sk_kMeans

April 15, 2019

1 k-means using sklearn

1.0.1 Simple algorithm for K-means clusturing

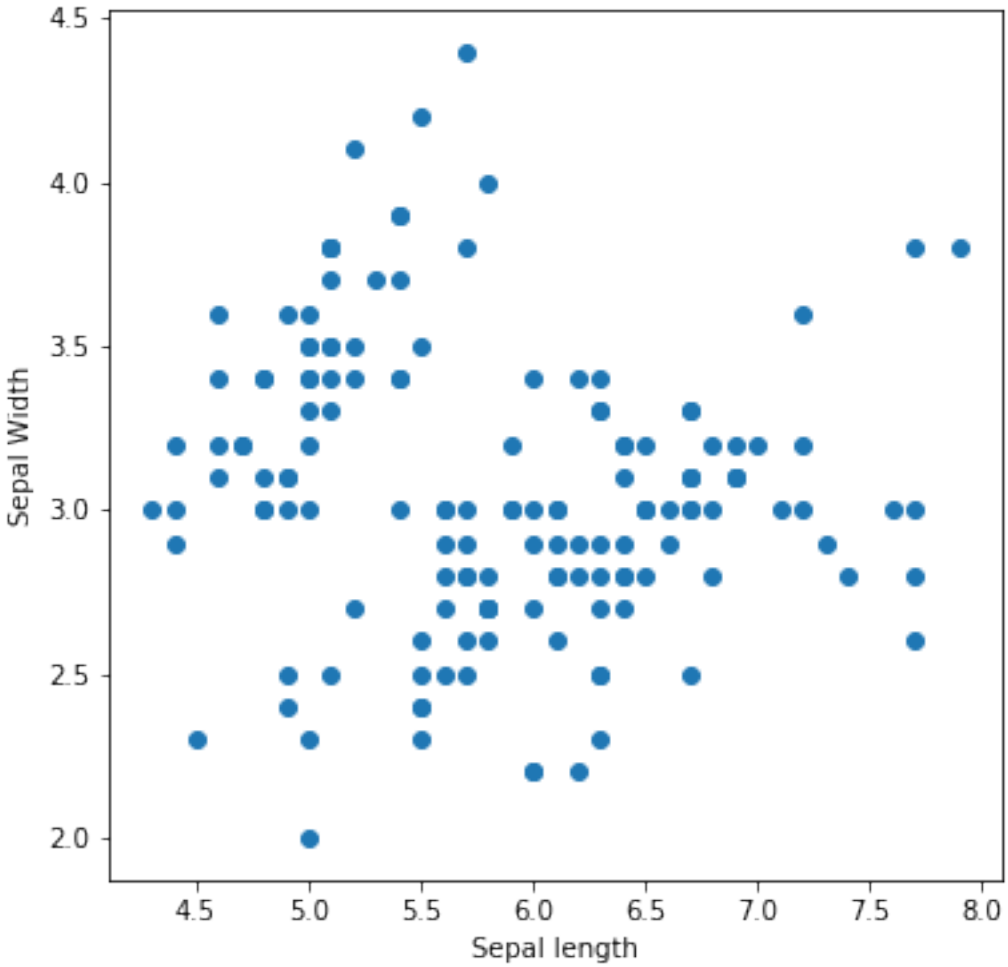
1. Find the Euclidean distance between each data instance and centroids of all the clusters
2. Assign the data instances to the cluster of the centroid with nearest distance
3. Calculate new centroid values based on the mean values of the coordinates of all the data instances from the corresponding cluster.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: dataset = datasets.load_iris()
        dataset.feature_names

Out[2]: ['sepal length (cm)',
         'sepal width (cm)',
         'petal length (cm)',
         'petal width (cm)']

In [3]: # Feature selection
        X = dataset.data[:, np.array([True, True, False, True])]
        plt.figure(figsize=(6, 6))
        plt.scatter( X[:, 0], X[:, 1])
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal Width')
        plt.show()
```




```
In [4]: # Model
        kmeans = KMeans(n_clusters=3)
        kmeans.fit(X)
```

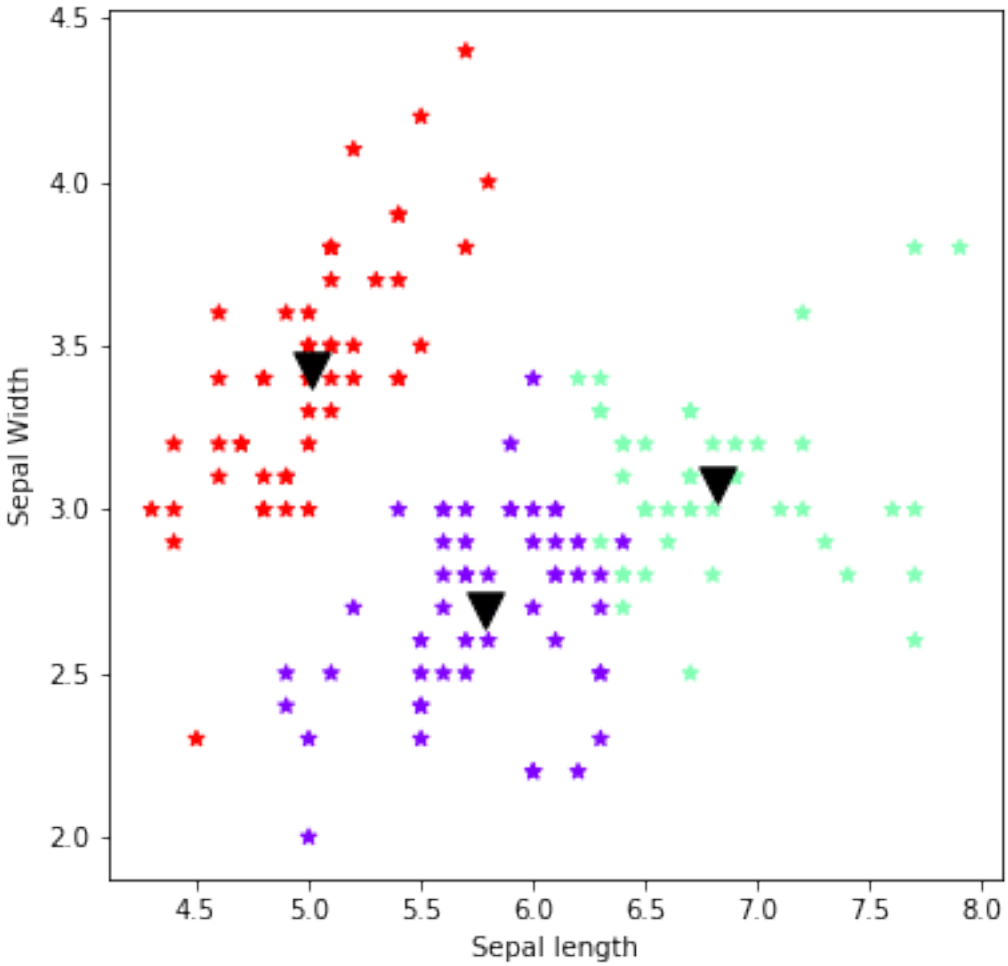
```
Out[4]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

```
In [5]: print(kmeans.cluster_centers_)
```

```
[[5.78518519 2.6962963  1.43148148]
 [6.82173913 3.07826087 1.96304348]
 [5.006      3.428      0.246     ]]
```

```
In [6]: markers = ["*", "v", "s"]
        plt.figure(figsize=(6, 6))
```

```
plt.scatter( X[:, 0], X[:, 1], c=kmeans.labels_, cmap='rainbow', marker="*")
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], color='black',
plt.xlabel('Sepal length')
plt.ylabel('Sepal Width')
plt.show()
```



1.1 References:

1. https://stackabuse.com/k-means-clustering-with-scikit-learn/
2. https://stackoverflow.com/questions/28296670/remove-a-specific-feature-in-scikit-learn