**Unsupervised Learning (K-Means Clustering)**

Let’s take a look at how we could go about classifying data using the K-Means algorithm with python. As always, we need to start by importing the required libraries.

import numpy as np  
import pandas as pd  
from matplotlib import pyplot as plt  
from sklearn.datasets.samples\_generator import make\_blobs  
from sklearn.cluster import KMeans

In this tutorial, we’ll generate our own data using the make\_blobs function from the sklearn.datasets module. The centers parameter specifies the number of clusters.

X, y = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)plt.scatter(X[:,0], X[:,1])

Even though we already know the optimal number of clusters, I figured we could still benefit from determining it using the **elbow method**. To get the values used in the graph, we train multiple models using a different number of clusters and storing the value of the intertia\_ property (WCSS) every time.

wcss = []for i in range(1, 11):  
 kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)  
 kmeans.fit(X)  
 wcss.append(kmeans.inertia\_)  
plt.plot(range(1, 11), wcss)  
plt.title('Elbow Method')  
plt.xlabel('Number of clusters')  
plt.ylabel('WCSS')  
plt.show()

Next, we’ll categorize the data using the optimum number of clusters (4) we determined in the last step. k-means++ ensures that you get don’t fall into the random initialization trap.

kmeans = KMeans(n\_clusters=4, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)  
pred\_y = kmeans.fit\_predict(X)plt.scatter(X[:,0], X[:,1])  
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='red')  
plt.show()

**Principal Component Analysis with Python**

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

Each of the principal components is chosen in such a way so that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has a maximum variance.

Click on the link

<https://www.geeksforgeeks.org/principal-component-analysis-with-python/>