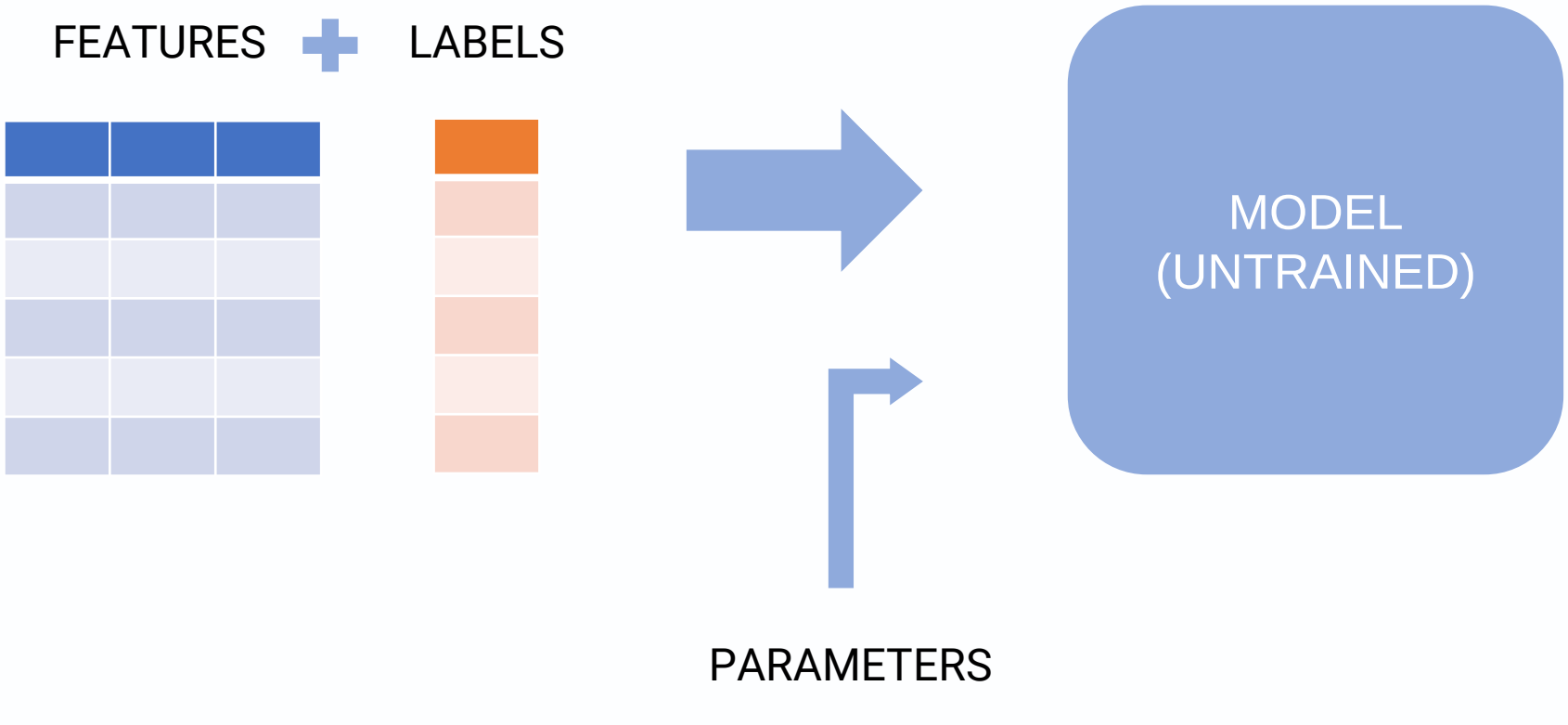
CLUSTERING

with Python Scikit-Learn

March 2023

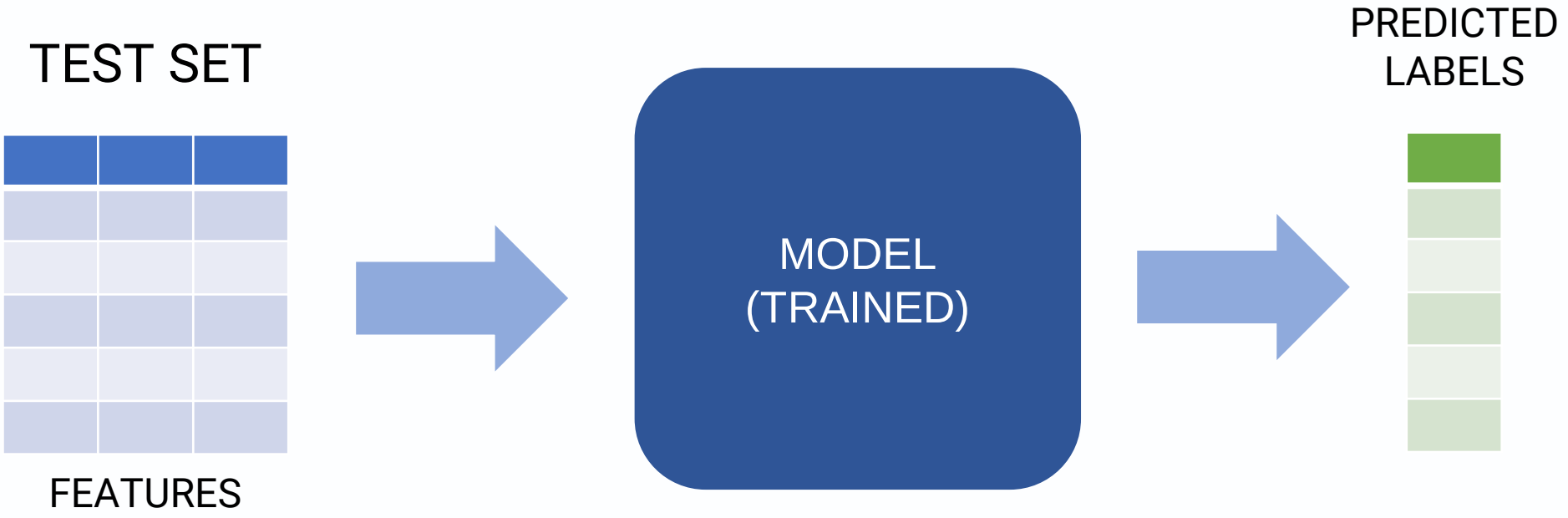
Overview

CLASSIFICATION



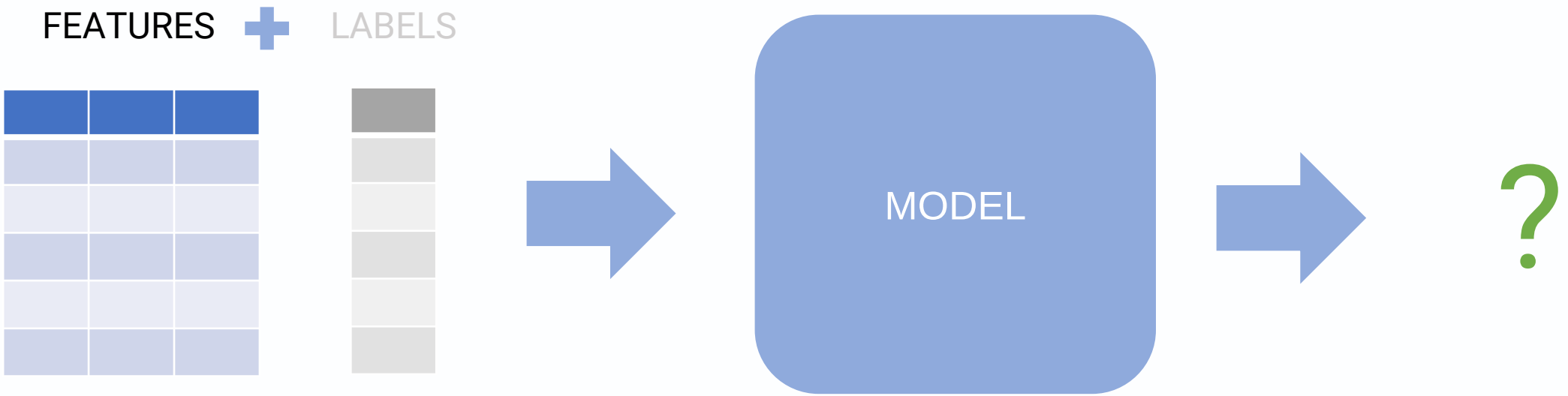
Overview

CLASSIFICATION



Overview

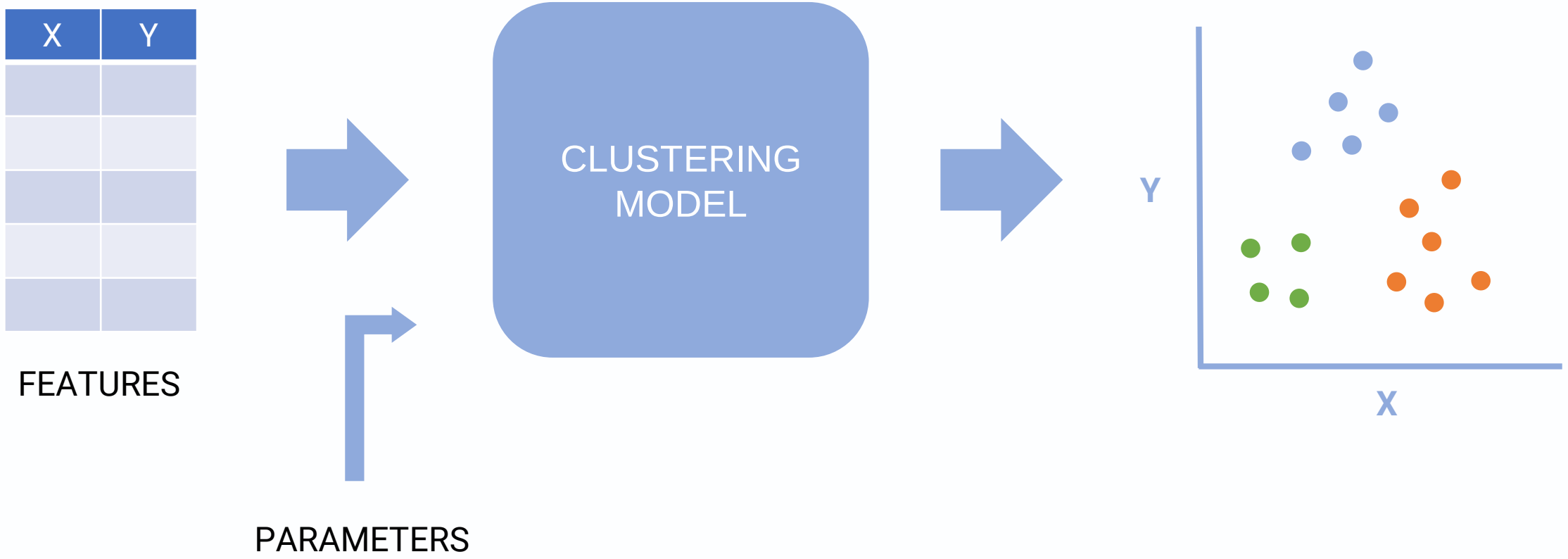
UNSUPERVISED LEARNING



What if there are no labels?

Overview

CLUSTERING



Using a Clustering Model

Model

- Source: from scratch or import from an existing package (e.g. sklearn)

Input: Training Set

- Features
- Parameters
  - e.g. : for kMeans, k is a parameter

Clustering Overview

1. Import the model from a package (sklearn)
2. Create an instance of the model
3. Train the model (often using the `fit()` function)

```
In [1]:  # KMeans
         from sklearn.cluster import KMeans
```

...

```
In [7]:  kmeans = KMeans(n_clusters=4)
         kmeans.fit(X)
```

Clustering Overview

You can create multiple models with different parameters

```
In [7]:  kmeans = KMeans(n_clusters=4)
         kmeans.fit(X)

Out[7]:  KMeans
         KMeans(n_clusters=4)
```

k = 4

```
In [20]:  kmeans = KMeans(n_clusters=2)
          kmeans.fit(X)

Out[20]:  KMeans
          KMeans(n_clusters=2)
```

k = 2

Hierarchical Clustering

Scikit Learn: AgglomerativeClustering

```
In [1]: from sklearn.cluster import AgglomerativeClustering
```

...

```
In [9]: agg_model = AgglomerativeClustering(n_clusters=3,
                                            metric='euclidean',
                                            linkage='ward')
        agg_model.fit(X)
```

```
Out[9]: AgglomerativeClustering
        AgglomerativeClustering(metric='euclidean', n_clusters=3)
```

KMeans

Scikit Learn: KMeans

In [1]:
```
# KMeans
from sklearn.cluster import KMeans
```

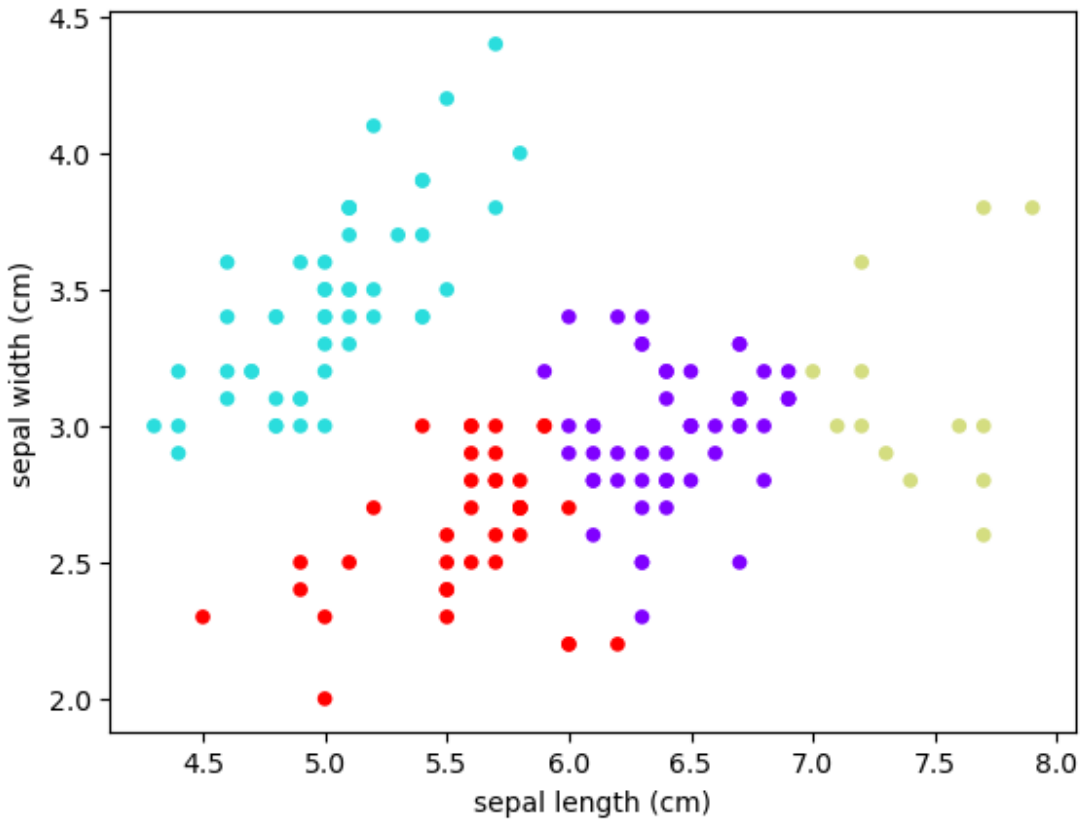
...

In [7]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
```

Scatter Plot

```
In [19]: X.plot.scatter("sepal length (cm)", "sepal width (cm)",
                        c = kmeans.labels_,
                        cmap="rainbow",
                        colorbar=False)
```

```
Out[19]: <Axes: xlabel='sepal length (cm)', ylabel='sepal width (cm)'>
```



Scatter Plot

dataframe.plot.scatter(“feature1”, “feature2”,

c = model.labels_,

cmap=“rainbow”, colorbar=False)

```
In [19]: X.plot.scatter("sepal length (cm)", "sepal width (cm)",
                        c = kmeans.labels_,
                        cmap="rainbow",
                        colorbar=False)
```

Clustering Overview

Note: we can cluster based on more than just two features!

- In our example, we only use two features from the iris dataset:
    - sepal length (cm)
    - sepal width (cm)
- Hard to visualize more than 2 features

```
In [4]:
# the two features to be used
col1 = "sepal length (cm)"
col2 = "sepal width (cm)"
```

```
In [5]:
# new dataframe with only two features
new_df = iris_df[[col1, col2]]
new_df
```

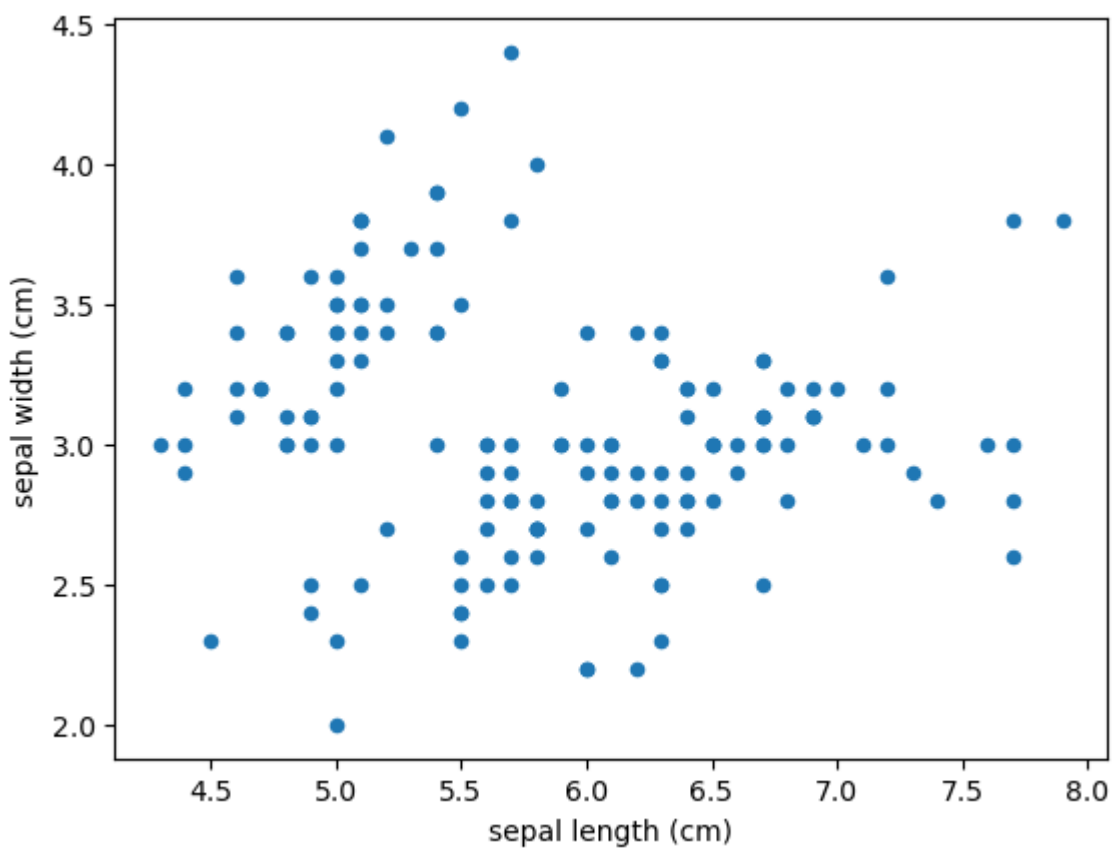
Out[5]:

	sepal length (cm)	sepal width (cm)
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6
...	...	...

Clustering Overview

Note: we can cluster based on more than just two features!

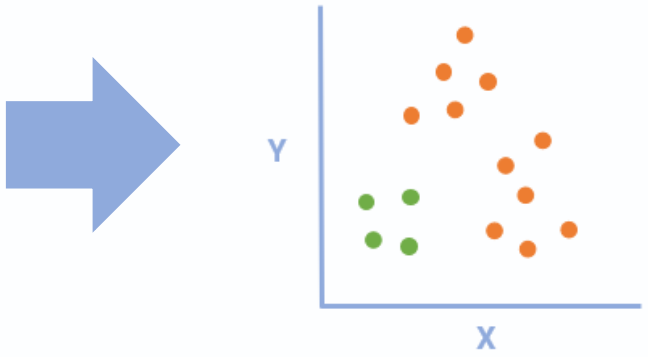
- In our example, we only use two features from the iris dataset:
    - sepal length (cm)
    - sepal width (cm)
- Hard to visualize more than 2 features



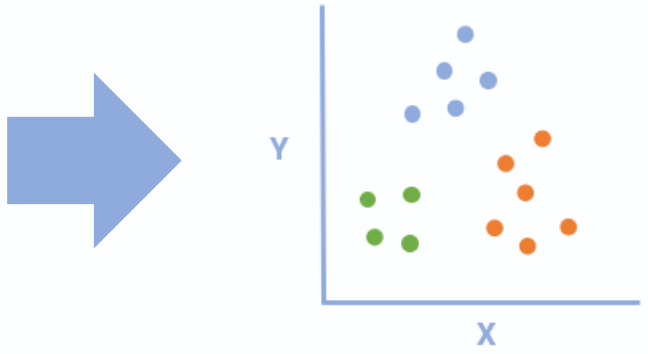
Overview

CLUSTERING

k = 2

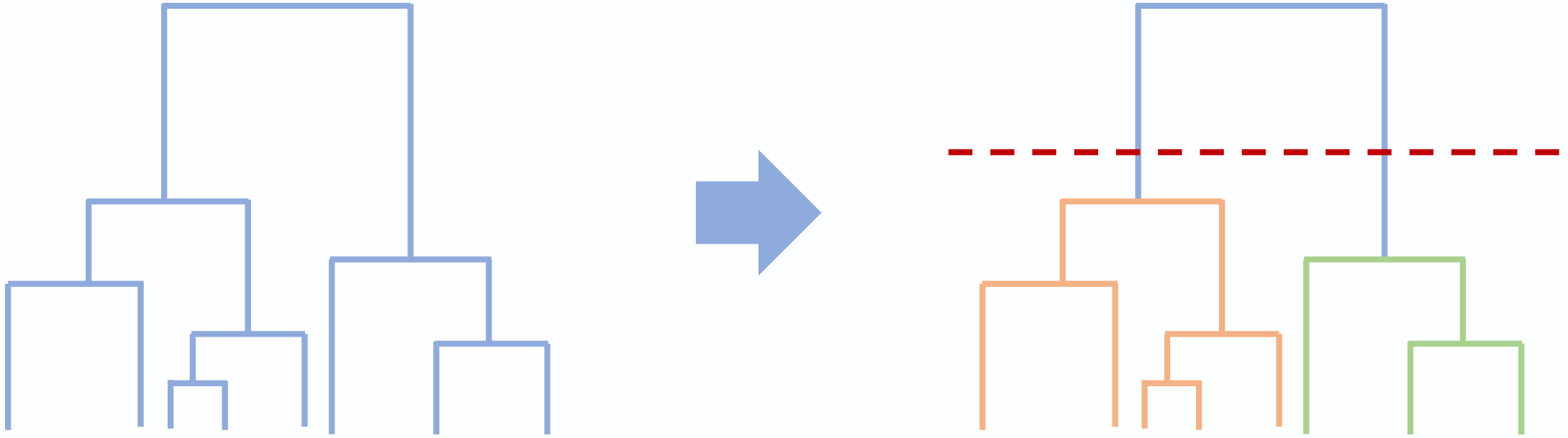


k = 3



How do we choose the number of clusters to use?

Hierarchical Clustering

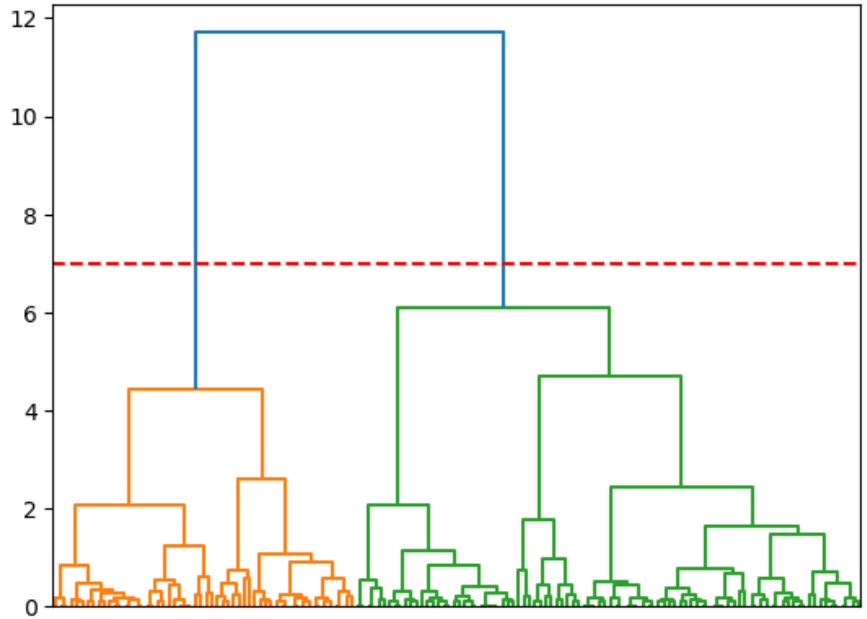


Visualize through dendrograms then make estimates from there

Hierarchical Clustering

Dendrogram Visualization

- Draw a horizontal line
- Number of clusters: lines that intersect with the horizontal line



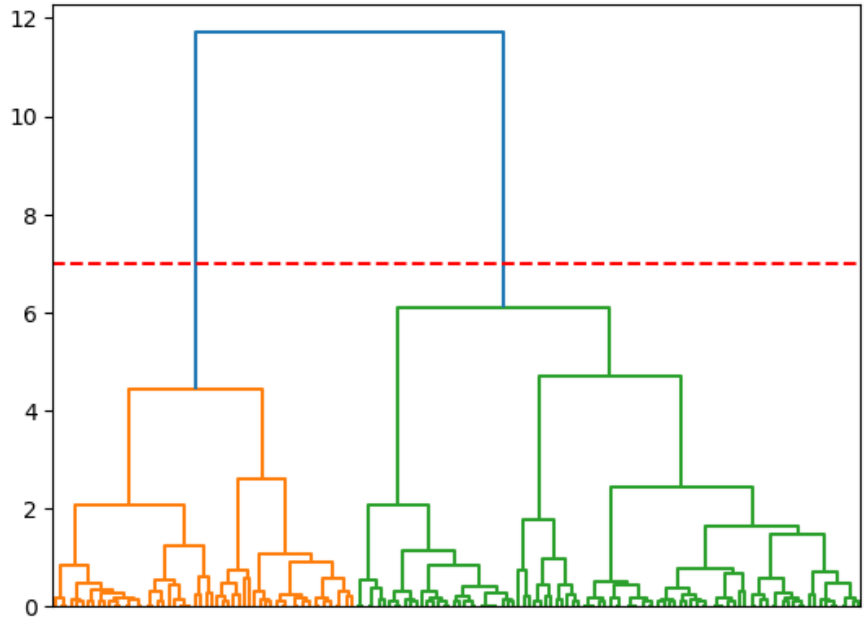

```
In [8]:  dendrogram = sch.dendrogram(sch.linkage(new_df, method='ward'))

         # plot horizontal line at y = 7
         plt.axhline(y=7, color='r', linestyle='--')
         plt.show()
```

Hierarchical Clustering

Dendrogram Visualization

- Horizontal line: y = 7
- Number of clusters: 2



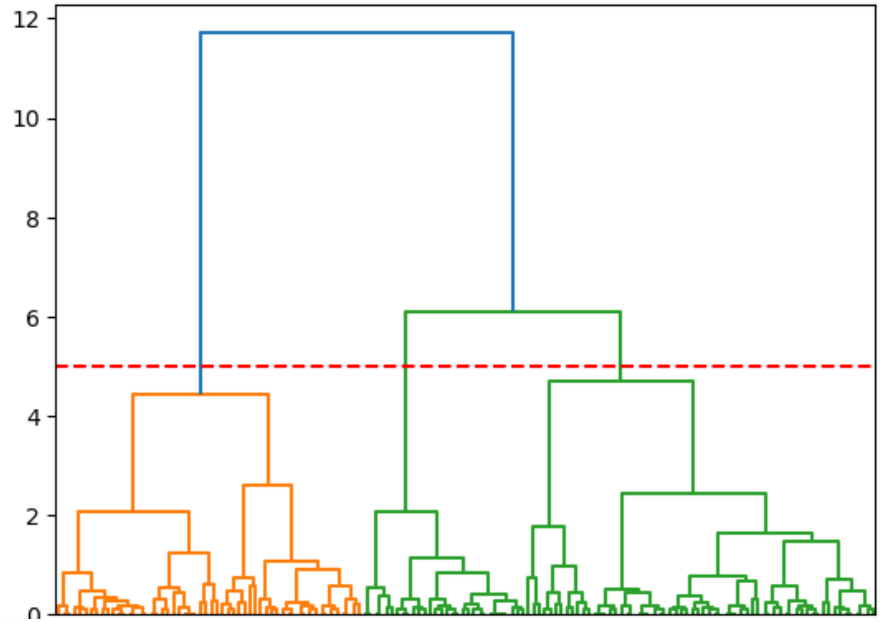

```
In [8]:
dendrogram = sch.dendrogram(sch.linkage(new_df, method='ward'))

# plot horizontal line at y = 7
plt.axhline(y=7, color='r', linestyle='--')
plt.show()
```

Hierarchical Clustering

Dendrogram Visualization

- Horizontal line: y = 5
- Number of clusters: 3



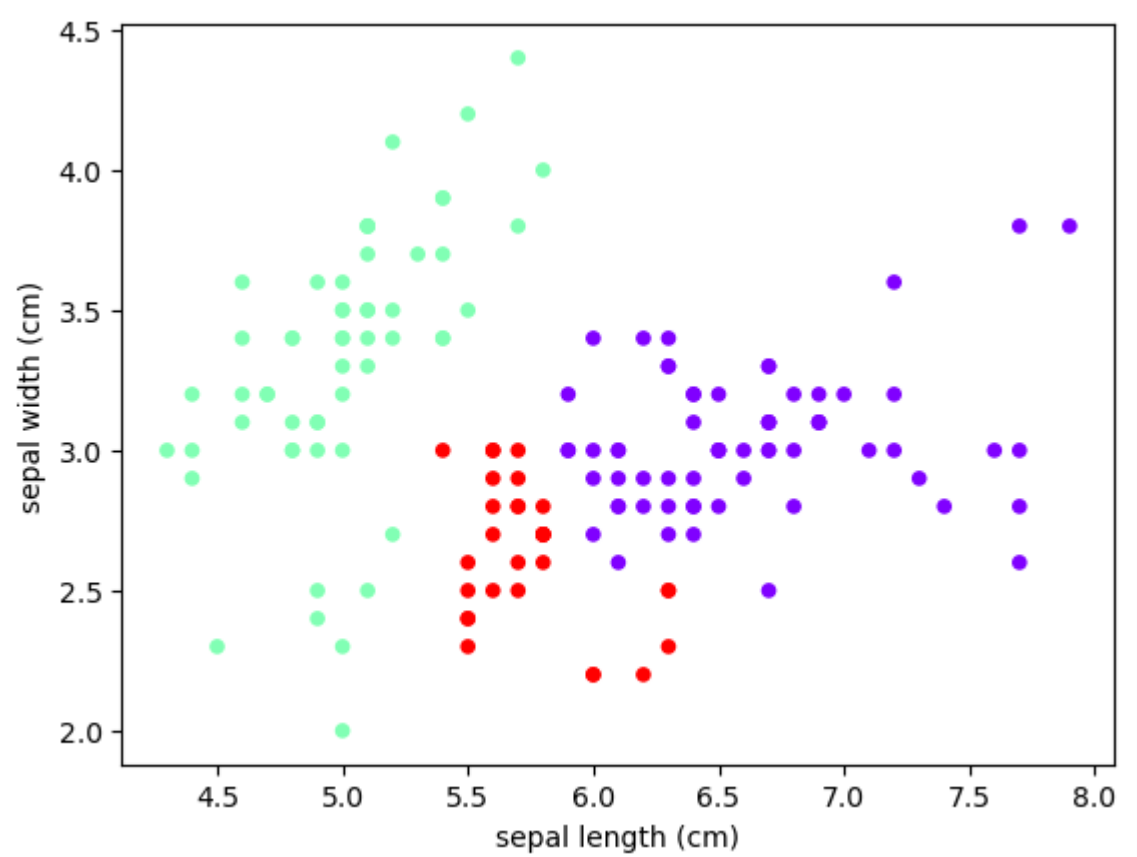

```
In [19]:  dendrogram = sch.dendrogram(sch.linkage(new_df, method='ward'))

          # plot horizontal line at y = 5
          plt.axhline(y=5, color='r', linestyle='--')
          plt.show()
```

Hierarchical Clustering

Dendrogram Visualization

- Horizontal line: y = 5
- Number of clusters: 3



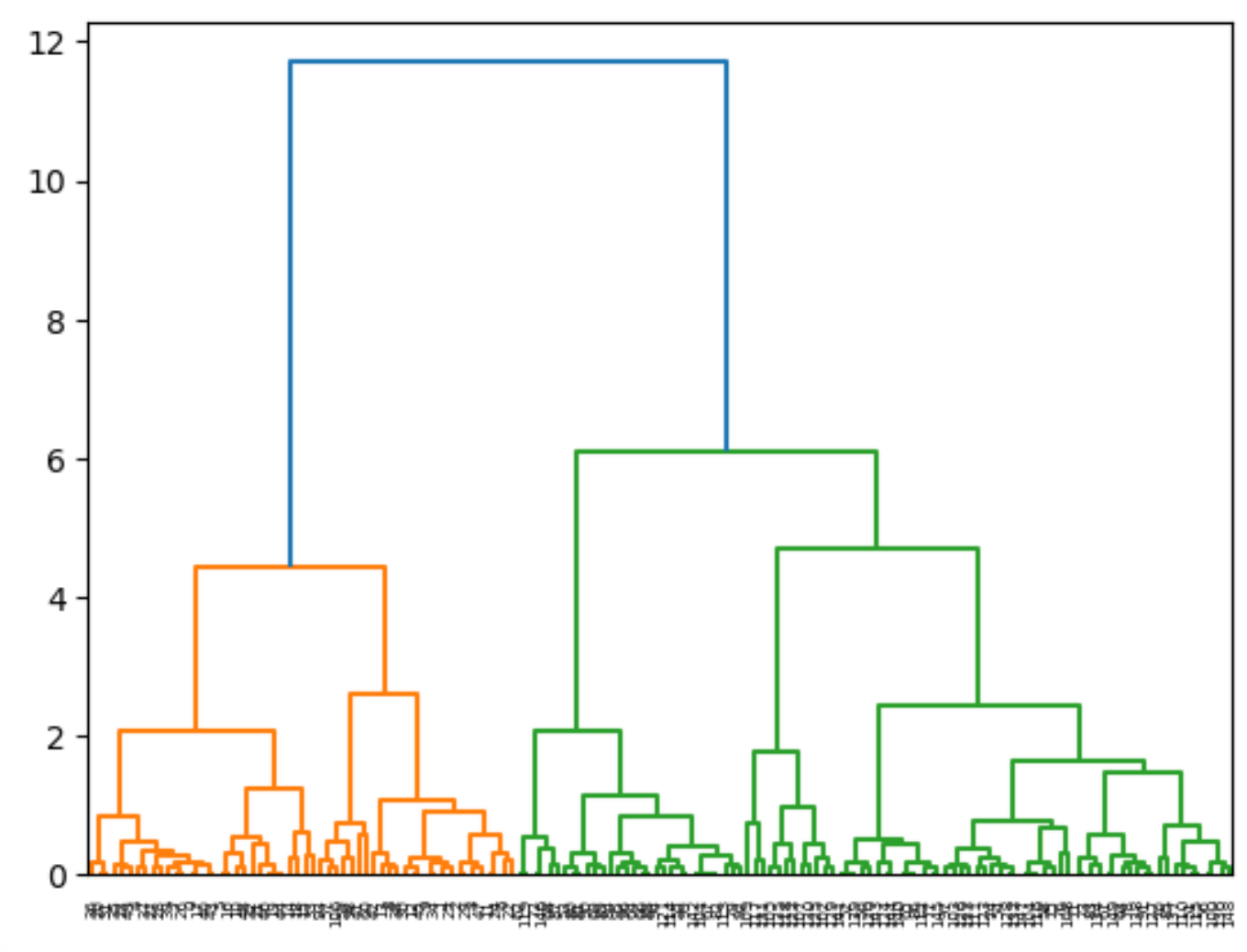
Hierarchical Clustering

Dendrogram Visualization

- Number of clusters: 4

- y = ?

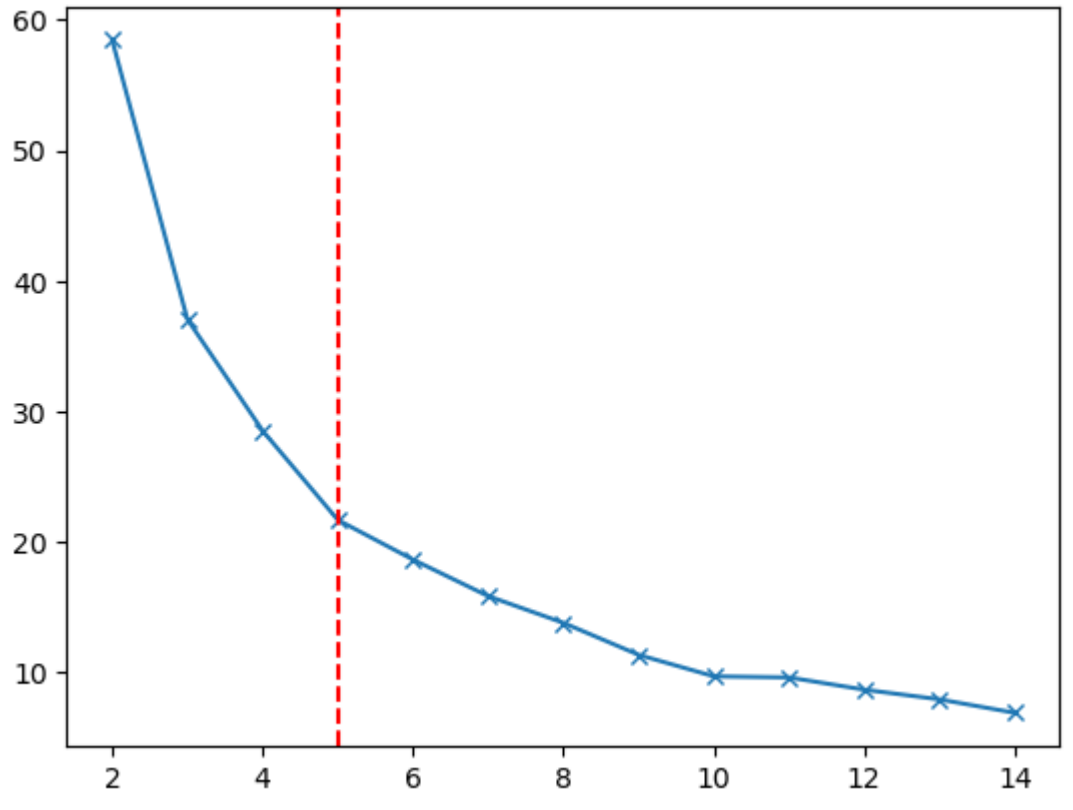
(where will the horizontal line be?)



KMeans

Methods to determine K:

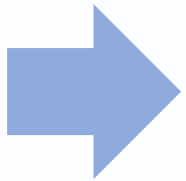
1. Silhouette Score
2. Elbow Method



KMeans: Silhouette Score

For each k in a range of k values:

1. Create a kMeans model
2. Fit the model with dataset
3. Get the instances' cluster labels
4. Get the silhouette score

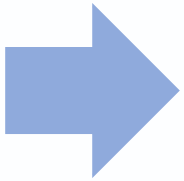


Get the k with the highest silhouette score

KMeans: Silhouette Score

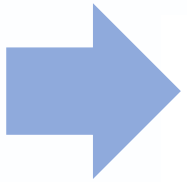
For each k in a range of k values:

1. Create a kMeans model
2. Fit the model with dataset
3. Get the instances' cluster labels
4. Get the silhouette score



Get the k with the highest silhouette score

```
for k in k_range:
    km_model = KMeans(n_clusters=k, n_init='auto')
    km_model.fit(new_df)
    km_labels = km_model.predict(new_df)
    avg = silhouette_score(new_df, km_labels)
```



```
K = 2
Score = 0.463
```

KMeans: Silhouette Score

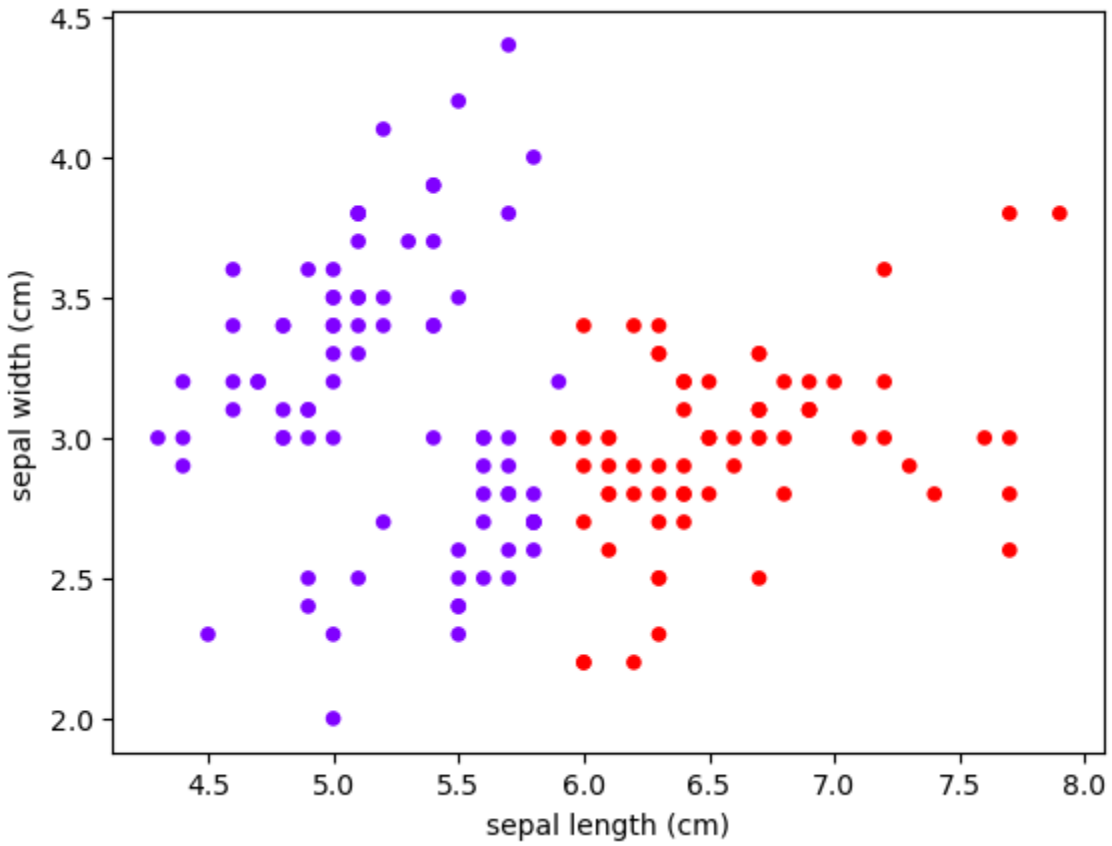
```
K = 2
Score = 0.463
```

```
km_model = KMeans(n_clusters=2,
                  n_init='auto', random_state=0)
km_model.fit(new_df)
```

]:

KMeans

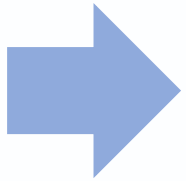
```
KMeans(n_clusters=2, n_init='auto', random_state=0)
```



KMeans: Elbow Method

For each k in a range of k values:

1. Create a kmeans model
2. Fit the model with dataset
3. Get and store inertia in a list



Plot the inertia values and estimate which k starts the “bend” (elbow) in the graph

KMeans: Elbow Method

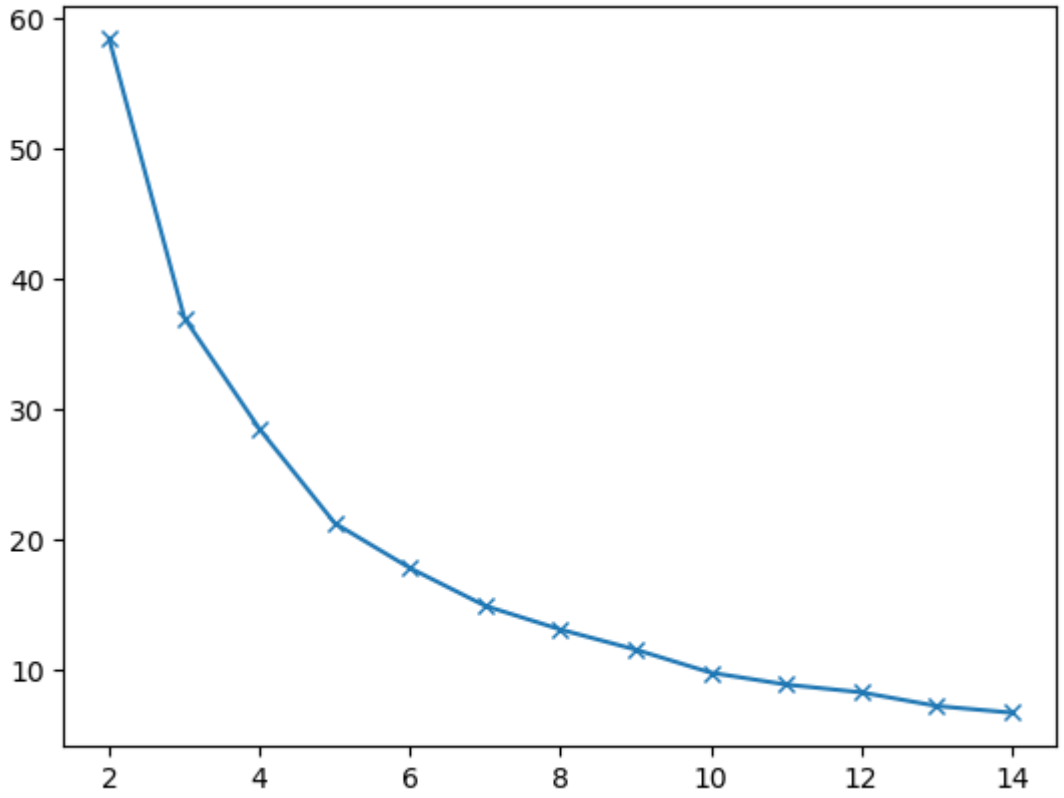
For each k in a range of k values:

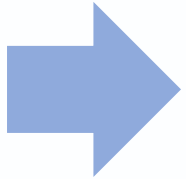
1. Create a kmeans model
2. Fit the model with dataset
3. Get and store inertia in a list

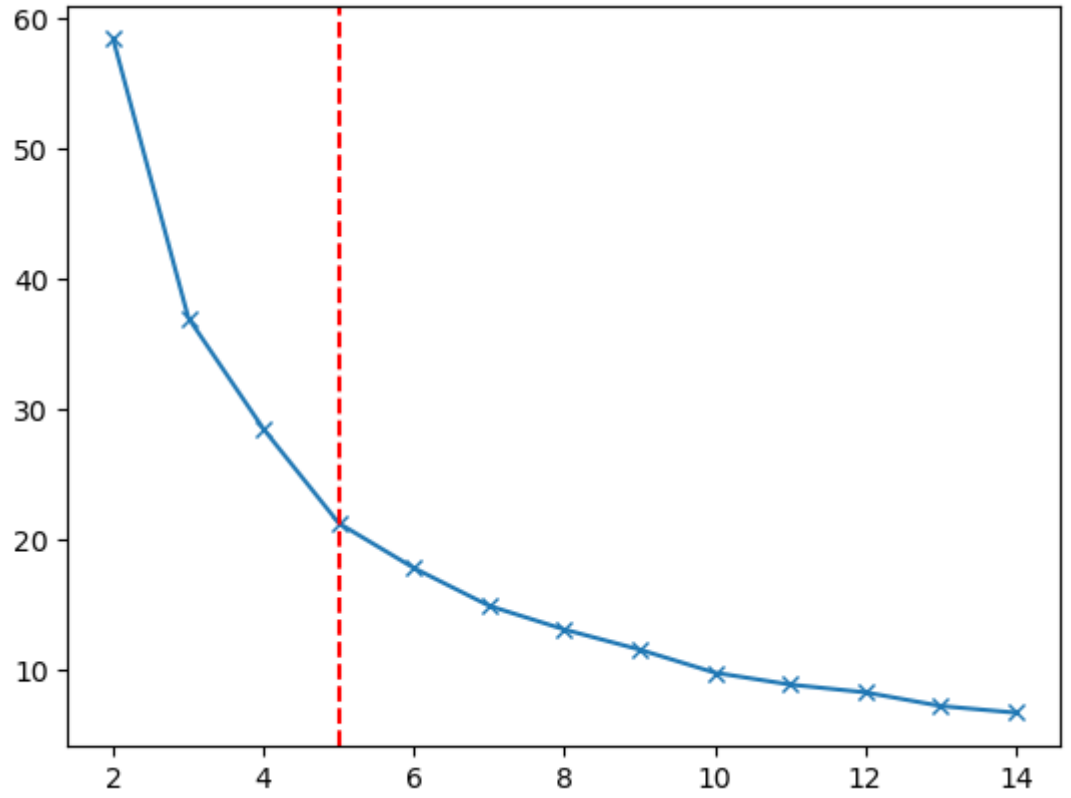
Plot the inertia values and estimate which k starts the “bend” (elbow) in the graph

```
for k in k_range:
    km_model = KMeans(n_clusters=k, n_init="auto", random_state=1)
    km_model.fit(new_df)
    k_inertia_list.append(km_model.inertia_)
```

KMeans: Elbow Method





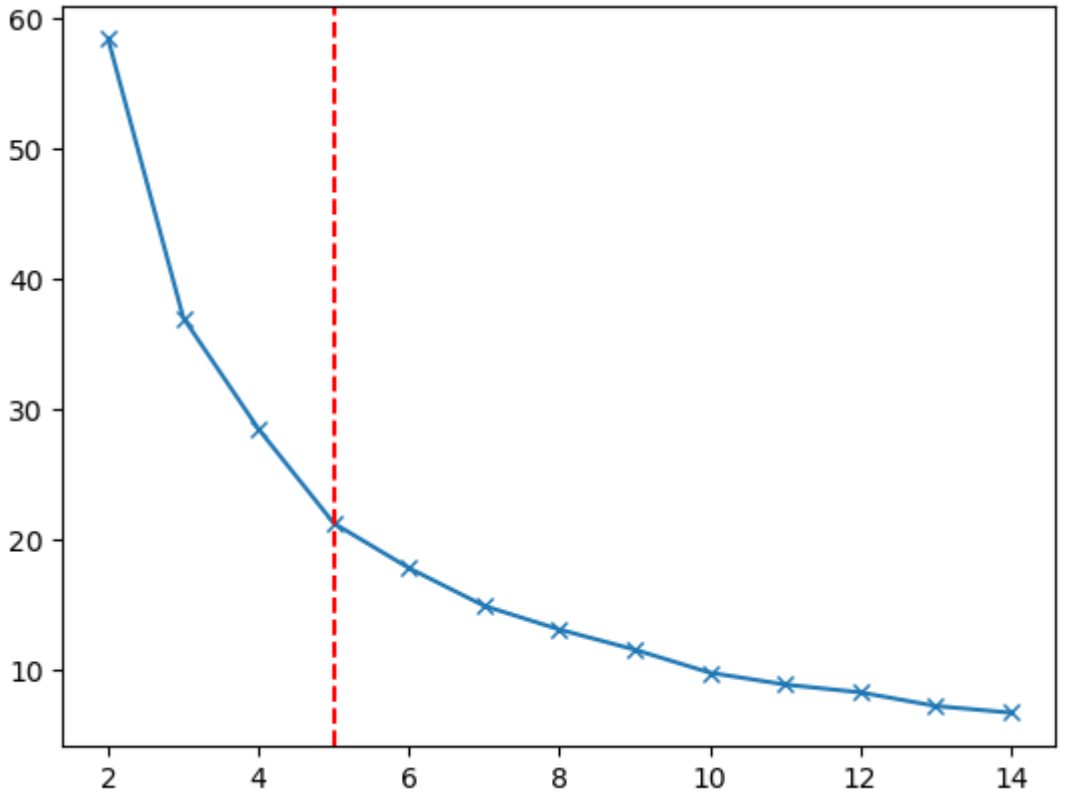


Estimate from elbow method: k = 5

KMeans: Elbow Method

```
plt.plot(k_list, k_inertia_list, 'x-')

# plot a vertical line at x = 5
plt.axvline(x=5, color='r', linestyle='--')
plt.show()
```

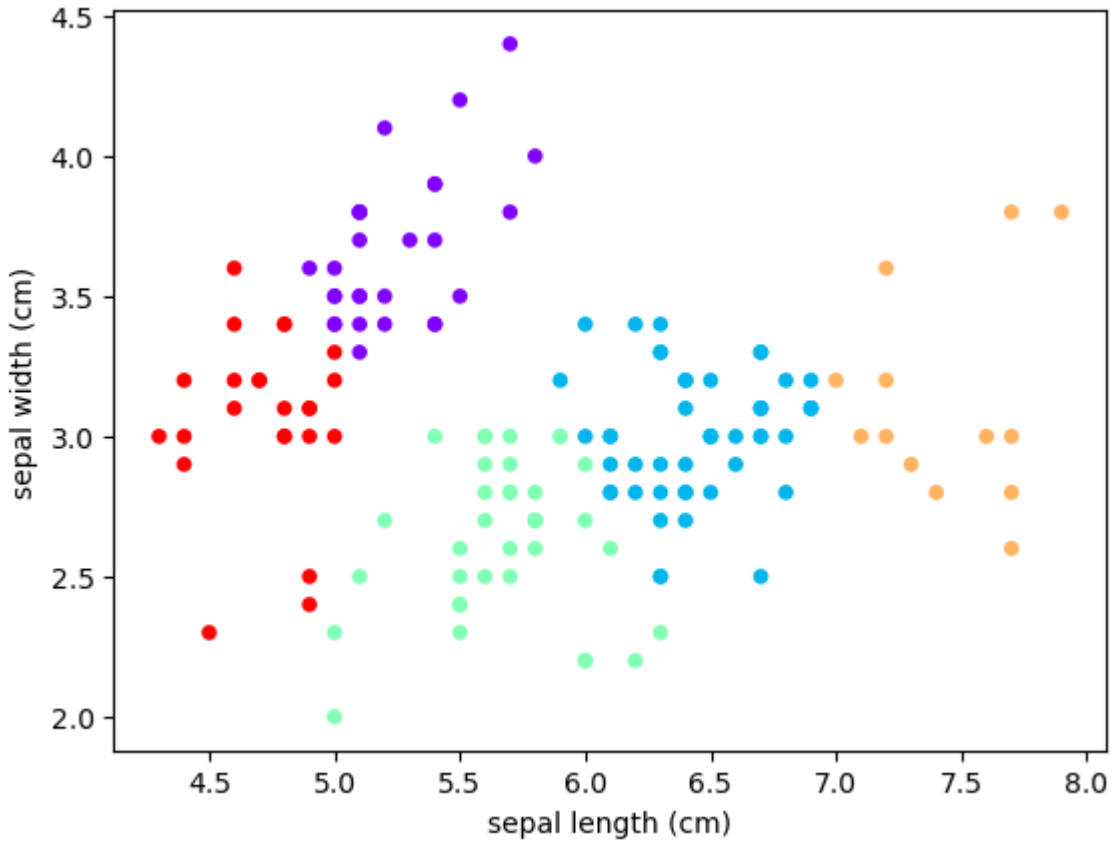


Estimate from elbow method: k = 5

KMeans: Elbow Method

```
km_model = KMeans(n_clusters=5,
                  n_init="auto",
                  random_state=1)
km_model.fit(new_df)
```

```
KMeans(n_clusters=5, n_init='auto', random_state=1)
```



Estimate from elbow method: k = 5

Overview

Iris Clustering Jupyter Notebook Outline:

1. Prepare dataset (use two features: sepal length and sepal width)
2. Hierarchical clustering
    1. Visualize dendrogram to get number of clusters
    2. Use Agglomerative Clustering and plot results
3. KMeans
    1. Use KMeans for best k based on Silhouette score and plot results
    2. Demonstrate elbow method, use KMeans, and plot results