<u>CSE-3019</u>

<u>DATA MINING</u>

LAB: L47 +L48

Name: Nimit Agrawal

REG. NO: 16BCE0170

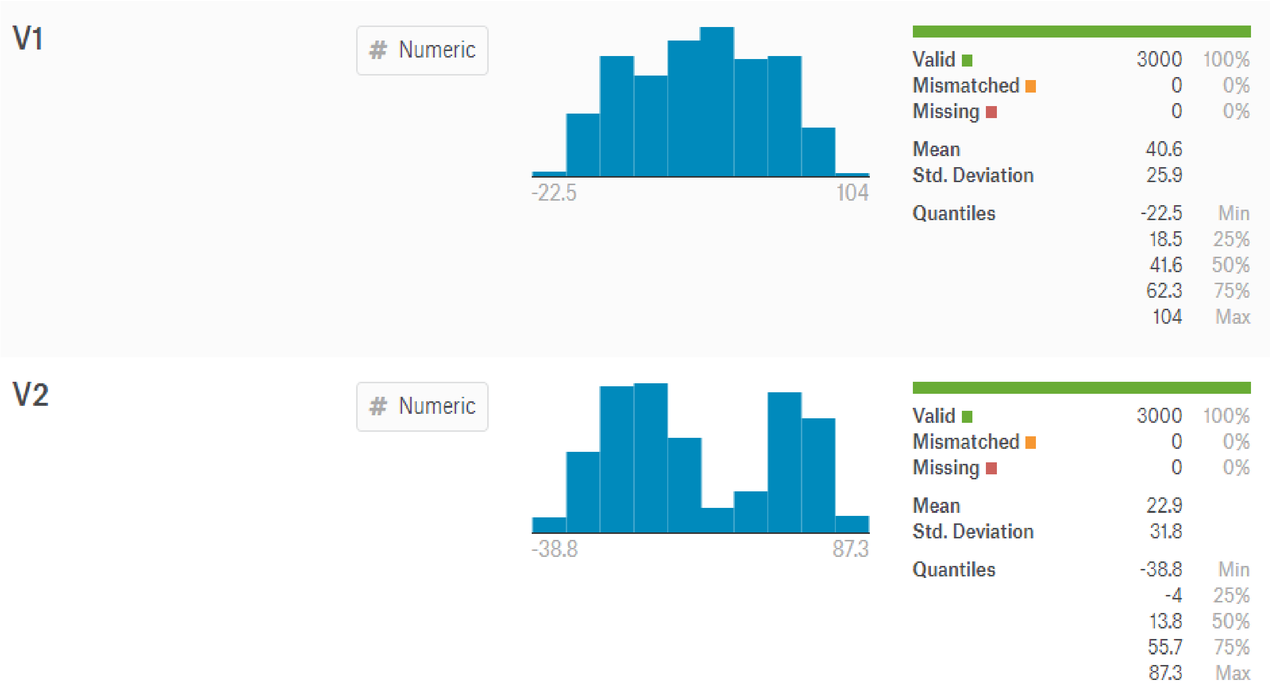
<u>Lab-5</u>

Choose an appropriate dataset for Clustering. (Min 20 records)

Write a python program to cluster the given dataset using K means clustering. obtain the result as with different K values.

Strengthen your output with appropriate graphical representation and inference.

Dataset: ***https://www.kaggle.com/hdriss/xclara#xclara.csv***



V1

Numeric

Valid 🟩	3000	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	40.6	
Std. Deviation	25.9	
Quantiles	-22.5	Min
	18.5	25%
	41.6	50%
	62.3	75%
	104	Max

V2

Numeric

Valid 🟩	3000	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	22.9	
Std. Deviation	31.8	
Quantiles	-38.8	Min
	-4	25%
	13.8	50%
	55.7	75%
	87.3	Max

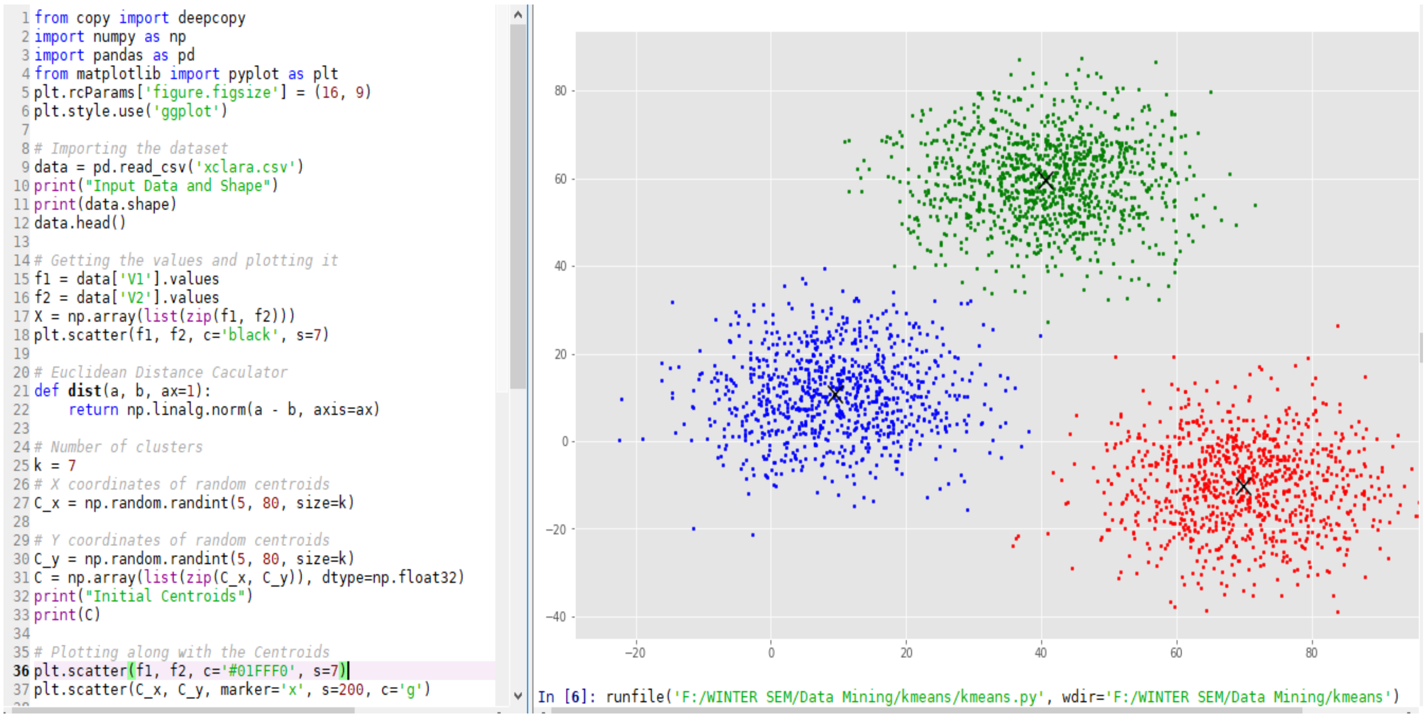
K means Code:

```
from copy import deepcopy
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

# Importing the dataset
data = pd.read_csv('xclara.csv')
print("Input Data and Shape")
print(data.shape)
data.head()

# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values

X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)

# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)
```

```
# Number of clusters
k = 6
# X coordinates of random centroids
C_x = np.random.randint(5, 80, size=k)

# Y coordinates of random centroids
C_y = np.random.randint(5, 80, size=k)

C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print("Initial Centroids")
print(C)



# Plotting along with the Centroids
plt.scatter(f1, f2, c='#01FFF0', s=7)
plt.scatter(C_x, C_y, marker='x', s=200, c='g')

# To store the value of centroids when it updates
C_old = np.zeros(C.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))

# Error func. - Distance between new centroids and old centroids
error = dist(C, C_old, None)
```

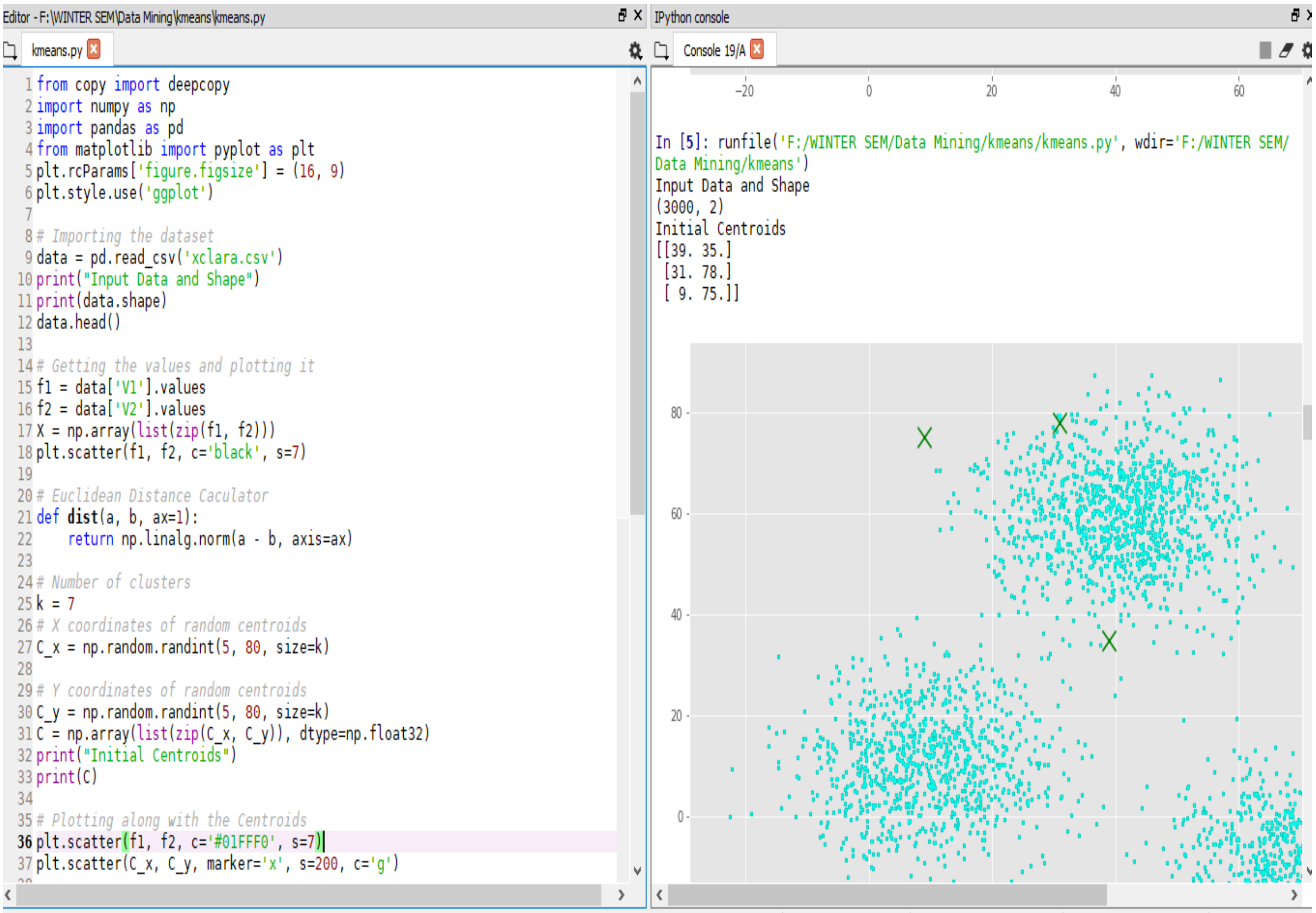
```
# Loop will run till the error becomes zero
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        distances = dist(X[i], C)
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(C)
    # Finding the new centroids by taking the average value
    for i in range(k):
        points = [X[j] for j in range(len(X)) if clusters[j] == i]
        C[i] = np.mean(points, axis=0)
    error = dist(C, C_old, None)

colors = ['r', 'g', 'b', 'y', 'c', 'm','p','o','w']
fig, ax = plt.subplots()

for i in range(k):
        points = np.array([X[j] for j in range(len(X)) if clusters[j] == i])
        ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])

ax.scatter(C[:, 0], C[:, 1], marker='x', s=200, c='#010101')
```

Code Snipets:



```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

# Importing the dataset
data = pd.read_csv('xclara.csv')
print("Input Data and Shape")
print(data.shape)
data.head()

# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)

# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

# Number of clusters
k = 7
# X coordinates of random centroids
C_x = np.random.randint(5, 80, size=k)

# Y coordinates of random centroids
C_y = np.random.randint(5, 80, size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print("Initial Centroids")
print(C)

# Plotting along with the Centroids
plt.scatter(f1, f2, c='#01FFF0', s=7)
plt.scatter(C_x, C_y, marker='x', s=200, c='g')
```

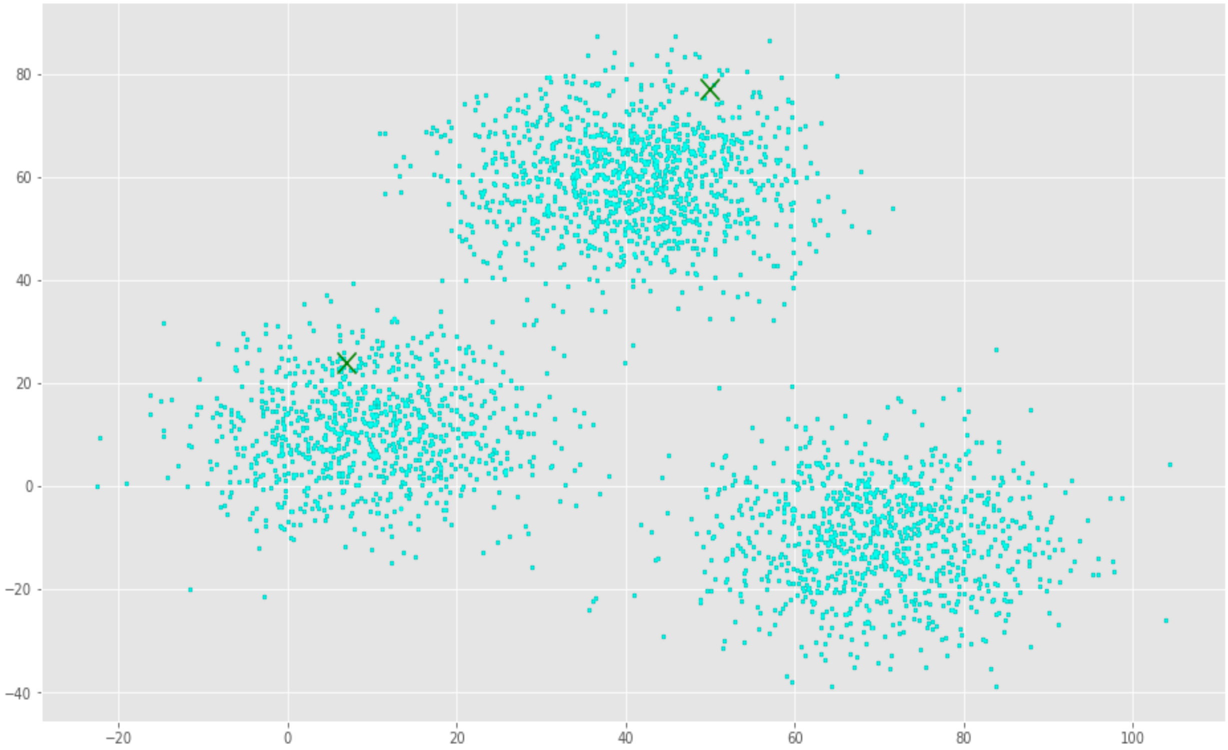
```
In [6]: runfile('F:/WINTER SEM/Data Mining/kmeans/kmeans.py', wdir='F:/WINTER SEM/Data Mining/kmeans')
```

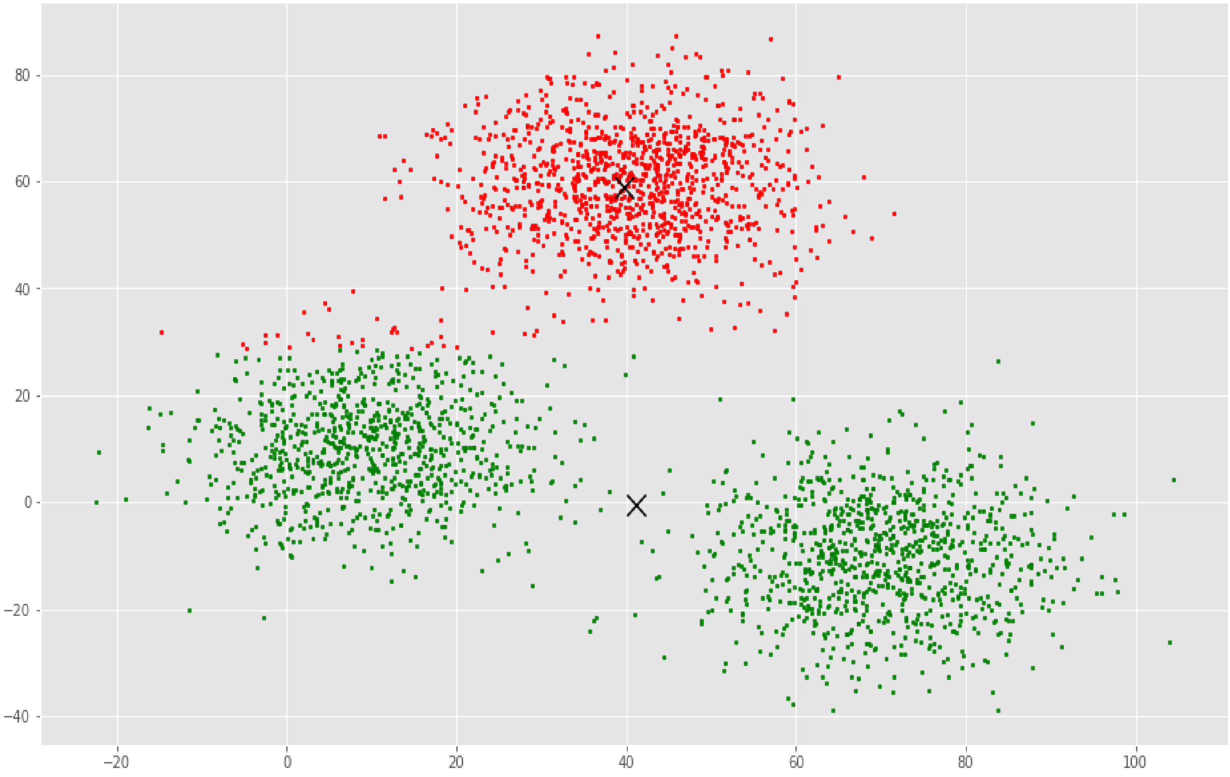


```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

# Importing the dataset
data = pd.read_csv('xclara.csv')
print("Input Data and Shape")
print(data.shape)
data.head()

# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)

# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

# Number of clusters
k = 7
# X coordinates of random centroids
C_x = np.random.randint(5, 80, size=k)

# Y coordinates of random centroids
C_y = np.random.randint(5, 80, size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print("Initial Centroids")
print(C)

# Plotting along with the Centroids
plt.scatter(f1, f2, c='#01FFF0', s=7)
plt.scatter(C_x, C_y, marker='x', s=200, c='g')
```

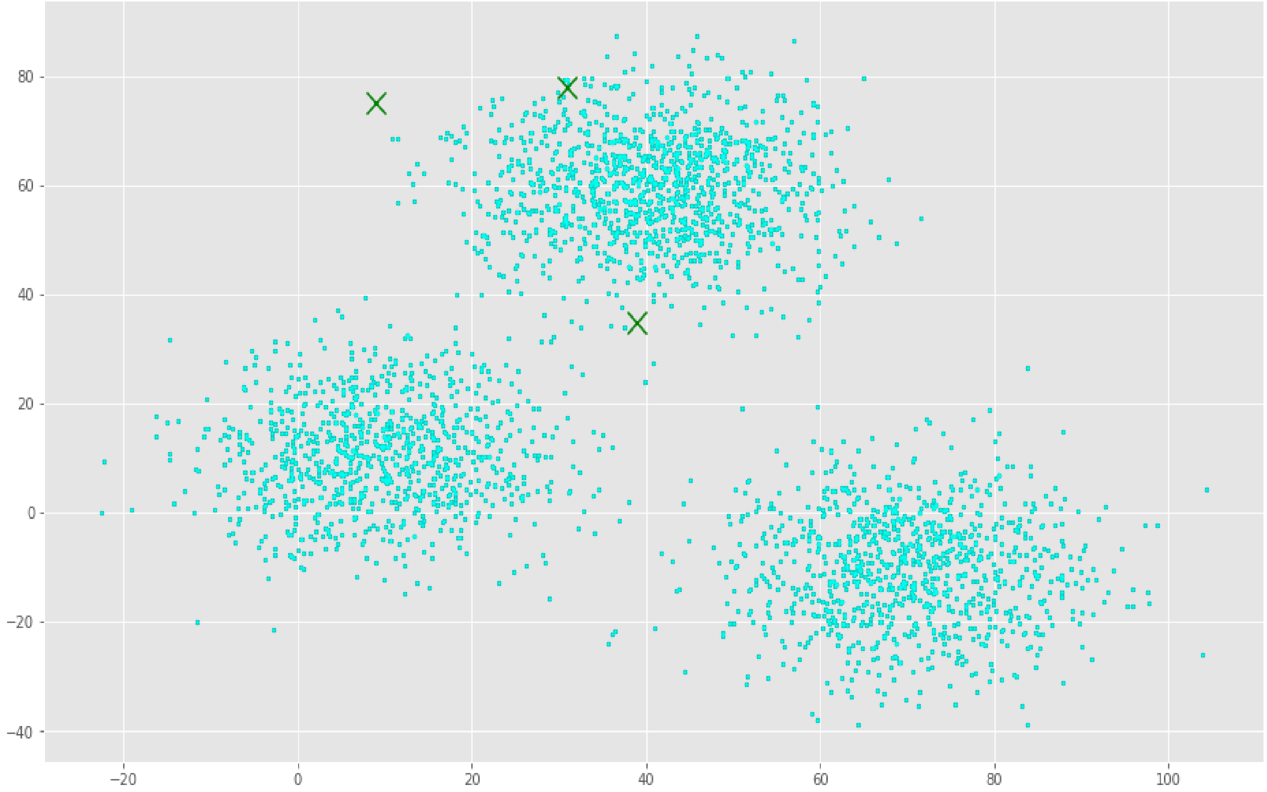
```
In [5]: runfile('F:/WINTER SEM/Data Mining/kmeans/kmeans.py', wdir='F:/WINTER SEM/
Data Mining/kmeans')
Input Data and Shape
(3000, 2)
Initial Centroids
[[39. 35.]
 [31. 78.]
 [ 9. 75.]]
```

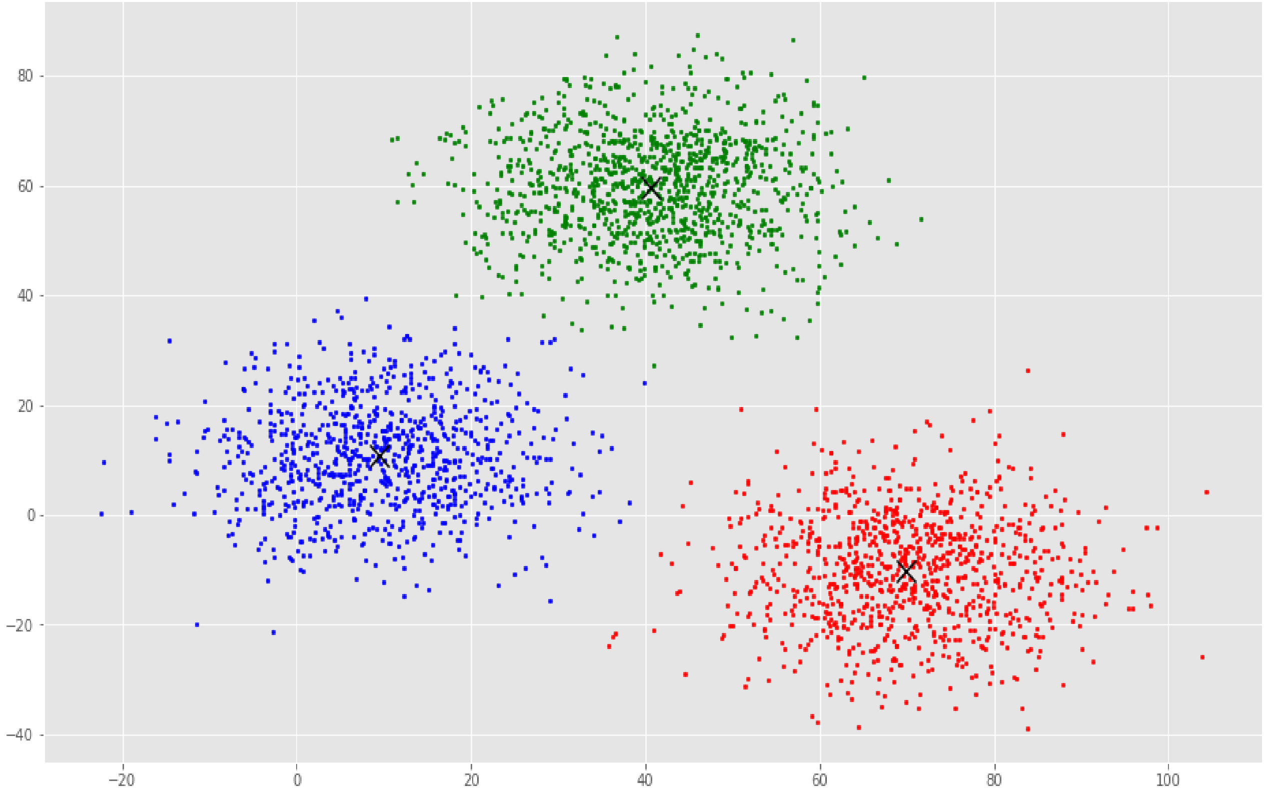
Number Of Clusters: 2



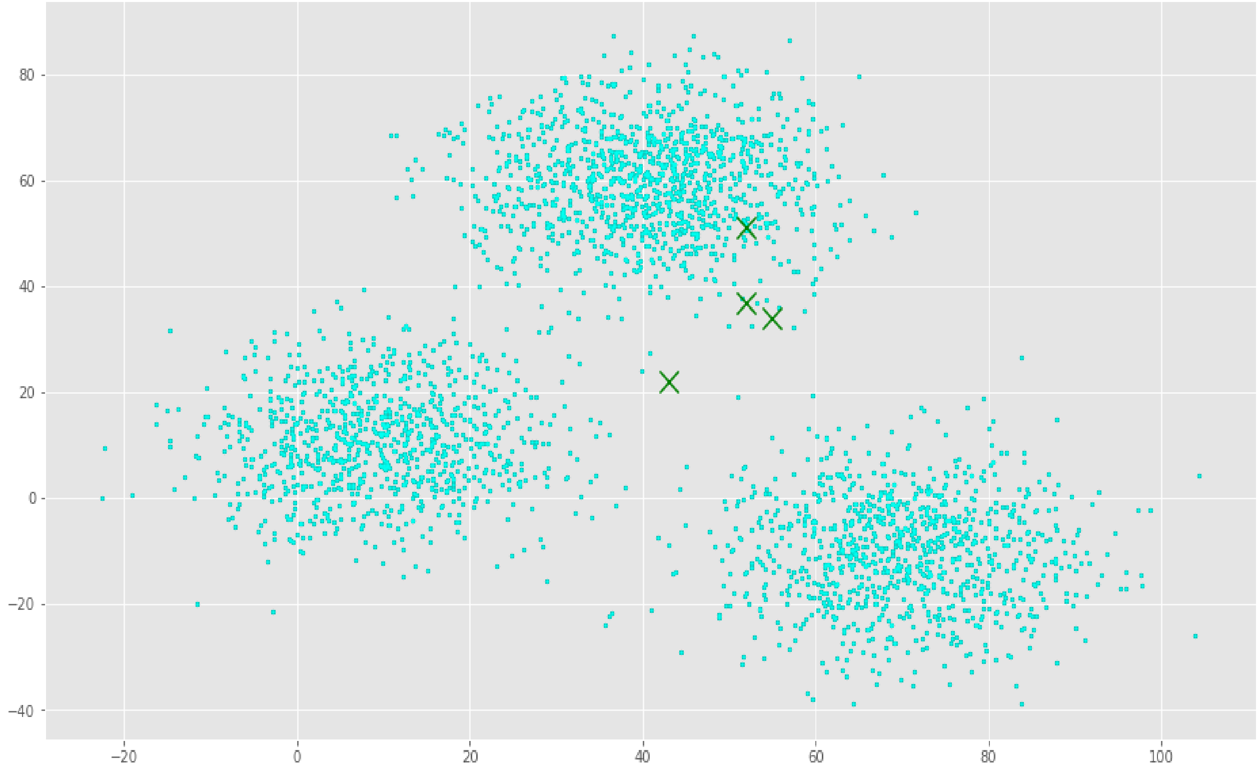


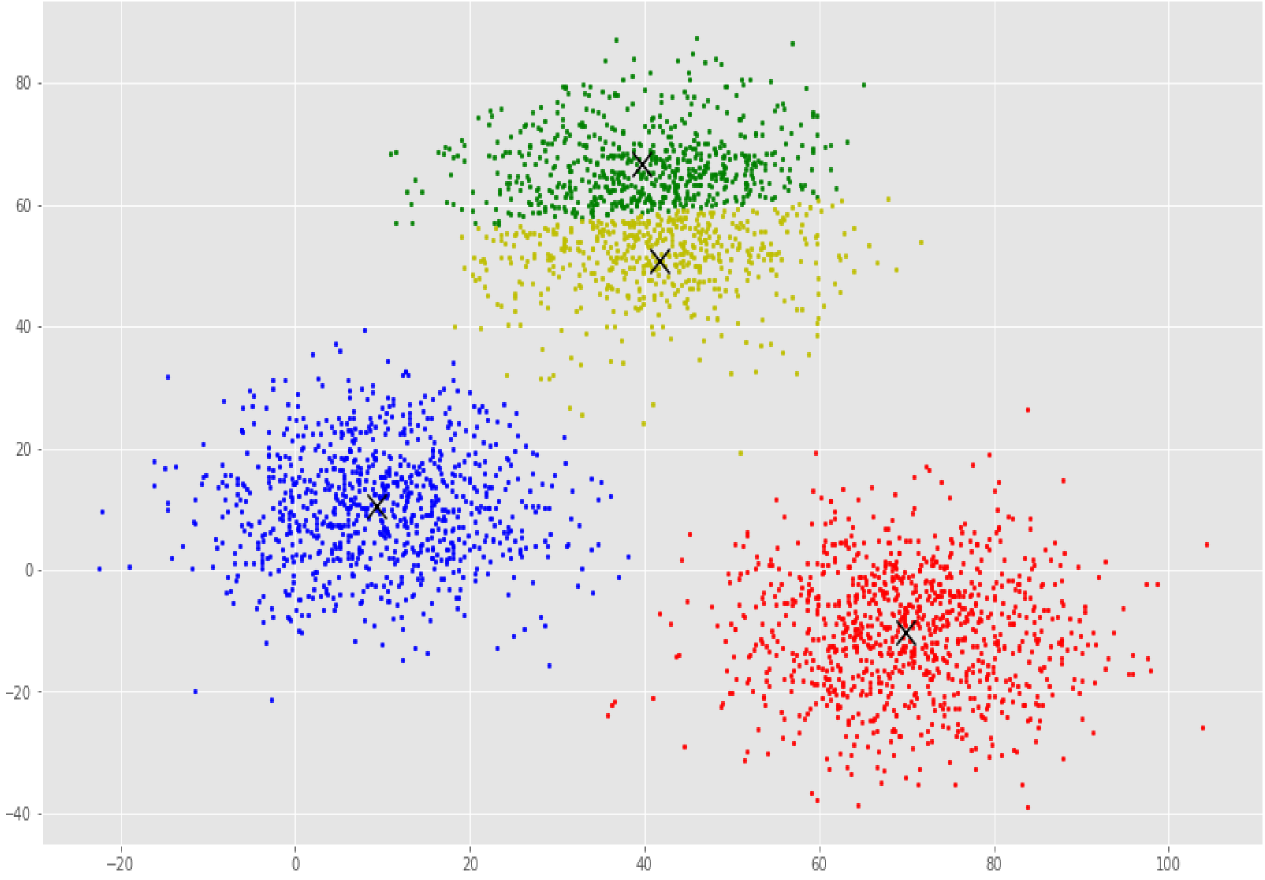
Number Of Clusters: 3



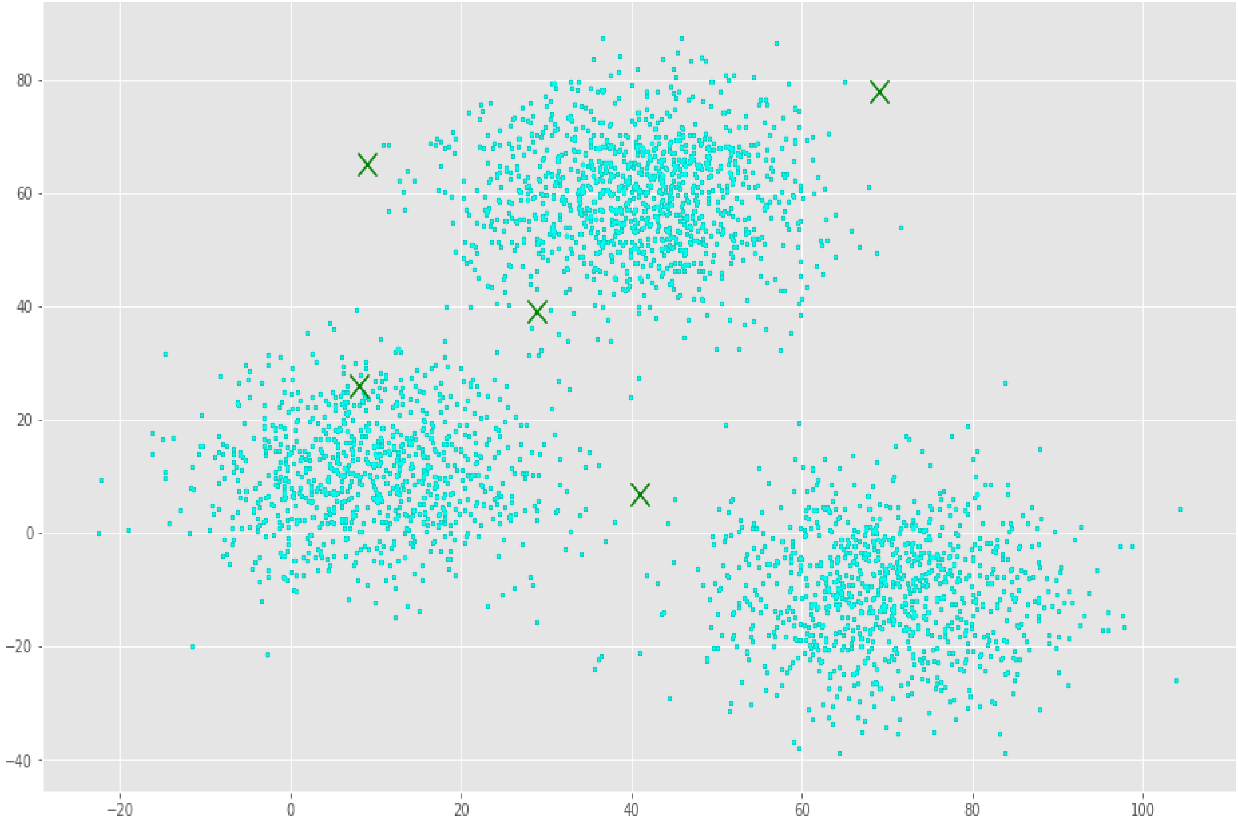


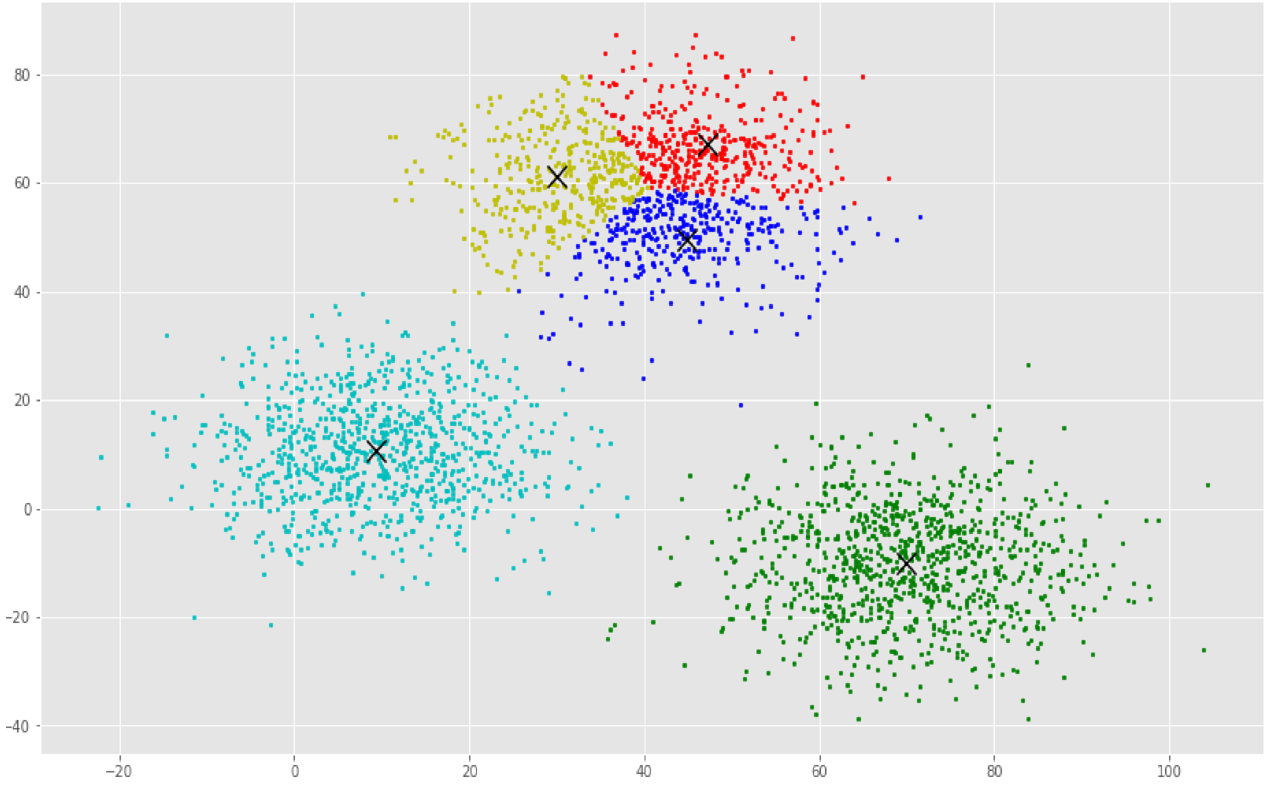
Number Of Clusters: 4



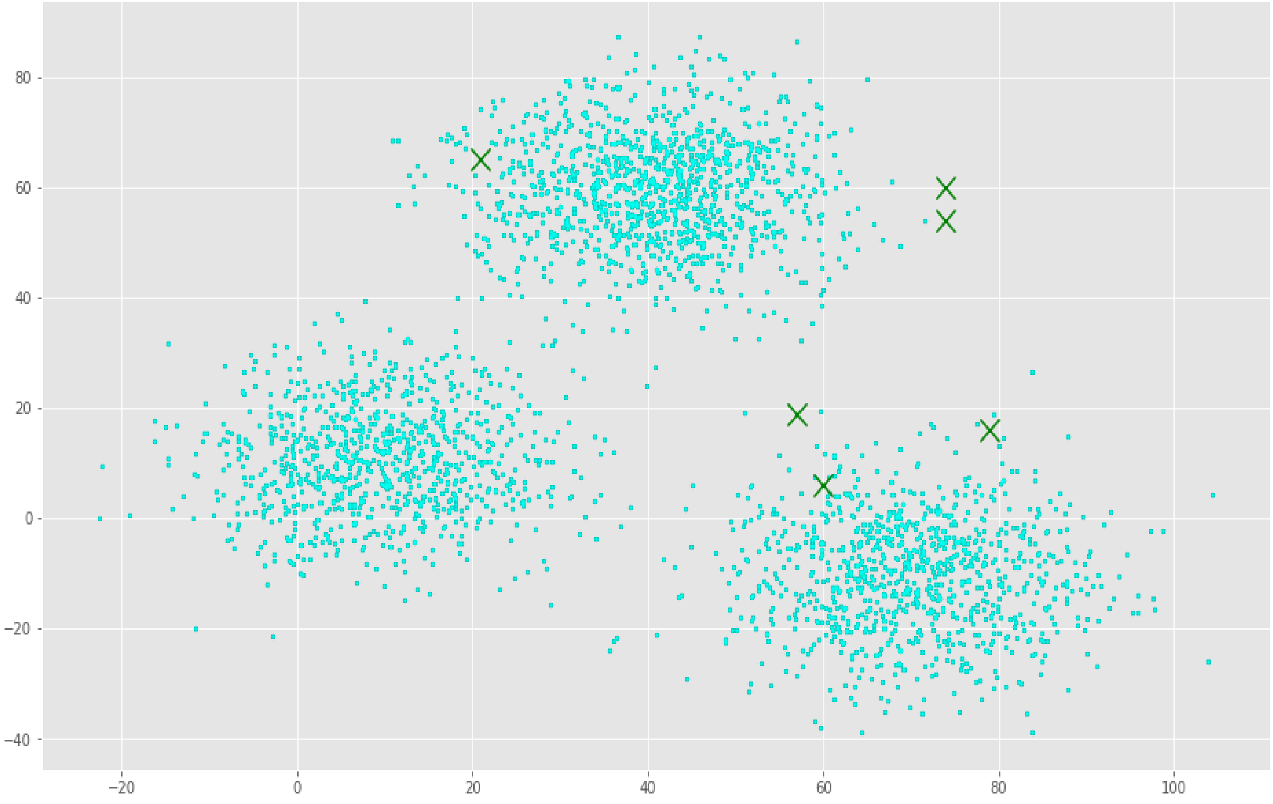


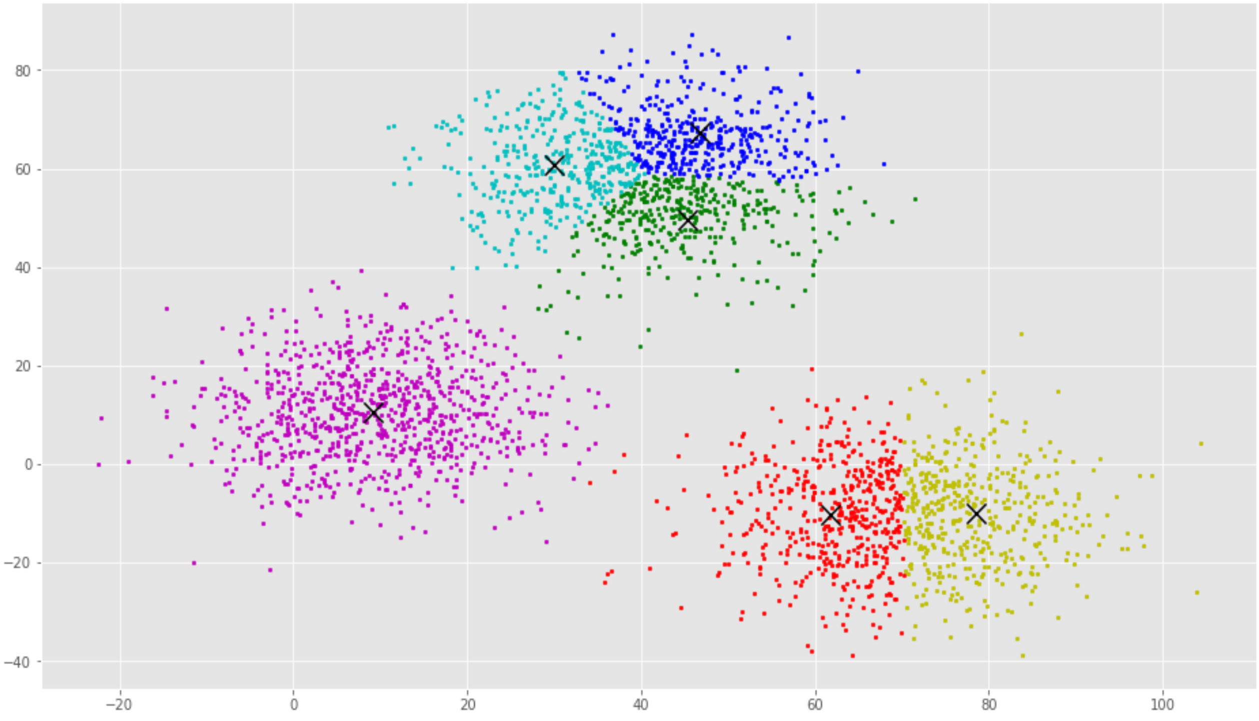
Number Of Clusters: 5





Number Of Clusters: 6





Inference:

Since the data set contains mainly 3 clusters, our output graphs also gives optimum number of clusters as 3 and 4.