**Clustering and Classifying Text**

**Clustering text?:**

How to use machine learning methods to move on to tasks such as clustering and classification.

So far we analyzed text to understand better what the text or corpus consists of.

* POS-tag or NER-tag, helped us know what kind of words were presented in our documents,
* topic-modelling, extracted the underlying topics which could be hidden in our texts.

We could **use** our **topic models** to **attempt** to **cluster** articles, but that isn't its purpose; we would be silly to expect great results if we tried this, too. For example, after we perform topic modeling, a document can be made of 30% topic 1, 30% topic 2, and 40% topic 3. In such a case, we cannot use this information to cluster.

**Clustering** is an **unsupervised** learning problem. We are not aware of the **clusters** or **groups** before we start assigning our data points to them (though we might have an idea of what we might find).

The idea is to group together data points in the same group, where points in the same group are more similar to each other than points in other groups. In our context, data points can be thought of as documents, or in some cases, words.

**Classification** is a similar task and is the problem of identifying to which of a set of categories (subpopulations) a new observation belongs, by a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into spam or non-spam classes, or the task of assigning newspaper articles to predetermined classes or groups.

Clustering text implies **a higher number of dimensions** in text analysis. In the Iris dataset, for example, there are only four features which we use to identify our classes or clusters. However, in the case of text, we have to deal with the entire vocabulary size when setting up our problem. Of course, we will do our best to reduce our dimensions using some of the techniques like **Singular Value Decomposition (SVD) and Latent Semantic Analysis (LSA/LSI)**

**Example Input:**

If X is a TF-IDF matrix generated from a text corpus with 1000 terms and 100 documents:

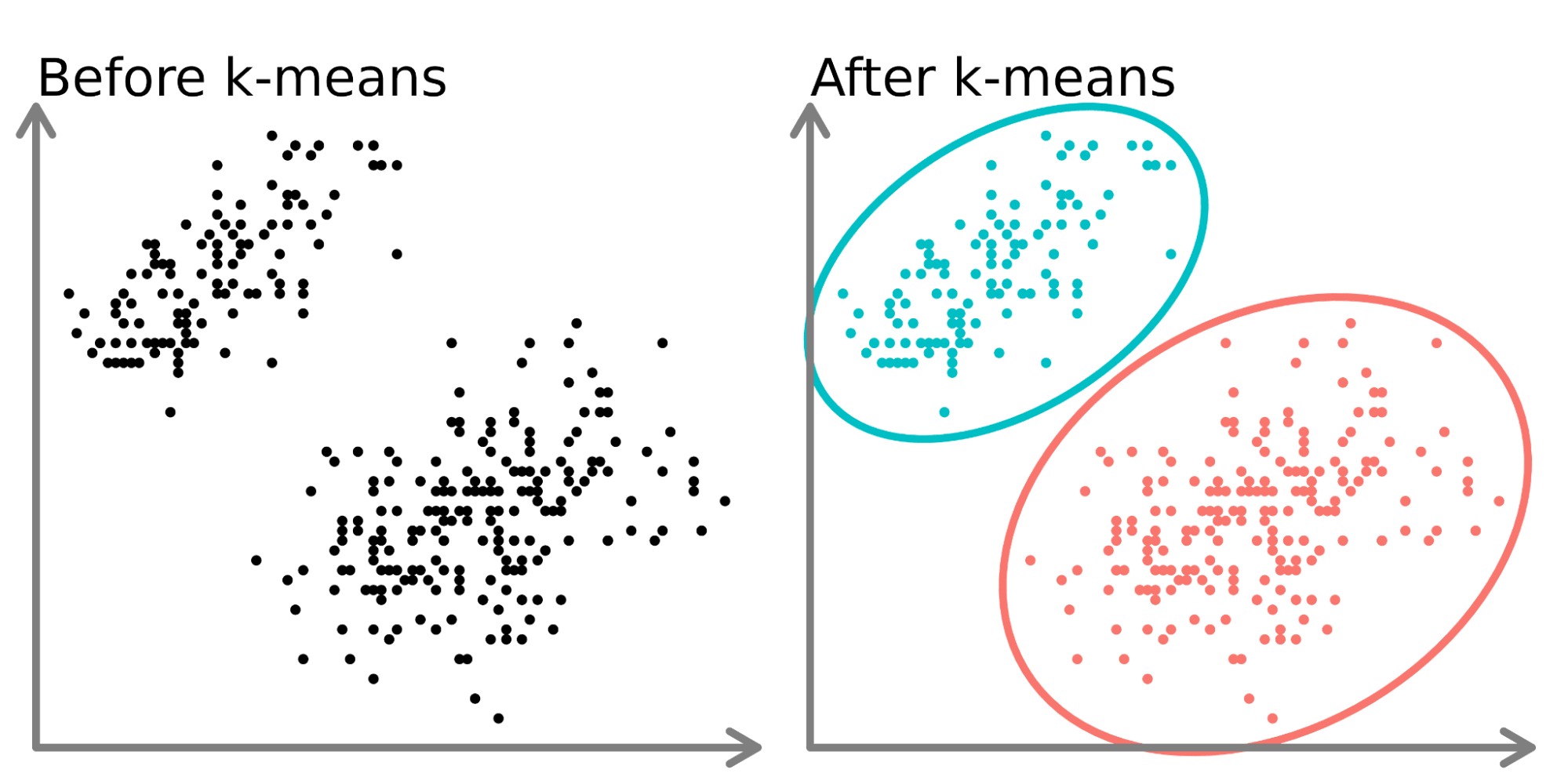
* After applying this pipeline with n\_components=5, X will be transformed into a 100×5 matrix, where each row represents a document in the reduced 5-dimensional semantic space.

**Clustering text using scikit-learn:**

While we perform our clustering and classifying tasks, you might often come across both **Word2Vec** and **Doc2Vec**, two ways of **representing** words and **documents as vectors**.

* Like every other text analysis algorithm, the most important step remains the **preprocessing** step — getting rid of our stop words and lemmatizing words.
* the next step is to convert our document into a **vector representation** we are most comfortable with
* **Reducing dimensions** to understand data and speeding up a machine learning (ML) Algorithm. For this we can used LSI, LDA, etc.

On sparse high-dimensional data such as text vectorized using the **Bag of Words** approach, k-means can initialize centroids on extremely isolated data points. Which leads to highly imbalanced clusters, depending on the random initialization.



To avoid this issue, one possibility is to increase the number of runs with independent random initiations **n\_init**. In such case the clustering with the best inertia (objective function of k-means) is chosen. A **n\_init=1** can still be used as long as the dimension of the vectorized space is reduced first to make k-means more stable. For such purpose we use [TruncatedSVD](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD), which works on term count/tf-idf matrices.

Using SVD to reduce the dimensionality of TF-IDF document vectors is often known as [latent semantic analysis](https://en.wikipedia.org/wiki/Latent_semantic_analysis) (LSA) in the information retrieval and text mining literature.

**Should you split your data into training and testing sets?**

The decision to split your data depends on what your goals are for clustering. If the goal is to cluster your data as the end of your analysis, then it is not necessary. If you are using the clusters as a feature in a supervised learning model or for prediction, then you will need to split your data before clustering.

**Homework:**

1. Use both Kmeans and Naïve Bayes on a dataset of **newspapers** and discuss the results and performance of both methods.
2. Compare the performance of algorithms using bag of words vs TFIDF
3. Test your best-performing model on new, unseen text inputs.