Practical Data Science: Clustering

Dr. Yongli Ren

(yongli.ren@rmit.edu.au)

Computer Science & IT

School of Science

Outline

- Part 1: Overview
- Part 2: k Means

Practical Data Science – COSC2670

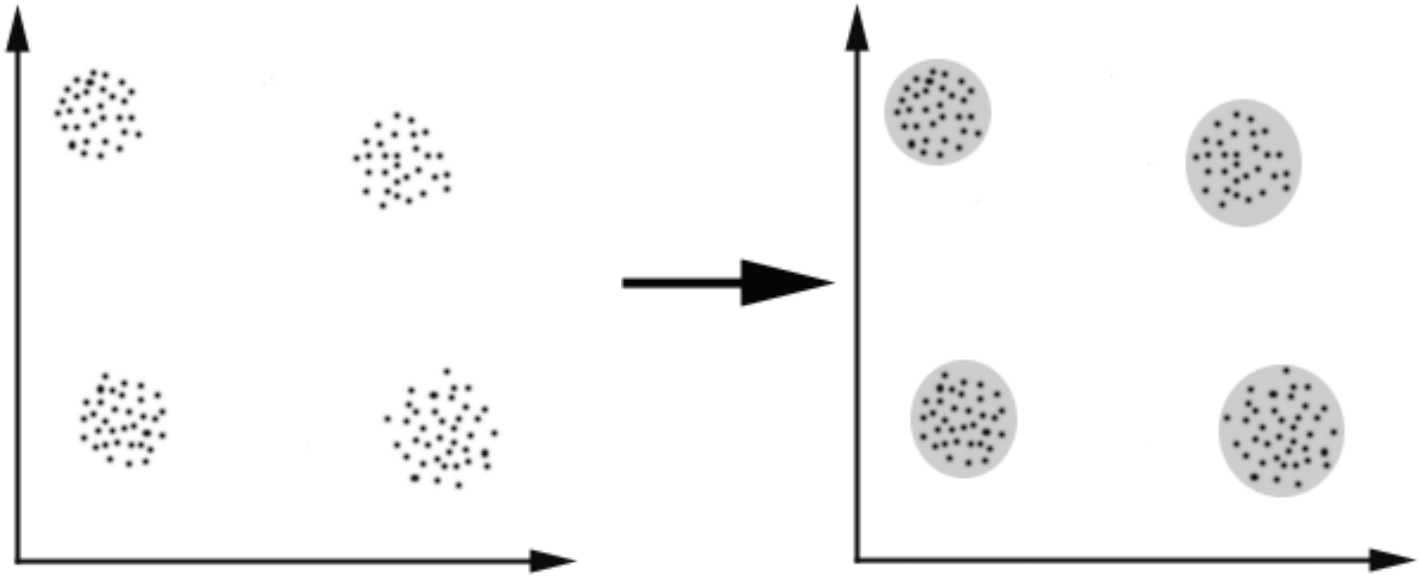
PART 1: OVERVIEW

What is Clustering?

- *Clustering* can be seen as the most important *unsupervised learning* problem
  - It deals with finding a *structure* in a collection of unlabelled data.
- A loose definition of clustering could be
  - "the process of organizing objects into groups whose members are similar in *some way*".
  - A *cluster* is therefore a collection of objects which
    - are "similar" between them and
    - are "dissimilar" to the objects belonging to other clusters.

What is Clustering?

- We can show this with a simple graphical example:



- The similarity criterion is *distance*:
  - two or more objects belong to the same cluster if they are "close"
    - according to *a given distance* (in this case geometrical distance).

What is Clustering - The Goal

- The goal of clustering is to determine *the intrinsic grouping* in a set of unlabelled data.
- But how to decide what constitutes a good clustering?
  - It can be shown that there is *no absolute "best" criterion*
    - which would be independent of the final aim of the clustering.
- Consequently, it is the user who must supply this criterion, in such a way that the result of the clustering will suit their needs.
- For instance, we could be interested
  - in finding representatives for homogeneous groups (*data reduction*),
  - in finding "natural clusters" and describe their unknown properties (*"natural" data types*),
  - in finding useful and suitable groupings (*"useful" data classes*) or
  - in finding unusual data objects (*outlier detection*).

Why Clustering?

- It's generally true that
  - *most large data sets don't have labels*,
  - so unless you sort through it all and give it labels, the supervised learning approach won't work.
- Instead, we must take the approach that will work with this data because
  - We can study the *distribution of the data* and infer truths about the data in different parts of the distribution.
  - We can study the *structure and values in the data* and infer new, more meaningful data and structure from it.

Why Clustering?

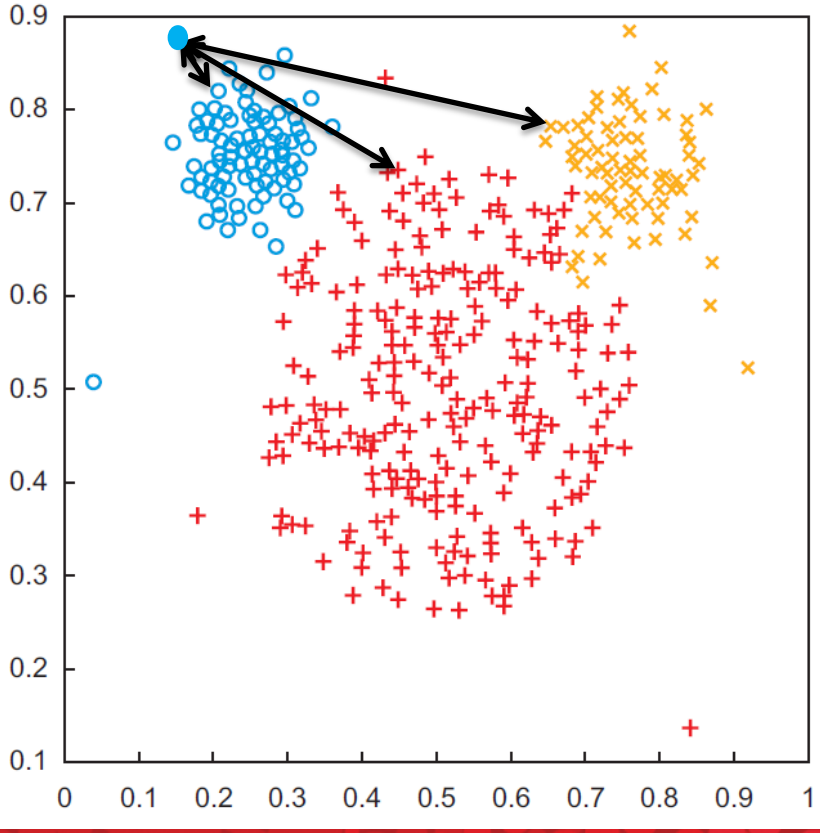
- Suppose for a moment you're building a website that
  - Recommends films to users based on
    - preferences they've entered and
    - films they've watched.
  - The chances are high that
    - if they watch many horror movies they're likely to want to know about new horror movies
    - and not so much about new teen romance films.
- By grouping together users
    - who've watched more or less the same films and
    - set more or less the same preferences,
  - *you can gain a good deal of insight into what else they might like to have recommended.*

Why Clustering?

- The general technique we're describing here is known as *clustering*.
- In this process,
  - we attempt to divide our data set into observation subsets, or *clusters*, wherein
    - *observations should be similar to those in the same cluster*
    - but *differ greatly from the observations in other clusters*.

Why Clustering?

- The figure below gives you a visual idea of what clustering aims to achieve.
- The circles in the top left of the figure are clearly close to each other while being farther away from the others.
- The same is true of the crosses in the top right.



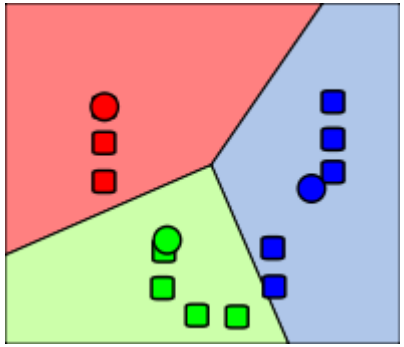
The goal of clustering is to divide a data set into "sufficiently distinct" subsets. In this plot for instance, the observations have been divided into three clusters.

Possible Applications

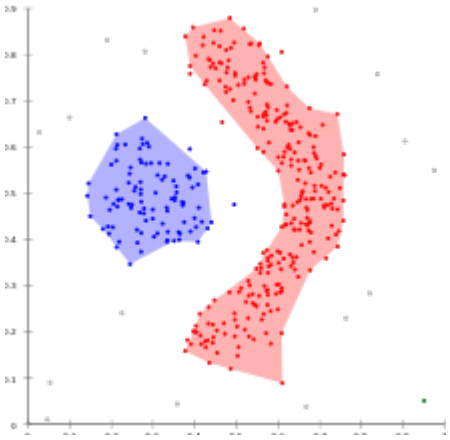
- Clustering algorithms can be applied in many fields, for instance:
  - *Marketing*: finding groups of customers with similar behaviour, given a large database of customer data containing their properties and past buying records;
  - *Biology*: grouping plants and animals given their features;
  - *Libraries*: book ordering;
  - *Insurance*: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
  - *City-planning*: identifying groups of houses according to their house type, value and geographical location;
  - *Earthquake studies*: clustering observed earthquake epicentres to identify dangerous zones;
  - *WWW*: clustering weblog data to discover groups of similar access patterns.

Two Common Clustering Techniques

- k Means
  - Aims to partition n observations into k clusters in which
    - *each observation belongs to the cluster* with the nearest mean,
      - Which serves as a prototype of the cluster.



- Density-based spatial clustering of applications with noise (DBSCAN)
  - It groups together points that are closely packed,
  - Marks as outliers points that lie alone in low-density regions.



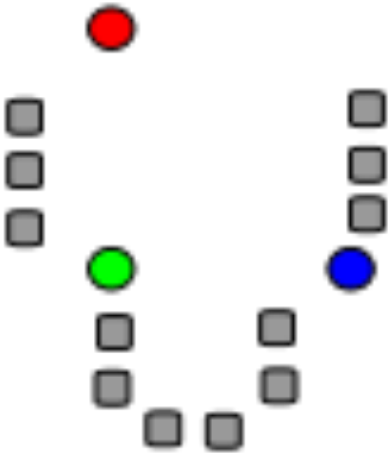
https://en.wikipedia.org/wiki/DBSCAN https://en.wikipedia.org/wiki/K-means_clustering

Practical Data Science – COSC2670

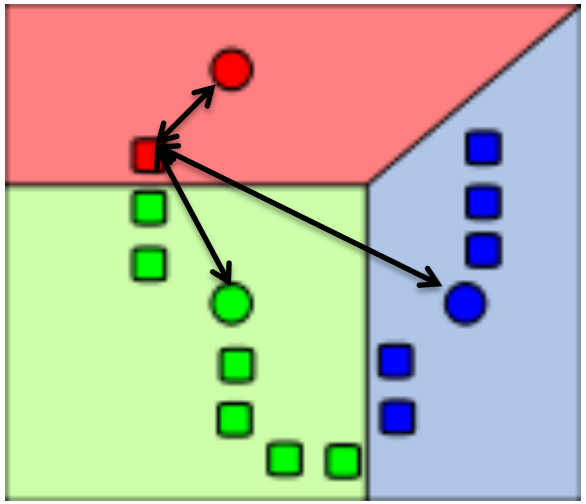
PART 2:
k MEANS

What is k-means clustering technique?

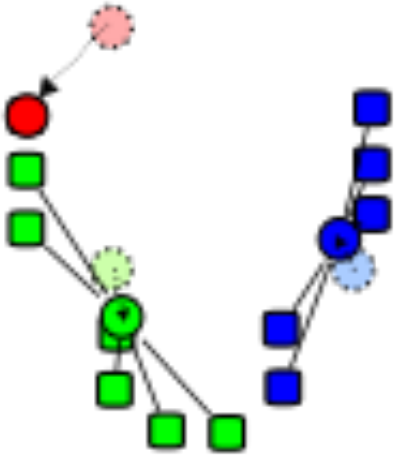
- k Means
  - Aims to partition n observations into k clusters in which
    - *each observation belongs to the cluster* with the nearest *mean*,
      - Which serves as a prototype of the cluster.
  - Assume Euclidean space/distance.
  - Start by picking k, the number of clusters
  - Initialize clustering by picking one point per cluster
    - For the moment, assume we pick the k points at random



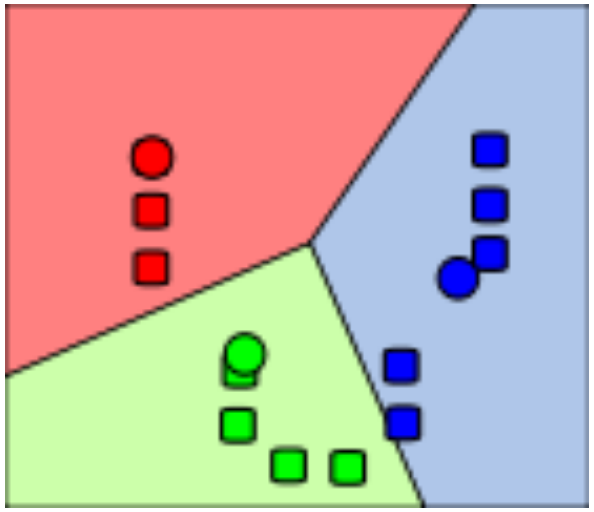
1. Initialise cluster centroids



2. Assign points to clusters whose current centroid is nearest

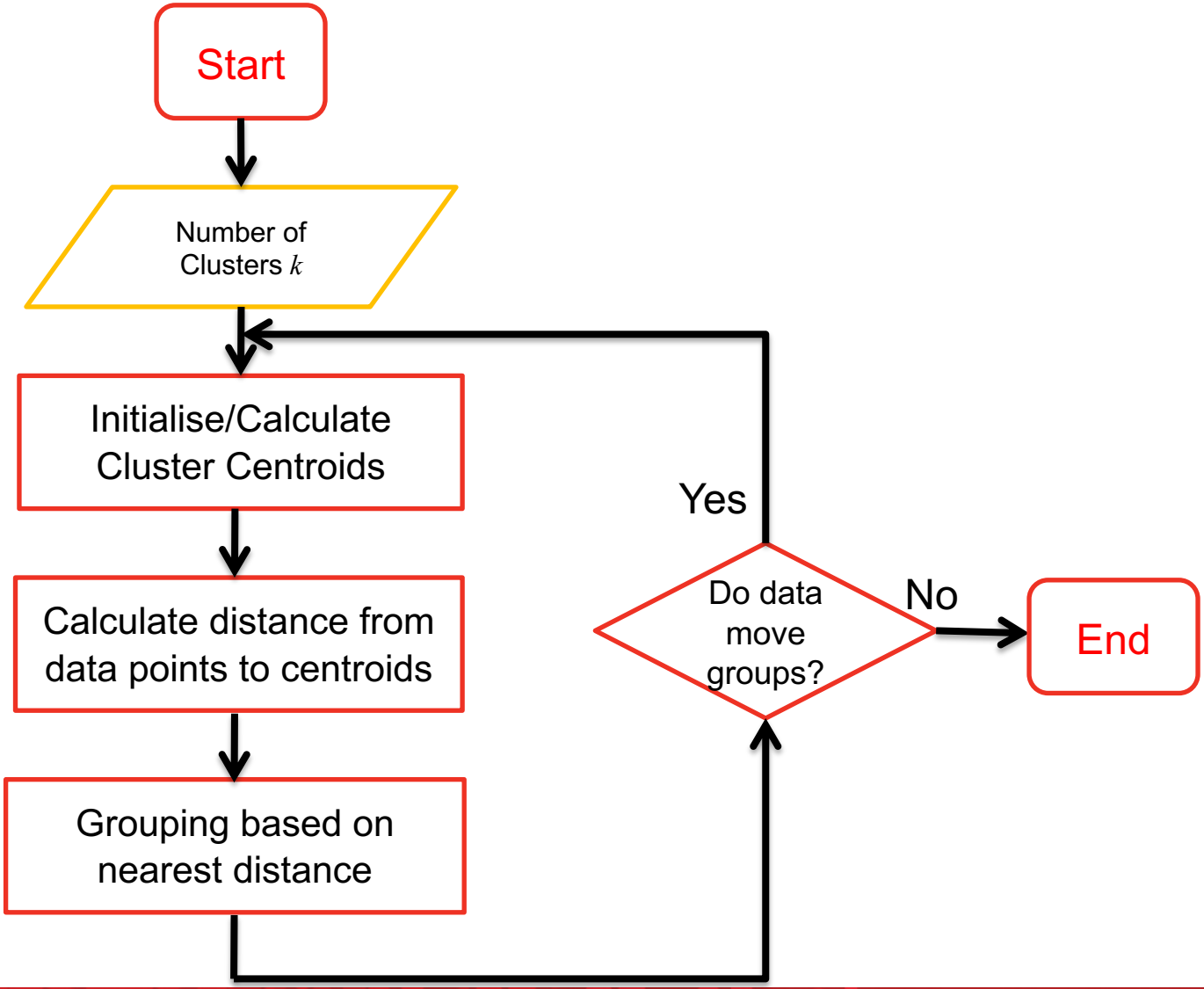


3. re-compute means (centroids)



4. Repeat step 2 and 3 until *convergence*: *Points don't move between clusters*

k Means Procedure



k Means

- You must provide the algorithm with the k parameter,
  - which is the number of clusters.
- Sometimes, this might be a limitation because you have to first investigate
  - which is the right k for the current dataset.
- The initial placement of the centroids is random.
- So, sometimes,
  - you need to run the algorithm several times so as not to find a local minimum.

k Means

- We have seen the theory behind the algorithm;
- Now, let's see it in practice.
- Let’s consider two 2-dimensional dummy datasets
  - that will explain what's going on better.
- Both datasets are composed of 2,000 samples so that you can also have an idea about the processing time.
  - The first one contains two (noisy) circles with the same origin but different radius;
  - The second one contains four blobs of points.

k Means

Load packages.

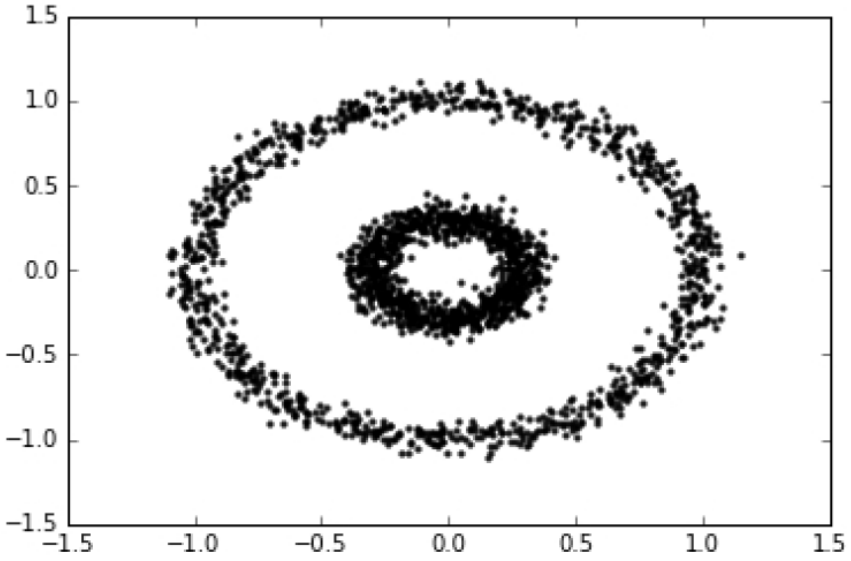
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

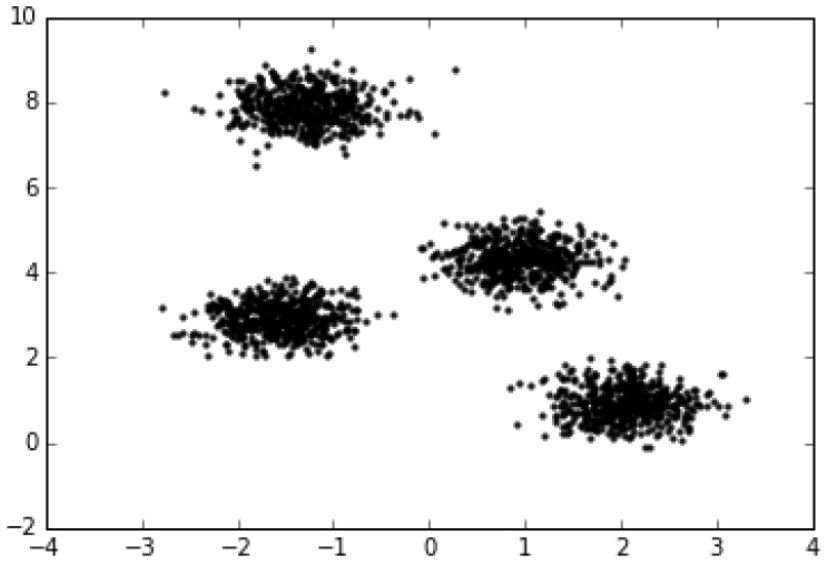
Generate the two dummy datasets by using *sklearn.datasets.make_circles* and *sklearn.datasets.make_blobs*

```
N_samples = 2000
dataset_1 = np.array(datasets.make_circles(n_samples=N_samples, noise=0.05, factor=0.3)[0])
dataset_2 = np.array(datasets.make_blobs(n_samples=N_samples, centers=4, cluster_std=0.4, random_state=0)[0])
```

```
plt.scatter(dataset_1[:,0], dataset_1[:,1], c='k', alpha=0.8, s=5.0)
plt.show()
plt.scatter(dataset_2[:,0], dataset_2[:,1], c='k', alpha=0.8, s=5.0)
plt.show()
```

Plot each dataset *matplotlib.pyplot.scatter*





k Means

- Dummy Dataset 1

```
from sklearn.cluster import KMeans
K_dataset_1 = 2
km_1 = KMeans(n_clusters=K_dataset_1)
labels_1 = km_1.fit(dataset_1).labels_
```

Load packages.

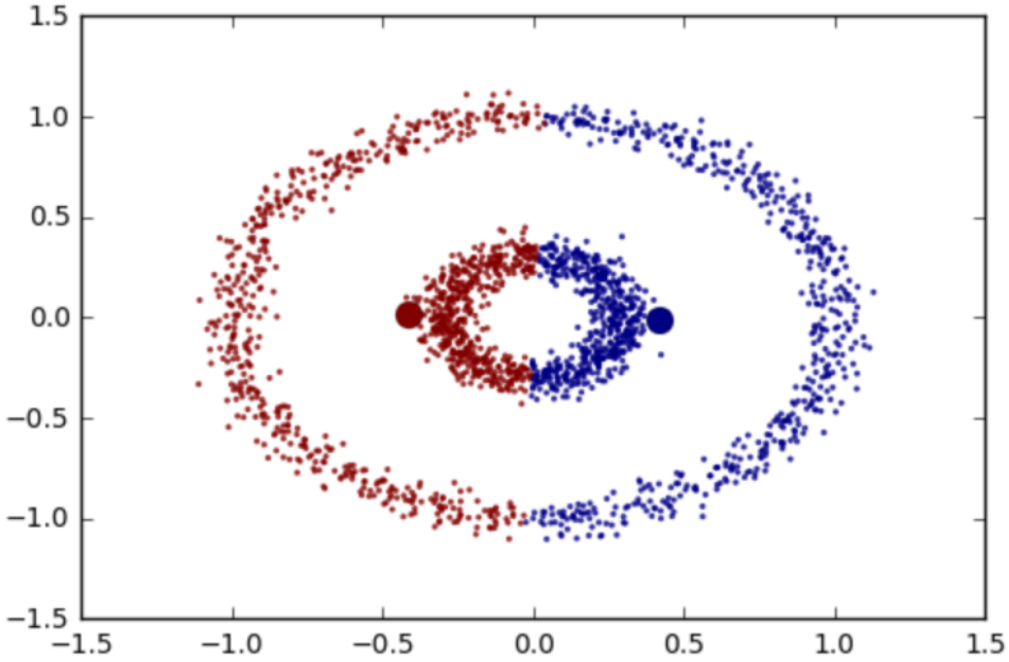
Set k

Build and feed the model with data (NO Labels)

Plot the clustering result

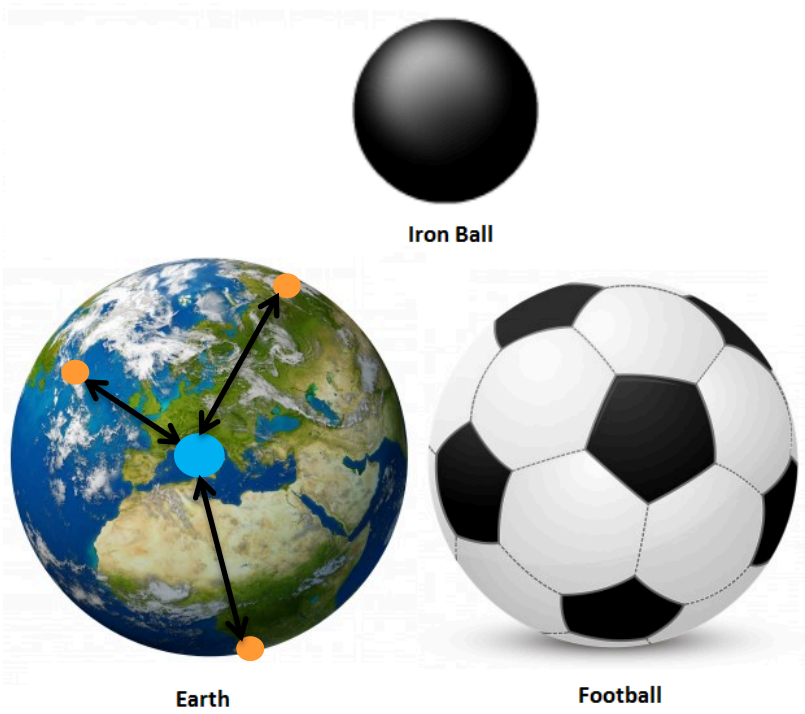
```
plt.scatter(dataset_1[:,0], dataset_1[:,1], c=labels_1, alpha=0.8, s=5.0, lw = 0)
plt.scatter(km_1.cluster_centers_[:,0], km_1.cluster_centers_[:,1], s=100, c=np.unique(labels_1), lw=0.2)
plt.show()
```

Plot the cluster center



k Means

- As we can see, k means is not performing well on dataset_1 (two circles),
  - Because it expect *spherical-shaped* data clusters.



- Now, let's see how it performs on a spherical-clustered data.

k Means

```
K_dataset_2 = 4
km_2 = KMeans(n_clusters=K_dataset_2)
labels_2 = km_2.fit(dataset_2).labels_
```

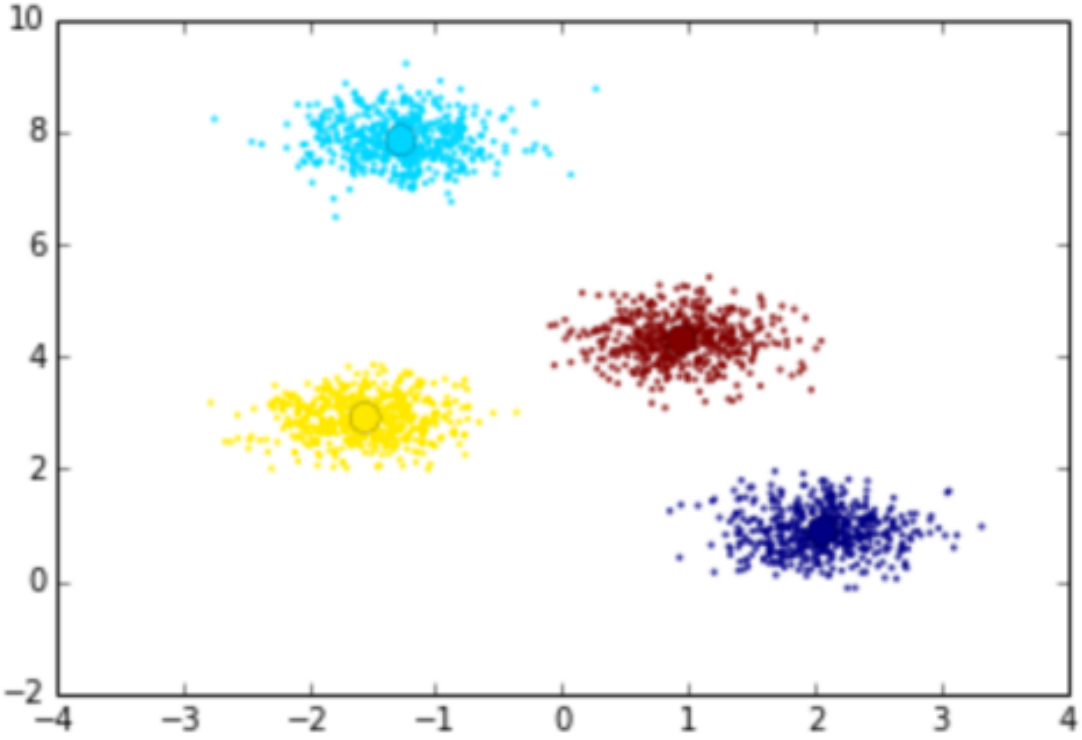
Set k

Build and feed the model with data (NO Labels)

Plot the clustering result

```
plt.scatter(dataset_2[:,0], dataset_2[:,1], c=labels_2, alpha=0.8, s=5.0, lw = 0)
plt.scatter(km_2.cluster_centers_[:,0], km_2.cluster_centers_[:,1], s=100, c=np.unique(labels_2), lw=0.2)
plt.show()
```

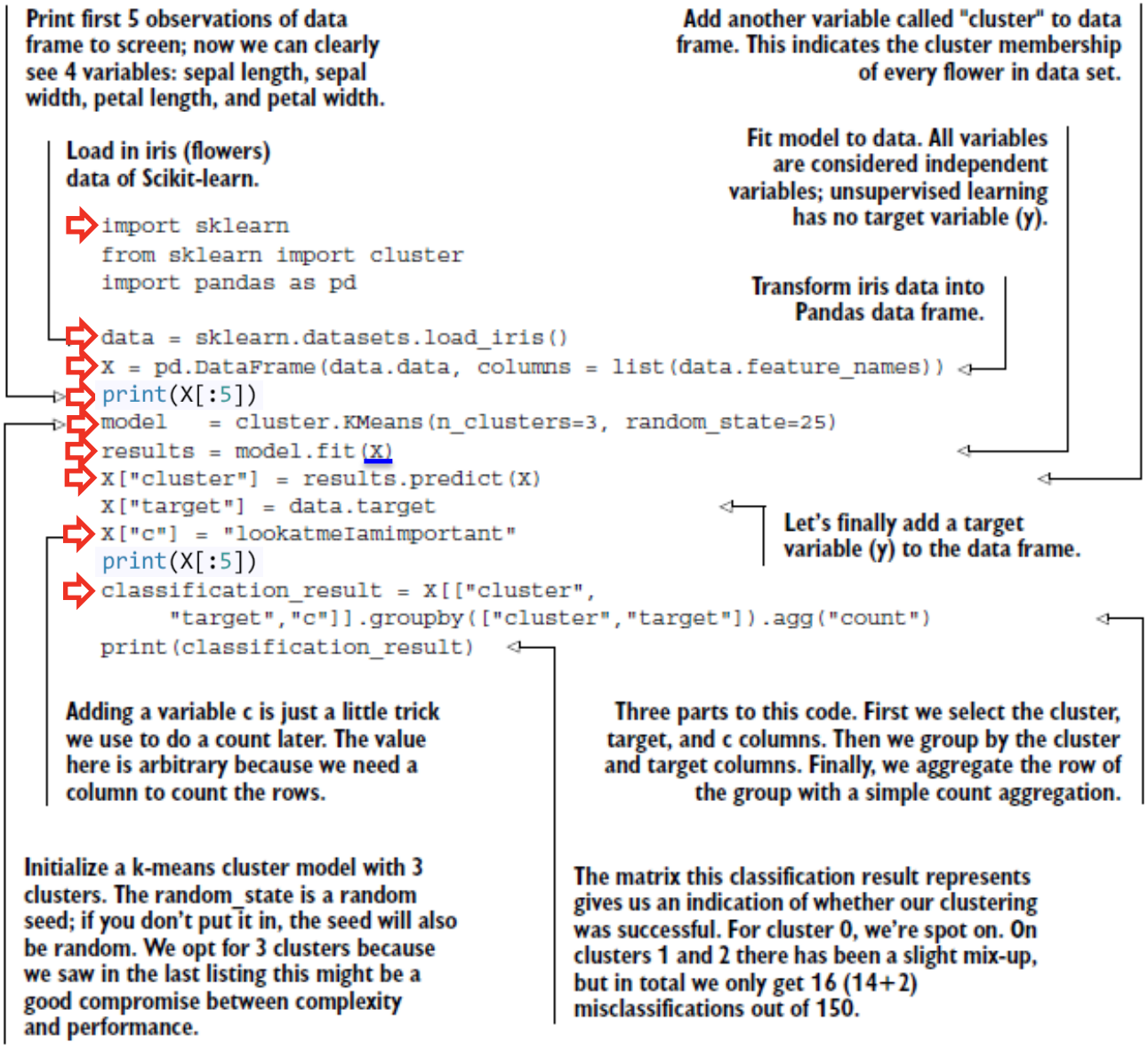
Plot the cluster center



On Iris Dataset

- As expected, the results is great on dataset_2.
  - The centroids and clusters are exactly what we had in mind while looking at the unlabelled dataset.
- Now, let's investigate how k means perform on real-world datasets
  - E.g. the Iris dataset

On Iris Dataset



On Iris Dataset

Print first 5 observations of data frame to screen; now we can clearly see 4 variables: sepal length, sepal width, petal length, and petal width.

Load in iris (flowers) data of Scikit-learn.

Add another variable called "cluster" to data frame. This indicates the cluster membership of every flower in data set.

Fit model to data. All variables are considered independent variables; unsupervised learning has no target variable (y).

Transform iris data into Pandas data frame.

```
import sklearn
from sklearn import cluster
import pandas as pd

data = sklearn.datasets.load_iris()
X = pd.DataFrame(data.data, columns = list(data.feature_names))
print(X[:5])
model   = cluster.KMeans(n_clusters=3, random_state=25)
results = model.fit(X)
X["cluster"] = results.predict(X)
X["target"] = data.target
X["c"] = "lookatmeIamimportant"
print(X[:5])
classification_result = X[["cluster",
      "target","c"]].groupby(["cluster","target"]).agg("count")
```

Let's finally add a target variable (y) to the data frame.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	cluster	target	c
0	5.1	3.5	1.4	0.2	0	0	lookatmeIamimportant
1	4.9	3.0	1.4	0.2	0	0	lookatmeIamimportant
2	4.7	3.2	1.3	0.2	0	0	lookatmeIamimportant
3	4.6	3.1	1.5	0.2	0	0	lookatmeIamimportant
4	5.0	3.6	1.4	0.2	0	0	lookatmeIamimportant

be random. We opt for 3 clusters because we saw in the last listing this might be a good compromise between complexity and performance.

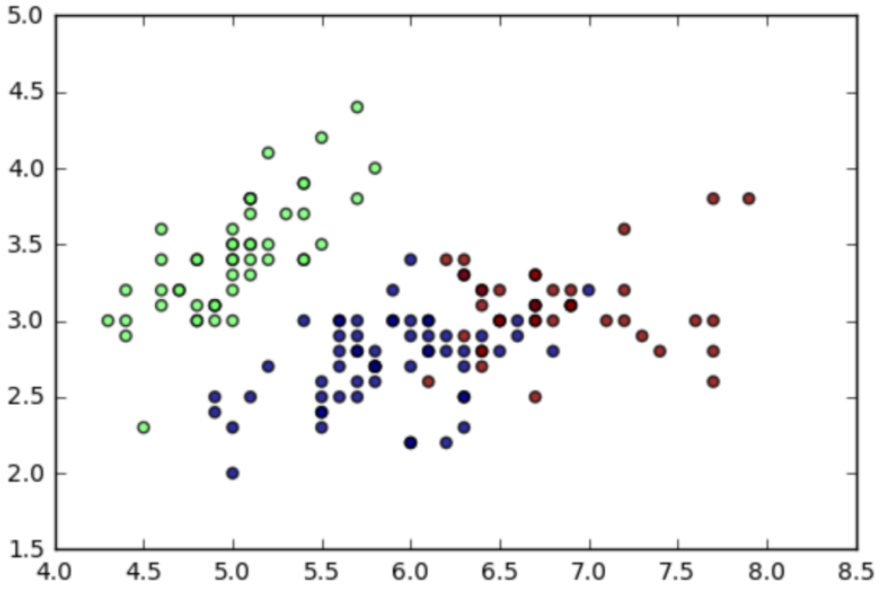
was successful. For cluster 0, we're spot on. On clusters 1 and 2 there has been a slight mix-up, but in total we only get 16 (14+2) misclassifications out of 150.

		c
cluster	target	
0	0	50
1	1	48
	2	14
2	1	2
	2	36

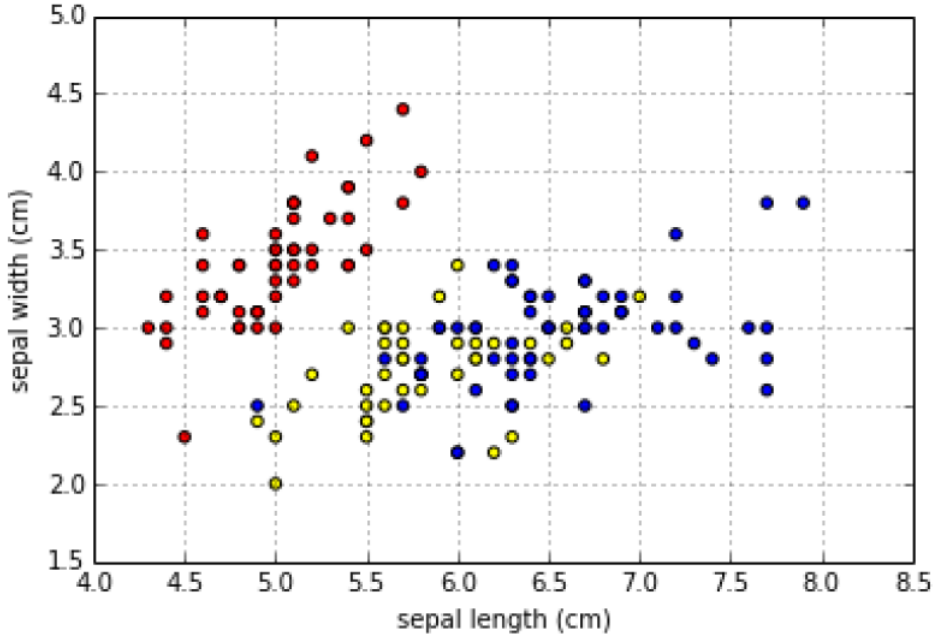
On Iris Dataset

```
import matplotlib.pyplot as plt


plt.scatter(X.loc[:, 0], X.loc[:, 1], c=results.labels_, alpha=0.8)
plt.show()
```



Clustering Results



Original

k Means
- *sklearn.cluster.KMeans*

sklearn.cluster.KMeans(n_clusters, init, n_init, max_iter, random_state)

- **n_clusters** : int, optional, default: 8
  - The number of clusters to form as well as the number of centroids to generate.
- **init** : {'k-means++', 'random' or an ndarray}
  - Method for initialization, defaults to 'k-means++':
  - 'k-means++' : selects initial cluster centers for k-means clustering in a smart way to speed up convergence. See section "Notes" in k_init for more details.
  - 'random': choose k observations (rows) at random from data for the initial centroids.
  - If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

k Means
- *sklearn.cluster.KMeans*

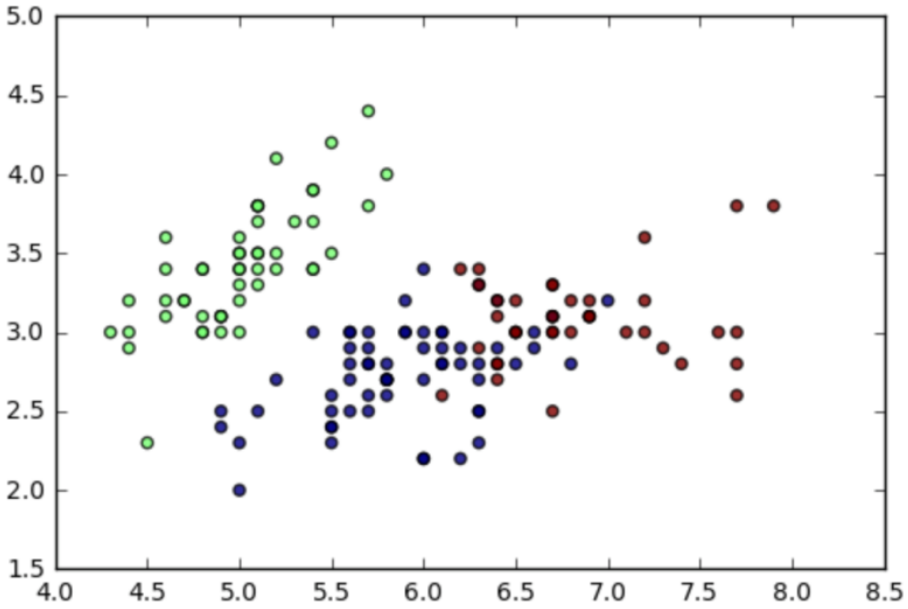
sklearn.cluster.KMeans(n_clusters, init, n_init, max_iter, random_state)

- **max_iter** : int, default: 300
  - Maximum number of iterations of the k-means algorithm for a single run.
- **n_init** : int, default: 10
  - Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.
- **random_state** : integer or numpy.RandomState, optional
  - The generator used to initialize the centers. If an integer is given, it fixes the seed. Defaults to the global numpy random number generator.

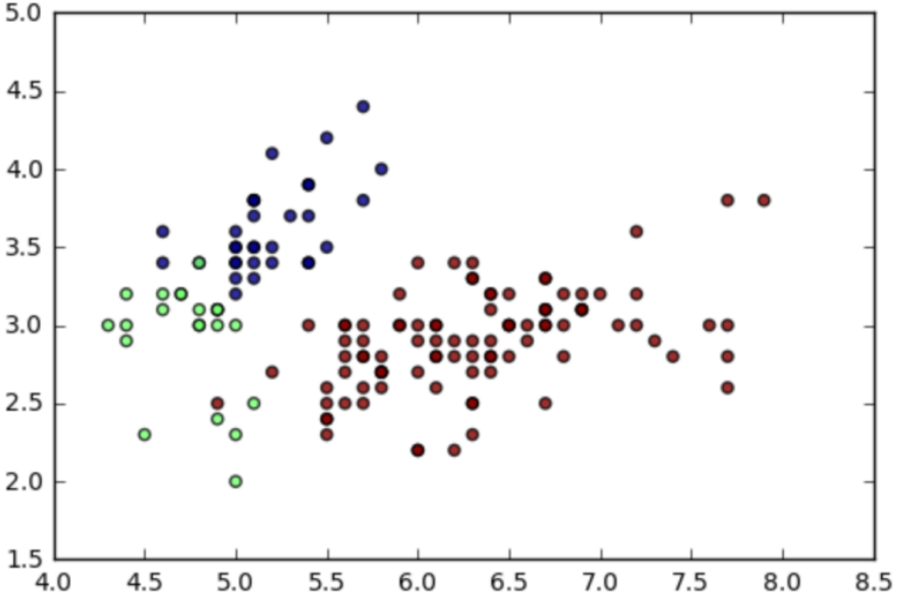
k Means

- Different Parameters

```
cluster.KMeans(n_clusters = 3)
```



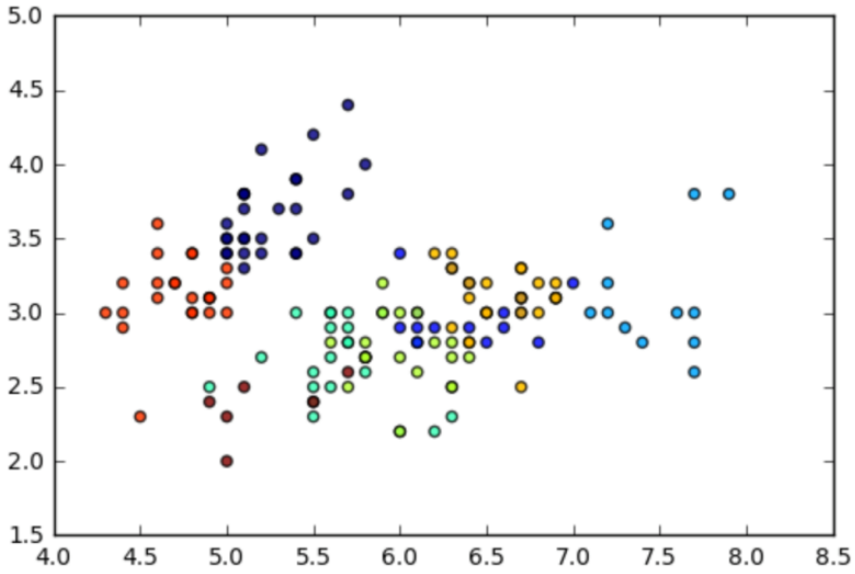

```
cluster.KMeans(n_clusters = 3,n_init = 1,
init='random')
```



k Means
- Different Parameters

```
model = cluster.KMeans(n_clusters = 8)
```

```
classification_result = X[['cluster','target','c']].groupby(['cluster', 'target']).agg('count')
print(classification_result)
```



k=8

		c
cluster	**target**	
0	**1**	20
1	**0**	24
2	**2**	24
3	**1**	22
	2	1
4	**1**	3
	2	15
5	**0**	26
6	**2**	10
7	**1**	5

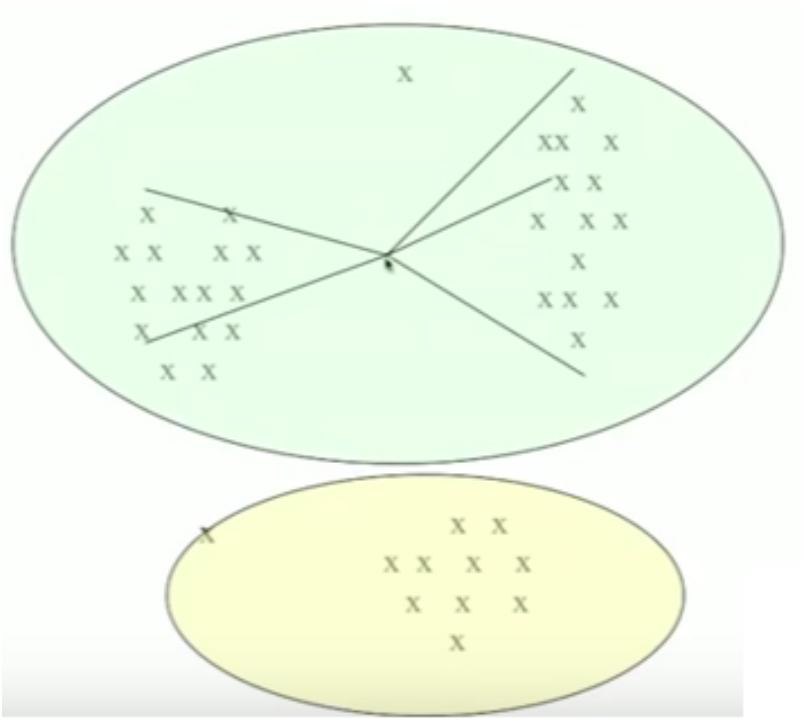
k=3

		c
cluster	**target**	
0	**0**	50
1	**1**	48
	2	14
2	**1**	2
	2	36

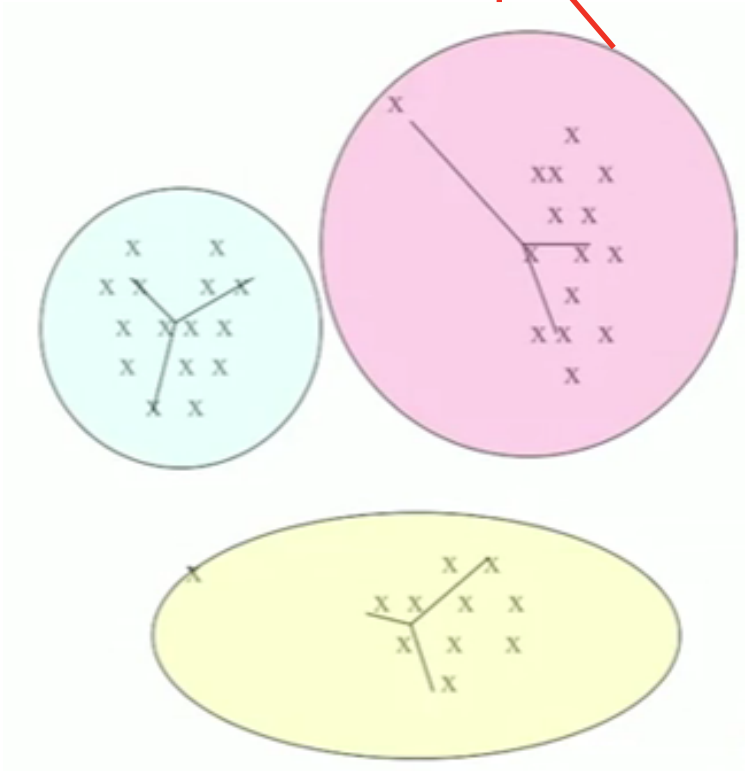
How to Select k?

- Try different k, looking at the change in *the average distance to centroid*, as k increases.

Too low: many large distances.



Just right: distance rather short

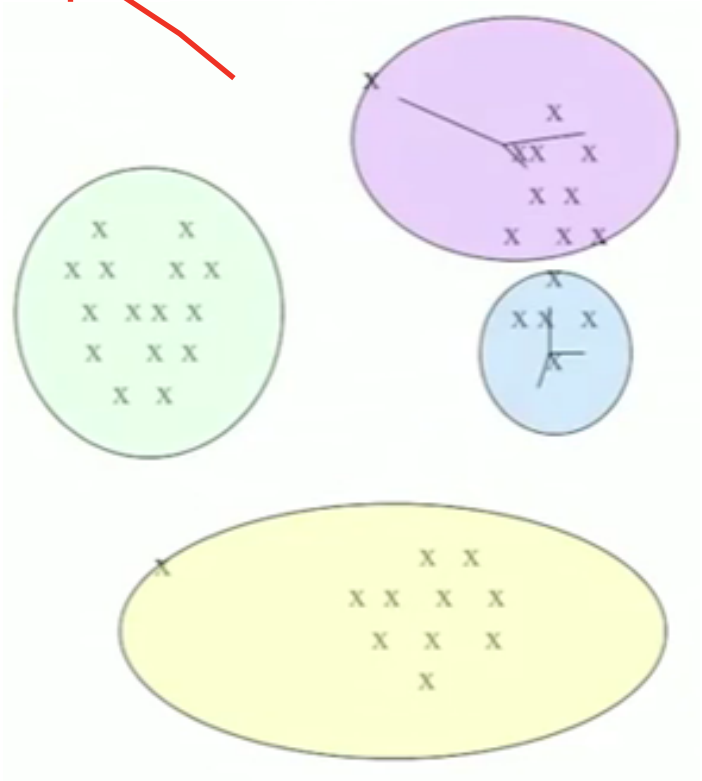


https://www.youtube.com/watch?v=RD0nNK51Fp8

How to Select k?

- What if we keep increasing k?

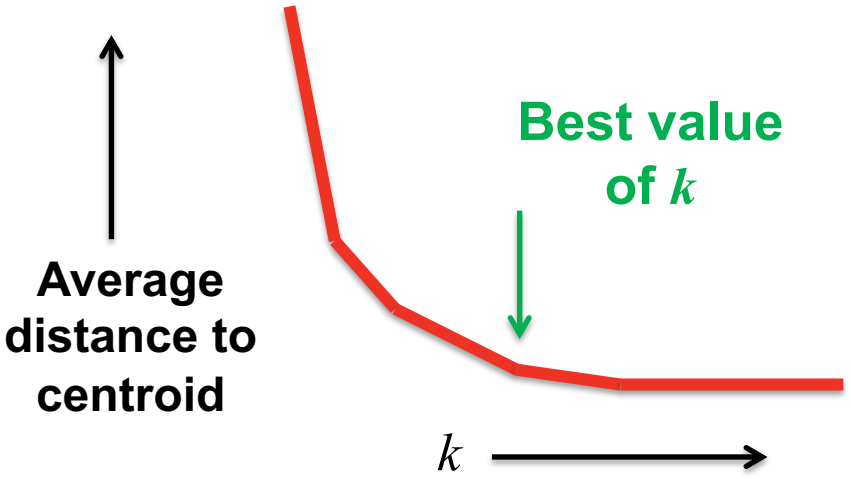
Too large: little improvement in average distance



https://www.youtube.com/watch?v=RD0nNK51Fp8

How to Select k?

- The average distance between *data points* and *centroids*
  - *falls rapidly until the right k,*
  - *then falls much more slowly.*

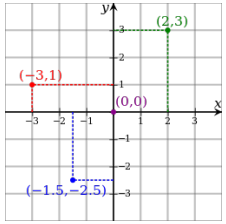


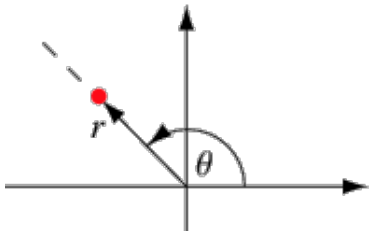
k Means: Advantages

- Fast, robust and easy to understand.
- Relatively efficient.
- Gives best result when data sets are distinct, or well separated from each other.

k Means: Disadvantages

- If there are two highly overlapping data sets then k means will not be able to resolve that there are two clusters.
- The learning algorithm is not invariant to non-linear transformations.
  - With different representation of data we get different results
    - (e.g. data represented in form of Cartesian co-ordinates and polar co-ordinates will give different results).





- Euclidean distance measures can unequally weight underlying factors.
- The learning algorithm provides the local optima of the squared error function.
- Randomly choosing the cluster centre cannot lead us to the fruitful result.
- Applicable only when the mean is defined i.e. fails for categorical data.
- Unable to handle noisy data and outliers.

References and Further Reading

- A. Boschetti and L. Massaron, *Python Data Science Essentials*, Chapters 4 and 6
- D. Cielen and A. Meysman and M. Ali, *Introducing Data Science*, Chapter 3

Data
Science

Thanks!