**Austin Lee**
SID: 862084022
Email: Alee235@ucr.edu
Dec 10, 2021
Github Link: https://github.com/austinslee/CS170-Project-2.git

Project 2 for CS 170 Fall 2021, with Dr. Eamonn Keogh

In completing this assignment, I consulted

- Consulted Project 2_full_briefing.mp4 and Project 2_full_briefing.pptx provided by Professor Keogh for pseudocode and help
- Consulted Project2.doc provided by Professor Keogh for instructions
- Consulted Project_2_sample_report.docx provided by Professor Keogh for report outline

All code is original, except for:
1) C++ Libraries: iostream, fstream, sstream, vector, climits, cmath, stdlib.h, stdio.h, algorithm

In this project, we are tasked with comparing and contrasting the nearest neighbor algorithm with Forward Selection and Backward Elimination on data sets of varying size and features.

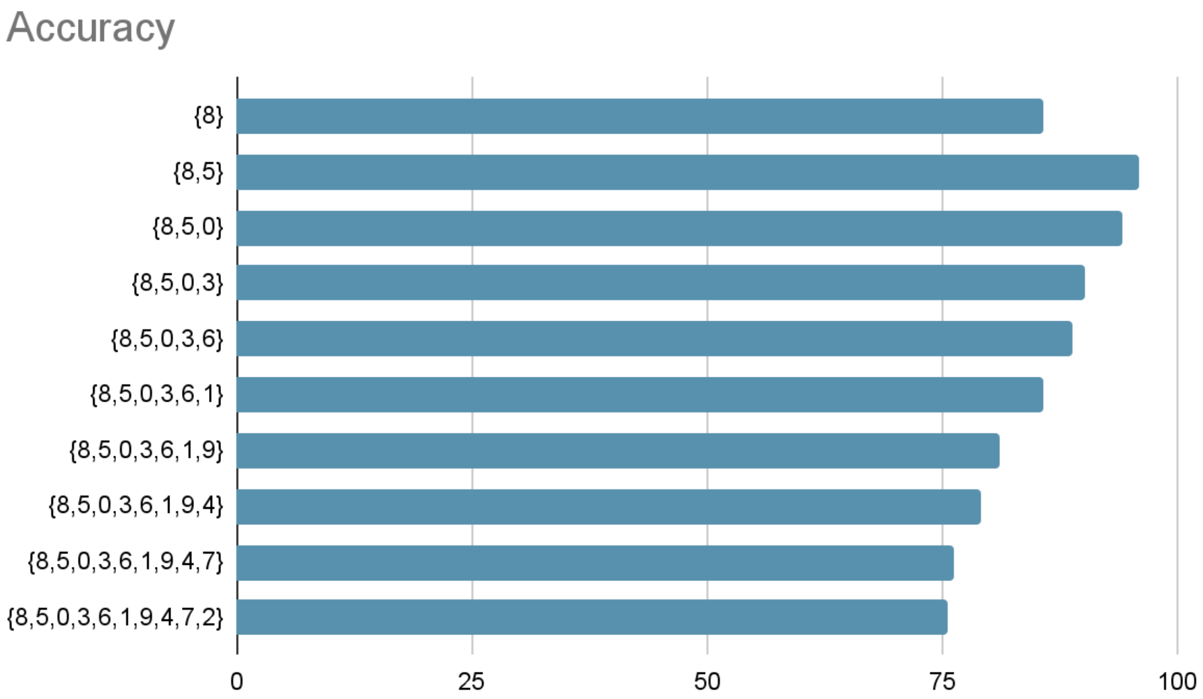
*I would like to  note that my program has 0 as the first feature so that means 1 would be the second feature, 2 would be the third feature, 3 would be the fourth feature, and so on. So when looking at my results, you would want to add 1 to see which feature it is.

**Small Data Set Analysis**

Forward Selection

In the table and graph below, we see the results of running the nearest neighbor algorithm with forward selection on the small dataset assigned to me, Ver_2_CS170_Fall_2021_Small_data_68.txt

| Feature Set | Accuracy (%) |
|---|---|
| {8} | 85.8 |
| {8,5} | 96 |
| {8,5,0} | 94.2 |
| {8,5,0,3} | 90.2 |
| {8,5,0,3,6} | 88.8 |
| {8,5,0,3,6,1} | 85.8 |
| {8,5,0,3,6,1,9} | 81.2 |
| {8,5,0,3,6,1,9,4} | 79.2 |
| {8,5,0,3,6,1,9,4,7} | 76.2 |
| {8,5,0,3,6,1,9,4,7,2} | 75.6 |



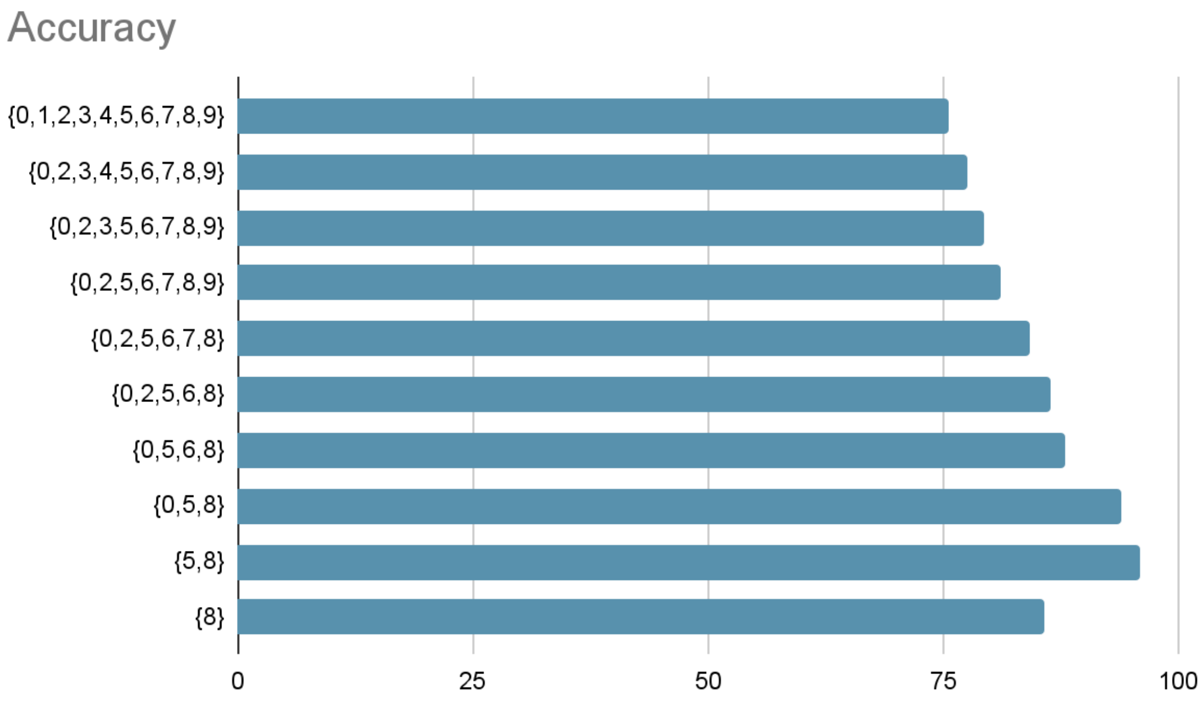
My program searches through the possible data sets by cycling through all the features to add. After the first iteration, I find that adding feature 8 yields the highest accuracy of 85.8% so I add that to my feature set. I then add feature 5 which grants a 10.2% increase in the accuracy. The next feature added is 0 which decreases the accuracy to 94.2% which would be a 1.8% decrease in accuracy. Every subsequent addition to the feature set decreases the accuracy until we end with an accuracy of 75.6% with all 10 features added in. This leads me to believe that the only 2 features that are relevant would be features 8 and 5. All other features only bring down the accuracy and are therefore probably irrelevant.

Backwards Elimination

Running nearest neighbor with backwards elimination on the same small data text gives me the following results.

| Feature Set | Accuracy (%) |
|---|---|
| {0,1,2,3,4,5,6,7,8,9} | 75.6 |
| {0,2,3,4,5,6,7,8,9} | 77.6 |
| {0,2,3,5,6,7,8,9} | 79.4 |

| | |
|---|---|
| {0,2,5,6,7,8,9} | 81.2 |
| {0,2,5,6,7,8} | 84.2 |
| {0,2,5,6,8} | 86.4 |
| {0,5,6,8} | 88 |
| {0,5,8} | 94.2 |
| {5,8} | 96 |
| {8} | 85.8 |



First, I will do a sanity check. The first tested subset, that is subset with all of the features, should have the same accuracy as the last tested subset in forward selection. With a score of 75.6%, I can conclude that this is the case. My program chose to remove feature 1 and increased the accuracy to 77.7%. It then chose to remove feature 4 to increase the accuracy to 79.4%. As my program removed more and more features, my accuracy continued to increase until I got to the feature subset {5,8}. After my program removed feature 5, my accuracy dropped down from 96% to 85.8%. Based on this result I can conclude that feature 5 is a relevant feature to the data set and that removing it will reduce the accuracy.

**Small Data Set Conclusion**

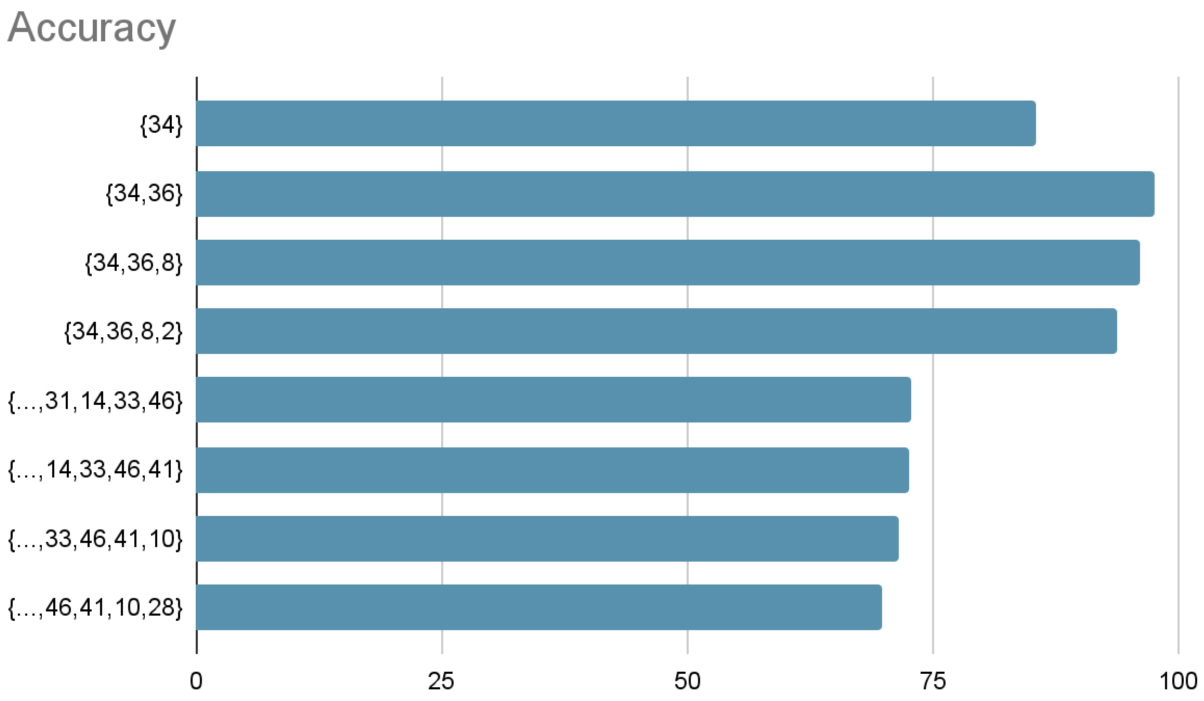
Both my forward selection and backwards elimination have found the same exact feature set, {5,8} with the same accuracy, 96%, so I can conclude with some degree of confidence that this is the best feature subset to use. And that there is no difference between forward selection and backward elimination for this small data set.

**Large Data Set Analysis**

Forward Selection

In the table and graph below, we see the results of running the nearest neighbor algorithm with forward selection on the Large dataset assigned to me, Ver_2_CS170_Fall_2021_Large_data_39.txt

| Feature Set | Accuracy (%) |
|---|---|
| {34} | 85.45 |
| {34,36} | 97.65 |
| {34,36,8} | 96.15 |
| {34,36,8,2} | 93.75 |
| {...,31,14,33,46} | 72.9 |
| {...,14,33,46,41} | 72.55 |
| {...,33,46,41,10} | 71.5 |
| {...,46,41,10,28} | 69.9 |



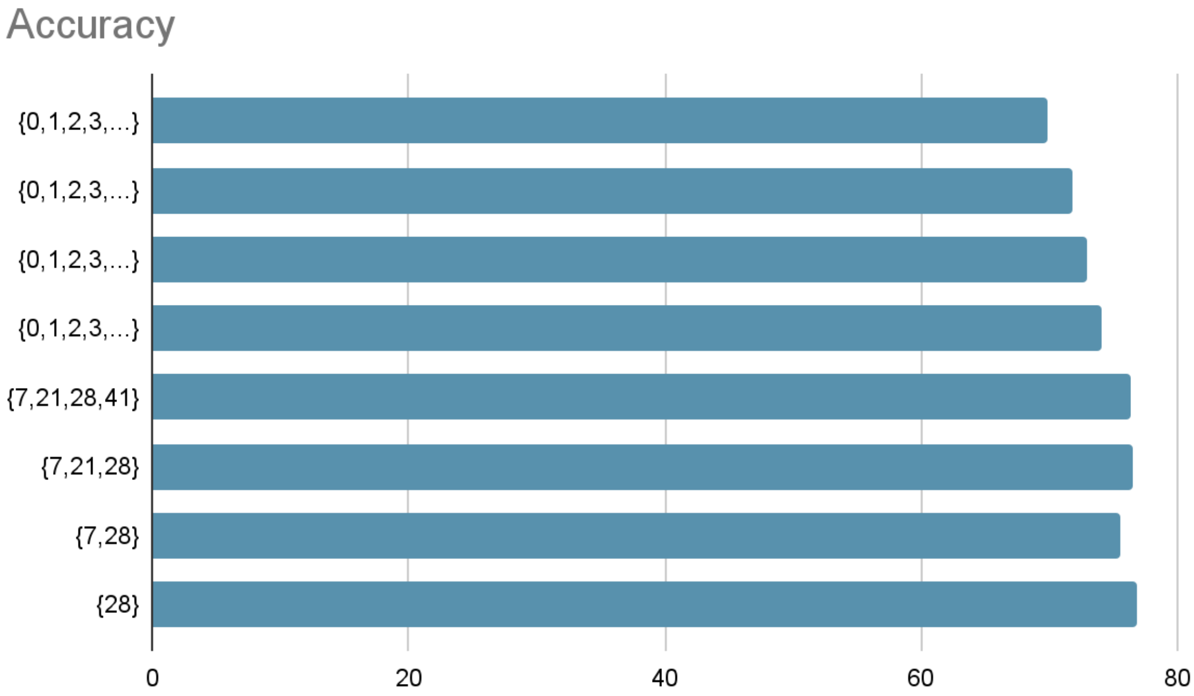
My program looks at the different accuracies and chooses feature 34 as the highest accuracy with 85.45%. My program then chooses feature 36 for a feature set of {34, 36} and an accuracy of 97.65%. My program then chooses features 8 for an accuracy of 96.15%. As my program chooses more and more features to add, I notice a downwards trend of accuracy. Every subsequent addition, only reduced the accuracy until I reached an accuracy of 69.9% with a feature set of {34,36,8,2,21,3,45,12,44,4,13,38,1,9,19,40,49,6,11,32,15,30,48,7,18,0,29,42,43,5,35,25,24,26,23,20,39,37,16,22,17,27,31,14,33,46,41,10,28}. I believe that all the features chosen after feature 36 are probably irrelevant because each addition only causes the accuracy to fall.

I would also like to note that there were too many subsets and accuracies to list them all on the graph and table so I listed the first four and last four results. I would also like to note that the feature subsets got too long at the end so I listed the last for features in the feature subset.

Backwards Elimination

Running nearest neighbor with backwards elimination on the same large data text gives me the following results.

| Feature Set | Accuracy (%) |
|---|---|
| {0,1,2,3,...} | 69.9 |
| {0,1,2,3,...} | 71.8 |
| {0,1,2,3,...} | 72.95 |
| {0,1,2,3,...} | 74.1 |
| {7,21,28,41} | 76.3 |
| {7,21,28} | 76.45 |
| {7,28} | 75.55 |
| {28} | 76.85 |



It was hard to communicate which numbers were removed in the table so I will just write which values were removed here. In the first feature set, no values were removed and it is all the features in one subset. In the second feature set, feature 45 was removed. In the third feature set, feature 28 was removed.

First, I will do a sanity check. The first tested subset, that is subset with all of the features, should have the same accuracy as the last tested subset in forward selection. With a score of 69.9%, I can conclude that this is the case.

My program started off with all of the features and had an accuracy of 69.9%. It then removed feature 45 and increased the accuracy to 71.8%. It then removed feature 28 to increase the accuracy to 72.95%. As my program removed more features, the accuracy continued to steadily increase until I reached feature subset {0,2,3,4,5,6,7,10,11,13,15,19,20,26,27,31,33,37,38,40,42,43,44,47,48,49} with an accuracy of 79.45%. After that as my program removed more features, the accuracy bounced back and forth between increasing and decreasing with a general trend of decreasing more. I eventually ended with a feature set of {28} and an accuracy of 76.85%.

**Large Data Set Conclusion**

My forward selection and backward elimination found completely different feature subsets with my forward selection finding feature subset {34, 36} with an accuracy of 97.65% and my backward elimination finding feature subset {0,2,3,4,5,6,7,10,11,13,15,19,20,26,27,31,33,37,38,40,42,43,44,47,48,49} with an accuracy of 79.45%. I can conclude that for this large data set, forward selection was much better and found a much higher accuracy feature set than backward selection by about 18.2%.

**Computational effort for search:**

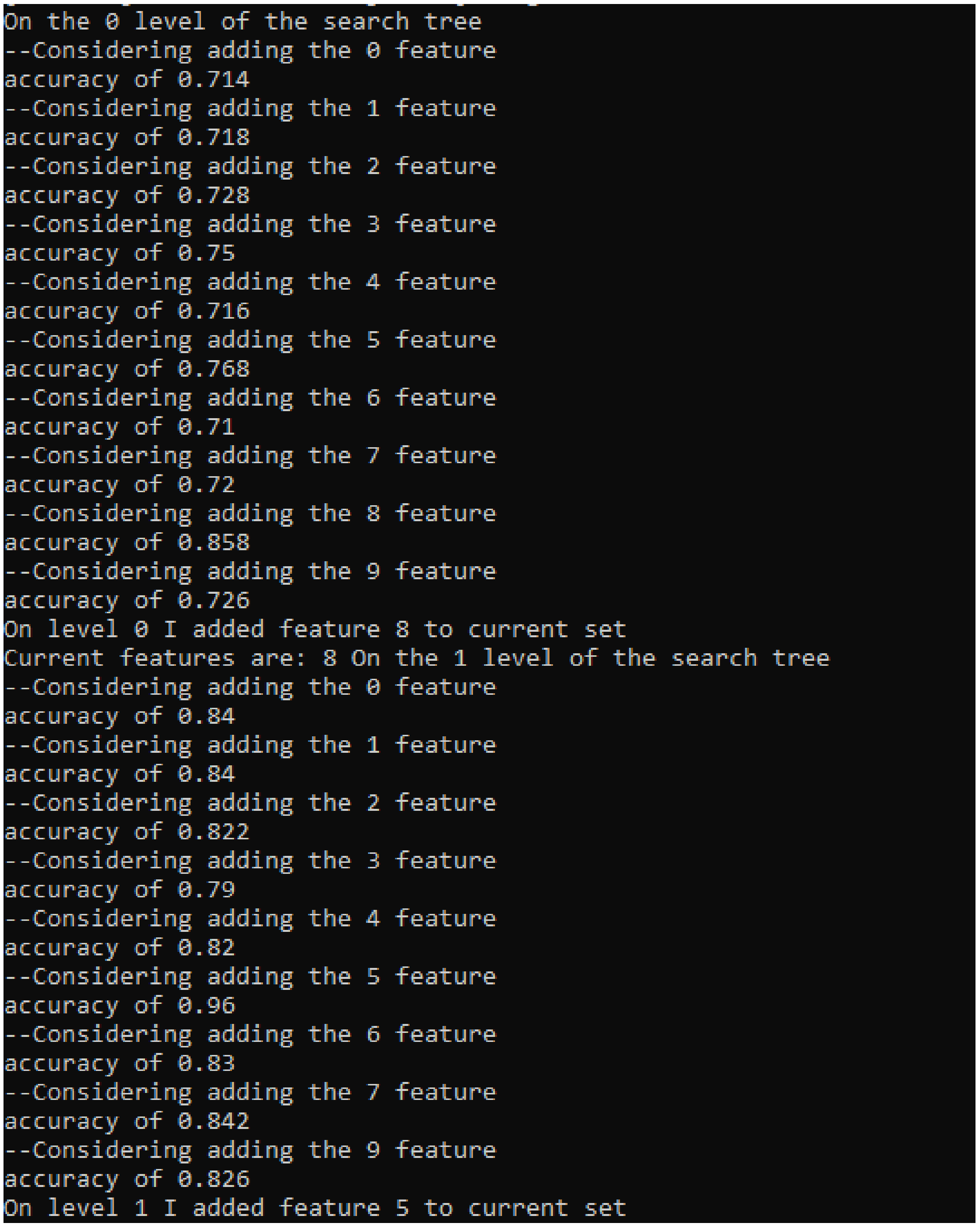
I ran my program with the g++ compiler with optimization flag -O3 set. In the table below, I report the computational time estimates for forward selection and backward elimination on both data sets.

|  | Small Dataset(10 features, 500 instances) | Large Dataset(50 features, 2000 instances) |
|---|---|---|
| Forward Selection | 1 Second | 25 Minutes 55 Seconds |
| Backward Elimination | 1 Second | 48 Minutes 31 Seconds |

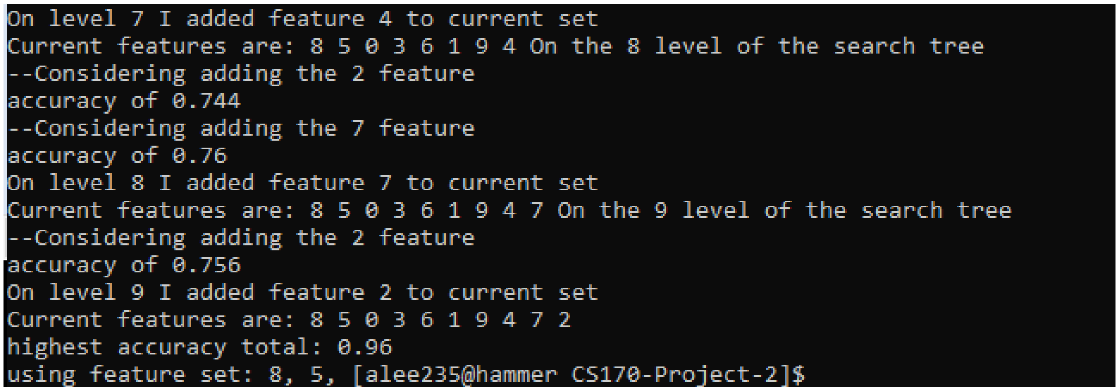
As you can see from the table, my program works quite quickly on small datasets, however, as the amount of data increases, my program quickly becomes quite slow at calculating the accuracies using Forward Selection and Backward Elimination. This is to be expected, as the number of entries and features increases, the amount of calculations needed also increases.

I would also like to point out that for the large dataset, my backward elimination was much slower than my forward selection with a time of 48 Minutes and 31 Seconds compared to 25 Minutes and 55 Second. There is a time difference of about 23 Minutes and 24 Seconds which means that forwards selection was almost twice as fast as my backwards elimination. I'm not sure if this is because my implementation of backwards elimination was inefficient compared to my implementation of forward selection. I wouldn't be surprised if this was the case. However, based solely on the data I gathered, I can theorize that backward elimination is much slower than forward selection on large datasets. While on small datasets, there is a negligible difference between them.

**Below is a trace of my algorithm. I am only showing Forward Selection on the small dataset. I have also cut off the middle of my trace to save space.**



```
On the 0 level of the search tree
--Considering adding the 0 feature
accuracy of 0.714
--Considering adding the 1 feature
accuracy of 0.718
--Considering adding the 2 feature
accuracy of 0.728
--Considering adding the 3 feature
accuracy of 0.75
--Considering adding the 4 feature
accuracy of 0.716
--Considering adding the 5 feature
accuracy of 0.768
--Considering adding the 6 feature
accuracy of 0.71
--Considering adding the 7 feature
accuracy of 0.72
--Considering adding the 8 feature
accuracy of 0.858
--Considering adding the 9 feature
accuracy of 0.726
On level 0 I added feature 8 to current set
Current features are: 8 On the 1 level of the search tree
--Considering adding the 0 feature
accuracy of 0.84
--Considering adding the 1 feature
accuracy of 0.84
--Considering adding the 2 feature
accuracy of 0.822
--Considering adding the 3 feature
accuracy of 0.79
--Considering adding the 4 feature
accuracy of 0.82
--Considering adding the 5 feature
accuracy of 0.96
--Considering adding the 6 feature
accuracy of 0.83
--Considering adding the 7 feature
accuracy of 0.842
--Considering adding the 9 feature
accuracy of 0.826
On level 1 I added feature 5 to current set
```



```
On level 7 I added feature 4 to current set
Current features are: 8 5 0 3 6 1 9 4 On the 8 level of the search tree
--Considering adding the 2 feature
accuracy of 0.744
--Considering adding the 7 feature
accuracy of 0.76
On level 8 I added feature 7 to current set
Current features are: 8 5 0 3 6 1 9 4 7 On the 9 level of the search tree
--Considering adding the 2 feature
accuracy of 0.756
On level 9 I added feature 2 to current set
Current features are: 8 5 0 3 6 1 9 4 7 2
highest accuracy total: 0.96
using feature set: 8, 5, [alee235@hammer CS170-Project-2]$
```

**Code**
Github link to code: https://github.com/austinslee/CS170-Project-2.git

I would like to note that to change which data set to test, go to main.cpp, scroll all the way down to int main(), and uncomment the lines with inFS.open() to change which data sets are opened. In order to change whether forward selection or backward elimination is used, go to main.cpp, scroll all the way down to int main() and uncomment feature_search_demo() or backwards_elim based on which one you want to do.