import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

#take data in form of data frame

#take the data in x,y as x and y direction

df = pd.DataFrame({

'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],

'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24] })

#number of centroid increase number of clearity

#randomly select centroid from data frame i=values in data frame,,k=randomly taking centroid

np.random.seed(200)

k = 3

# centroids[i] = [x, y]

centroids = { i+1: [np.random.randint(0, 80), np.random.randint(0, 80)]

for i in range(k)

}

#color the centroid and plot it

#3 color for 3 centroid

#scatter points in x,y direction

fig = plt.figure(figsize=(8,8))

plt.scatter(df['x'], df['y'], color='k')

colmap = {1: 'r', 2: 'g', 3: 'b'}

for i in centroids.keys():

plt.scatter(\*centroids[i], color=colmap[i])

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()

#euclidean distance=sqrt(variable all - centroid)^2

def assignment(df, centroids):

for i in centroids.keys():

# sqrt((x1 - x2)^2 - (y1 - y2)^2)

df['distance\_from\_{}'.format(i)] = (np.sqrt( (df['x'] - centroids[i][0]) \*\* 2

+ (df['y'] - centroids[i][1]) \*\* 2))

#after calculating euclidean distance which ever variable and centroid distance is close to centroid is choosen to in that group of centroid then after centroid is get update .

#all values of euclidean distance are stored in centroid\_distance\_cols

#selecting the closest point near to centroid by minimum edistance

centroid\_distance\_cols = ['distance\_from\_{}'.format(i) for i in centroids.keys()]

#idxmin

df['closest'] = df.loc[:, centroid\_distance\_cols].idxmin(axis=1)

#lambda x means collecting values nearear to centroid in 1 group and color them

df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance\_from\_')))

df['color'] = df['closest'].map(lambda x: colmap[x])

return df

#use of assignment defination df=values of x,y

df = assignment(df, centroids)

print(df.head())

#plot figure

fig = plt.figure(figsize=(8,8))

plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')

for i in centroids.keys():

plt.scatter(\*centroids[i], color=colmap[i])

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()

# after randomly choose centroid and calc. ED and taking minimum dist. and collecting all update new centroid

import copy

#colletion of old centroid for new centroid

old\_centroids = copy.deepcopy(centroids)

#update centroid by taking mean

def update(k):

for i in centroids.keys():

centroids[i][0] = np.mean(df[df['closest'] == i]['x'])

centroids[i][1] = np.mean(df[df['closest'] == i]['y'])

return k

centroids = update(centroids)

df = assignment(df, centroids)

# Plot results

fig = plt.figure(figsize=(8,8))

plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')

for i in centroids.keys():

plt.scatter(\*centroids[i], color=colmap[i])

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()

#repeat all this step upto centroid will not change

while True:

closest\_centroids = df['closest'].copy(deep=True)

centroids = update(centroids)

df = assignment(df, centroids)

if closest\_centroids.equals(df['closest']):

break

#plot it

fig = plt.figure(figsize=(8,8))

plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')

for i in centroids.keys():

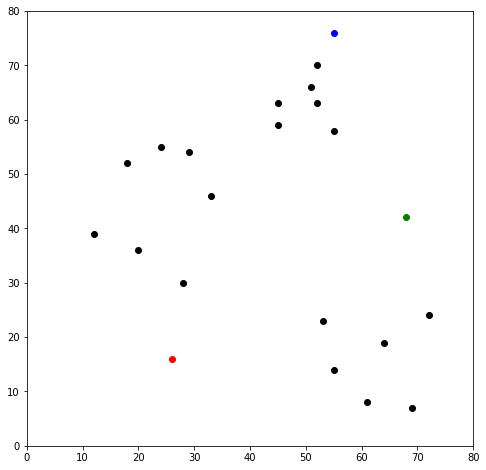
plt.scatter(\*centroids[i], color=colmap[i])

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()

**output:**

****

x y distance\_from\_1 distance\_from\_2 distance\_from\_3 closest color

0 12 39 26.925824 56.080300 56.727418 1 r

1 20 36 20.880613 48.373546 53.150729 1 r

2 28 30 14.142136 41.761226 53.338541 1 r

3 18 52 36.878178 50.990195 44.102154 1 r

4 29 54 38.118237 40.804412 34.058773 3 b

