Big Data Homework 3

2.3 b)

KMeans – All Features (Purity = 0.51708)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Case (1)** | **Control (2)** | **Unknown (3)** |
| Cluster 0 | 14.54918033 | 48.62869198 | 18.02721088 |
| Cluster 1 | 85.45081967 | 50.84388186 | 78.62811791 |
| Cluster 2 | 0 | 0.52742616 | 3.344671202 |
|  | 100 | 100 | 100 |

KMeans – Filtered Features (Purity – 0.89134)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Case (1)** | **Control (2)** | **Uknown (3)** |
| Cluster 0 | 1.536885246 | 100 | 1.230769231 |
| Cluster 1 | 98.46311475 | 0 | 29.53846154 |
| Cluster 2 | 0 | 0 | 69.23076923 |
|  | 100 | 100 | 100 |

2.4 b)

GMM – All Features (Purity = 0.50651)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Case (1)** | **Control (2)** | **Unknown (3)** |
| Cluster 0 | 19.15983607 | 43.88185654 | 17.68707483 |
| Cluster 1 | 75.30737705 | 43.24894515 | 65.53287982 |
| Cluster 2 | 5.532786885 | 12.86919831 | 16.78004535 |
|  | 100 | 100 | 100 |

GMM – Filtered Features (Purity = 0.65574)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Case (1)** | **Control (2)** | **Unknown (3)** |
| Cluster 0 | 80.22540984 | 0 | 26.56410256 |
| Cluster 1 | 0.204918033 | 97.78481013 | 57.23076923 |
| Cluster 2 | 19.56967213 | 2.215189873 | 16.20512821 |
|  | 100 | 100 | 100 |

2.5 a)

In the streaming k-means setting, the data comes in batches and in real time. The cluster centers take into account this new batch and readjust accordingly. Exactly by how much they adjust is determined by the half-life. A large half-life means centers adjust slowly whereas a small half life means that they adjust very rapidly (i.e. are forgetful)

The update rule is given by:

Where C\_t is the cluster center before the new batch, N\_t is the number of points in that cluster, C’\_t is the center of the new batch, N’\_t is the number of points in the new batch, alpha is the weight constant (can be interpreted as the half life described above).

Here is a quick high level step-by-step of the algorithm:

1. Assign the new data points to its nearest cluster
2. Update the weight alpha by time unit
3. Update the clusters
4. If a cluster is no longer needed, we separate the largest cluster into two separate clusters.

The pros of streaming kmeans are that: It can use live streaming data and capture dynamic changes in the data sources as time passes. I.e. if the distribution of one of the clusters changes after some time, streaming kmeans will do a better job of adapting to this new change by focusing on this newer distribution thanks to its ability to be “forgetful”.

Cons of streaming kmeans: Since the algorithm doesn’t see all the data at once like typical kmeans, the chance of error is greater. For example, maybe towards the very end, our data source seems to have some noise. Streaming kmeans will fit that noise more aggressively than traditional kmeans since it’s weighted to emphasize newer data points.

Forgetfulness in general gives us the ability to assign a weight to how much we want to weigh the old data compared to the new, incoming data. For example, if a data source is changing over time, we want streaming kmeans to be forgetful so it will effectively model the newer data.

2.5c)