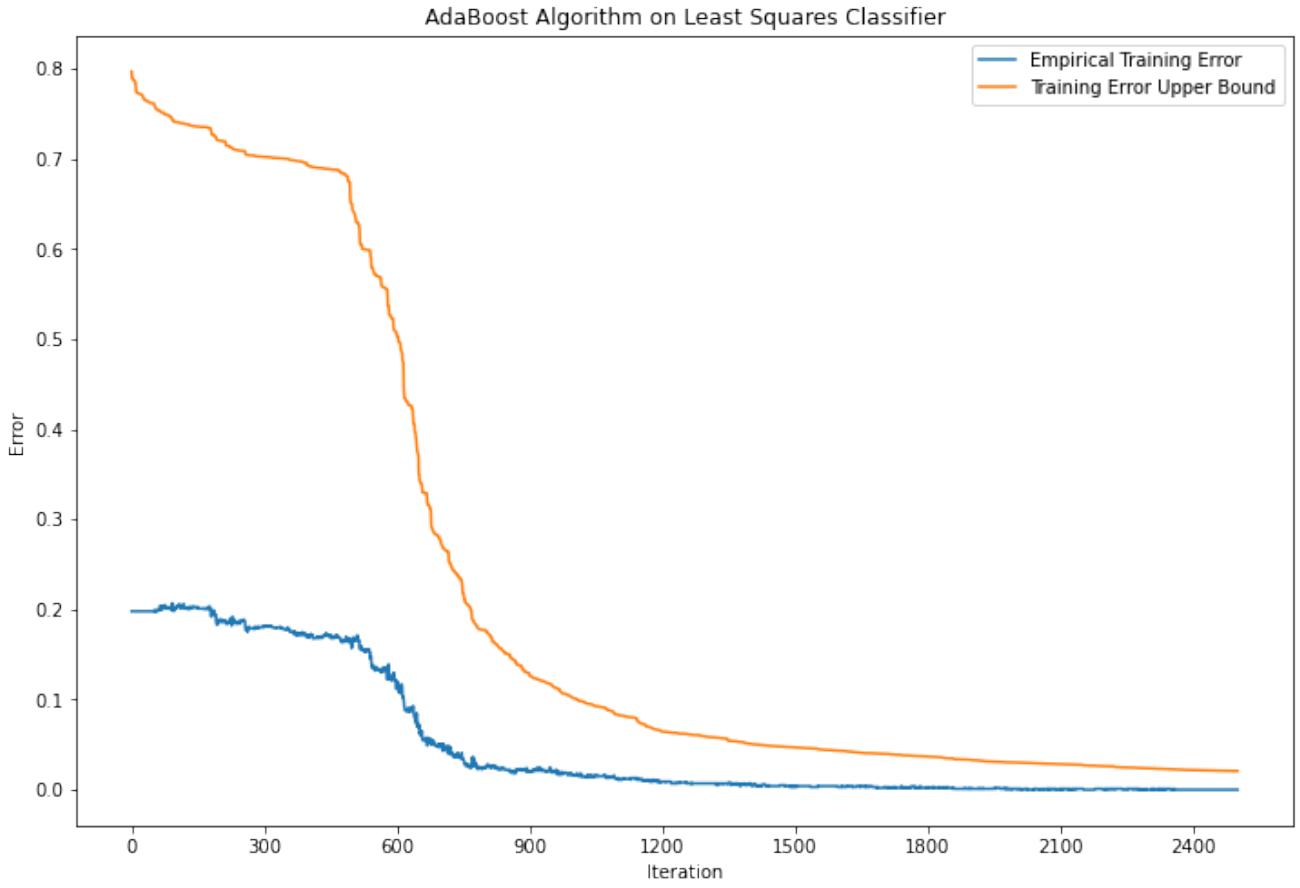
Question 1

Question 1(a)

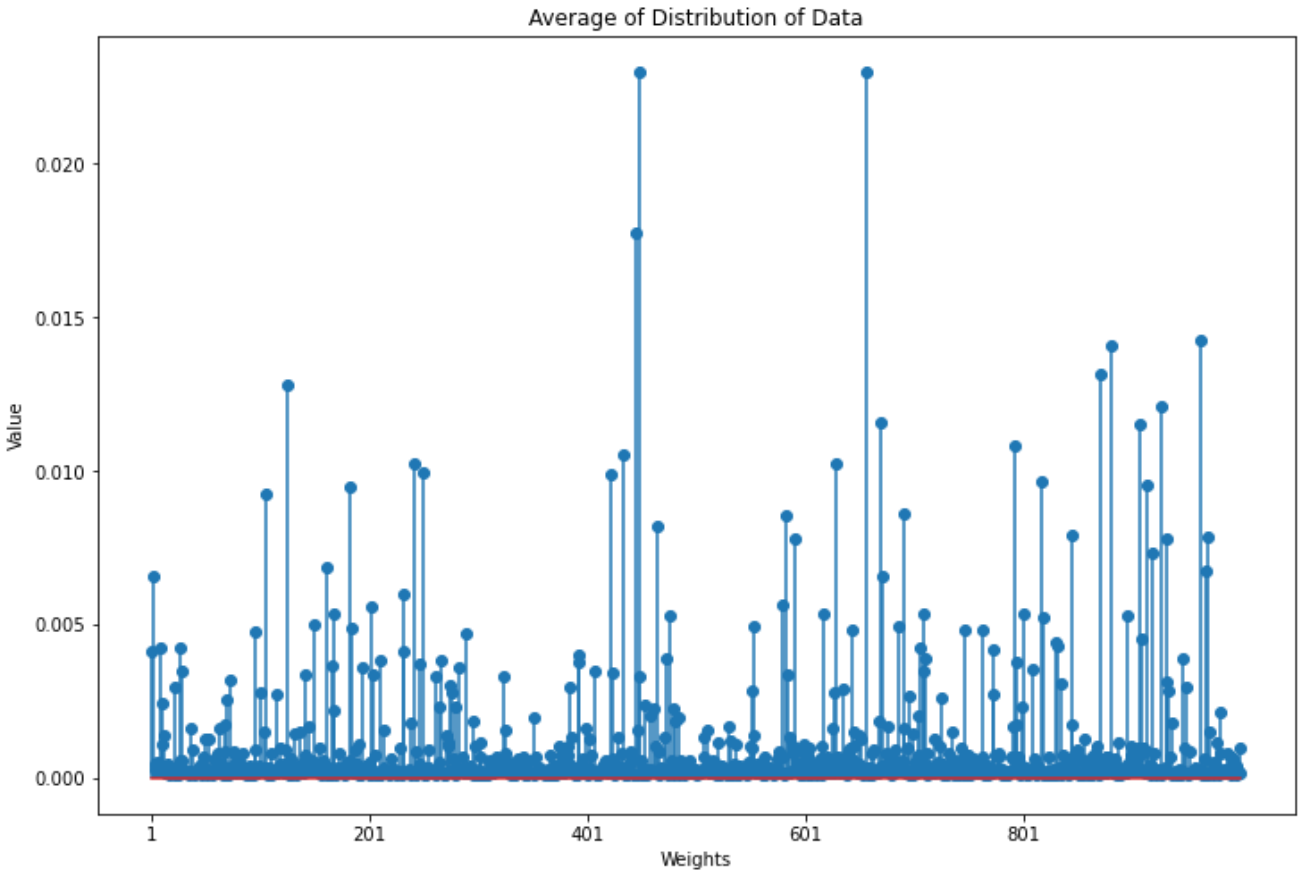
In [8]:
```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('AdaBoost Algorithm on Least Squares Classifier')
plt.xlabel('Iteration')
plt.ylabel('Error')
plt.plot(error_1, label = 'Empirical Training Error')
plt.plot(bound_1, label = 'Training Error Upper Bound')
plt.legend()
plt.show()
```



It can be inferred from the graph that Error becomes almost flat towards the end of graph and comes very close to 0 thus nearing convergence.

Question 1(b)

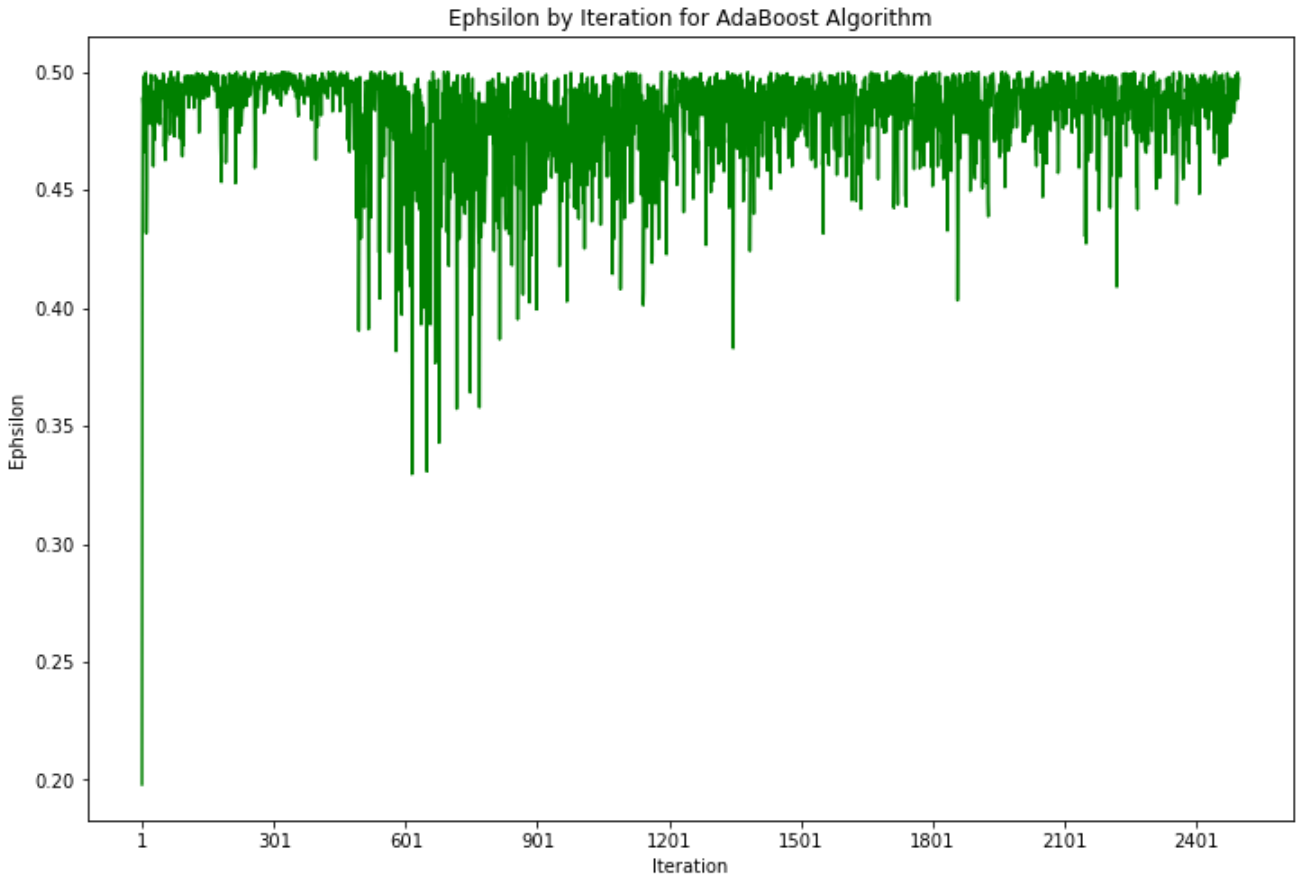
In [11]:
```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('Average of Distribution of Data')
plt.xlabel('Weights')
plt.ylabel('Value')
plt.stem(numbers, weights)
plt.xticks(np.arange(1, 1001, 200))
plt.show()
```



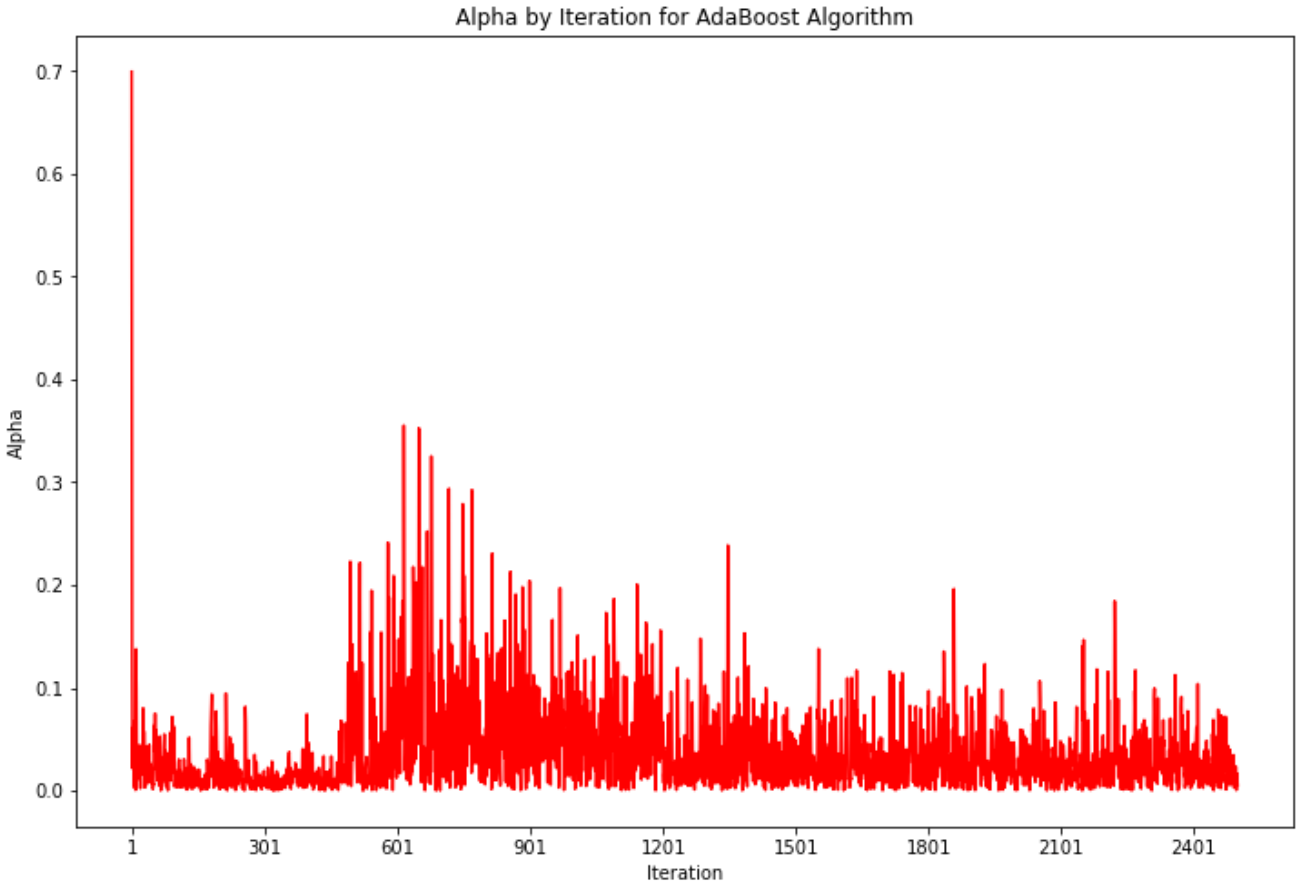
It can be seen from the graph that the Top 2 weights are at the indices 656 and 447. The points at these indices contribute most to the model.

Question 1(c)

In [12]:
```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('Ephsilon by Iteration for AdaBoost Algorithm')
plt.xlabel('Iteration')
plt.ylabel('Ephsilon')
plt.plot(ephsilon_1, label = 'Ephsilon', color = 'green')
plt.xticks(np.arange(1, 2501, 300))
plt.show()
```



In [13]:
```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('Alpha by Iteration for AdaBoost Algorithm')
plt.xlabel('Iteration')
plt.ylabel('Alpha')
plt.plot(alpha_1, label = 'Ephsilon', color = 'red')
plt.xticks(np.arange(1, 2501, 300))
plt.show()
```



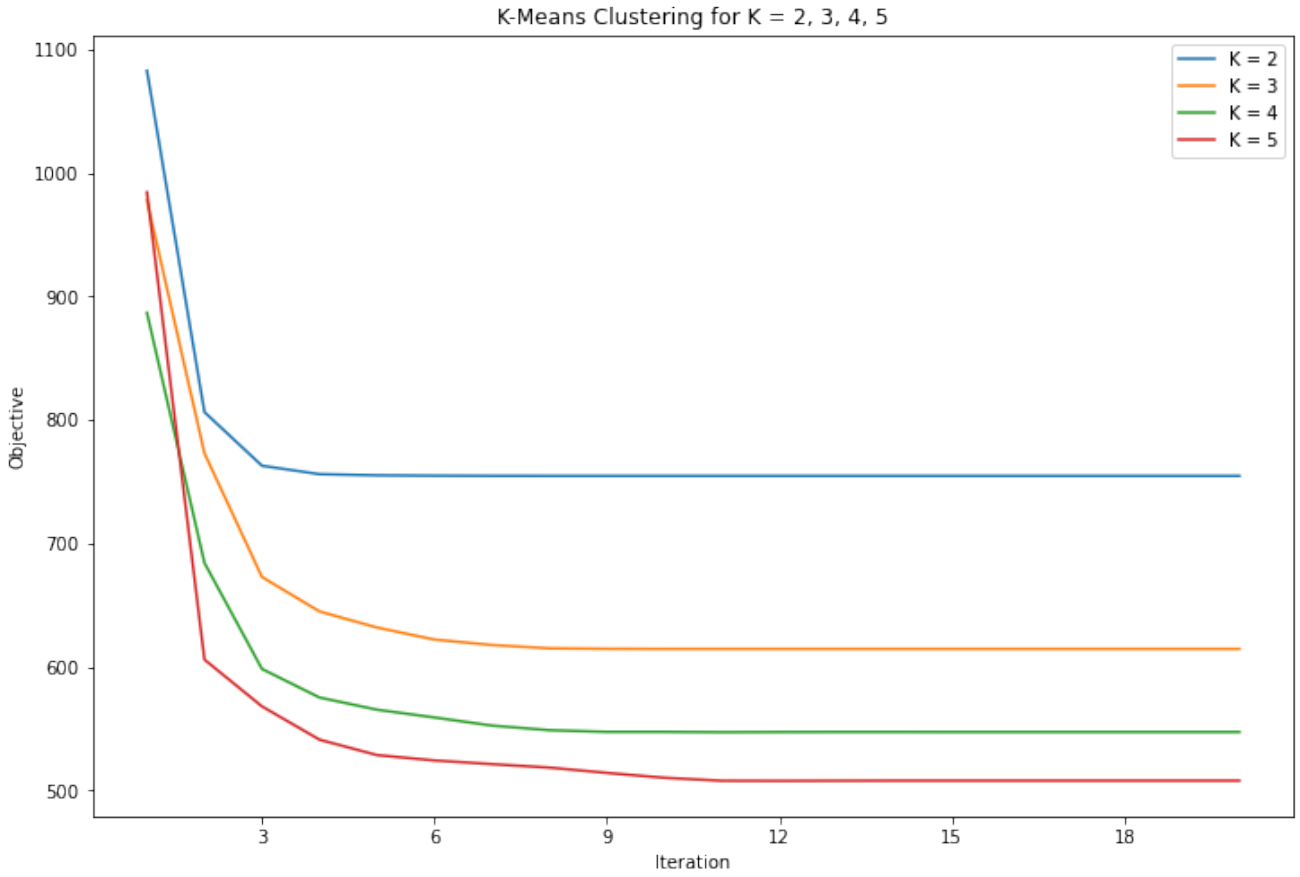
It can be inferred from the graph that Ephsilon and Alpha behave almost inversely.

Question 2

Question 2(a)

In [20]:

```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('K-Means Clustering for K = 2, 3, 4, 5')
plt.xlabel('Iteration')
plt.ylabel('Objective')
plt.plot(df2['K = 2'], label = 'K = 2')
plt.plot(df2['K = 3'], label = 'K = 3')
plt.plot(df2['K = 4'], label = 'K = 4')
plt.plot(df2['K = 5'], label = 'K = 5')
plt.legend()
plt.show()
```

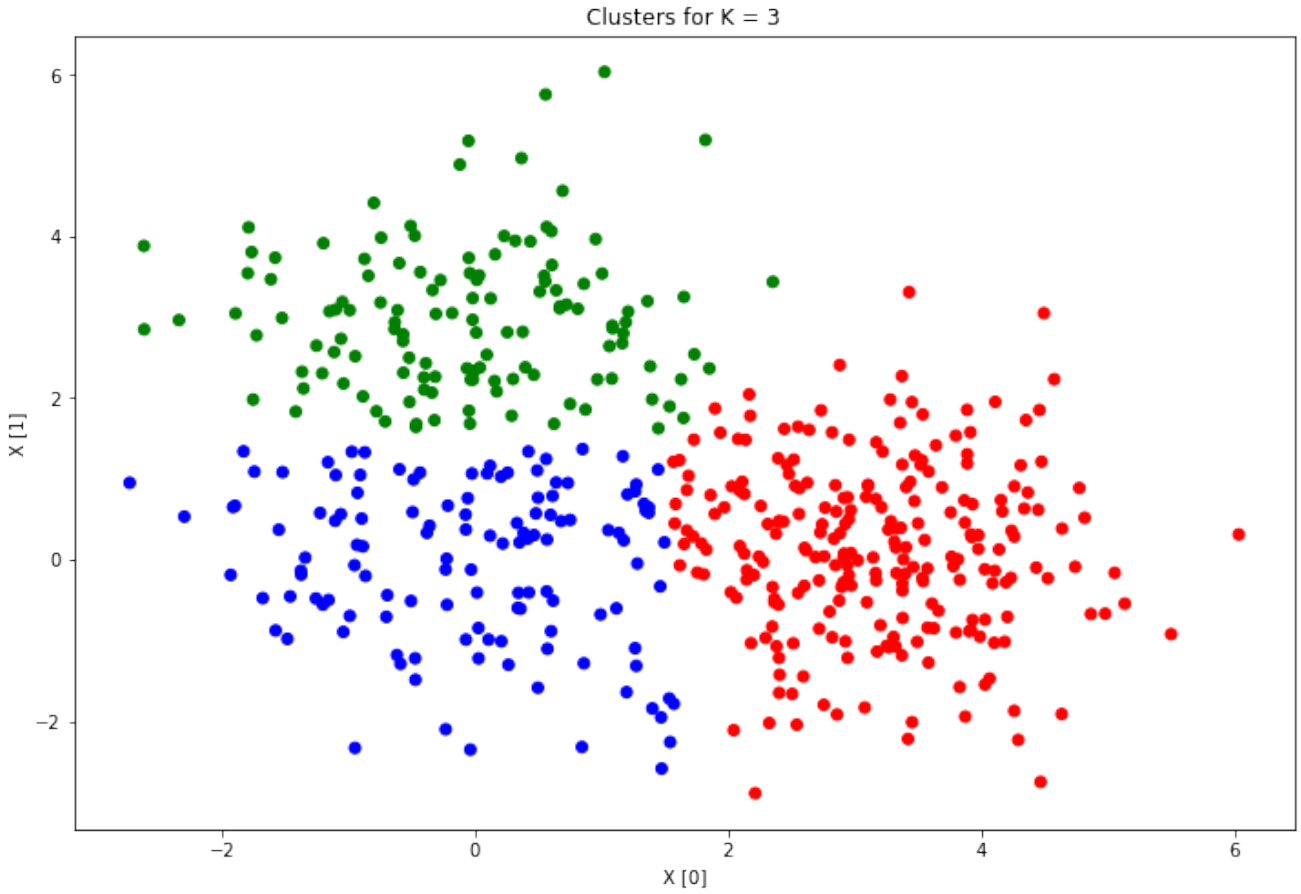


It can be inferred from the graph that the objective function decreases as the number of clusters increase from 2 to 5.

Question 2(b)

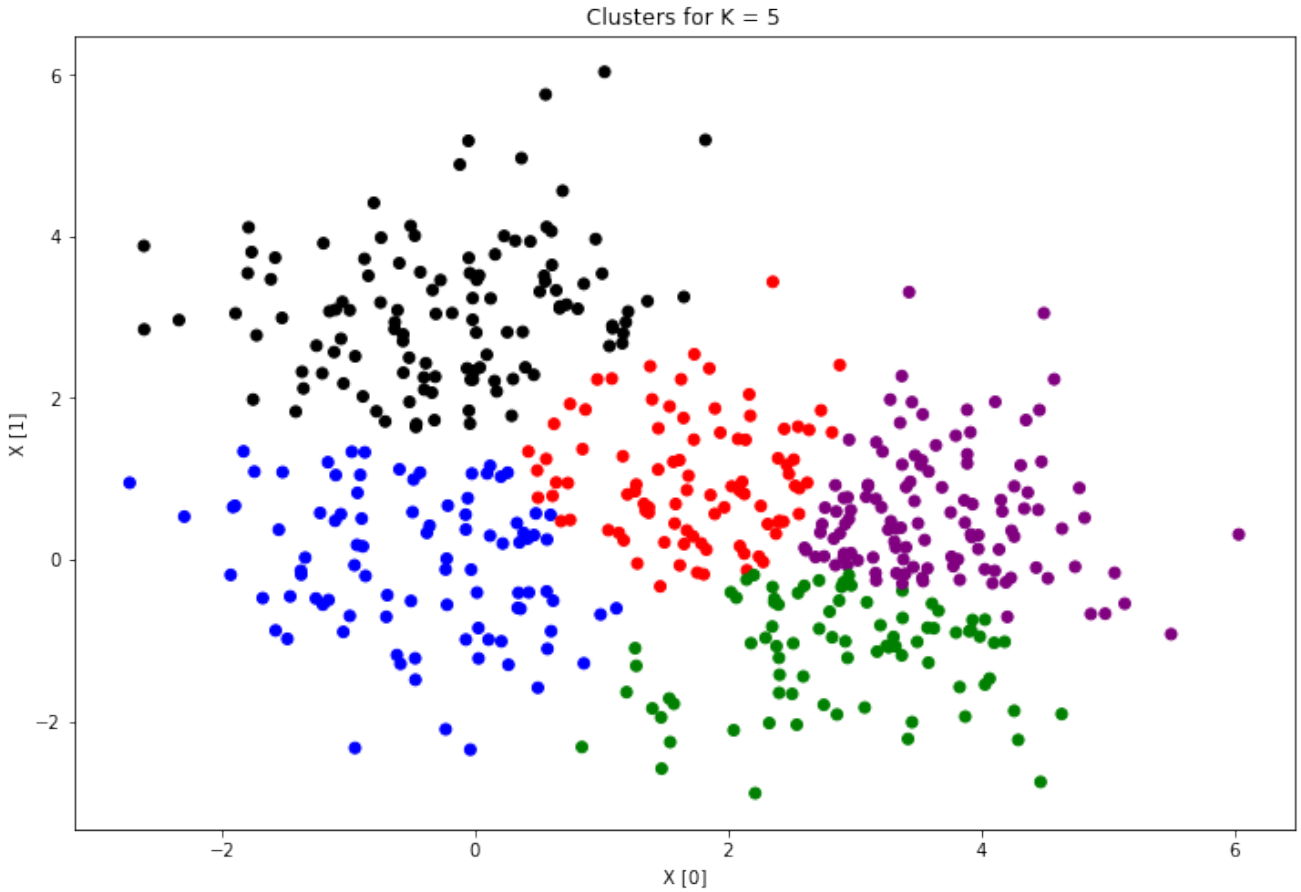
In [22]:

```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
plt.title('Clusters for K = 3')
plt.xlabel('X [0]')
plt.ylabel('X [1]')
colors = ['red', 'green', 'blue']
plt.scatter(dist[:,0].flatten(), dist[:,1].flatten(), c = c3.flatten(), cmap
plt.show()
```



In [23]:

```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
plt.title('Clusters for K = 5')
plt.xlabel('X [0]')
plt.ylabel('X [1]')
colors = ['red', 'green', 'blue', 'purple', 'black']
plt.scatter(dist[:,0].flatten(), dist[:,1].flatten(), c = c5.flatten(), cmap =
plt.show()
```



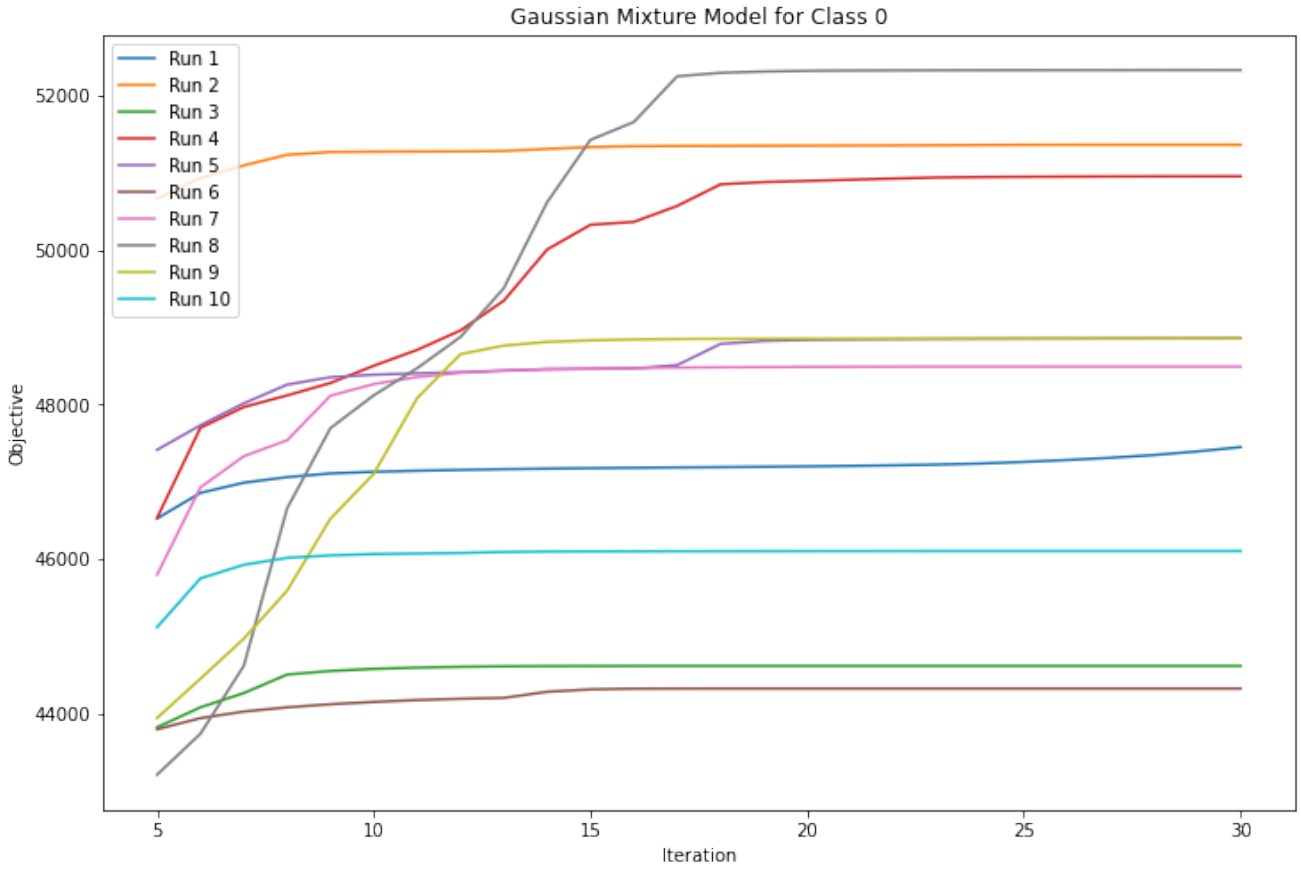
The graphs show distinct clusters being formed in both cases and it can be said that the case of K = 5 further subdivides the clusters from K = 3.

Question 3

Question 3(a)

In [27]:

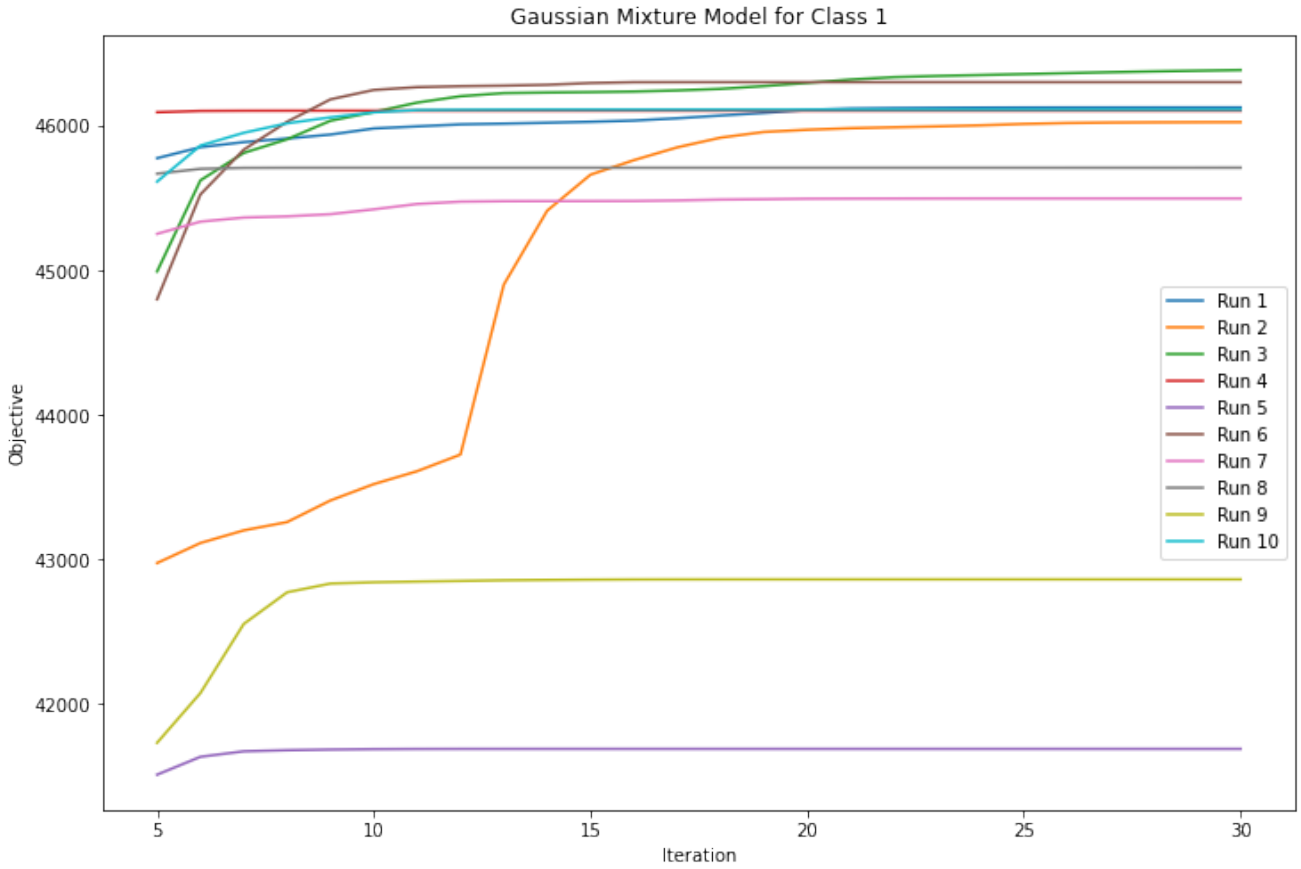
```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('Gaussian Mixture Model for Class 0')
plt.xlabel('Iteration')
plt.ylabel('Objective')
plt.plot(df3['Run 1'], label = 'Run 1')
plt.plot(df3['Run 2'], label = 'Run 2')
plt.plot(df3['Run 3'], label = 'Run 3')
plt.plot(df3['Run 4'], label = 'Run 4')
plt.plot(df3['Run 5'], label = 'Run 5')
plt.plot(df3['Run 6'], label = 'Run 6')
plt.plot(df3['Run 7'], label = 'Run 7')
plt.plot(df3['Run 8'], label = 'Run 8')
plt.plot(df3['Run 9'], label = 'Run 9')
plt.plot(df3['Run 10'], label = 'Run 10')
plt.xticks(np.arange(5, 31, 5))
plt.legend()
plt.show()
```



Highest Objective for Class 0 is observed for Run Number 8.

In [29]:

```
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(12, 8)
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.title('Gaussian Mixture Model for Class 1')
plt.xlabel('Iteration')
plt.ylabel('Objective')
plt.plot(df4['Run 1'], label = 'Run 1')
plt.plot(df4['Run 2'], label = 'Run 2')
plt.plot(df4['Run 3'], label = 'Run 3')
plt.plot(df4['Run 4'], label = 'Run 4')
plt.plot(df4['Run 5'], label = 'Run 5')
plt.plot(df4['Run 6'], label = 'Run 6')
plt.plot(df4['Run 7'], label = 'Run 7')
plt.plot(df4['Run 8'], label = 'Run 8')
plt.plot(df4['Run 9'], label = 'Run 9')
plt.plot(df4['Run 10'], label = 'Run 10')
plt.xticks(np.arange(5, 31, 5))
plt.legend()
plt.show()
```



Highest Objective for Class 1 is observed for Run Number 3.

Question 3(b)

In [25]:
```
print("\033[1m" + '1-Gaussian Mixture Model' + "\033[0m")
print('Prediction Accuracy = ' + str(accuracy_1))
print(confusion_matrix_1.to_markdown() + '\n')
print("\033[1m" + '2-Gaussian Mixture Model' + "\033[0m")
print('Prediction Accuracy = ' + str(accuracy_2))
print(confusion_matrix_2.to_markdown() + '\n')
print("\033[1m" + '3-Gaussian Mixture Model' + "\033[0m")
print('Prediction Accuracy = ' + str(accuracy_3))
print(confusion_matrix_3.to_markdown() + '\n')
print("\033[1m" + '4-Gaussian Mixture Model' + "\033[0m")
print('Prediction Accuracy = ' + str(accuracy_4))
print(confusion_matrix_4.to_markdown() + '\n')
```

```
1-Gaussian Mixture Model
Prediction Accuracy = 77.39130434782608
|   Actual |   0 |   1 |
|---------:|----:|----:|
|        0 | 180 |  98 |
|        1 |   6 | 176 |

2-Gaussian Mixture Model
Prediction Accuracy = 80.43478260869566
|   Actual |   0 |   1 |
|---------:|----:|----:|
|        0 | 198 |  80 |
|        1 |  10 | 172 |

3-Gaussian Mixture Model
Prediction Accuracy = 84.34782608695653
|   Actual |   0 |   1 |
|---------:|----:|----:|
|        0 | 216 |  62 |
|        1 |  10 | 172 |

4-Gaussian Mixture Model
Prediction Accuracy = 82.3913043478261
|   Actual |   0 |   1 |
|---------:|----:|----:|
|        0 | 209 |  69 |
|        1 |  12 | 170 |
```

It can be inferred that the Accuracy increases as k goes from 1 to 3 but then decreases as k goes to 4. Thus k = 3 looks to be ideal for GMM.