PREDICTING IRIS FLOWER SPECIES WITH K-MEANS CLUSTERING IN PYTHON

Belen Sanchez Follow
Oct 22, 2018 · 3 min read ★

Photo by Anton Scherbakov on Unsplash

Clustering is an unsupervisedlearning method that allows us to group set of objects based on similar characteristics. In general, it can help you find meaningful structure among your data, group similar data together and discover underlying patterns.

One of the most common clustering methods is K-means algorithm. The goal of this algorithm isto partition the data into set such that the total sum of squared distances from each point to the mean point of the cluster is minimized.

K means works through the following iterative process:

1. Pick a value for k (the number of clusters to create)
2. Initialize k 'centroids' (starting points) in your data
3. Create your clusters. Assign each point to the nearest centroid.
4. Make your clusters better. Move each centroid to the center of its cluster.
5. Repeat steps 3–4 until your centroids converge.

How to apply it?

For the following example, I am going to use the Iris data set of scikit learn. This data consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). It has four features from each sample: length and width of sepals and petals.

1. To start let's import the following libraries.

```
from sklearn import datasets

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans
```

2. Load the data.

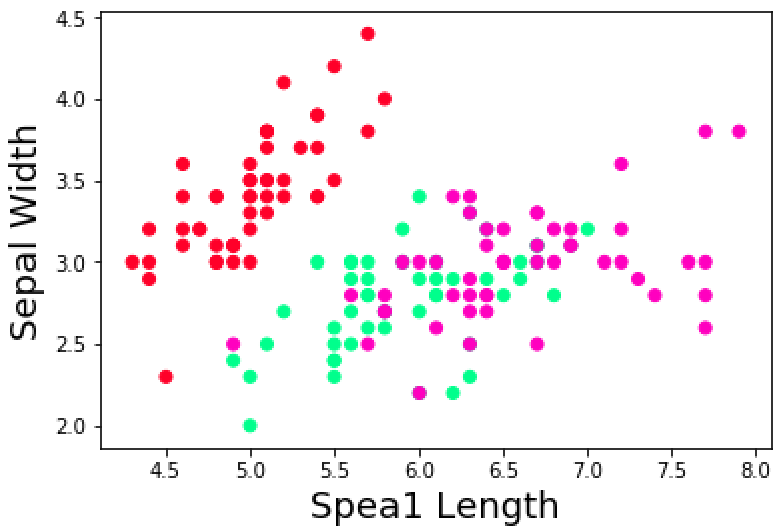
```
iris = datasets.load_iris()
```

3. Define your target and predictors.

```
X = iris.data[:, :2]

y = iris.target
```

4. Let’s have a look at our data through a scatter plot.

```
plt.scatter(X[:,0], X[:,1], c=y, cmap='gist_rainbow')
plt.xlabel('Speal Length', fontsize=18)
plt.ylabel('Sepal Width', fontsize=18)
```



5. Now, let’s instantiate and fit our K means cluster model. We are going to use three clusters and a random state of 21.

```
km = KMeans(n_clusters = 3, n_jobs = 4, random_state=21)

km.fit(X)
```

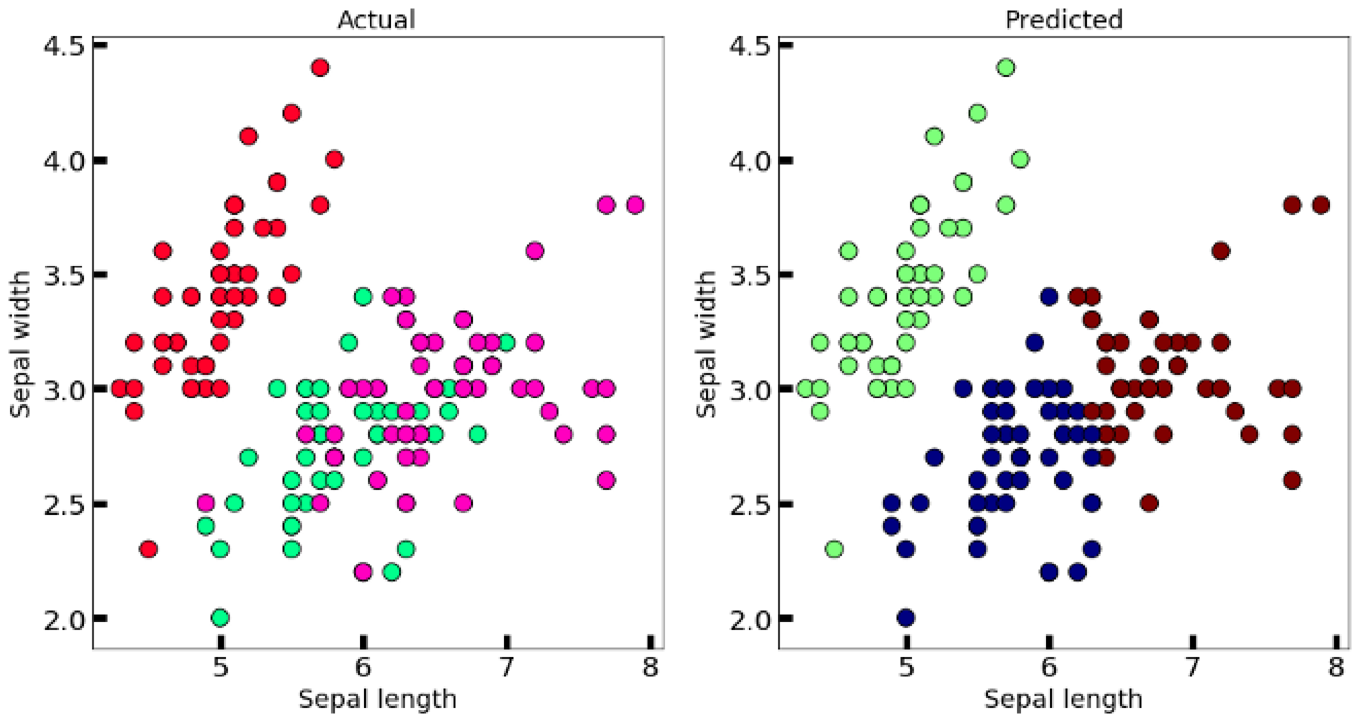
6. With the following code you can identify the center points of the data.

```
centers = km.cluster_centers_

print(centers)
```

```
Output
[[5.77358491 2.69245283]

[5.006      3.418    ]

[6.81276596 3.07446809]]
```

7. Now, let’s compare our original data versus our clustered results using the following code.

```
#this will tell us to which cluster does the data observations
belong.

new_labels = km.labels_

# Plot the identified clusters and compare with the answers

fig, axes = plt.subplots(1, 2, figsize=(16,8))

axes[0].scatter(X[:, 0], X[:, 1], c=y, cmap='gist_rainbow',

edgecolor='k', s=150)

axes[1].scatter(X[:, 0], X[:, 1], c=new_labels, cmap='jet',

edgecolor='k', s=150)

axes[0].set_xlabel('Sepal length', fontsize=18)

axes[0].set_ylabel('Sepal width', fontsize=18)

axes[1].set_xlabel('Sepal length', fontsize=18)

axes[1].set_ylabel('Sepal width', fontsize=18)

axes[0].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20)

axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20)

axes[0].set_title('Actual', fontsize=18)

axes[1].set_title('Predicted', fontsize=18)
```



Here is a list of the main advantages and disadvantages of this algorithm.

Advantages:

- K-Means is simple and computationally efficient.
- It is very intuitive and their results are easy to visualize.

Disadvantages:

- K-Means is highly scale dependent and is not suitable for data of varying shapes and densities.
- Evaluating results is more subjective. It requires much more human evaluation than trusted metrics.

Machine Learning