Alex Reibman
CS378

a. **Describe the datasets and your quality metrics.**

I used two datasets—iris.data and nutrients.txt. The iris dataset is found on the UCI machine learning database website. It details the characteristics of 150 species of iris flowers. The other dataset I used, nutrients.txt lists the nutrient levels for 27 varieties of food.

The quality metric I used was the Sum of Squared Errors (SSE). SSE measures size of the squared difference between the distance of a data point to its cluster and the distance between an average data point to its cluster. Ideally, each data point will have an equal distance from the centroid as the average distance from each centroid. Hence, smaller a smaller SSE will mean more reliable clustering.

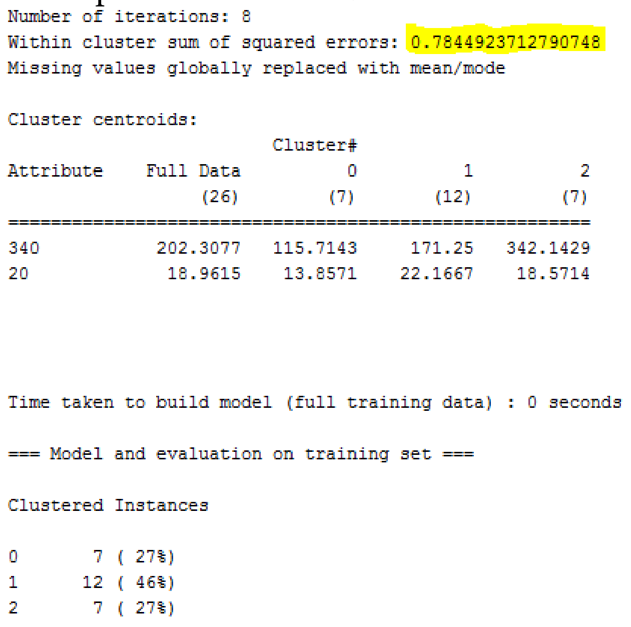
b. **Describe your experiment setup such as how you preprocessed the data (if any), how you chose the parameters for the selected algorithms (if any), and why.**

Iris.data was actually converted from an .arff file found in the weka library called Iris2d.arff. To preprocess this, all I needed to do was remove the comments and attribute tags found at the top of the file. Otherwise, iris.data required no preprocessing. With nutrients.txt, I had to manually format some of the data. I deleted the comments at the top, but the bulk of the preprocessing was focused around turning the file into a readable comma delimited database. The main problem was that the file was created as tab delimited but saved as space delimited. Hence, I had to do some find and replace operations to turn the extra spaces into commas.

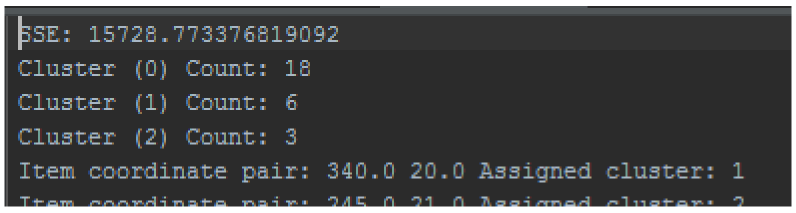
For this clustering experiment, I chose to only find clusters for the first two columns of attributes that give numerical data. For instance, for iris.data, I use the sepal length and width. The reason I chose these attributes is because the iris dataset is quite popular, so I thought it would be a good test dataset. Given there are 3 species of iris flowers, we can test the reliability of the algorithm to see if it is able to evenly divide the dataset into 3 clusters. For nutrients.txt, I use Energy in calories and Protein in grams. I chose energy and protein because it might be interesting to see which clusters can be identified as "superfoods" (that is, foods with low calories and high amounts of protein) or junk food (high calories low protein).

c. **Present the experiment results. They should not be a simple copy-and-paste from Weka output, but rather presented in a tabular or chart format for easy comparison.**

The experiment results for nutrients.txt were as follows:



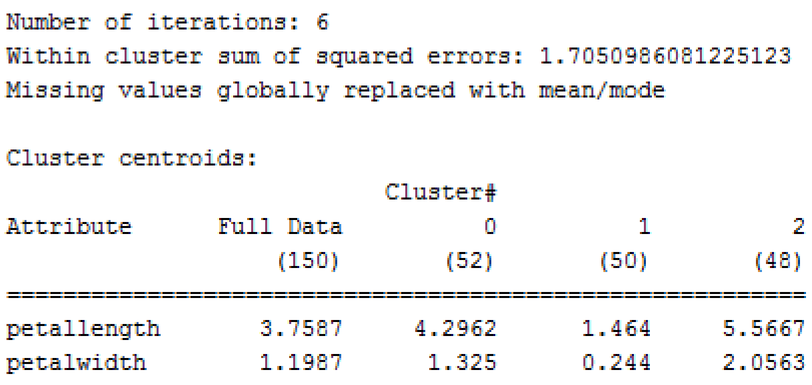
```
Number of iterations: 8
Within cluster sum of squared errors: 0.7844923712790748
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute     Full Data         0          1          2
                   (26)       (7)       (12)        (7)
=========================================================
340            202.3077  115.7143     171.25   342.1429
20              18.9615   13.8571    22.1667    18.5714




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       7 ( 27%)
1      12 ( 46%)
2       7 ( 27%)
```



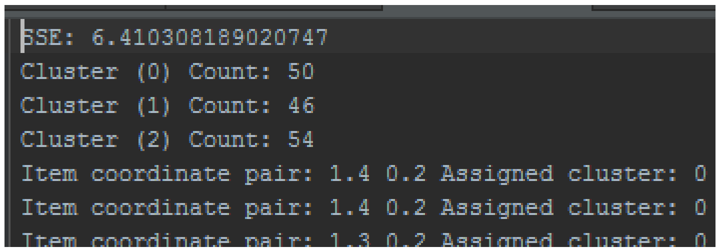
```
SSE: 15728.773376819092
Cluster (0) Count: 18
Cluster (1) Count: 6
Cluster (2) Count: 3
Item coordinate pair: 340.0 20.0 Assigned cluster: 1
Item coordinate pair: 345.0 21.0 Assigned cluster: 2
```

| Algorithm | Weka | Assignment |
|---|---|---|
| SSE | 0.7844 | 15728.773 |
| Cluster 0 | 7 | 18 |
| Cluster 1 | 12 | 6 |
| Cluster 2 | 7 | 3 |

The experiment results for Iris.data were as follows:



```
Number of iterations: 6
Within cluster sum of squared errors: 1.7050986081225123
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute       Full Data         0          1          2
                    (150)      (52)       (50)       (48)
=========================================================
petallength        3.7587    4.2962      1.464     5.5667
petalwidth         1.1987     1.325      0.244     2.0563
```



```
SSE: 6.410308189020747
Cluster (0) Count: 50
Cluster (1) Count: 46
Cluster (2) Count: 54
Item coordinate pair: 1.4 0.2 Assigned cluster: 0
Item coordinate pair: 1.4 0.2 Assigned cluster: 0
Item coordinate pair: 1.3 0.2 Assigned cluster: 0
```

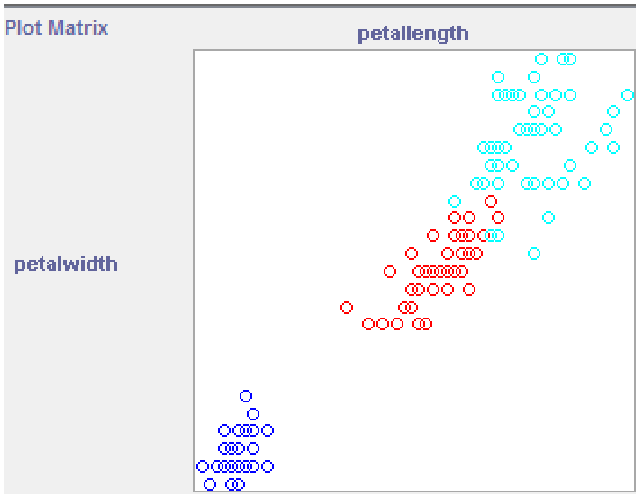
| Algorithm | Weka | Assignment |
|---|---|---|
| SSE | 1.705 | 6.410 |
| Cluster 0 | 52 | 50 |
| Cluster 1 | 50 | 46 |
| Cluster 2 | 48 | 54 |

**d. Discuss the insights and conclusions from your experiments. For example, do different clustering methods make a difference in terms of quality or performance for the particular datasets you selected? And why? How data preprocessing might help? Are there conditions or general types of datasets that will make certain algorithms more suitable than others?**

Iris.data revealed some interesting insights, but nutrients.txt did not. For instance, with iris.data, there was a near perfect three way split between the clusters which is indicative of the fact that there are 50 of each species of iris. This means that the algorithm was efficient at identifying the algorithm. Additionally, the SSE between Weka and the assignment's algorithm was pretty close.

Nutrients.txt however was less reliable. This is primarily because the clusters in nutrients.txt are fairly scattered, so clustering does not give much interesting unknown information. Additionally, the SSE is much lower for Weka. I suspect that this is because Weka's SSE measure is indexed, so the assignment's measure of SSE is substantially higher because I did not index the attributes.

Using Weka's alternative clustering methods reveal that kMeans is actually quite effective at identifying clusters. With Iris.data, kMeans evenly distributes the clusters between each of the 3 iris varieties, but hierarchical clustering and EM do not do a good job. XMeans is actually pretty effective at showing that there are 2 large clusters in the set. Petal width by length normally shows 2 large clusters which means two iris varieties are very similar to one another.



As a general procedure, preprocessing is an effective way to ensure clustering will be effective. For instance, removing outliers can ensure more effective clustering methods. Additionally, preprocessing can indicate what sort of clustering method will be effective. Supposing we know the number of clusters but not the data points that fall into them, algorithms kMeans is effective at placing points into clusters. Alternatively, if we are interested in finding how to cluster certain data points in order to find out what clusters they belong to, xMeans might be more effective. Iris.data is a good dataset because it can be used to serve both of these purposes. For instance, we can use kMeans to classify all of the data points into nearly perfectly even clusters. Additionally, we can use xMeans to show that two of the species of iris plants are actually very similar to one another.