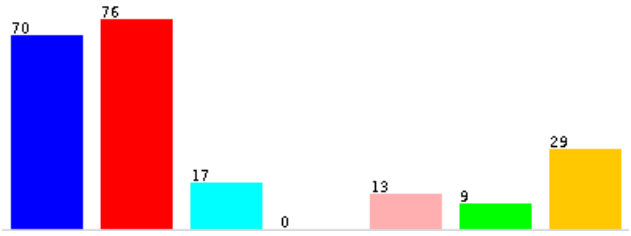
1. For the *glass.arff* file, there is 10 attributes (and the class attribute) and 214 instances within the dataset. The attributes are listed below:

- ID number (ID)
- Refractive index (RI)
- Sodium (Na)
- Magnesium (Mg)
- Aluminium (Al)
- Silicon (Si)
- Potassium (K)
- Calcium (Ca)
- Barium (Ba)
- Iron (Fe)
- Type of glass (Type)

All of these attributes are numeric up to type, which is nominal and its choices are listed below:

- building window float
- building window not float
- vehicle window float
- vehicle window not float
- container
- tableware
- headlamps

All of the chemicals are a percentage from 0-100 in terms of weight in oxide. ID number is an ordered list from 1 to 214. Refractive index is any number.
The class attribute is Type and is distributed as shown in figure 1.



- Blue (70) is building windows with float
- Red (76) building windows with no float
- Light blue (17) vehicle windows with float
- Pink (13) containers
- Green (9) tableware
- Yellow (29) headlamps

Figure 1

Based on the visualization tab, I couldn't notice any direct collerations between attributes. It also has no missing data values and so there is no strategy in place to deal with this.

iris.arff

There is 4 attributes in this dataset, and a class attribute, and it has 150 instances of data within it. The attributes are listed below:

- Sepallength
- Sepalwidth
- Petallength
- Petalwidth
- Class

All of these attributes, bar 'Class', are numerical. 'Class' is nominal and its choices are:

- Iris-setosa
- Iris-versicolor
- Iris-virginica

The numerical attributes are all centimeter measurements.
The class attribute is 'Class' and is distrubuted as shown in figure 2.
This shows the classification is 33.3… reccuring for each of the class types.

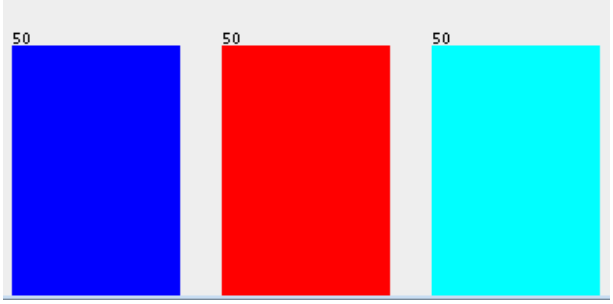


Figure 2

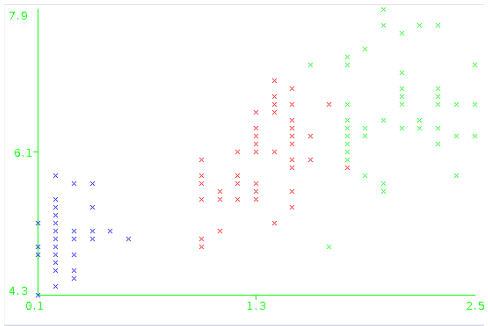


Figure 3

I noticed a correlation between petal width/petal length, sepal length/petal length, and petal width/sepal length . The petal width by sepal length is figure 3. No missing values in this dataset.

2.

(a) Running the nearest neighbour classifier on the *glass.arff* file, with cross validation to 10, and KNN = 1, the accuracy was 70.5607% accurate, classifying 151 instances correct and 63 incorrectly.

(b) With KNN=5, and cross validation still 10, nearest neighbour was 67.757% accurate, with 145 correctly classified instances, and 69 incorrect. This shows that more neighbours results in less accuracy in classification.

(c) Running the nearest neighbour classifier on the *iris.arff* file, with cross validation to 10 and KNN=1, the accuracy was much higher than in the *glass.arff*, with 143 instances being correctly classified, out of 150, 95.333% accurate.
With KNN=5 and everything else the same, the number of correctly classified instances does not change, the accuracy is the same. This means that the 5 nearest neighbours to the *new instance* are all consistent, and so the outcome is also consistent. Some of the absolute percentages change between these but are negligible enough to ignore. This could be due to the fact that there are only 3 classifications for iris, instead of 9 for the glass dataset.
The difference in results from these two datasets when the KNN value is changed is stark. Where the glass dataset loses 6 correctly classified instances, the iris doesn't lose any.

(d) **TP rate** is the rate of true positives (in which the classifier correctly predicted the class attribute, this is the ideal outcome). For *glass.arff* the average TP rate (when KNN=1) was 0.706 (calculated by adding all the TP rates of all the classes and dividing by the number of choices, in this case 7). When the KNN was increased to 5, the TP rate slightly lowered down to 0.678 on average.
The **FP rate** is opposite of TP, thus is the false positives, we want to try to minimise this the best we can – we don't want incorrect classifications. For the glass dataset with KNN=1 the average FP rate is 0.109, which is significantly lower than the TP rate for this classification. For KNN=5 the average was 0.142. This is slightly higher than when it is only checking one nearest neighbour, as is expected, but this is still reasonable.
Precision is calculated by checking how often the classification was actually right in its prediction. It is calculated by dividing the TP rate with the amount of predicted 'yes'. Obviously, we want this as close to 1 as possible. For the glass dataset this when KNN was set to 1 average was 0.709, meaning it was over 2/3 chance of it predicting correctly. When KNN was set to 5, the average precision rate 0.635, meaning it was just under 2/3 correctly predicted.
Recall is how often when it was 'yes', was 'yes' predicted, so the opposite of precision. Again, this should be as high as possible. For KNN=1, it was on average 0.706, the same as the TP rate, meaning that there was no instances that a yes was overlooked. The same goes for KNN=5, the average was 0.678.
F-measure is a measure of the test's accuracy. It takes into account the precision and the recall of the classification with 1 being the best and 0 the worst. For KNN=1, the f-measure is 0.704 which is a relatively solid score. It decreases when KNN is set to 5 however 0.651 as both the precision rate and recall rate are lower respectively.
ROC Curve summarizes the performance of the nearest neighbour classifier over all possible thresholds. This is generated by plotting the TP rate against FP rate and varying the classifiers to find the average area under the curve. 1 being the best, and demonstrating a perfect classifier, and 0 being the opposite. An average of 0.5 would indicate random guessing. For KNN=1, the average ROC area is 0.792, which shows a good classifier. Based on the TP and FP rate of the third class, the *vehicle window with float* class, this is poorly classified and is the reason for such a low average. The area under this is 0.59, just a little above random guessing. The ROC area is much higher for the third class when the KNN is 5, it is now 0.642, showing that as it was able to search more neighbours, it was able to classify this with more accuracy. The overall average for 5 neighbours is 0.853, which is a lot better than when it was looking to its closest neighbour only.
Confusion matrix describes the performance of the nearest neighbour classifier. It is split into 7x7 matrix (for each classification) each with a corresponding number. This number shows the classifications guess at what class it belonged to. It allows you to see how many guesses were incorrect in an easy to read format. For some of the classes, when KNN was changed it improved the number of correct guesses, and for some classes it decreased and spreads the guesses over more classes.

3.
(a)

No. of attributes	**Names of attributes in set**	**Accuracy**
9	{RI, Na. Mg. Al, Si, K, Ca, Ba, Fe}	70.5607
8	{RI, Na, Mg, Al, Si, K, Ca, Ba}	77.1028
7	{RI, Na, Mg, Al, K, Ca, Ba}	77.5701
6	**{RI, Na, Mg, K, Ca, Ba}**	**78.9720**
5	{RI. Na, Mg, K, Ca}	78.0374
4	{RI, Mg, K, Ca}	77.1028
3	{RI, K, Ca}	73.8318
2	{RI, K}	64.9533
1	{K}	49.0654
0	{}	35.514

(b) The best set of attributes is when there are 6, Refractive Index, Sodium, Magnesium, Potassium, Calcium, Barium.

(c) The best accuracy set is not unbiased, as it is calculated using actual test data. For an unbiased estimate on future data, a training set should be used.

(d) A **wrapper method** is one that goes through each subset of data, running the classifier to determine the best attribute set. It is accurate, however is slow and uses a large amount of computation power as the number of attributes increases.
A **filter method** is one that selects attributes regardless of the model. Based on general features of the dataset, like correlation, to try predict the best set of attributes. These methods tend to suppress the least interesting variables and focus on the interesting ones. However they don't take into account the relationship between variables, which could influence the correlation of the class variable.

Backwards elimination is a wrapper method.

Running the CfsSubsetEval on weka, and cross validation to 10, seed is 1, I found that it predicted 6 variables 100% of the time. These being Refractive Index, Magnesium, Aluminium, Potassium, Calcium, and Barium. This is similar to the set I found by backward elimination, with one change. Sodium is in place of Aluminium in this.

4.

(a) The process of **discretization** is putting values into the buckets so that there are a limited number of possible states. These buckets are ordered and values are discrete.

Equal width is an unsupervices binning technique that divides the data into equal size intervals like shown in Figure 4.

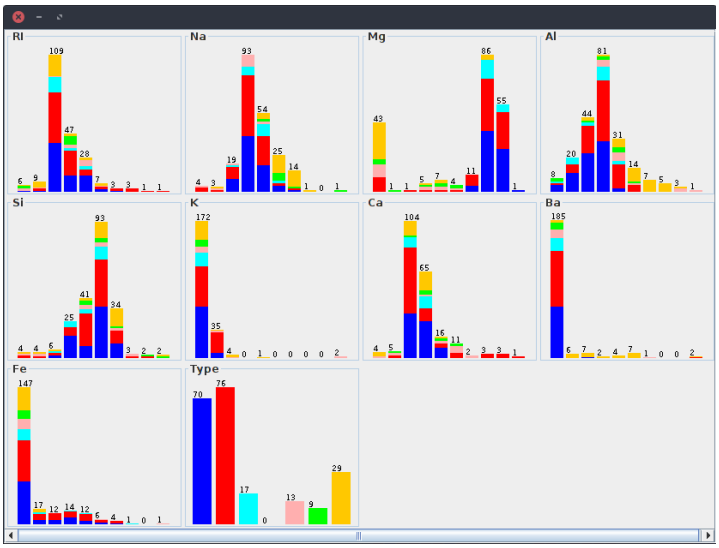


Figure 4

Figure 5

Equal frequency divides the data into groups where each group contains the approximately the same number of values. This is shown in Figure 5.

The difference between these two techniques is that one divides the data into equal size intervals, the other ensures they are all approximately the same value, as is shown in Figure 5. Sometimes it is just not possible to get the data to be the same in terms of class distribution, as you can see in the last histograms of Figure 5.

(b) Applying supervised discretization to the *iris* dataset I can see that petal width and petal length are easily predictable attributes to predict the class attribute from. You can determine which class it belongs to easily, with a low false positive rate.

(c) Applying supervised discretization to the *glass* dataset I found that there were no easily predicted attributes, they all were pretty evenly spread out.

(d) If they have a single bar in the histogram, it shows that they all contained this attribute. For example in the *glass* dataset, they all had iron and silicon in them in each of the types in the class attribute.