**PRACTICAL – 8**

**#WAP for K-Means clustering**

## Initialisation

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

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] })

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)

}

fig = plt.figure(figsize=(5, 5))

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

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

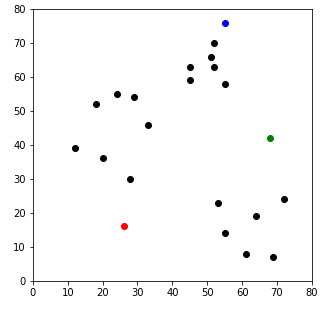
**for** i **in** centroids.keys():

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

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()



**## Assignment Stage**

**def** assignment(df, centroids):

**for** i **in** centroids.keys():

df['distance\_from\_{}'.format(i)] = (

np.sqrt( (df['x'] - centroids[i][0]) \*\* 2 + (df['y'] - centroids[i][1]) \*\* 2) )

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

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

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

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

**return** df

df = assignment(df, centroids)

**print**(df.head())

fig = plt.figure(figsize=(5, 5))

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

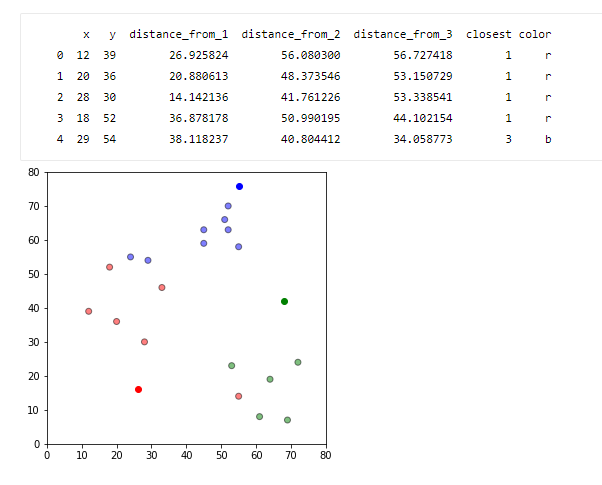
**for** i **in** centroids.keys():

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

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()



**## Update Stage**

**import** **copy**

old\_centroids = copy.deepcopy(centroids)

**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)

fig = plt.figure(figsize=(5, 5))

ax = plt.axes()

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)

**for** i **in** old\_centroids.keys():

old\_x = old\_centroids[i][0]

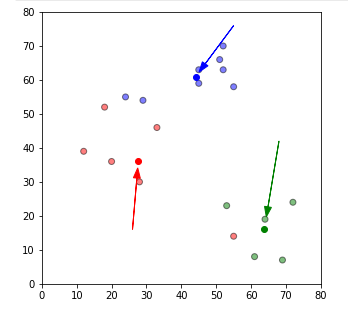
old\_y = old\_centroids[i][1]

dx = (centroids[i][0] - old\_centroids[i][0]) \* 0.75

dy = (centroids[i][1] - old\_centroids[i][1]) \* 0.75

ax.arrow(old\_x, old\_y, dx, dy, head\_width=2, head\_length=3, fc=colmap[i], ec=colmap[i])

plt.show()



## Repeat Assigment Stage

df = assignment(df, centroids)

# Plot results

fig = plt.figure(figsize=(5, 5))

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

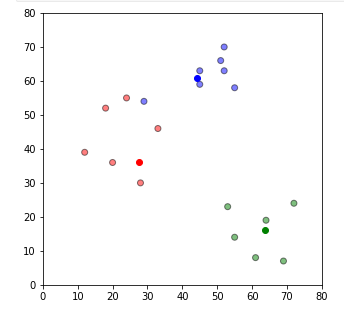
**for** i **in** centroids.keys():

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

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()



**# Continue until all assigned categories don't change any more**

**while** True:

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

centroids = update(centroids)

df = assignment(df, centroids)

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

**break**

fig = plt.figure(figsize=(5, 5))

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

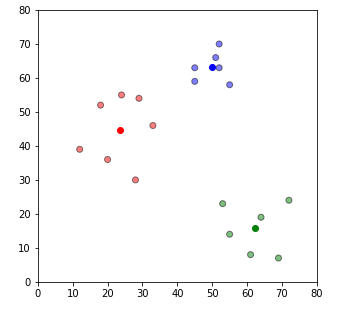
**for** i **in** centroids.keys():

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

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()



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] } )

**from** **sklearn.cluster** **import** KMeans

kmeans = KMeans(n\_clusters=3)

kmeans.fit(df)

labels = kmeans.predict(df)

centroids = kmeans.cluster\_centers\_

fig = plt.figure(figsize=(5, 5))

colors = map(**lambda** x: colmap[x+1], labels)

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

**for** idx, centroid **in** enumerate(centroids):

plt.scatter(\*centroid, color=colmap[idx+1])

plt.xlim(0, 80)

plt.ylim(0, 80)

plt.show()

