Introduction / Aim

The aim of this study is to develop classifiers that can accurately use medical data to forecast the potential onset of diabetes. In addition, the study aims to evaluate the performance of different types of classifiers based on accuracy, understandability / human readability, speed, and the impact of pre-processing techniques. This study is important because these classifiers can help medical professionals identify patients who are at risk of developing type 2 diabetes mellitus. Type 2 diabetes is a potentially life-threatening and chronic disease. These classifiers can help prevent adult-onset diabetes by identifying at-risk individuals so they can take early measures like increasing exercise and maintaining a healthy diet. Also, the study is important because it clearly identifies which types of classifiers and pre-processing techniques are most effective for this dataset. In the future, the strategies used to create the most effective classifiers can be reused for similar datasets.

Data

The data consists of 768 examples made up of 8 attributes and 1 class. The examples come from individuals from Pima indian heritage. The class identifies if the individual in the example tested positive (“yes”) or negative (“no”) for diabetes. There were 500 “yes” examples and 268 “no” examples.
The 8 attributes were:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

A pre-processing technique called Correlation-based Feature Selection (CFS) was used to select a subset of the 8 original features that are highly correlated with the class but uncorrelated with each other. In essence, Feature Selection searches for the aspects of data that are most useful for analysis and ignores irrelevant and redundant data in order to reduce computation and increase the performance of training classifiers. Feature Selection was performed on this dataset by conducting a Best-First Search attribute selection using a software program called Weka.
This process selected these 5 attributes:

1. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
2. 2-Hour serum insulin (mu U/ml)
3. Body mass index (weight in kg/(height in m)^2)
4. Diabetes pedigree function
5. Age (years)

Results and Discussion

Accuracy for a classifier is determined by using 10-fold stratified cross validation, which splits up the data into 10 comparable folds. Each fold consists of the same (within 1) number of “yes” examples and the same (within 1) number of “no” examples. Then, each classifier is trained on 9 of the 10 folds, and the remaining fold is used to test the accuracy by comparing the result of the classifier to the actual value for each example in the testing fold. This is repeated so that each fold is used as a testing fold, and the accuracies are averaged. The average accuracy of each classifier is shown in the figures below.

Weka Classifiers	**1NN**	**5NN**	**NB**	**ZeroR**	**1R**	**DT**	**MLP**	**SVM**	**RF**
No Feature Selection	67.8385	74.4792	75.1032	65.1042	70.8333	71.7448	75.3906	76.3021	74.8698
CFS	69.0104	74.4792	76.3021	65.1042	70.8333	73.3073	75.7813	76.6927	75.9115

Figure 1: The table shows accuracy of the 9 different types of classifiers processed by Weka.

My Classifiers	**My1NN**	**My5NN**	**MyNB**
No Feature Selection	73.9068	76.5491	74.9736
CFS	72.4197	75.6408	76.0228

Figure 2: The table shows accuracy of the 3 classifiers that were implemented by me in Java.

Of the Weka classifiers, there does not exist a classifier that is clearly more accurate than the others. The best accuracy was the Support Vector Machine (SVM) using CFS with an accuracy of 76.6927%. However, the difference between the most accurate classifier and the next 3 best is extremely low - less than 1% different. For this reason, factors like understandability and speed are considered to determine the best performing classifiers.

Understandability is the ability for a human to look at the classifier and decipher how it came to its decision. Of the four most accurate classifiers: (1) Support Vector Machine - CFS, (2) Naive Bayes - CFS, (3) Random Forest - CFS, (4) Multi-Layer Perceptron - CFS , only (2) Naive Bayes is easy for humans to understand. The Support Vector Machine classifier can be difficult to understand because it creates a d-dimensional (where d is the number of attributes) space that is impossible for the human brain to comprehend. Random Forest can be difficult to understand because it is a combination of many decision trees which can make it hard to follow. Multi-Layer perceptrons are inherently difficult to understand due to a hidden layer of perceptrons and changing weights and biases that obfuscate the impact that different attributes have. Alternatively, it is easy for humans to understand Bayes Theorem, which is the basis of how the Naive Bayes classifier is created. Further, it is easy for a human to understand why the classifier chooses yes or no when they can see the probability of “yes” or “no” given the input data. Notably, the other remaining (less accurate) classifiers are even more understandable.

The k-Nearest Neighbor classifiers are understandable because it’s easy for humans to see how input data is “close” to the training data examples that determine the class of the new example. It is easy for humans to understand ZeroR because only the ratio of the classes is considered. Similarly, it is easy for humans to understand OneR because it is easy to solve the one rule that defines the classifier. Finally, Decision Trees are usually easy to follow because a human simply checks the conditions of each node until they reach a leaf node. In other words, a decision tree is easy to understand because there is an exact path/solution to why/how an example was classified the way it was.

Another important factor for evaluating the performance of classifiers is speed/complexity. Weka measures the time to build a classifier. Unsurprisingly, the less understandable classifiers usually took longer to build. It is important to note that in this dataset the time to build is relatively low; however, with increasingly large datasets the time to build a classifier becomes more significant. Simple/quick classifiers had a time to build of 0 seconds. Again, even though the time to build the classifiers appears very low in written context, the difference is definitely noticeable in practice - especially in larger datasets.

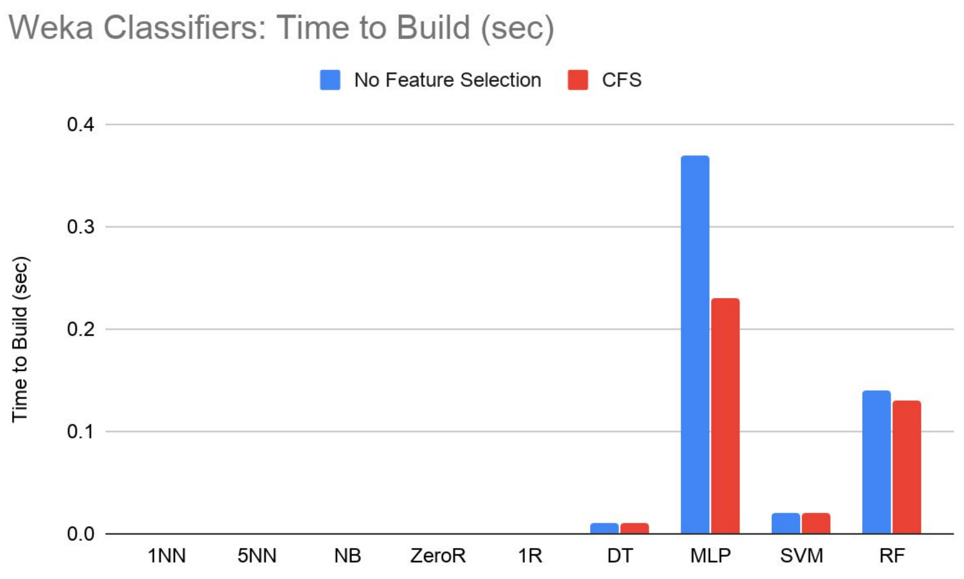


Figure 3: The chart shows the time to build each classifier in Weka.

It is important to discuss the impact of Correlation-based Feature Selection on the performance of the classifiers. CFS had either a positive or no effect on the accuracy of all of the Weka classifiers, as shown in the figure below.

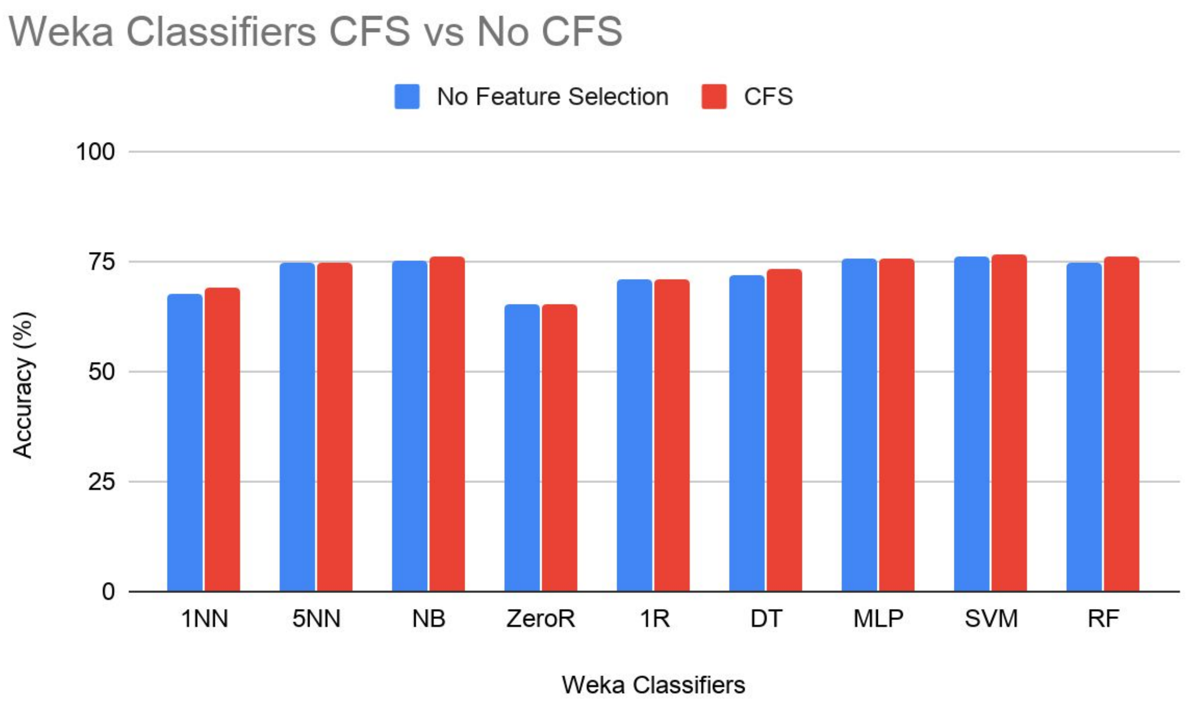
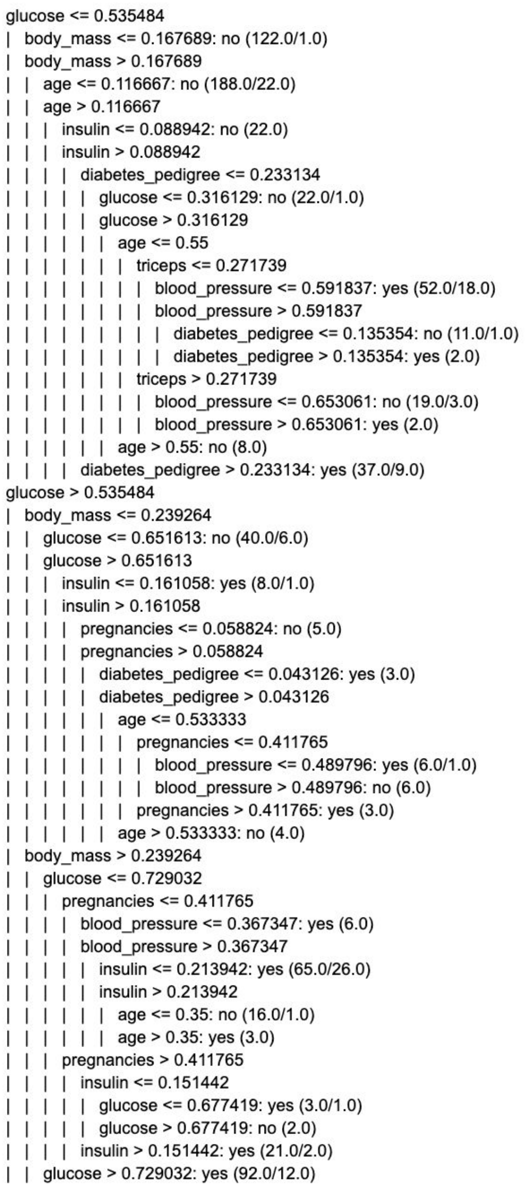


Figure 4: The chart compares the Weka Classifiers with and without Feature Selection.

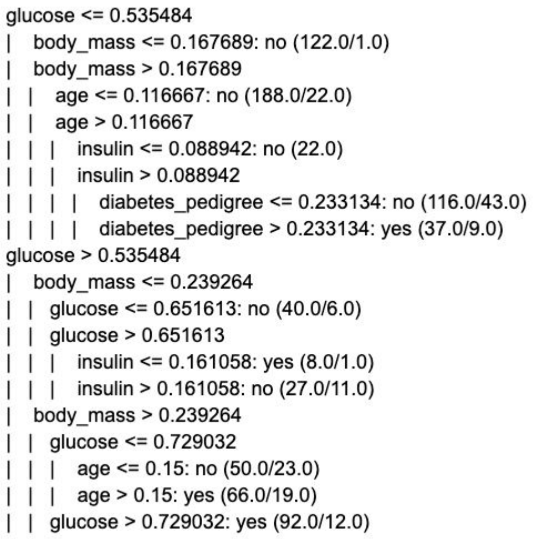
It is not surprising that feature selection was mostly beneficial for the accuracy of all of the Weka classifiers because the goal of Feature Selection is to select only the most meaningful attributes of the data. By reducing the number of attributes and selecting for attributes that are highly correlated with the class there is less noise in the data. Thus, it is easy to follow that when there is less noise in the data, classifiers become more accurate.

Feature selection was also beneficial because it reduced the time to build for the most complex Weka Classifiers, as seen in Figure 3. This is not surprising because feature selection reduces the total amount of data that is used to train the classifier, so there is less processing required.

Similarly, it can be shown that feature selection also reduces the complexity of the data. Again, this is not surprising because less data to process makes it easier for humans to comprehend. For example, 5-dimensional space (5 attributes selected by CFS) is relatively easier to comprehend than 8-dimensional space (8 attributes in original data). This benefit is most clearly shown by the difference in complexity between the Decision Tree classifiers in figure 5. Feature Selection makes it much easier to read and understand how the classifier processes new example data. In fact, the “No Feature Selection” Decision Tree has a smaller size (21) than the “CFS” Decision Tree has leaves/endpoints (27).



```
glucose <= 0.535484
|   body_mass <= 0.167689: no (122.0/1.0)
|   body_mass > 0.167689
|   |   age <= 0.116667: no (188.0/22.0)
|   |   age > 0.116667
|   |   |   insulin <= 0.088942: no (22.0)
|   |   |   insulin > 0.088942
|   |   |   |   diabetes_pedigree <= 0.233134
|   |   |   |   |   glucose <= 0.316129: no (22.0/1.0)
|   |   |   |   |   glucose > 0.316129
|   |   |   |   |   |   age <= 0.55
|   |   |   |   |   |   |   triceps <= 0.271739
|   |   |   |   |   |   |   |   blood_pressure <= 0.591837: yes (52.0/18.0)
|   |   |   |   |   |   |   |   blood_pressure > 0.591837
|   |   |   |   |   |   |   |   |   diabetes_pedigree <= 0.135354: no (11.0/1.0)
|   |   |   |   |   |   |   |   |   diabetes_pedigree > 0.135354: yes (2.0)
|   |   |   |   |   |   |   triceps > 0.271739
|   |   |   |   |   |   |   |   blood_pressure <= 0.653061: no (19.0/3.0)
|   |   |   |   |   |   |   |   blood_pressure > 0.653061: yes (2.0)
|   |   |   |   |   |   age > 0.55: no (8.0)
|   |   |   |   diabetes_pedigree > 0.233134: yes (37.0/9.0)
glucose > 0.535484
|   body_mass <= 0.239264
|   |   glucose <= 0.651613: no (40.0/6.0)
|   |   glucose > 0.651613
|   |   |   insulin <= 0.161058: yes (8.0/1.0)
|   |   |   insulin > 0.161058
|   |   |   |   pregnancies <= 0.058824: no (5.0)
|   |   |   |   pregnancies > 0.058824
|   |   |   |   |   diabetes_pedigree <= 0.043126: yes (3.0)
|   |   |   |   |   diabetes_pedigree > 0.043126
|   |   |   |   |   |   age <= 0.533333
|   |   |   |   |   |   |   pregnancies <= 0.411765
|   |   |   |   |   |   |   |   blood_pressure <= 0.489796: yes (6.0/1.0)
|   |   |   |   |   |   |   |   blood_pressure > 0.489796: no (6.0)
|   |   |   |   |   |   |   pregnancies > 0.411765: yes (3.0)
|   |   |   |   |   |   age > 0.533333: no (4.0)
|   body_mass > 0.239264
|   |   glucose <= 0.729032
|   |   |   pregnancies <= 0.411765
|   |   |   |   blood_pressure <= 0.367347: yes (6.0)
|   |   |   |   blood_pressure > 0.367347
|   |   |   |   |   insulin <= 0.213942: yes (65.0/26.0)
|   |   |   |   |   insulin > 0.213942
|   |   |   |   |   |   age <= 0.35: no (16.0/1.0)
|   |   |   |   |   |   age > 0.35: yes (3.0)
|   |   |   pregnancies > 0.411765
|   |   |   |   insulin <= 0.151442
|   |   |   |   |   glucose <= 0.677419: yes (3.0/1.0)
|   |   |   |   |   glucose > 0.677419: no (2.0)
|   |   |   |   insulin > 0.151442: yes (21.0/2.0)
|   |   glucose > 0.729032: yes (92.0/12.0)
```



```
glucose <= 0.535484
|   body_mass <= 0.167689: no (122.0/1.0)
|   body_mass > 0.167689
|   |   age <= 0.116667: no (188.0/22.0)
|   |   age > 0.116667
|   |   |   insulin <= 0.088942: no (22.0)
|   |   |   insulin > 0.088942
|   |   |   |   diabetes_pedigree <= 0.233134: no (116.0/43.0)
|   |   |   |   diabetes_pedigree > 0.233134: yes (37.0/9.0)
glucose > 0.535484
|   body_mass <= 0.239264
|   |   glucose <= 0.651613: no (40.0/6.0)
|   |   glucose > 0.651613
|   |   |   insulin <= 0.161058: yes (8.0/1.0)
|   |   |   insulin > 0.161058: no (27.0/11.0)
|   body_mass > 0.239264
|   |   glucose <= 0.729032
|   |   |   age <= 0.15: no (50.0/23.0)
|   |   |   age > 0.15: yes (66.0/19.0)
|   |   glucose > 0.729032: yes (92.0/12.0)
```

Figure 5: These diagrams show the Decision Trees for “No Feature Selection” (Left) and “CFS” (Right). “No Feature Selection” has 27 leaves and size 53. “CFS” has 11 leaves and size 21.

However, the impact that feature selection had on the accuracy of my classifiers that I implemented in Java was much more complicated. In fact, CFS hurt the accuracy of my k-Nearest Neighbor classifier, but was beneficial for my Naive Bayes classifier.

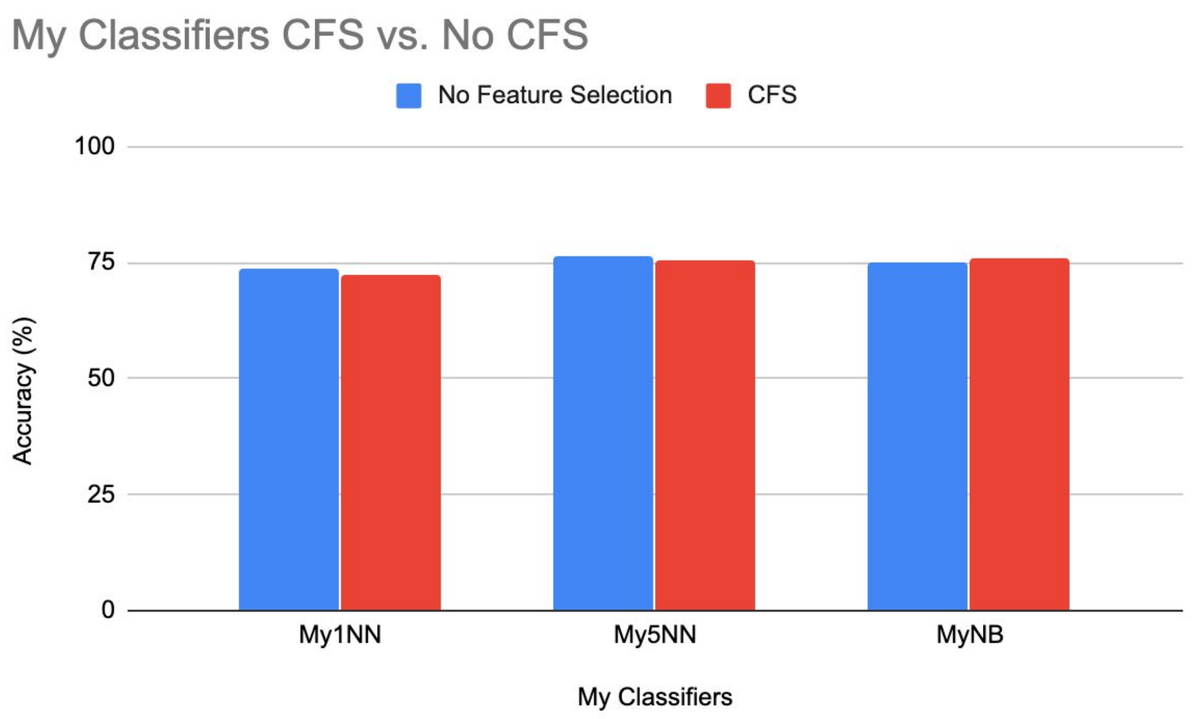


Figure 6: This chart shows the impact that CFS had on the accuracy of my classifiers.

It was unexpected to see that CFS hurt the accuracy of my kNN classifier, especially when it is shown that CFS benefits the accuracy of the 1NN classifier in Weka. An explanation for this difference may lie in the distance/evaluation measures that are used. My implementation of the k-Nearest Neighbor classifier used euclidean distance to measure how close a test example is to the training examples. If Correlation-based Feature Selection implemented by Weka uses a different distance/evaluation measure to select the best attributes, then the impact of CFS may have unexpected results. In other words, if CFS evaluates the “best features” differently from how the euclidean distance evaluates the “most similar” features, then the feature selection could have a negative impact on the kNN classifier’s accuracy. Further, it would make sense that Weka’s kNN classifier uses the same evaluation measure as Weka’s CFS evaluation measure. If those measures are the same, then it would follow that CFS has a positive impact on Weka’s kNN classifier (which it does) because CFS would reduce the noise of other attributes.

This explanation is consistent with why both implementations of the Naive Bayes classifier benefit from CFS by about +1% accuracy. Naive Bayes doesn’t use a distance or evaluation measure to predict the class. Instead, Naive Bayes uses probability based on the normal distribution of the training data. Thus, there is no difference in an evaluation measure that would lead to a divergence. Naive Bayes benefits in both examples because there is less noise from other attributes.

Conclusion

In conclusion, there are multiple classifiers that are relatively accurate and could be confidently used to forecast the potential onset of type 2 diabetes. The 3 most accurate classifiers were Support Vector Machine, Naive Bayes, and Random Forest; however, most other classifiers that were tested had a similar accuracy. The most accurate classifiers were able to correctly forecast the onset of diabetes around 75% of the time. Classifiers that are most difficult for a human to easily understand, are also the most complex and take longer to build. Interestingly, it was generally consistent that the more complex classifiers were also more accurate. In this way, it is important to find a balance between complexity and accuracy. More importantly, new problems will value accuracy vs. complexity/speed differently, so finding a good balance will be unique for each problem. For this problem, the Naive Bayes classifier is impressively accurate for how simple and fast it is.

Correlation-based Feature Selection is generally beneficial for the performance of classifiers. For all classifiers CFS clearly simplifies the classifier so that it is easier to understand and faster to build. Feature selection can drastically improve the understandability of classifiers like Decision Trees. In general, CFS was also beneficial for improving the accuracy of classifiers. The only instance where CFS was detrimental, was on the accuracy of my implementations of the k-Nearest Neighbor classifier. Thus, when performing feature selection it is important to fully understand how/why certain features are selected and how that relates to the process of how the classifier is built.

This study started to find a correlation between the time to build a classifier, and the perceived complexity/understandability of the classifier. In the future, it would be beneficial to explore this correlation further. For example, it would be interesting to train these classifiers on a much larger dataset in order to see more variability in the time to build each classifier. With this, this future study could help answer if the time to build increases as perceived understandability or complexity increases.

In addition, future work could explore the impact of different feature selection techniques. For example, it is still unknown for certain why my implementation of k-Nearest Neighbor was less accurate when CFS was used on the dataset. The explanation hypothesized in this study has yet to be tested. Thus, it would be beneficial for future work to explore why exactly my implementation of kNN did not benefit from CFS. It would be beneficial to explore how different feature selection techniques affect different classifiers.

Reflection

The most important thing I took away from this assignment was how simple classifiers can be just as good as complex classifiers. I first became interested in machine learning because I was introduced to neural networks and how interesting they are because they resemble human brains and have been used to do really amazing things. However, I didn’t realize just how powerful simple techniques can be in machine learning. For example, this assignment showed how the Naive Bayes classifier is the second-most accurate classifier and is barely less accurate than the Support Vector Machine. This is extremely impressive considering how Naive Bayes uses Bayes Theorem which is a very understandable mathematical principle. In this way, Naive Bayes is easier to implement, faster to train, and simpler to explain to others. The importance of a fast and speed and understandable classifier should not be undervalued because, in the end humans make the final decision. Obviously, when a classifier is fast and easy to explain, a human is more likely to trust it. Thus, when a simple classifier like Naive Bayes (or kNN, or oneR, or any other simple classifier) proves to be highly accurate, it can clearly be a very powerful asset.