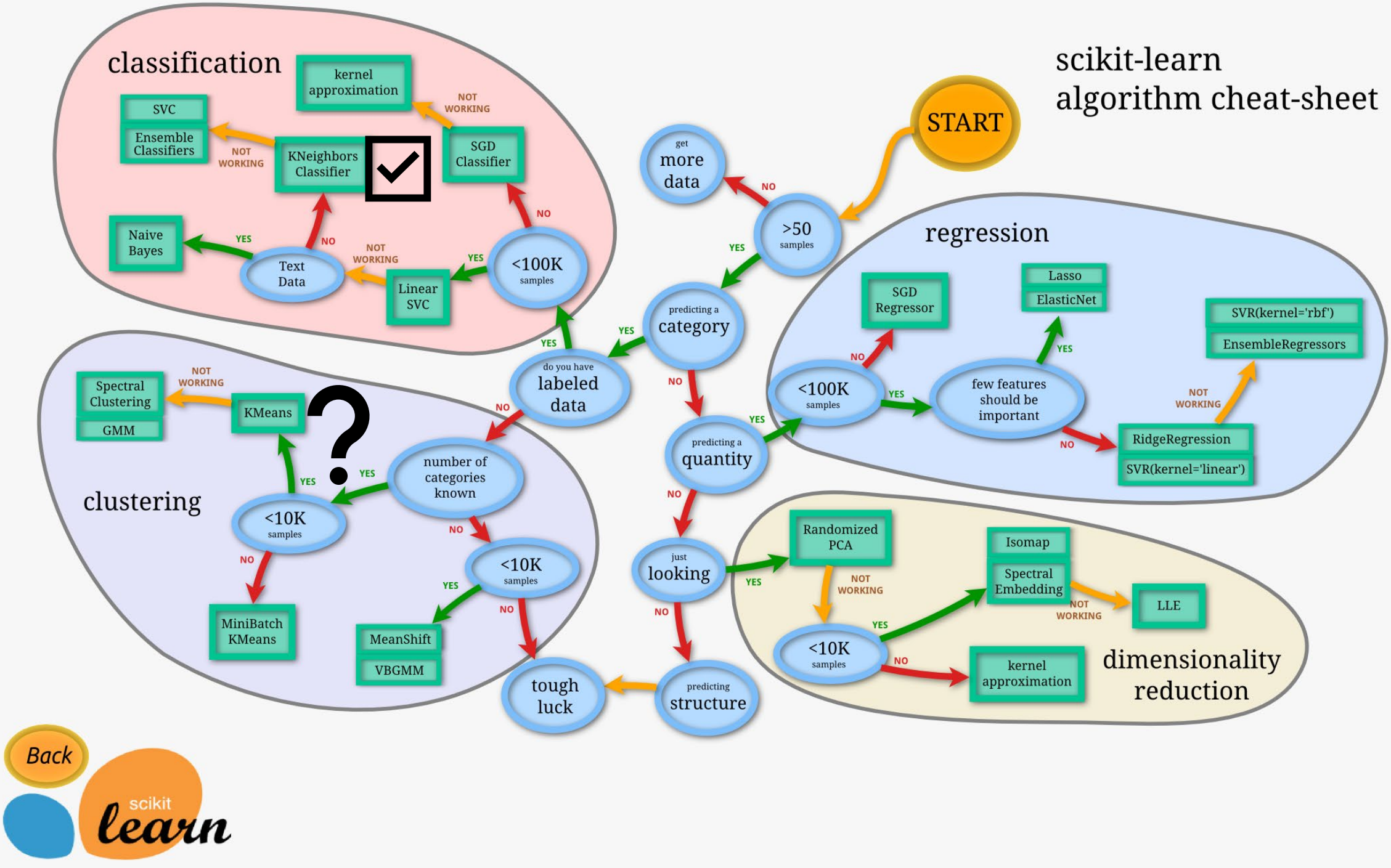
EM 538-001: PRACTICAL MACHINE LEARNING FOR ENGINEERING ANALYTICS

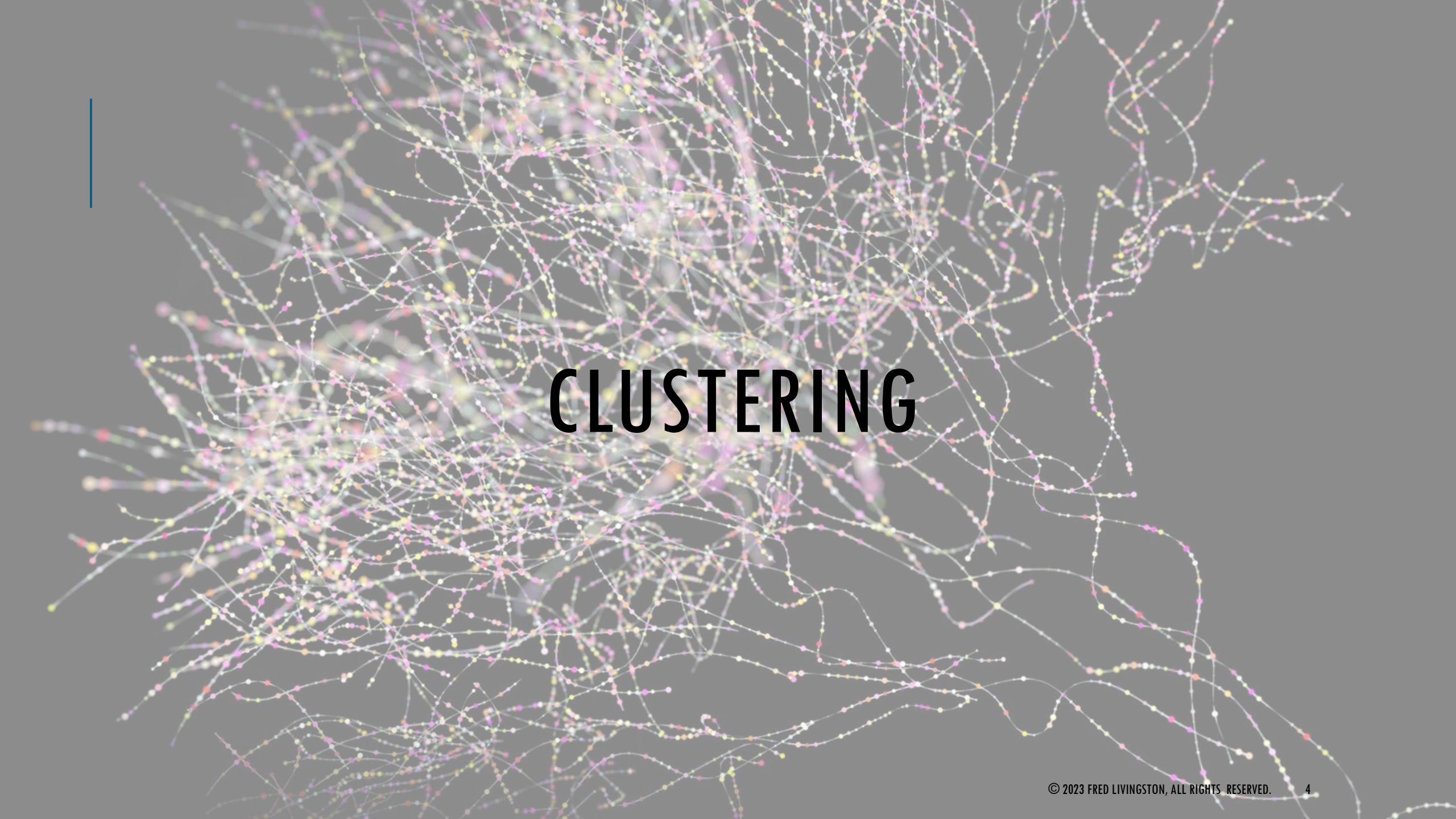
LECTURE 008

Fred Livingston, Ph.D.

NEAREST NEIGHBOOR MACHINE LEARNING MODELS

- Unsupervised Learning – Clustering
- Kmean Worked Example
- Advance Clustering Models


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
NO
do you have labeled data
YES
NO
<100K samples
YES
NO
Linear SVC
NOT WORKING
Text Data
YES
NO
Naive Bayes
YES
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
kernel approximation
NOT WORKING
SGD Classifier
classification
predicting a quantity
YES
NO
<100K samples
NO
YES
SGD Regressor
Lasso
ElasticNet
few features should be important
YES
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
regression
number of categories known
YES
NO
<10K samples
YES
NO
KMeans
NOT WORKING
Spectral Clustering
GMM
MiniBatch KMeans
<10K samples
YES
NO
MeanShift
VBGMM
clustering
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
<10K samples
YES
NO
kernel approximation
dimensionality reduction
predicting structure
tough luck
Back
scikit learn

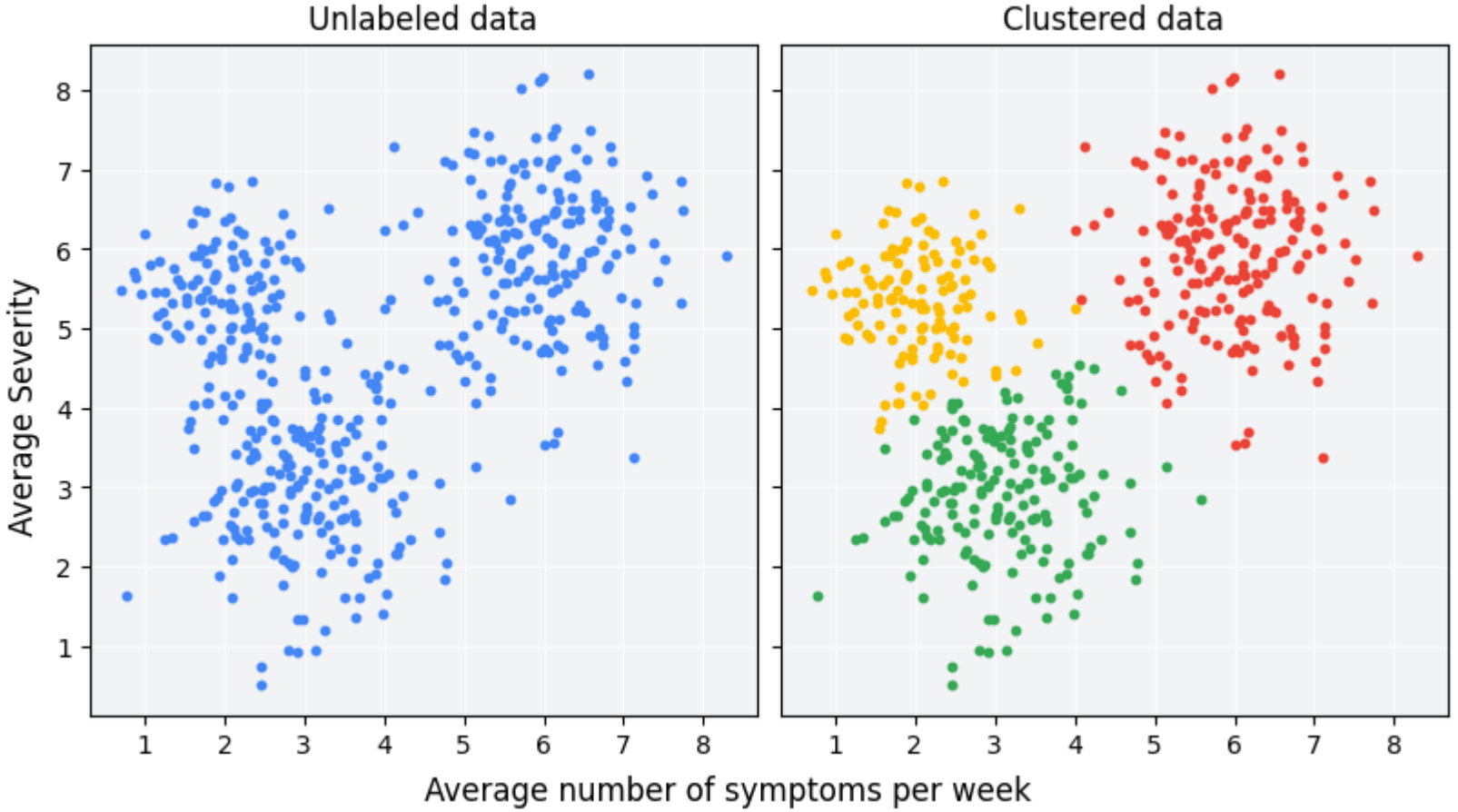


CLUSTERING

WHAT IS CLUSTERING

Clustering is an unsupervised machine learning technique designed to group **unlabeled examples** based on their similarity to each other.

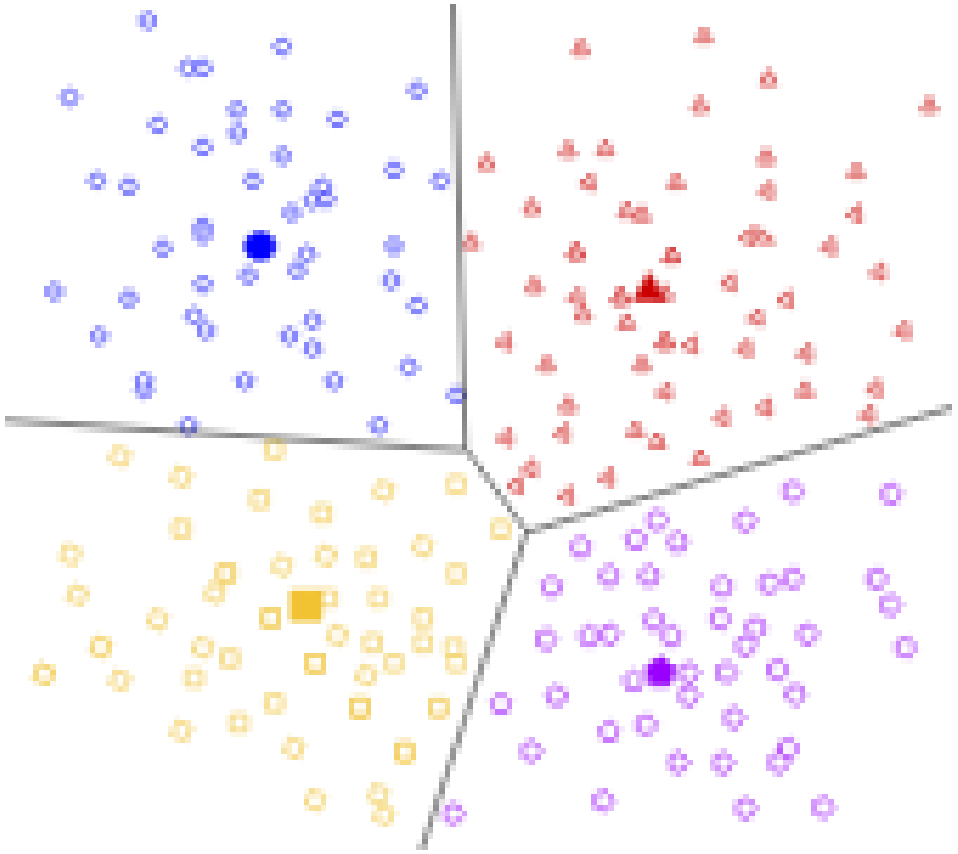
WHAT IS CLUSTERING



After clustering, each group is assigned a unique label called a **cluster ID**. Clustering is powerful because it can simplify large, complex datasets with many features to a single cluster ID.

CENTROID-BASED CLUSTERING

The centroid of a cluster is the arithmetic mean of all the points in the cluster. **Centroid-based clustering** organizes the data into non-hierarchical clusters. Centroid-based clustering algorithms are efficient but sensitive to initial conditions and outliers. Of these, k-means is the most widely used. It requires users to define the number of centroids, *k*, and works well with clusters of roughly equal size.



K-MEANS CLUSTERING

As previously mentioned, many clustering algorithms don't scale to the datasets used in machine learning, which often have millions of examples. For example, agglomerative or divisive hierarchical clustering algorithms look at all pairs of points and have complexities of O(n^2log(n)) and O(n^2), respectively.

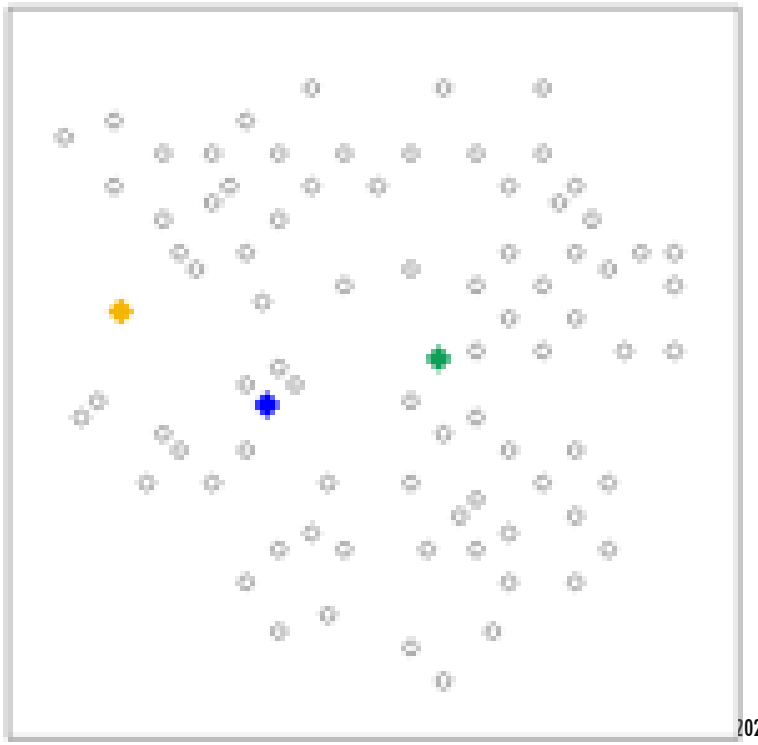
k-means scales as O(nk), where k is the number of clusters chosen by the user. This algorithm groups points into k clusters by minimizing the distances between each point and its cluster's centroid

As a result, k-means effectively treats data as composed of a number of roughly circular distributions, and tries to find clusters corresponding to these distributions. But real-world data contains outliers and density-based clusters and might not match the assumptions underlying k-means.

K-MEANS CLUSTERING ALGORITHM

The algorithm follows these steps:

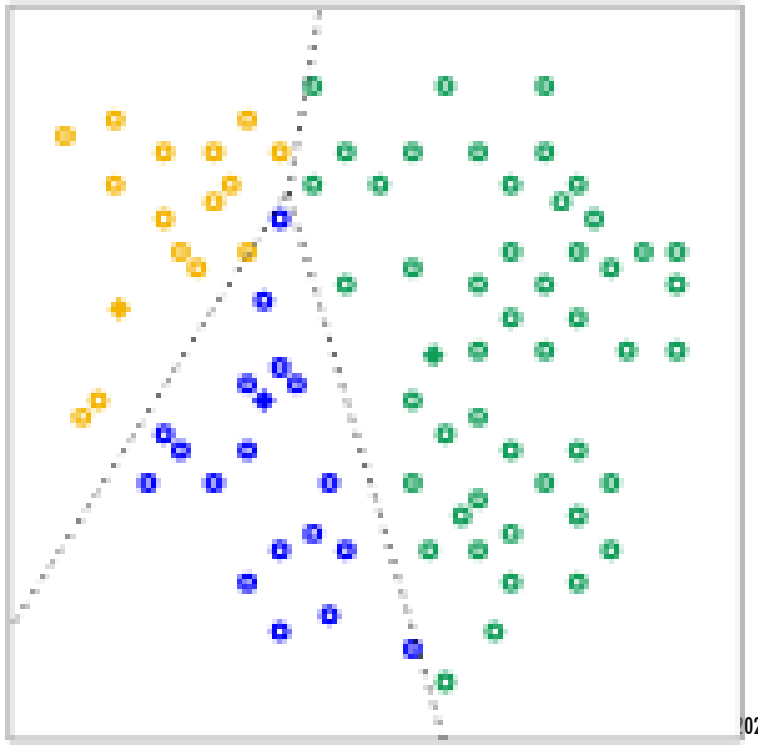
1. Provide an initial guess for **k**, which can be revised later. For this example, we choose **k=3**.
2. Randomly choose **k** centroids.



K-MEANS CLUSTERING ALGORITHM

The algorithm follows these steps:

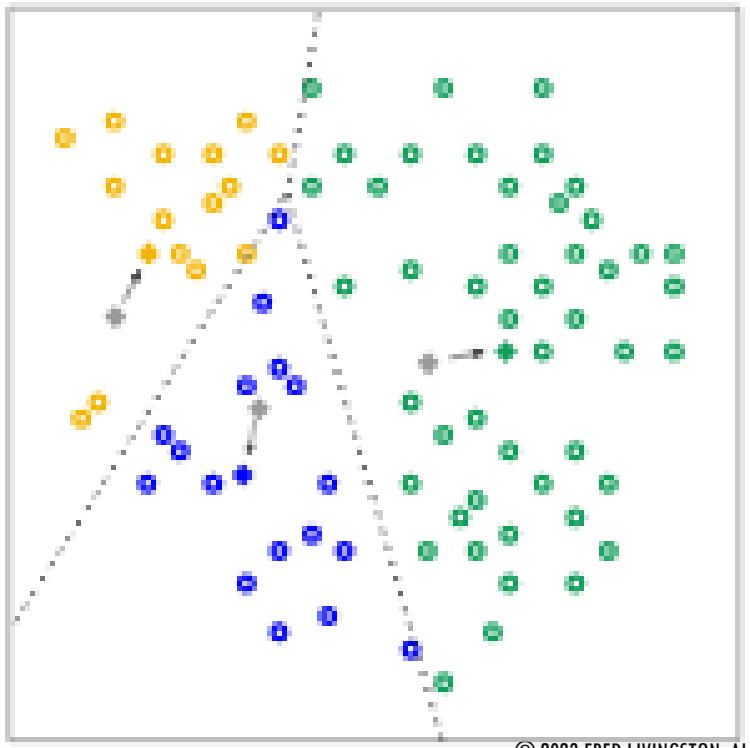
3. Assign each point to the **nearest centroid** to get **k** initial clusters.



K-MEANS CLUSTERING ALGORITHM

The algorithm follows these steps:

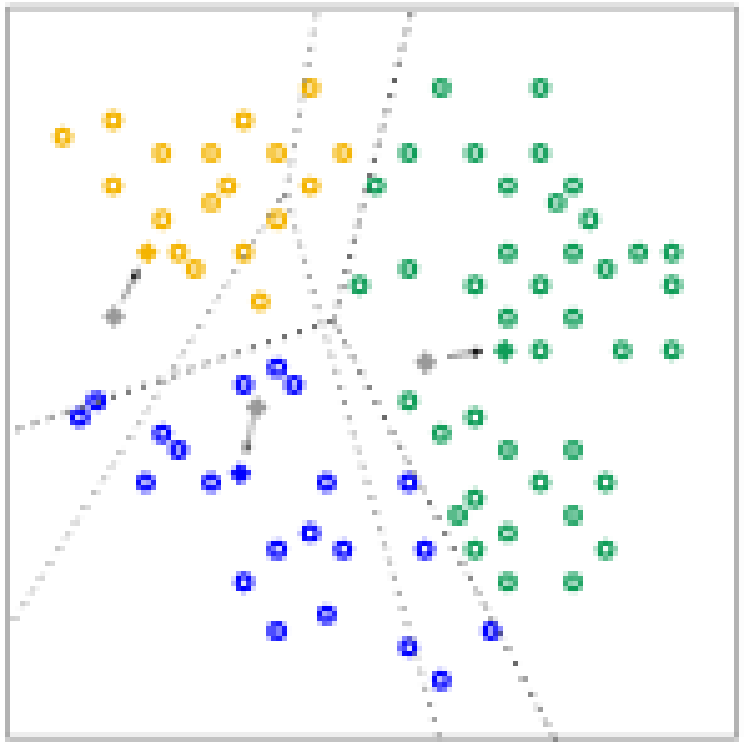
4. For each cluster, calculate a new centroid by taking the mean position of all points in the cluster. The arrows show the change in centroid positions.



K-MEANS CLUSTERING ALGORITHM

The algorithm follows these steps:

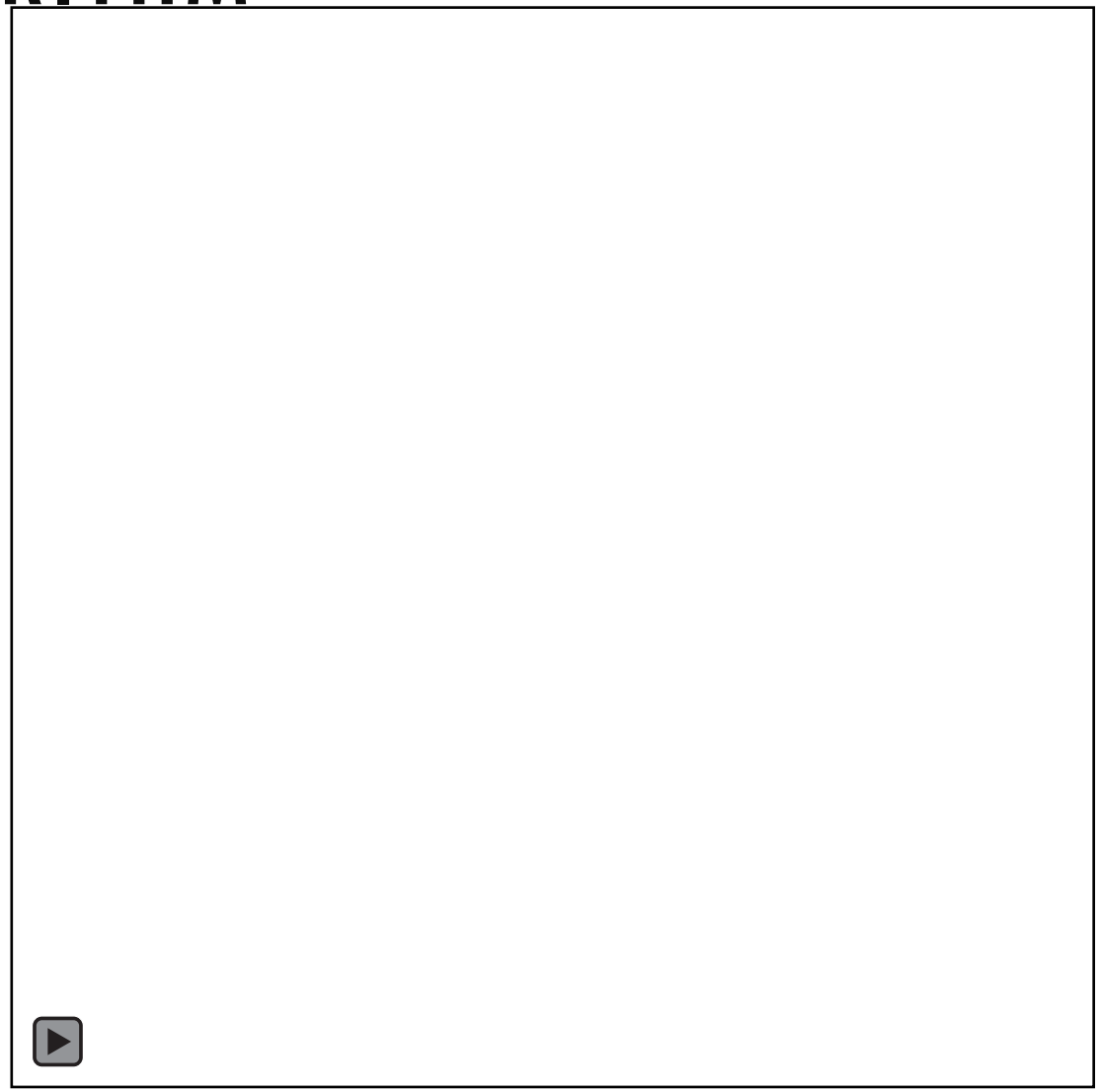
5. Reassign each point to the nearest new centroid.



K-MEANS CLUSTERING ALGORITHM

The algorithm follows these steps:

6. Repeat steps 4 and 5, recalculating centroids and cluster membership, until points no longer change clusters. In the case of large datasets, you can stop the algorithm before convergence based on other criteria.



0-kmean_clustering_scikit.ipynb ●

G: > My Drive > Academics > Machine Learning > Spring 2025 > Lectures > Lecture 008 > lecture8_code_examples > 0-kmean_clustering_scikit.ipynb > Load and Prepare Datasets > import seaborn as sns

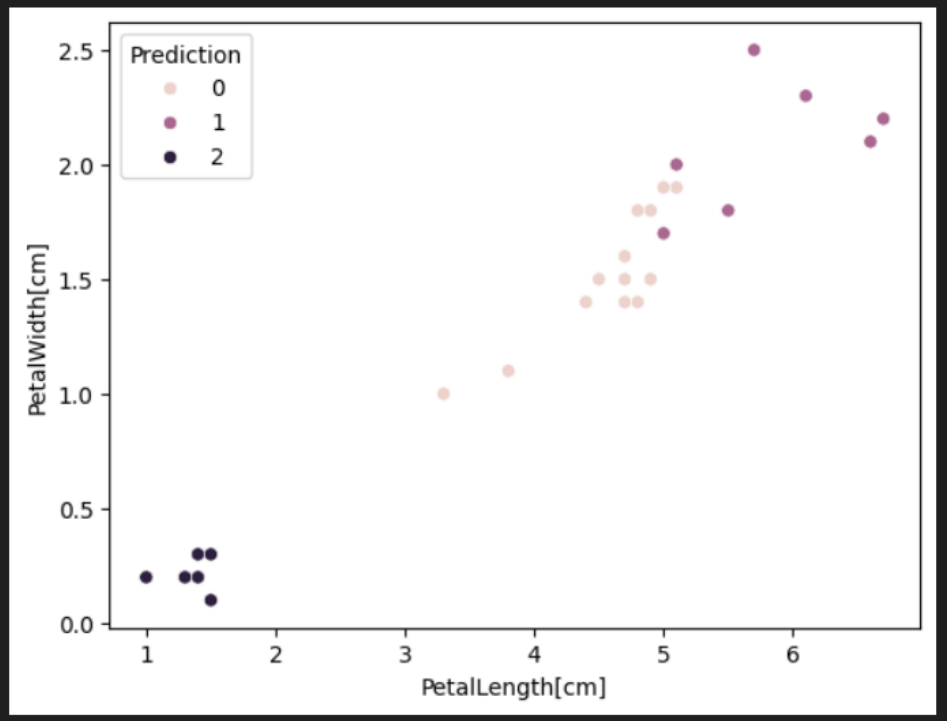
Generate | + Code | + Markdown | Run All | Restart | Clear All Outputs | View data | Jupyter Variables | Outline | ... | pyml (Python 3.11.9)

144	6.7	3.3	5.7	2.5	1
110	6.5	3.2	5.1	2.0	1
38	4.4	3.0	1.3	0.2	2

```
import seaborn as sns
sns.scatterplot(data=X_test, y="PetalWidth[cm]", x="PetalLength[cm]", hue='Prediction')
```

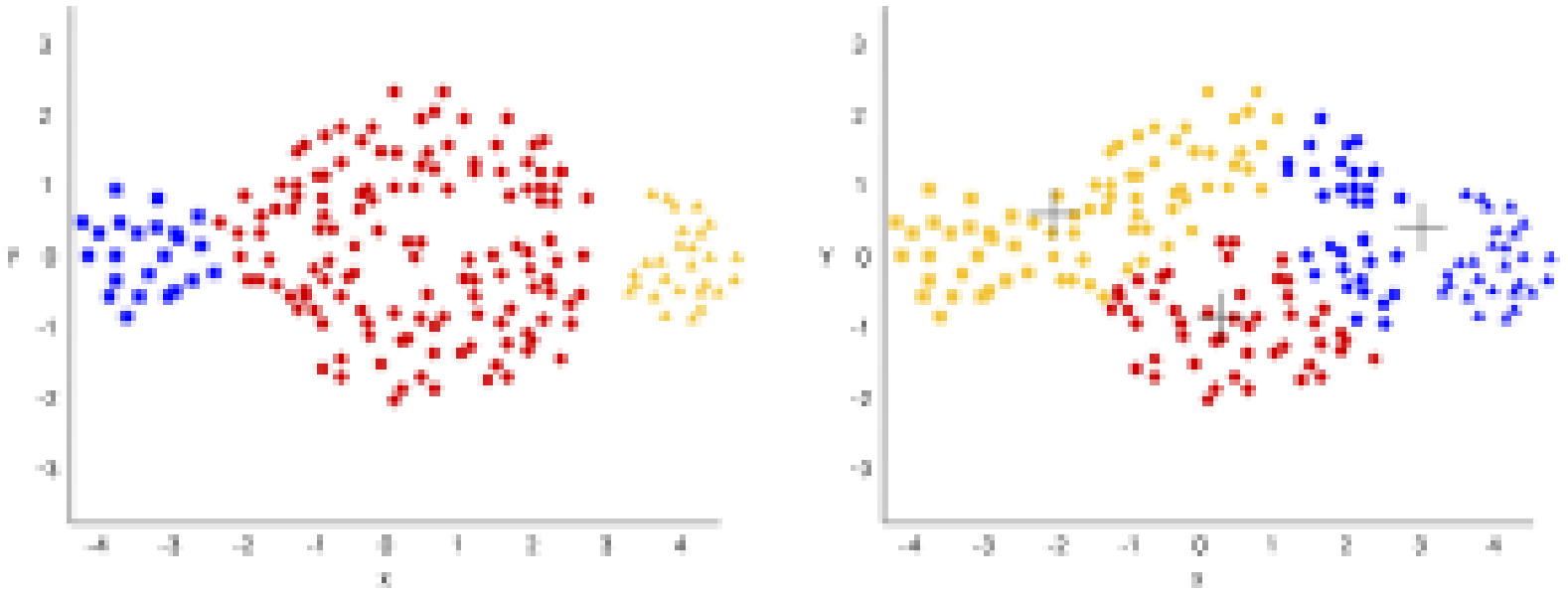
[132] ✓ 0.2s Python

```
<Axes: xlabel='PetalLength[cm]', ylabel='PetalWidth[cm]'>
```

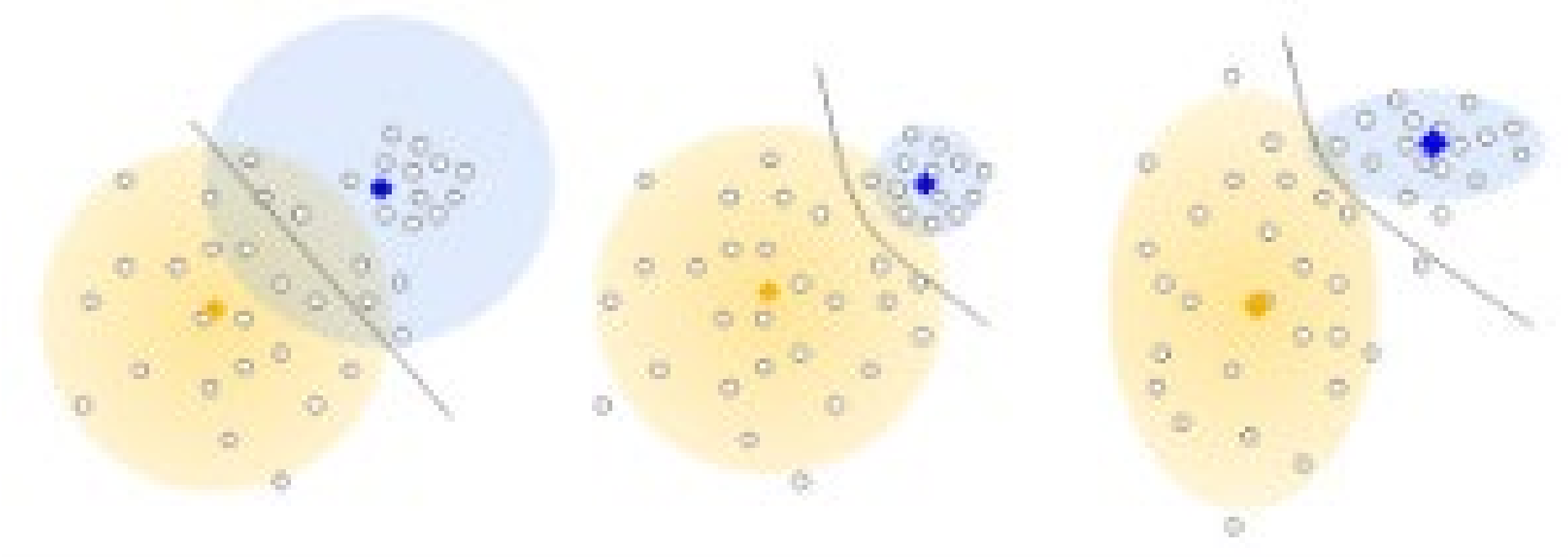


GENERALIZING K-MEANS

A straightforward implementation of k-means can struggle with clusters of different densities and sizes. The left side shows the clusters we'd expect to see, while the right side shows the clusters proposed by k-means.



GENERALIZING K-MEANS



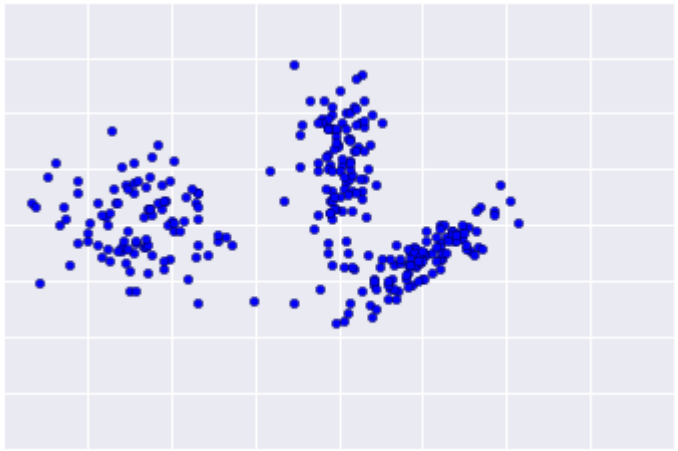
Plain k-means

Varying widths across clusters

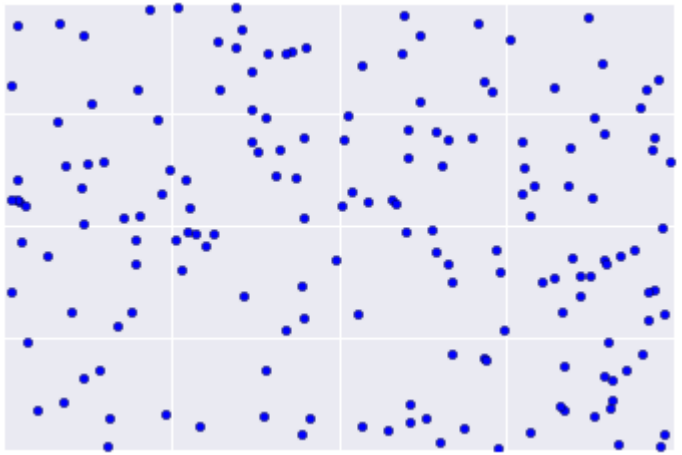
Varying widths across clusters & dimensions

EVALUATING RESULTS

Because clustering is unsupervised, no ground truth is available to verify results. The absence of truth complicates assessments of quality.

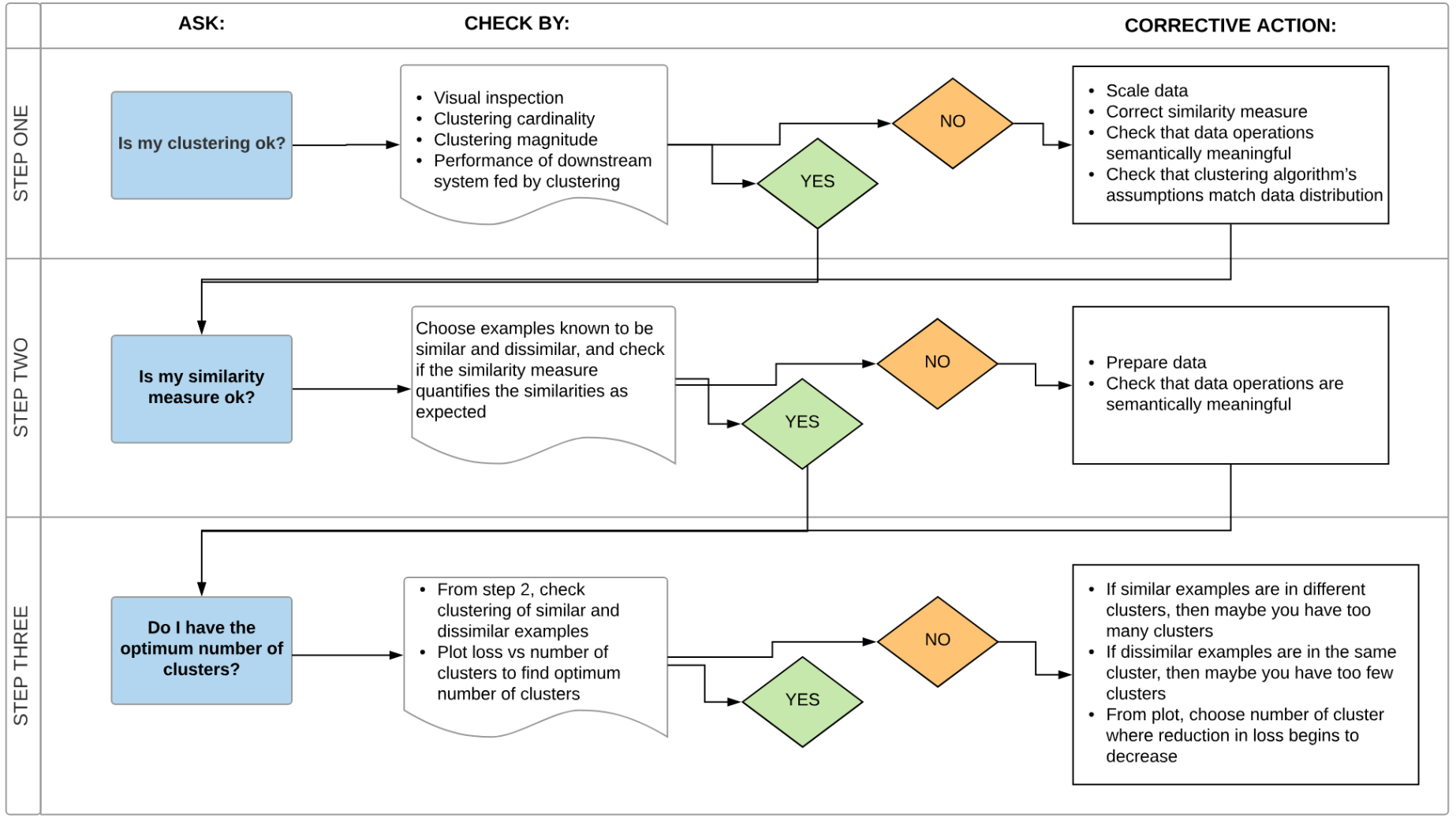


Ideal



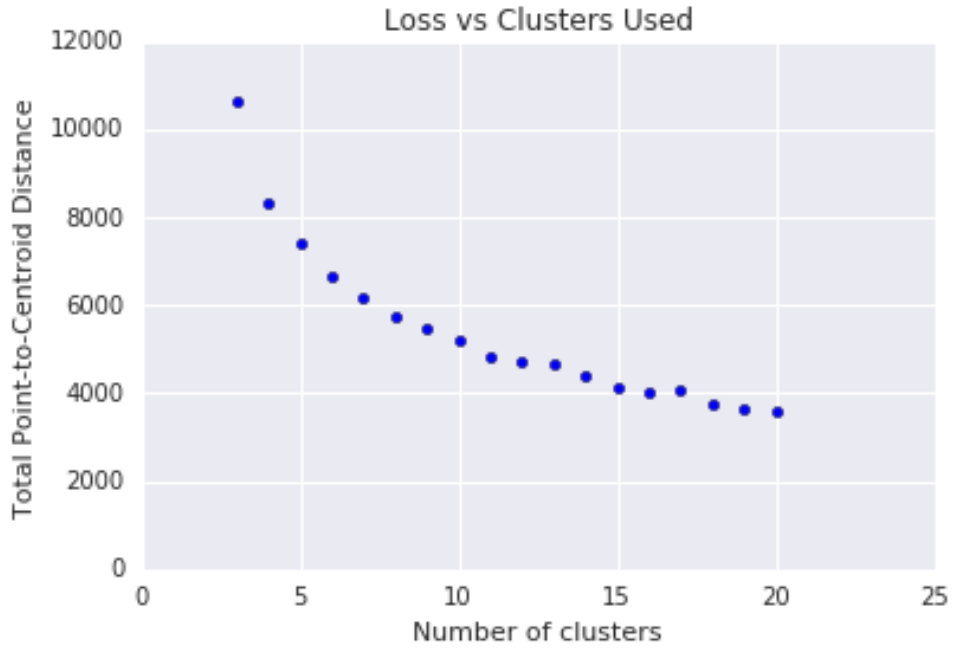
Realist Data Plot

BEST PRACTICES



FIND THE OPTIMAL NUMBER OF CLUSTERS

k-means requires you to decide the number of clusters k beforehand. How do you determine an optimal k? Try running the algorithm with increasing values of and note the sum of all cluster magnitudes. As k increases, clusters become smaller, and the total distance of points from centroids decreases. We can treat this total distance as a loss. Plot this distance against the number of clusters.



ADVANTAGES OF K-MEANS

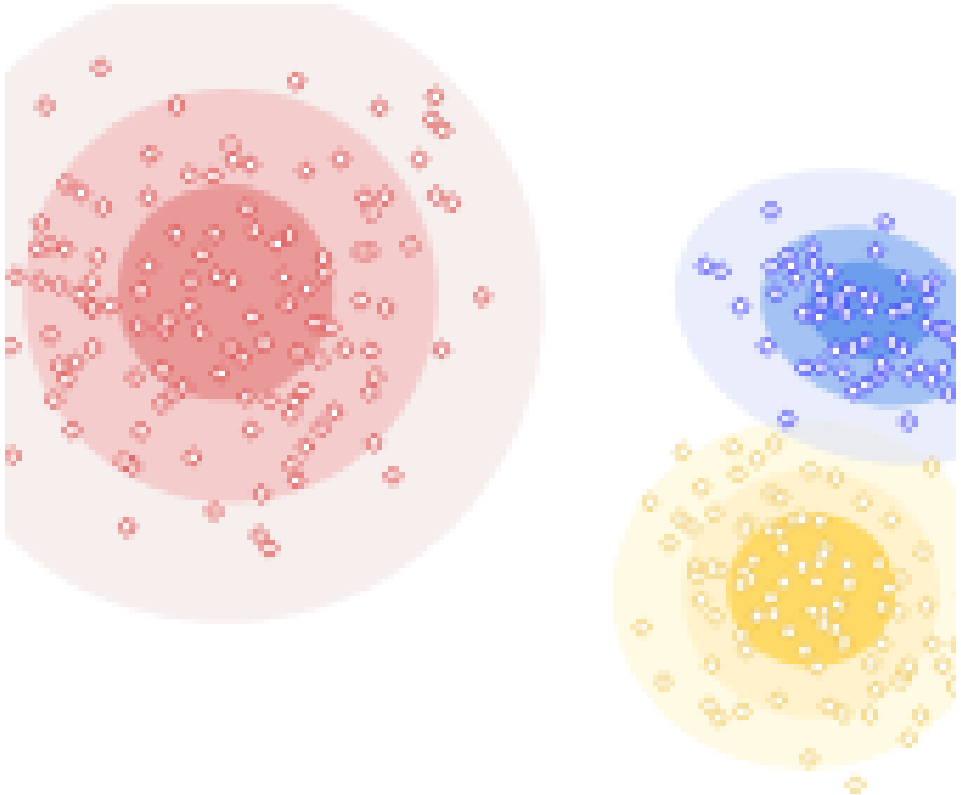
- Relatively simple to implement.
- Scales to large data sets.
- Always converges.
- Allows warm-starting the positions of centroids.
- Smoothly adapts to new examples.
- Can be generalized to clusters of different shapes and sizes, such as elliptical clusters.

DISADVANTAGES OF K-MEANS

- K must be chosen manually
- Results depend on initial value
- Difficulty clustering data of varying sizes and densities without generalization
- Allows warm-starting the positions of centroids.
- Difficulty clustering outliers.
- Difficulty scaling with number of dimensions

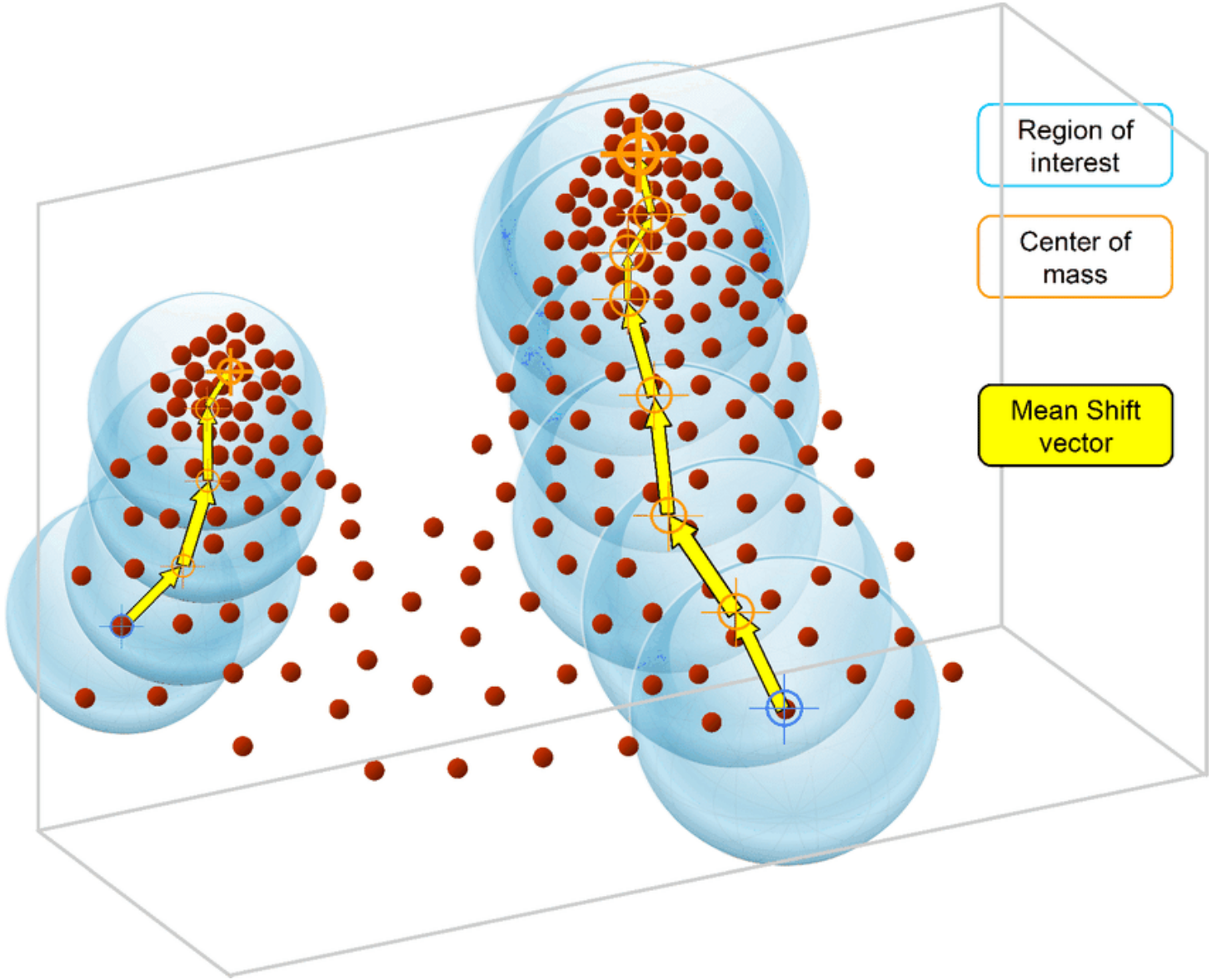
DISTRIBUTION-BASED CLUSTERING

This clustering approach assumes data is composed of probabilistic distributions, such as **Gaussian distributions**. As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases. The bands show that decrease in probability.



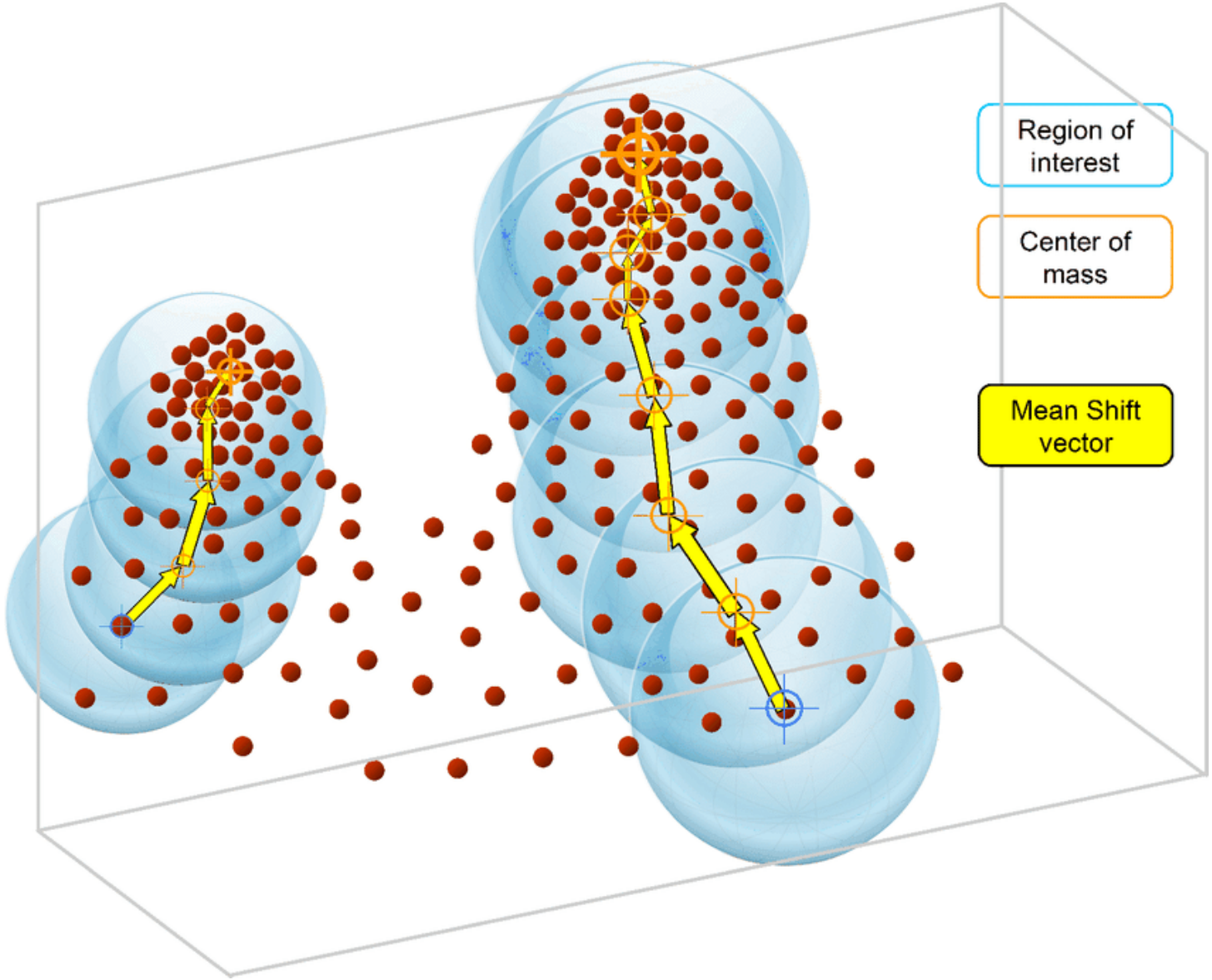
MEANSHIFT

Unlike the K-Means algorithm, MeanShift algorithm does not require specifying the number of clusters. The algorithm itself automatically determines the number of clusters, which is a pretty big advantage over K-Means if you are not sure about patterns in data.



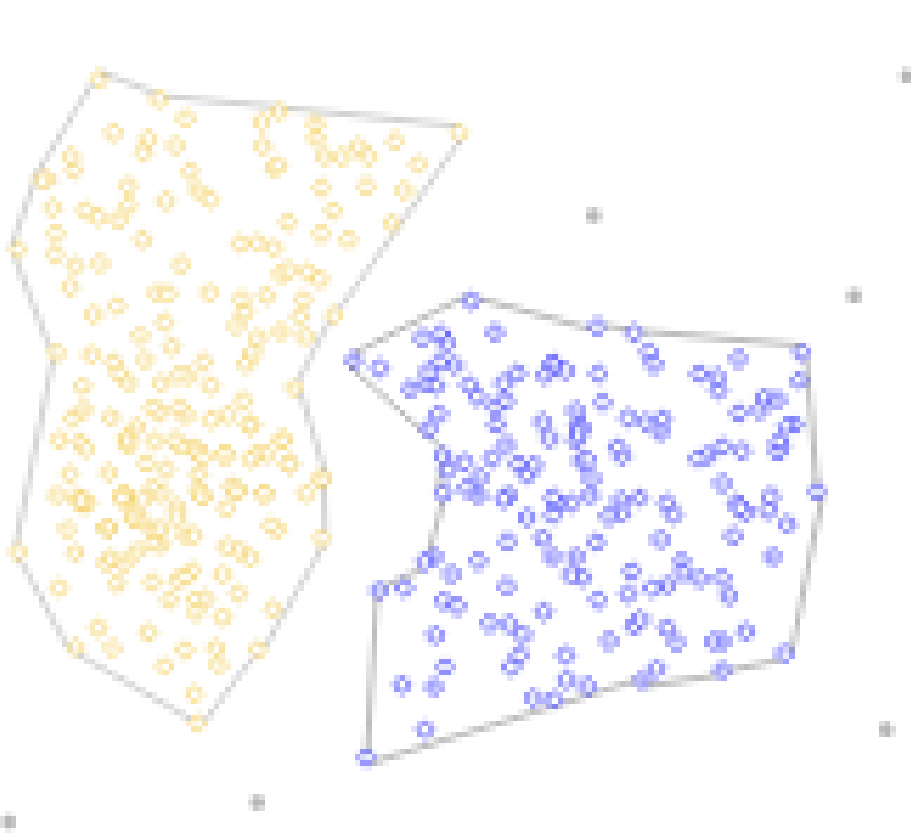
MEANSHIFT

Unlike the K-Means algorithm, MeanShift algorithm does not require specifying the number of clusters. The algorithm itself automatically determines the number of clusters, which is a pretty big advantage over K-Means if you are not sure about patterns in data.



DENSITY-BASED CLUSTERING

Density-based clustering connects contiguous areas of high example density into clusters. This allows for the discovery of any number of clusters of any shape. Outliers are not assigned to clusters. These algorithms have difficulty with clusters of different density and data with high dimensions.



DBSCAN

DBSCAN or **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise is an unsupervised clustering algorithm that works on the premise that clusters are dense spaces in the region separated by lower-density regions.

The biggest advantage of this algorithm over K-Means and MeanShift is that it is robust to outliers meaning it will not include outliers data points in any cluster.

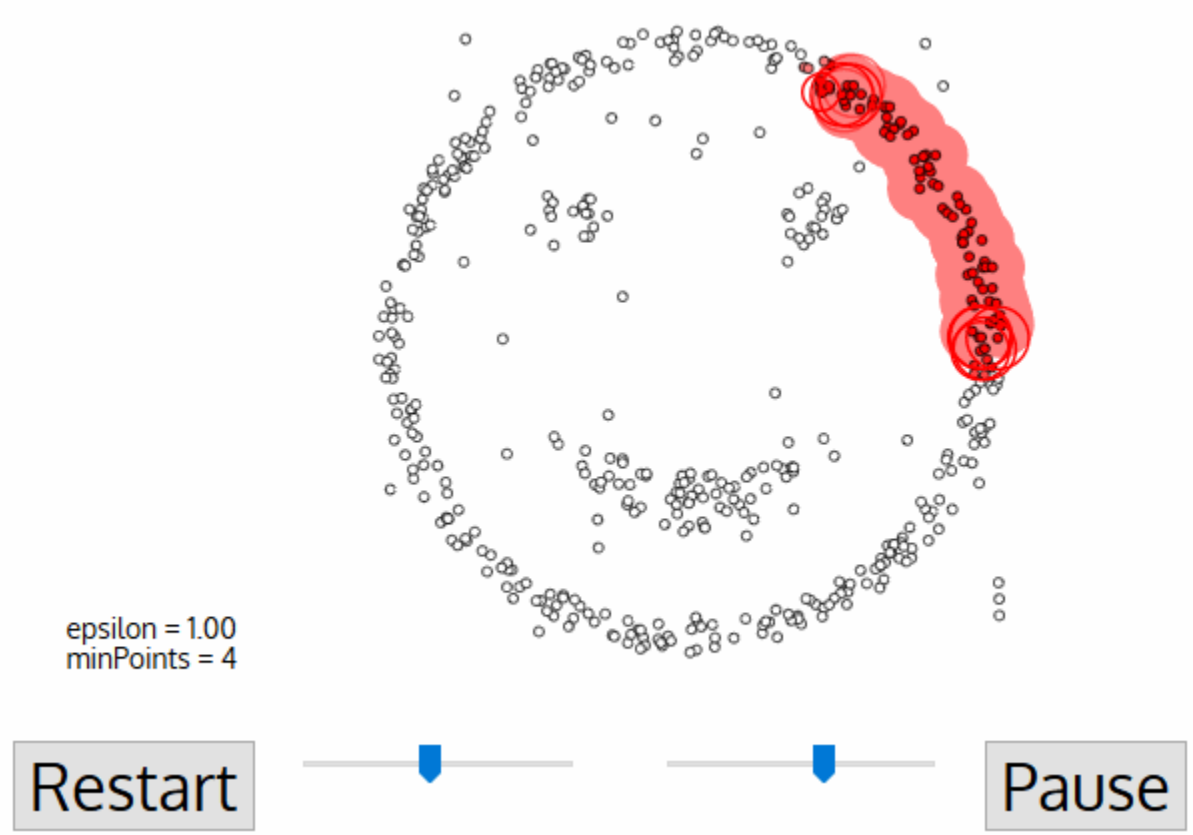
DBSCAN algorithms require only two parameters from the user:

- The radius of the circle to be created around each data point, also known as `epsilon`
- minPoints which defines the minimum number of data points required inside that circle for that data point to be classified as a Core point.

DBSCAN

Every data point is surrounded by a circle with a radius of epsilon, and DBSCAN identifies them as being either a Core point, Border point, or Noise point. A data point is considered to be a Core point if the circle that surrounds it has a minimum number of points specified by minPoints parameter.

It is considered a Border Point if the number of points is lower than the minimum required, and it is considered Noise if there are no additional data points located within an epsilon radius of any data point. Noise data points are not categorized in any cluster (basically, they are outliers).



0-kmean_clustering_scikit.ipynb ● 1-other_clustering_scikit.ipynb ●

G: > My Drive > Academics > Machine Learning > Spring 2025 > Lectures > Lecture 008 > lecture8_code_examples > 1-other_clustering_scikit.ipynb > ...

Generate + Code + Markdown | Run All Restart Clear All Outputs | View data Jupyter Variables Outline ··· pyml (Python 3.11.9)

```
    plt.show()
```

[ ] Python

DBSCAN clustering algorithm

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

db = DBSCAN(eps=0.3, min_samples=10).fit(X)
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)
```

[ ] Python

```
unique_labels = set(labels)
core_samples_mask = np.zeros_like(labels, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True

colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = labels == k
```


Q/A?

