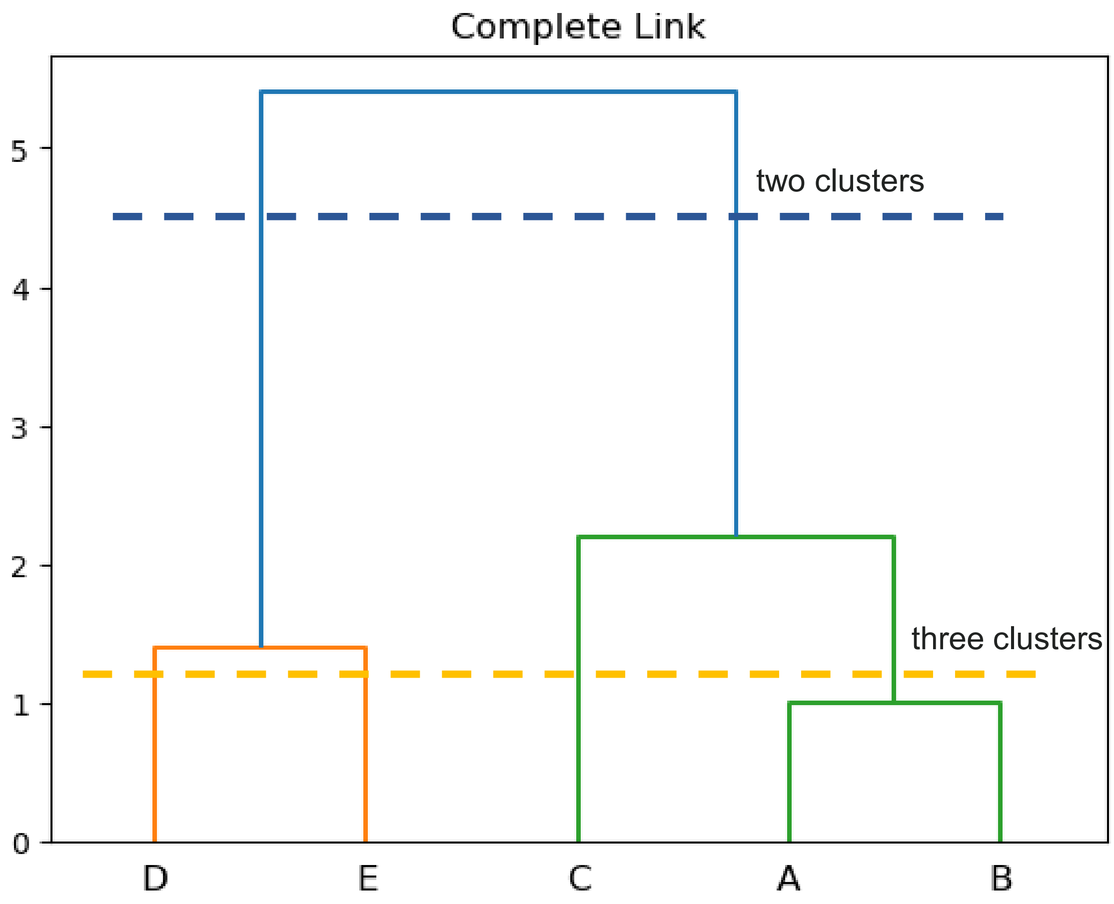
	ABC	DE
ABC	0	
DE	5.4	0

d[(A,B),(D,E)] = max {d((A,B)D), d(A,B)E))}
= max {4.1, 5.4}

d[(C,(D,E))] = max {d(C,D), d (C,E)}
= max {2.8, 4.2}

d[(A,B,C),(D,E) = max {d((D,E)(A,B), ((D,E,(C))
= max {5.4, 4.2}

RESULT

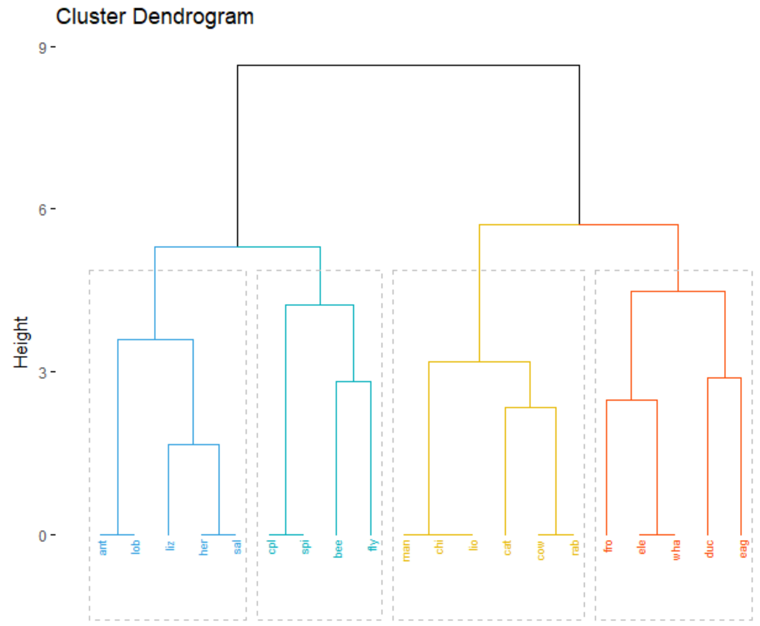


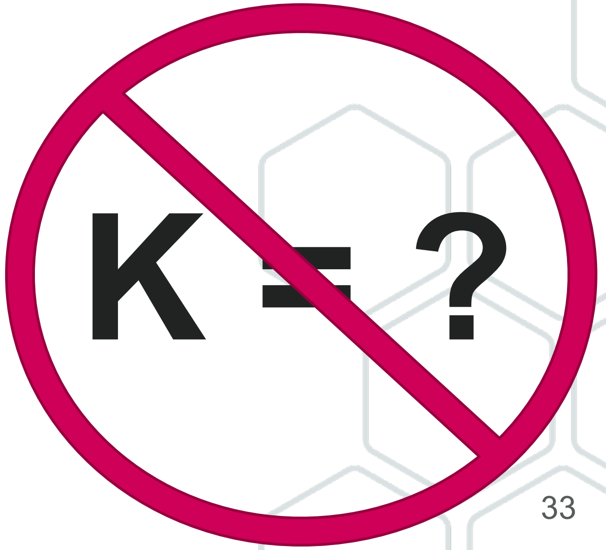
- Dendrogram: y-axis denotes when in the agglomerative algorithm two clusters get merged

- Y-axis also shows how far apart the merged clusters are → pay attention to the length of the branches

What are the strengths?

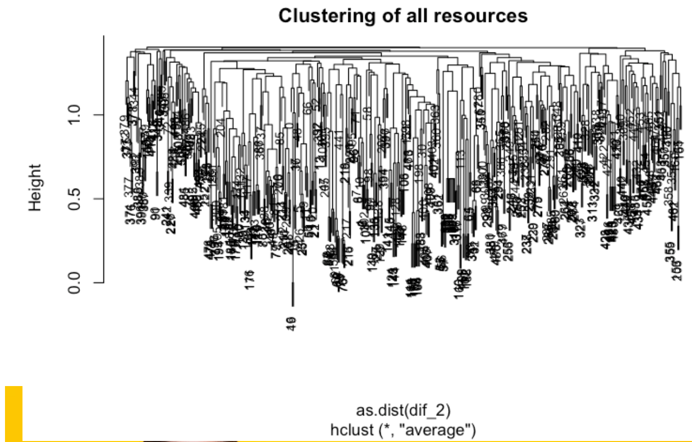
- Easy to understand and implement
- Most appealing output
- Can handle non-convex clusters
- No need to specify the number of clusters!





What are the limitations?

- Mathematically simple…but computationally expensive!
- Hard to visualize results with a large dataset
- Heavily driven by heuristics and arbitrary decisions
- Algorithm can’t undo previous step



K-Means vs Hierarchical clustering

	K-Means	Hierarchical clustering
Time complexity	O(n)	$O(n^2)$
Hyperparameters Tuning	Must specify the number of clusters (k) and retrain model for each k	No need to specify k value, can perform split wherever
Data structure	Better performance when dealing with convex clusters	Generates better results when dealing with non-convex clusters
Types/variations	Many variations (e.g., K-median, K-medoid)	Two approaches: Agglomerative and Divisive
Result Robustness	Result may be different on different runs	Same parameters generate the same result every time

References

Geron, A (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.

Müller, A.C, and Guido, S (2017). Introduction to Machine Learning with Python: A Guide for Data Scientists, O'Reilly Media, Inc.

Reddy, C.K., and Vinzamuri, B (2015). A Survey of Partitional and Hierarchical Clustering Algorithms.

- Outlier detection using clustering (Outlier detection image) https://blogs.sap.com/2020/12/16/outlier-detection-by-clustering/
- Image compression using K-Means clustering (parrot image) https://medium.com/@agarwalvibhor84/image-compression-using-k-means-clustering-8c0ec055103f
- Clustering Algorithms: Types of Clustering (images) https://developers.google.com/machine-learning/clustering/clustering-algorithms
- K-Means clustering (image with centroids as triangles)
- K-Means pseudocode (image) https://www.cms.waikato.ac.nz/~abifet/book/chapter_9.html#rfig9-1
- K-Means Elbow Method and Silhouette Analysis (image) https://stackabuse.com/k-means-elbow-method-and-silhouette-analysis-with-yellowbrick-and-scikit-learn/

UK Data Service

Material for Tuesday the 2nd of November

GitHub:

https://github.com/UKDataServiceOpen/ML_Workshop

UK Data Service

Survey

- When you leave the webinar, please complete our short survey
- Just click on ‘continue’ to access the survey.

Thank You.

Email:
louise.capener@Manchester.ac.uk
nadia.kennar@manchester.ac.uk,

Twitter:
@CapenerLouise
@NadiaKennar