K Means Clustering Algorithm Implementation with Python

Here we considered a dataset of students with name, roll number and marks. You can find the dataset below

Initially we load some basic libraries and our dataset.

from sklearn.cluster import KMeans

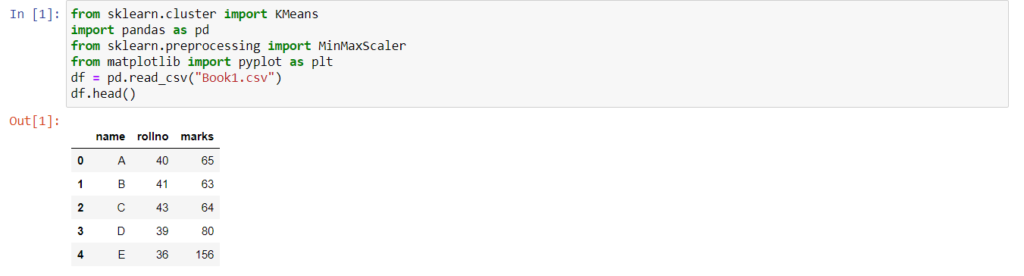
import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

df = pd.read\_csv("Book1.csv")

df.head()

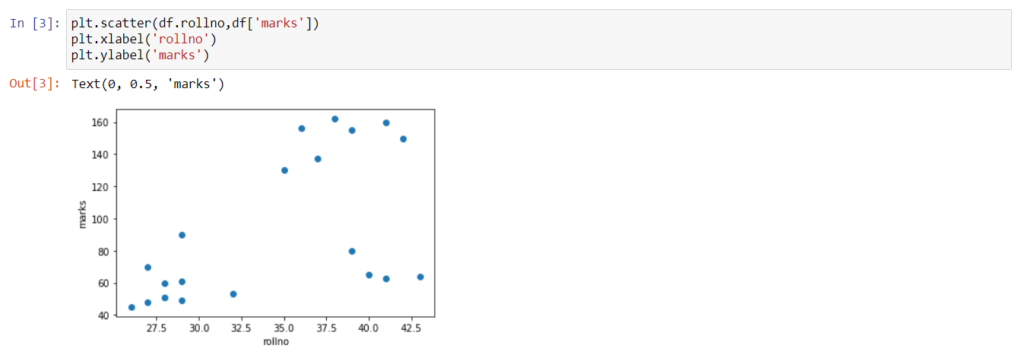


To understand our data we can use visualization and plot a graph to find out the groups/clusters that could be formed. From below mention plot it is clear that we can easily create 3 clusters.

plt.scatter(df.rollno,df['marks'])

plt.xlabel('rollno')

plt.ylabel('marks')

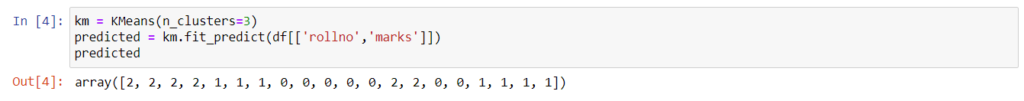


Hence we use Kmeans method and mention n\_clusters equals to 3. After that we use fit\_predict method to predict which sample will be assigned to which cluster. Here cluster 1 is ‘0’, cluster 2 is ‘1’ and cluster 3 is ‘2’.

km = KMeans(n\_clusters=3)

predicted = km.fit\_predict(df[['rollno','marks']])

predicted



Now we will add these predictions to our original dataset and after that we can split the main data frame into 3 data frames based on cluster(0,1 and 2). Then we can plot those data framed points on graph with different colors to see whether we go it all right(the predictions). But from below mentioned plot we can see that 2 sample/data point which should belong to cluster green are actually colored blue, that means we added them(those 2 sample points) in a wrong cluster.

df['cluster']=predicted

df.head()

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

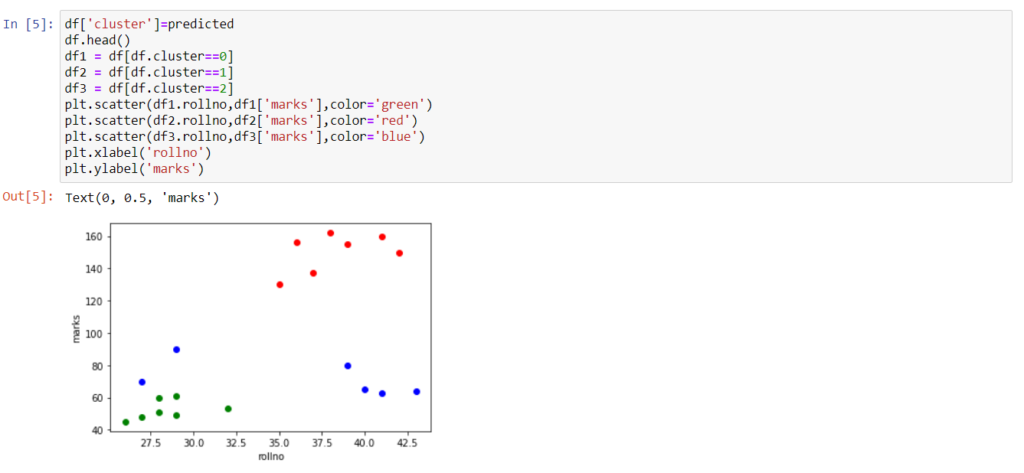
plt.scatter(df1.rollno,df1['marks'],color='green')

plt.scatter(df2.rollno,df2['marks'],color='red')

plt.scatter(df3.rollno,df3['marks'],color='blue')

plt.xlabel('rollno')

plt.ylabel('marks')



The sample points got added into a wrong group because our data is not properly scaled, so now we need to scale our data using normalization. For that we need the help of MinMaxscaler to scale our data in range of 0 and 1. After that we again try to take the predictions using Kmeans and fit\_predict methods as we did previously on unscaled data.

scale = MinMaxScaler()

scale.fit(df[['marks']])

df['marks'] = scale.transform(df[['marks']])

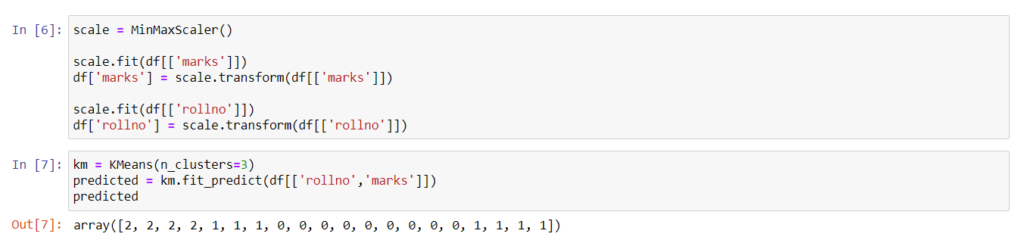
scale.fit(df[['rollno']])

df['rollno'] = scale.transform(df[['rollno']])

km = KMeans(n\_clusters=3)

predicted = km.fit\_predict(df[['rollno','marks']])

predicted



We will now delete or drop the previously added cluster column and add a new one with updated predictions(with scaled data) to our dataset. Now when we try to plot the clusters, we can observe that each sample point of our dataset is clustered correctly.

df = df.drop(['cluster'], axis='columns')

df['cluster']=predicted

df.head()

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

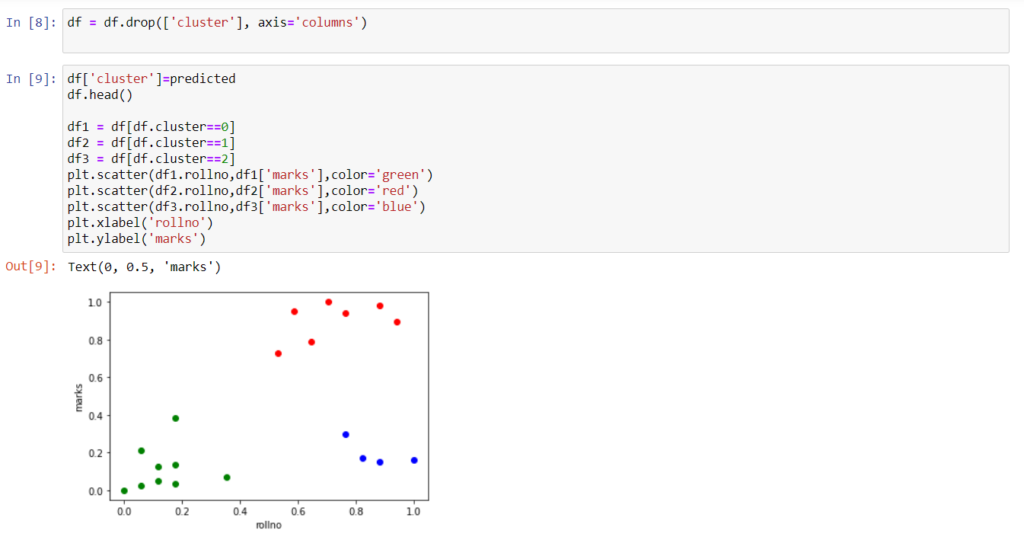
plt.scatter(df1.rollno,df1['marks'],color='green')

plt.scatter(df2.rollno,df2['marks'],color='red')

plt.scatter(df3.rollno,df3['marks'],color='blue')

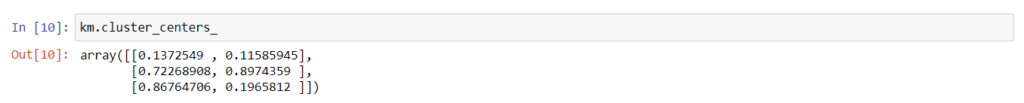
plt.xlabel('rollno')

plt.ylabel('marks')



We can also generate the centroids of our clusters by using *cluster\_centers\_*

km.cluster\_centers\_



The generated centroid values can also be represented on our plot using the following code. Centroids of each cluster is represented by a black ‘\*’ marker.

plt.scatter(df1.rollno,df1['marks'],color='green')

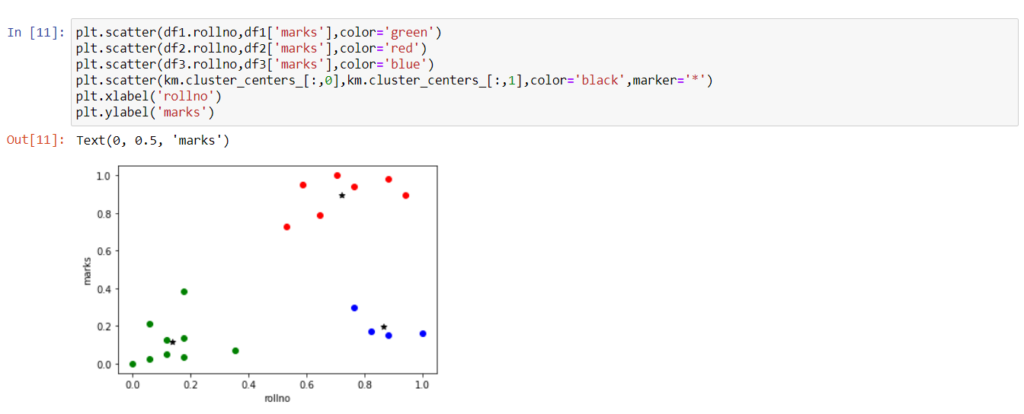
plt.scatter(df2.rollno,df2['marks'],color='red')

plt.scatter(df3.rollno,df3['marks'],color='blue')

plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],color='black',marker='\*')

plt.xlabel('rollno')

plt.ylabel('marks')



CATEGORIES[MACHINE LEARNING](https://5minutesengineering.com/category/machine-learning/)