K-Means Clustering

K-Means Clustering is a concept that falls under Unsupervised Learning. This algorithm can be used to find groups within unlabeled data. To demonstrate this concept, I’ll review a simple example of K-Means Clustering in Python.

Topics to be covered:

* Creating the DataFrame for two-dimensional data-set
* Finding the centroids for 3 clusters, and then for 4 clusters
* Adding a graphical user interface (GUI) to display the results

from pandas import DataFrame

Data = {'x': [25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],

'y': [79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]

}

df = DataFrame(Data,columns=['x','y'])

print (df)

### K-Means Clustering in Python – ****3**** clusters

Once you created the DataFrame based on the above data, you’ll need to import 2 additional Python modules:

Assign 3 clusters as follows:

KMeans(n\_clusters=**3**).fit(df)

from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

Data = {'x': [25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],

'y': [79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]

}

df = DataFrame(Data,columns=['x','y'])

kmeans = KMeans(n\_clusters=3).fit(df)

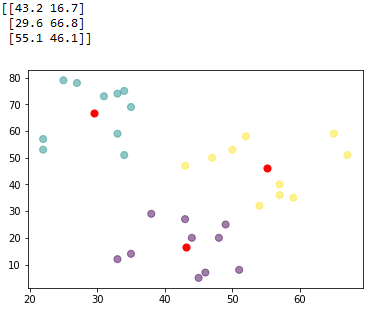
centroids = kmeans.cluster\_centers\_

print(centroids)

plt.scatter(df['x'], df['y'], c= kmeans.labels\_.astype(float), s=50, alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)

Run the code in Python and you would see 3 clusters with 3 distinct centroids:



Note that the center of each cluster (in red) represents the mean of all the observations that belong to that cluster.

As you may also see, the observations that belong to a given cluster are closer to the center of that cluster, in comparison to the centers of other clusters.

### K-Means Clustering in Python – ****4**** clusters

Let’s now see what would happen if you use 4 clusters instead. In that case, the only thing you’ll need to do is to change the n\_clusters from 3 to 4:

KMeans(n\_clusters=**4**).fit(df)

And so, your full Python code for 4 clusters would look like this:

from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

Data = {'x': [25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],

'y': [79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]

}

df = DataFrame(Data,columns=['x','y'])

kmeans = KMeans(n\_clusters=4).fit(df)

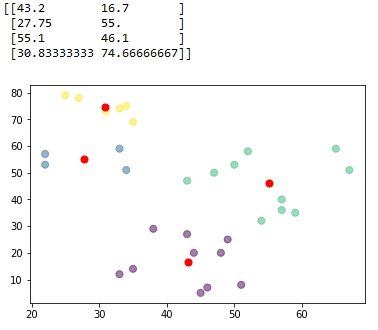
centroids = kmeans.cluster\_centers\_

print(centroids)

plt.scatter(df['x'], df['y'], c= kmeans.labels\_.astype(float), s=50, alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)

Run the code, and you’ll now see 4 clusters with 4 distinct centroids:



### Tkinter GUI to Display the Results

This is the code that you can use (for 3 clusters):

from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import tkinter as tk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

Data = {'x': [25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],

'y': [79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]

}

df = DataFrame(Data,columns=['x','y'])

kmeans = KMeans(n\_clusters=3).fit(df)

centroids = kmeans.cluster\_centers\_

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 100, height = 100)

canvas1.pack()

label1 = tk.Label(root, text=centroids, justify = 'center')

canvas1.create\_window(70, 50, window=label1)

figure1 = plt.Figure(figsize=(5,4), dpi=100)

ax1 = figure1.add\_subplot(111)

ax1.scatter(df['x'], df['y'], c= kmeans.labels\_.astype(float), s=50, alpha=0.5)

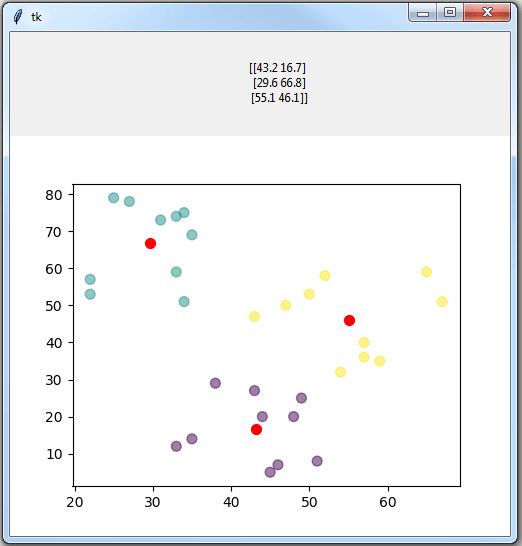
ax1.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)

scatter1 = FigureCanvasTkAgg(figure1, root)

scatter1.get\_tk\_widget().pack(side=tk.LEFT, fill=tk.BOTH)

root.mainloop()

And this is what you’ll get when running the code in Python:



### More Advanced Tkinter GUI

In the final section of this tutorial, I’ll share the code to create a more advanced tkinter GUI that will allow you to:

* Import an Excel file with two-dimensional data-set
* Type the number of clusters needed
* Display the clusters and centroids

Here is the full Python code:

import tkinter as tk

from tkinter import filedialog

import pandas as pd

from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 400, height = 300, relief = 'raised')

canvas1.pack()

label1 = tk.Label(root, text='k-Means Clustering')

label1.config(font=('helvetica', 14))

canvas1.create\_window(200, 25, window=label1)

label2 = tk.Label(root, text='Type Number of Clusters:')

label2.config(font=('helvetica', 8))

canvas1.create\_window(200, 120, window=label2)

entry1 = tk.Entry (root)

canvas1.create\_window(200, 140, window=entry1)

def getExcel ():

global df

import\_file\_path = filedialog.askopenfilename()

read\_file = pd.read\_excel (import\_file\_path)

df = DataFrame(read\_file,columns=['x','y'])

browseButtonExcel = tk.Button(text=" Import Excel File ", command=getExcel, bg='green', fg='white', font=('helvetica', 10, 'bold'))

canvas1.create\_window(200, 70, window=browseButtonExcel)

def getKMeans ():

global df

global numberOfClusters

numberOfClusters = int(entry1.get())

kmeans = KMeans(n\_clusters=numberOfClusters).fit(df)

centroids = kmeans.cluster\_centers\_

label3 = tk.Label(root, text= centroids)

canvas1.create\_window(200, 250, window=label3)

figure1 = plt.Figure(figsize=(4,3), dpi=100)

ax1 = figure1.add\_subplot(111)

ax1.scatter(df['x'], df['y'], c= kmeans.labels\_.astype(float), s=50, alpha=0.5)

ax1.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)

scatter1 = FigureCanvasTkAgg(figure1, root)

scatter1.get\_tk\_widget().pack(side=tk.RIGHT, fill=tk.BOTH)

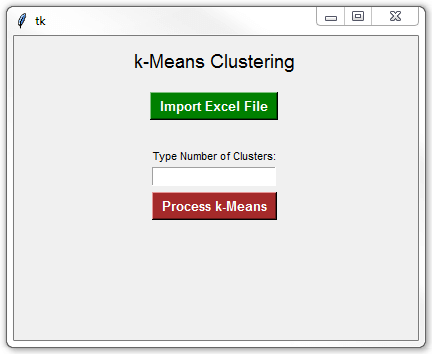
processButton = tk.Button(text=' Process k-Means ', command=getKMeans, bg='brown', fg='white', font=('helvetica', 10, 'bold'))

canvas1.create\_window(200, 170, window=processButton)

root.mainloop()

Before you run the above code, you’ll need to store your two-dimensional data-set in an Excel file. For example, I stored the date-set that we saw at the beginning of this post in an Excel file:

Then, run the Python code, and you’ll see the following GUI:



And this is the result that I got:

