# K-Means Clustering with Scikit-learn

This example implements K-means clustering with Scikit-learn. Since this is an unsupervised learning use case, we use just the features of the Iris dataset to cluster the observations into labels.

```
# import packages
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets
from sklearn.model_selection import train_test_split

# load dataset
data = datasets.load_iris()

# get the dataset features
X = data.data

# create the model. Since we know that the Iris dataset has 3 classes, we
set n_clusters = 3
kmeans = KMeans(n_clusters=3, random_state=0)

# fit the model on the training set
kmeans.fit(X)
'Output':
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)

# predict the closest cluster each sample in X belongs to.
y_kmeans = kmeans.predict(X)

# plot clustered labels
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, cmap='viridis')

# plot cluster centers
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.7);
plt.show()
```

The code to plot the clustered labels and the cluster centers should be executed in the same notebook. The plot of clusters made by the K-means algorithm is shown in Figure 25-4.

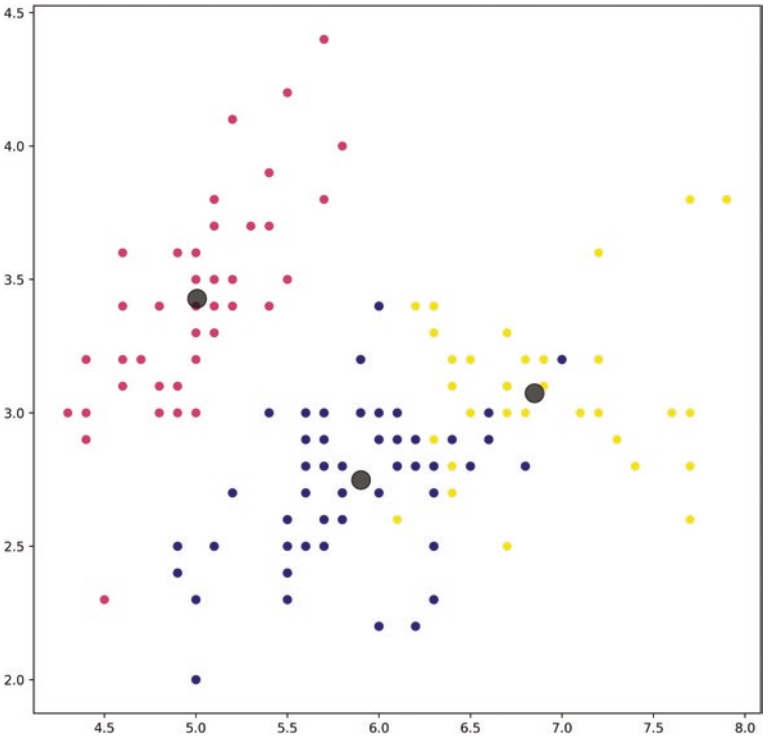


***Figure 25-4.*** *Plot of K-means clusters and their cluster centers*

# Hierarchical Clustering

Hierarchical clustering is another clustering algorithm for finding homogeneous sub-groups or classes within a dataset. However, as opposed to *k*-means, we do not need to make an a priori assumption of the number of clusters in the dataset before running the algorithm.

The two main techniques for performing hierarchical clustering are

- Bottom-up or agglomerative
- Top-down or divisive