Project 2 CS205: Introduction to Artificial Intelligence, Dr. Eamonn Keogh

Shruti Jawale
SID# [redacted]
Email: sjawa006@ucr.edu
Birthday Sept 5 2001
June 15, 2023

Jakin Chan
SID# [redacted]
Email: jchan419@ucr.edu
Birthday: Jan 8 2001
June 15, 2023

In completing this assignment, we consulted:

- https://www.udacity.com/blog/2021/05/how-to-read-from-a-file-in-cpp.html
- https://cplusplus.com/reference/sstream/stringstream/stringstream/
- https://en.cppreference.com/w/cpp/container/vector/push_back
- https://en.cppreference.com/w/cpp/container/vector
- https://en.wikipedia.org/wiki/Cross-validation_(statistics)
- Keogh's ML1 and ML2 slides
- Keogh's CS205 Project 2 instructions and sample report
- CS170 project2 description of leave-one-out validator
- https://cplusplus.com/reference/cmath/fabs/
- https://www.geeksforgeeks.org/cpp-functions-pass-by-reference/
- https://www.geeksforgeeks.org/cpp-function-call-by-value/
- https://www.tutorialkart.com/cpp/cpp-vector-size/#gsc.tab=0
- https://stackoverflow.com/questions/31162367/significance-of-ios-basesync-with-stdiofalse-cin-tienull
- https://www.geeksforgeeks.org/setbegin-setend-c-stl/
- https://stackoverflow.com/questions/7631996/remove-an-element-from-a-vector-by-value-c
- https://en.cppreference.com/w/cpp/container/vector/erase
- https://stackoverflow.com/questions/798046/digit-limitation-from-decimal-point-in-c
- https://www.kaggle.com/datasets/rounakbanik/pokemon?resource=download
- https://gamerant.com/pokemon-types-how-many/

Packages/Libraries used

- chrono library for time tracking

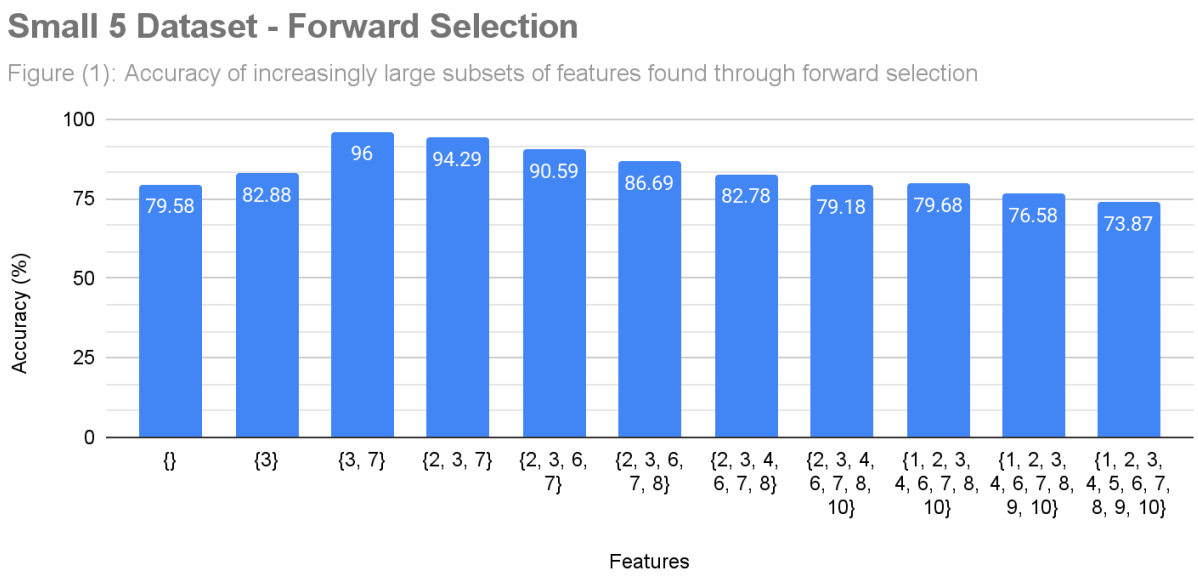
Outline of Report

Cover Page (This page)	1
Report	2-7
Program Trace	8-16
Code Also see: https://github.com/JakinChan200/CS205Project2	17-23

Project Report

In essence, forward selection is an algorithm whereby one starts with no features, and through leave-one out nearest neighbor validation, the features with the highest accuracy are added. This is repeated until all of the features are added. At the end, the combination of features with the highest accuracy will be kept. The backwards elimination algorithm uses leave-one out nearest neighbor validation as well, but to decide what features to drop.

In Figure (1) below, an example can be seen with forward selection on the small dataset 5. This dataset consists of 10 total features and 1000 records. In the figure below, features {3, 7} have the highest accuracy at 96%, with features {2, 3, 7} at 94.29% and {2, 3, 6, 7} at 90.59% the next highest accuracies. This means that features '3' and '7', based on the data in Figure (1), most likely have the largest correlation for classification for the dataset.

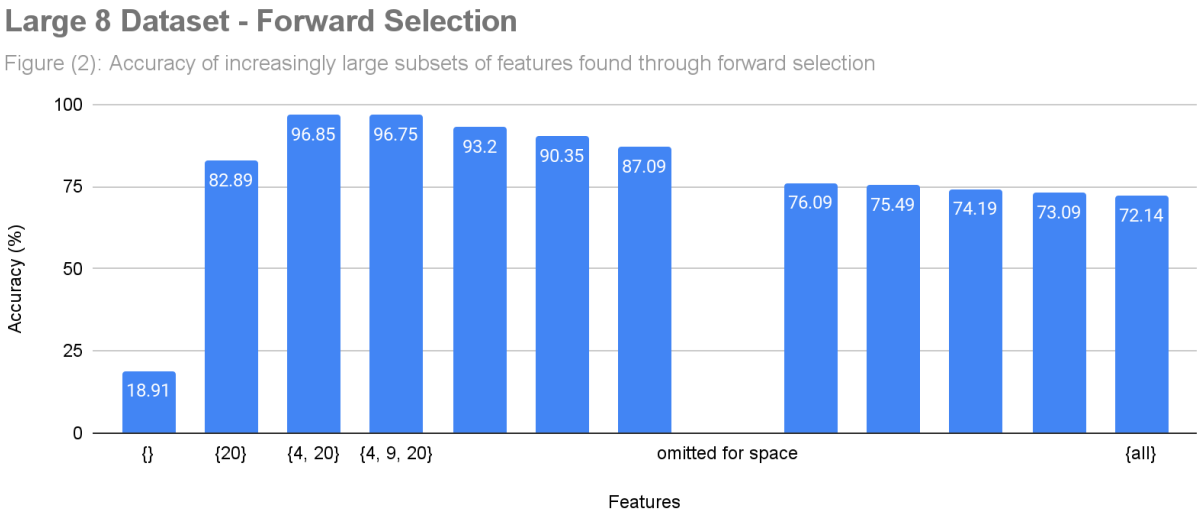


Backwards Elimination was also run on the small 5 dataset. Like the forward selection, the resulting feature list consisted of {3, 7} with an accuracy of 96%.

Conclusion for Small Data Set

Throughout both forward selection and backward elimination, the feature list {3, 7} results in the global maxima in terms of accuracy. This leads us to surmise that features '3' and '7' are some of the best features for classification. If this model were to be deployed, we would see a an accuracy of roughly 96%

The next dataset to be run is the Large 8 dataset. This dataset contains 40 features and 2000 records. In Figure (2) below, the results are displayed. From running forward selection feature selection on this dataset, the features with the highest accuracy is {4, 20} with an accuracy of 96.85%. The second and third highest accuracy features are {4, 9, 20} at 96.75% and {4, 9, 17, 20} with an accuracy of 93.2%. Features {4, 20} and {4, 9, 20} have a difference of only .13%, just a little over a tenth of a percent. With such minute differences, some may consider the two feature subsets to be practically equal, but since features {4, 20} is simpler, that is preferred. The interesting thing is that the base accuracy with no features is 18.91%.



When the Large 8 Dataset was run through backwards elimination, the feature subset {4, 20} was also outputted as the global maxima with an accuracy of 96.85%. This falls in line with the results of the forward selection search above.

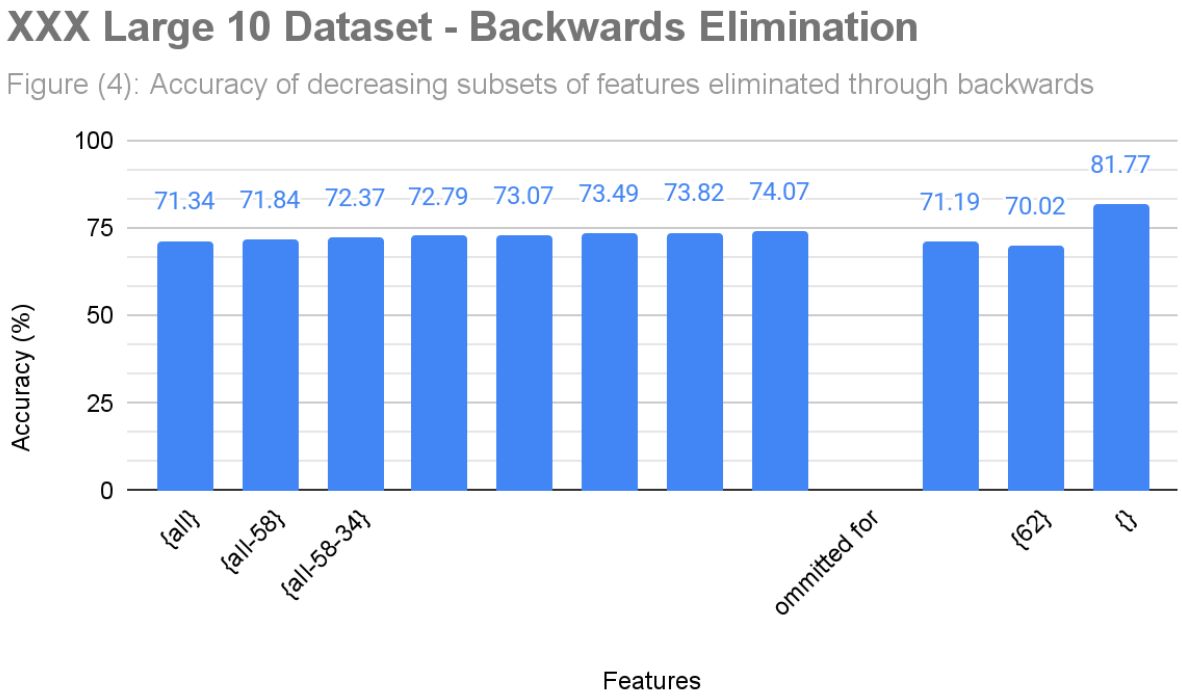
Conclusion for Large Data Set

In both forward selection and backwards elimination, the features {4, 20} were found to have the highest accuracy of 96.85%. This leads us to believe that the features '4' and '20' are some of the best features to aid in the classification of the data instances. If this model were to be deployed, one should expect an accuracy rate similar to 96.85%.

In Figure (3) below, the XXX large 10 dataset is used. It contains 80 total features and 4000 records. In the figure, the feature list with the highest accuracy is {58, 78} with an accuracy of 97.47%. This means that features '58' and '78' are the most likely to be highly correlated with the classification of the records. With the inclusion of features {58, 78}, the drop in accuracy as features are added is gradual. It is interesting to note that the inclusion of feature '58' causes a jump of 12.02% in accuracy.

This is interesting because in Figure (4), the first feature to be eliminated is feature '58'. In this case, the empty feature list has the highest accuracy rate. This result implies that choosing a specific classification every time provides better accuracy than considering any features. The accuracy rate of 81.77% is correct as out of 4000 records, 3270 are classified as 2.0000000e+00. 3270/4000 works out to 0.8175, or 81.75%, which is close to what we got.

As for feature '58', the accuracy of the classification continues to gradually climb after feature '58' was eliminated. This is a stark contrast from Figure (3), where the inclusion of feature '58' caused the accuracy of the classifier to jump 12.02%. There could be a number of reasons why it was considered a very good feature from the forward selection above, but a very bad feature from the backwards elimination algorithm. One such reason is that unrelated features can cause noise and hurt the accuracy of search.



Conclusion for XXX Large Data Set
In forward Selection Search, the feature subset {58, 78} resulted in the global maxima for accuracy. But in backwards elimination search, a subset of no features resulted in the global maxima for accuracy. More time and effort would be necessary to figure out exactly where the discrepancy lies. If this classifier would be deployed, the feature subset {58, 78} is recommended with an expected accuracy of 97.47%.

Computational effort for search
The search was implemented in C++, and on a laptop with an Intel Core i7-11800H and 16GB of RAM. Do note that the "-O4" compiler flag was used for all of the runs.

Datasets and their Runtime			
	Small 5	Large 8	XXXLarge 10
Forward selection	279 milliseconds	5845 milliseconds	20.29 minutes
Backward elimination	344 milliseconds	8718 milliseconds	35.02 minutes

Figure (5): Runtime of Algorithms

According to Figure 5, backwards elimination consistently performs worse than forward selection. This makes sense because forward selection has an inverse ratio between the amount of features to consider and the number of features currently running on nearest neighbor while backwards elimination has a directly proportional relationship. This makes sense because when the algorithm has more features that it is running nearest neighbor on, it has less features to consider. Backwards elimination, on the other hand, has both being equal. Since the search tries the nearest neighbor on each feature combination, backwards elimination results in more computations.

Processing a Real-World Classification Dataset

I did a forward selection on The Complete Pokemon Dataset, which was taken from the Kaggle library. I used the data to classify the type of pokemon for several hundred pokemon. The dataset I used is a slightly modified version of the dataset provided in the Kaggle library.

I deleted the categorical data that I felt was unnecessary, aka the english name and the japanese name… etc. I took the categorical words of type 1 from the dataset and converted them into numbers. These are the specific conversions I made: normal=1, fire=2, water=3, grass=4, electric=5, ice=6, fighting=7, poison=8, ground=9, flying=10, psychic=11, bug=12, rock=13, ghost=14, dark=15, dragon=16, steel=17, fairy=18. I added the similarly modified type 2 from the dataset into our own dataset as feature 28 to see if there was a correlation between type 1 and type 2. For the empty spots in the type 2 column, I put a 0. I kept all of the 18 weaknesses columns, one for each of the 18 possible types of weakness, in addition to several extra columns like attack, defense, sp_attack, sp_defense, percentage of males, hp, speed, and the weight in kg. All the columns are numerical values, most of which are discrete and some are continuous.

The dataset has 29 features (not including the class attribute) with 800 instances.

Feature Information:

1. against_bug (discrete)
2. against_dark (discrete)
3. against_dragon (discrete)
4. against_electric (discrete)
5. against_fairy (discrete)
6. against_fight (discrete)
7. against_fire (discrete)
8. against_flying (discrete)
9. against_ghost (discrete)
10. against_grass (discrete)
11. against_ground (discrete)
12. against_ice (discrete)
13. against_normal (discrete)
14. against_poison (discrete)
15. against_psychic (discrete)
16. against_rock (discrete)
17. against_steel (discrete)
18. against_water (discrete)
19. attack (discrete)
20. base_egg_steps (discrete)
21. defense (discrete)
22. height_m (continuous)
23. hp (discrete)
24. percentage_male (continuous)
25. sp_attack (discrete)
26. sp_defense (discrete)
27. speed (discrete)
28. type2 (categorical converted to discrete)
29. weight_kg (continuous)

I ran a forward selection on the Pokemon Dataset with early abandoning after three consecutive decreases in accuracy and got a final accuracy of 95.12% for the features: {1, 7, 8, 9, 12, 14, 16, 17, 18, 28, 3}.

I ran a backward elimination on the Pokemon Dataset with early abandoning after three consecutive decreases in accuracy and got a final accuracy of 28.38% for the features: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27}.

When I ran forward selection, the first feature it chose was feature 9 (against_ghost), for which the value represents that pokemon's weakness against ghost types. This makes sense, since the type of a Pokemon is likely to be correlated to what type it is weak against. Having what type a pokemon is weak against is a good feature to have since certain types are generally consistently weak against specific other types of pokemon. The next feature chosen was feature 18, against_water which makes sense for a similar reason, the type of a Pokemon is likely to be much more affected by whether it is weak to water types than other weaknesses. Water types are probably a common enough weakness to have for multiple other types of pokemon, which is why it works well for predicting the classification of a given Pokemon.

I noticed that the forward selection chose feature 3 at the end after it chose feature 28 instead of earlier. I scanned through the entire output for it and saw that feature 3 had an initial bad accuracy but as other features got added, the accuracy with the addition of feature 3 increased gradually. That shows me that sometimes a feature does not look like it will lead to an accurate classification, but it does eventually lead to an accurate classification when other features are combined with it.

For backward elimination, the reason why the final accuracy was so low is because it’s possible that a combination of good features creates a bad feature as well as a combination of bad features could create a good looking feature. Specifically, the backward elimination eliminated a bunch of features and ended up with only the bad features. The final accuracy somehow still includes those bad features. I say bad features here since the forward selection process did not consider adding those features.

The features that were never chosen in the forward selection: features 4, 5, 6, 10, 11, 13, 15, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29. This makes sense because most of those are the features that I added on purpose that I knew would not give me a good accuracy.

Overall, looking at the features our program selected for each search, we can see that some features do not ever lead to an accurate classification. Ten of the features that the forward selection never considered are the same ones that I expected to not lead to an accurate classification. Additionally, my prediction that maybe feature 28, the secondary type of the pokemon, would have had some effect on the type of the Pokemon, and it was in the final accuracy feature list for the forward selection.

Output Traceback

All tracebacks below are with early abandonment. For traceback without early abandonment or the code, see https://github.com/JakinChan200/CS205Project2

Forward Selection with Small Dataset 5 with early abandonment after 3 levels of accuracy decreasing

We processed small dataset 5 and got the following output:
The dataset has 10 features (not including the class attribute) with 1000 instances.

What size dataset are you planning to use?
1: small
2: large
3: XXXlarge
4: Custom Test Data
5: Pokemon Dataset
1
What dataset file do you want to use?
Enter a valid number from 1-33:
5
Datasets/CS170_small_Data__5.txt
Type the number of the algorithm you want to run.
1. Forward Selection
2. Backwards Elimination
1
Using Features(s) {} while considering 1, the accuracy is 65.67%
Using Features(s) {} while considering 2, the accuracy is 66.47%
Using Features(s) {} while considering 3, the accuracy is 82.88%
Using Features(s) {} while considering 4, the accuracy is 69.57%
Using Features(s) {} while considering 5, the accuracy is 67.47%
Using Features(s) {} while considering 6, the accuracy is 65.57%
Using Features(s) {} while considering 7, the accuracy is 72.07%
Using Features(s) {} while considering 8, the accuracy is 67.97%
Using Features(s) {} while considering 9, the accuracy is 68.97%
Using Features(s) {} while considering 10, the accuracy is 67.87%
For level 1, the index chosen is 3 with an accuracy of 82.88%
The current feature list is: {3}

Using Features(s) {3} while considering 1, the accuracy is 82.58%
Using Features(s) {3} while considering 2, the accuracy is 81.88%
Using Features(s) {3} while considering 4, the accuracy is 82.18%
Using Features(s) {3} while considering 5, the accuracy is 85.19%
Using Features(s) {3} while considering 6, the accuracy is 83.18%
Using Features(s) {3} while considering 7, the accuracy is 96%
Using Features(s) {3} while considering 8, the accuracy is 84.58%
Using Features(s) {3} while considering 9, the accuracy is 81.98%
Using Features(s) {3} while considering 10, the accuracy is 83.48%
For level 2, the index chosen is 7 with an accuracy of 96%
The current feature list is: {3, 7}

Using Features(s) {3, 7} while considering 1, the accuracy is 91.69%
Using Features(s) {3, 7} while considering 2, the accuracy is 94.29%

Using Features(s) {3, 7} while considering 4, the accuracy is 92.49%
Using Features(s) {3, 7} while considering 5, the accuracy is 93.39%
Using Features(s) {3, 7} while considering 6, the accuracy is 92.59%
Using Features(s) {3, 7} while considering 8, the accuracy is 92.69%
Using Features(s) {3, 7} while considering 9, the accuracy is 92.19%
Using Features(s) {3, 7} while considering 10, the accuracy is 92.79%

numLevelsDecreasing = 1
For level 3, the index chosen is 2 with an accuracy of 94.29%
The current feature list is: {2, 3, 7}

Using Features(s) {2, 3, 7} while considering 1, the accuracy is 89.69%
Using Features(s) {2, 3, 7} while considering 4, the accuracy is 89.69%
Using Features(s) {2, 3, 7} while considering 5, the accuracy is 88.69%
Using Features(s) {2, 3, 7} while considering 6, the accuracy is 90.59%
Using Features(s) {2, 3, 7} while considering 8, the accuracy is 88.39%
Using Features(s) {2, 3, 7} while considering 9, the accuracy is 88.79%
Using Features(s) {2, 3, 7} while considering 10, the accuracy is 90.59%

For level 4, the index chosen is 6 with an accuracy of 90.59%
The current feature list is: {2, 3, 6, 7}

Using Features(s) {2, 3, 6, 7} while considering 1, the accuracy is 84.98%
Using Features(s) {2, 3, 6, 7} while considering 4, the accuracy is 86.39%
Using Features(s) {2, 3, 6, 7} while considering 5, the accuracy is 85.79%
Using Features(s) {2, 3, 6, 7} while considering 8, the accuracy is 86.69%
Using Features(s) {2, 3, 6, 7} while considering 9, the accuracy is 86.29%
Using Features(s) {2, 3, 6, 7} while considering 10, the accuracy is 85.99%

numLevelsDecreasing = 3
For level 5, the index chosen is 8 with an accuracy of 86.69%
The current feature list is: {2, 3, 6, 7, 8}

Using Features(s) {2, 3, 6, 7, 8} while considering 1, the accuracy is 81.68%
Using Features(s) {2, 3, 6, 7, 8} while considering 4, the accuracy is 82.78%
Using Features(s) {2, 3, 6, 7, 8} while considering 5, the accuracy is 80.58%
Using Features(s) {2, 3, 6, 7, 8} while considering 9, the accuracy is 81.78%
Using Features(s) {2, 3, 6, 7, 8} while considering 10, the accuracy is 81.98%
The highest accuracy features are: {3, 7} with an accuracy of: 96%

Time: 1570 ms with early abandoning after three consecutive decreases.
Time: 1690 ms without early abandoning.

Forward Selection on Large Dataset 8 traceback with early abandonment after 3 levels of accuracy decreasing
We processed large dataset 8 and got the following output:
The dataset has 20 features (not including the class attribute) with 2000 instances.

What size dataset are you planning to use?
1: small
2: large
3: XXXlarge
4: Custom Test Data
5: Pokemon Dataset
2
What dataset file do you want to use?
Enter a valid number from 1-33:
8
Datasets/CS170_large_Data__8.txt
Type the number of the algorithm you want to run.
1. Forward Selection
2. Backwards Elimination
1
Using Features(s) {} while considering 1, the accuracy is 69.48%
Using Features(s) {} while considering 2, the accuracy is 68.63%
Using Features(s) {} while considering 3, the accuracy is 69.08%
Using Features(s) {} while considering 4, the accuracy is 73.24%
Using Features(s) {} while considering 5, the accuracy is 69.03%
Using Features(s) {} while considering 6, the accuracy is 69.43%
Using Features(s) {} while considering 7, the accuracy is 69.53%
Using Features(s) {} while considering 8, the accuracy is 68.48%
Using Features(s) {} while considering 9, the accuracy is 70.04%
Using Features(s) {} while considering 10, the accuracy is 69.38%
Using Features(s) {} while considering 11, the accuracy is 70.14%
Using Features(s) {} while considering 12, the accuracy is 70.79%
Using Features(s) {} while considering 13, the accuracy is 67.68%
Using Features(s) {} while considering 14, the accuracy is 70.24%
Using Features(s) {} while considering 15, the accuracy is 69.98%
Using Features(s) {} while considering 16, the accuracy is 70.19%
Using Features(s) {} while considering 17, the accuracy is 69.73%
Using Features(s) {} while considering 18, the accuracy is 70.94%
Using Features(s) {} while considering 19, the accuracy is 69.78%
Using Features(s) {} while considering 20, the accuracy is 82.89%
For level 1, the index chosen is 20 with an accuracy of 82.89%
The current feature list is: {20}

Using Features(s) {20} while considering 1, the accuracy is 82.99%
Using Features(s) {20} while considering 2, the accuracy is 83.09%
Using Features(s) {20} while considering 3, the accuracy is 82.79%
Using Features(s) {20} while considering 4, the accuracy is 96.85%
Using Features(s) {20} while considering 5, the accuracy is 82.79%
Using Features(s) {20} while considering 6, the accuracy is 82.04%
Using Features(s) {20} while considering 7, the accuracy is 83.19%
Using Features(s) {20} while considering 8, the accuracy is 81.49%
Using Features(s) {20} while considering 9, the accuracy is 85.84%
Using Features(s) {20} while considering 10, the accuracy is 82.19%
Using Features(s) {20} while considering 11, the accuracy is 83.54%

Using Features(s) {20} while considering 12, the accuracy is 83.09%
Using Features(s) {20} while considering 13, the accuracy is 82.24%
Using Features(s) {20} while considering 14, the accuracy is 80.94%
Using Features(s) {20} while considering 15, the accuracy is 83.14%
Using Features(s) {20} while considering 16, the accuracy is 82.69%
Using Features(s) {20} while considering 17, the accuracy is 83.49%
Using Features(s) {20} while considering 18, the accuracy is 82.04%
Using Features(s) {20} while considering 19, the accuracy is 81.89%
For level 2, the index chosen is 4 with an accuracy of 96.85%
The current feature list is: {4, 20}

//Deleted lines to save space

Using Features(s) {4, 8, 9, 17, 20} while considering 1, the accuracy is 85.79%
Using Features(s) {4, 8, 9, 17, 20} while considering 2, the accuracy is 87.04%
Using Features(s) {4, 8, 9, 17, 20} while considering 3, the accuracy is 86.04%
Using Features(s) {4, 8, 9, 17, 20} while considering 5, the accuracy is 85.24%
Using Features(s) {4, 8, 9, 17, 20} while considering 6, the accuracy is 87.04%
Using Features(s) {4, 8, 9, 17, 20} while considering 7, the accuracy is 86.69%
Using Features(s) {4, 8, 9, 17, 20} while considering 10, the accuracy is 86.49%
Using Features(s) {4, 8, 9, 17, 20} while considering 11, the accuracy is 85.74%
Using Features(s) {4, 8, 9, 17, 20} while considering 12, the accuracy is 85.84%
Using Features(s) {4, 8, 9, 17, 20} while considering 13, the accuracy is 85.44%
Using Features(s) {4, 8, 9, 17, 20} while considering 14, the accuracy is 87.09%
Using Features(s) {4, 8, 9, 17, 20} while considering 15, the accuracy is 85.84%
Using Features(s) {4, 8, 9, 17, 20} while considering 16, the accuracy is 86.99%
Using Features(s) {4, 8, 9, 17, 20} while considering 18, the accuracy is 85.74%
Using Features(s) {4, 8, 9, 17, 20} while considering 19, the accuracy is 85.54%
The highest accuracy features are: {20, 4} with an accuracy of: 96.85%

Time: 1828 ms with early abandoning after three consecutive decreases.
Time: 5620 ms without early abandoning.

Forward Selection with XXX Large Dataset 10 with early abandonment after 3 levels of accuracy decreasing
We processed XXXL dataset 10 and got the following output:
The dataset has 80 features (not including the class attribute) with 4000 instances.

What size dataset are you planning to use?
1: small
2: large
3: XXXlarge
4: Custom Test Data
5: Pokemon Dataset
3
What dataset file do you want to use?
Enter a valid number from 1-24:
10
Datasets/CS170_XXXlarge_Data__10.txt
Type the number of the algorithm you want to run.
1. Forward Selection
2. Backwards Elimination
1
Using Features(s) {} while considering 1, the accuracy is 70.22%
Using Features(s) {} while considering 2, the accuracy is 70.94%
Using Features(s) {} while considering 3, the accuracy is 69.79%
Using Features(s) {} while considering 4, the accuracy is 70.14%
Using Features(s) {} while considering 5, the accuracy is 69.07%
Using Features(s) {} while considering 6, the accuracy is 70.27%
Using Features(s) {} while considering 7, the accuracy is 70.52%
Using Features(s) {} while considering 8, the accuracy is 70.44%
Using Features(s) {} while considering 9, the accuracy is 70.44%
Using Features(s) {} while considering 10, the accuracy is 70.64%
Using Features(s) {} while considering 11, the accuracy is 70.89%
Using Features(s) {} while considering 12, the accuracy is 71.02%
Using Features(s) {} while considering 13, the accuracy is 69.29%
Using Features(s) {} while considering 14, the accuracy is 70.99%
Using Features(s) {} while considering 15, the accuracy is 69.87%
Using Features(s) {} while considering 16, the accuracy is 69.34%
Using Features(s) {} while considering 17, the accuracy is 69.54%
Using Features(s) {} while considering 18, the accuracy is 69.14%
Using Features(s) {} while considering 19, the accuracy is 69.59%
Using Features(s) {} while considering 20, the accuracy is 68.22%
Using Features(s) {} while considering 21, the accuracy is 69.84%
Using Features(s) {} while considering 22, the accuracy is 70.14%
Using Features(s) {} while considering 23, the accuracy is 71.12%
Using Features(s) {} while considering 24, the accuracy is 69.39%
Using Features(s) {} while considering 25, the accuracy is 69.47%
Using Features(s) {} while considering 26, the accuracy is 71.09%
Using Features(s) {} while considering 27, the accuracy is 69.87%
Using Features(s) {} while considering 28, the accuracy is 70.09%
Using Features(s) {} while considering 29, the accuracy is 70.24%
Using Features(s) {} while considering 30, the accuracy is 69.57%
Using Features(s) {} while considering 31, the accuracy is 69.02%
Using Features(s) {} while considering 32, the accuracy is 71.04%
Using Features(s) {} while considering 33, the accuracy is 70.52%
Using Features(s) {} while considering 34, the accuracy is 68.99%

Using Features(s) {} while considering 35, the accuracy is 69.94%
Using Features(s) {} while considering 36, the accuracy is 69.87%
Using Features(s) {} while considering 37, the accuracy is 69.82%
Using Features(s) {} while considering 38, the accuracy is 69.94%
Using Features(s) {} while considering 39, the accuracy is 70.24%
Using Features(s) {} while considering 40, the accuracy is 70.34%
Using Features(s) {} while considering 41, the accuracy is 69.59%
Using Features(s) {} while considering 42, the accuracy is 69.69%
Using Features(s) {} while considering 43, the accuracy is 71.19%
Using Features(s) {} while considering 44, the accuracy is 70.09%
Using Features(s) {} while considering 45, the accuracy is 71.12%
Using Features(s) {} while considering 46, the accuracy is 69.67%
Using Features(s) {} while considering 47, the accuracy is 69.07%
Using Features(s) {} while considering 48, the accuracy is 70.39%
Using Features(s) {} while considering 49, the accuracy is 69.69%
Using Features(s) {} while considering 50, the accuracy is 70.47%
Using Features(s) {} while considering 51, the accuracy is 69.47%
Using Features(s) {} while considering 52, the accuracy is 70.14%
Using Features(s) {} while considering 53, the accuracy is 70.72%
Using Features(s) {} while considering 54, the accuracy is 69.42%
Using Features(s) {} while considering 55, the accuracy is 70.39%
Using Features(s) {} while considering 56, the accuracy is 70.82%
Using Features(s) {} while considering 57, the accuracy is 71.29%
Using Features(s) {} while considering 58, the accuracy is 73.74%
Using Features(s) {} while considering 59, the accuracy is 70.74%
Using Features(s) {} while considering 60, the accuracy is 69.89%
Using Features(s) {} while considering 61, the accuracy is 70.04%
Using Features(s) {} while considering 62, the accuracy is 70.99%
Using Features(s) {} while considering 63, the accuracy is 70.92%
Using Features(s) {} while considering 64, the accuracy is 69.12%
Using Features(s) {} while considering 65, the accuracy is 70.04%
Using Features(s) {} while considering 66, the accuracy is 70.52%
Using Features(s) {} while considering 67, the accuracy is 71.47%
Using Features(s) {} while considering 68, the accuracy is 70.32%
Using Features(s) {} while considering 69, the accuracy is 69.17%
Using Features(s) {} while considering 70, the accuracy is 70.37%
Using Features(s) {} while considering 71, the accuracy is 71.27%
Using Features(s) {} while considering 72, the accuracy is 70.49%
Using Features(s) {} while considering 73, the accuracy is 70.47%
Using Features(s) {} while considering 74, the accuracy is 68.82%
Using Features(s) {} while considering 75, the accuracy is 70.42%
Using Features(s) {} while considering 76, the accuracy is 69.62%
Using Features(s) {} while considering 77, the accuracy is 70.02%
Using Features(s) {} while considering 78, the accuracy is 85.45%
Using Features(s) {} while considering 79, the accuracy is 69.94%
Using Features(s) {} while considering 80, the accuracy is 69.42%
For level 1, the index chosen is 78 with an accuracy of 85.45%
The current feature list is: {78}

//Deleted lines to save space

Using Features(s) {26, 45, 58, 75, 78} while considering 60, the accuracy is 87.17%
Using Features(s) {26, 45, 58, 75, 78} while considering 61, the accuracy is 87.45%

Using Features(s) {26, 45, 58, 75, 78} while considering 62, the accuracy is 88.05%
Using Features(s) {26, 45, 58, 75, 78} while considering 63, the accuracy is 87.67%
Using Features(s) {26, 45, 58, 75, 78} while considering 64, the accuracy is 87.37%
Using Features(s) {26, 45, 58, 75, 78} while considering 65, the accuracy is 87.37%
Using Features(s) {26, 45, 58, 75, 78} while considering 66, the accuracy is 87.82%
Using Features(s) {26, 45, 58, 75, 78} while considering 67, the accuracy is 88.1%
Using Features(s) {26, 45, 58, 75, 78} while considering 68, the accuracy is 87.92%
Using Features(s) {26, 45, 58, 75, 78} while considering 69, the accuracy is 89%
Using Features(s) {26, 45, 58, 75, 78} while considering 70, the accuracy is 87.6%
Using Features(s) {26, 45, 58, 75, 78} while considering 71, the accuracy is 87.45%
Using Features(s) {26, 45, 58, 75, 78} while considering 72, the accuracy is 88.25%
Using Features(s) {26, 45, 58, 75, 78} while considering 73, the accuracy is 87.15%
Using Features(s) {26, 45, 58, 75, 78} while considering 74, the accuracy is 88.1%
Using Features(s) {26, 45, 58, 75, 78} while considering 76, the accuracy is 87.45%
Using Features(s) {26, 45, 58, 75, 78} while considering 77, the accuracy is 87.52%
Using Features(s) {26, 45, 58, 75, 78} while considering 79, the accuracy is 87.82%
Using Features(s) {26, 45, 58, 75, 78} while considering 80, the accuracy is 88.45%
The highest accuracy features are: {78, 58} with an accuracy of: 97.47%

Time: 51690 ms with early abandoning after three consecutive decreases.
Time: 1168252 ms without early abandoning.

Forward Selection for the Pokemon Dataset with early abandonment after 3 levels of decreasing accuracy
We processed Pokemon dataset and got the following output:
The dataset has 29 features (not including the class attribute) with 800 instances.

What size dataset are you planning to use?
1: small
2: large
3: XXXlarge
4: Custom Test Data
5: Pokemon Dataset
Using Pokemon Dataset...
pokemon3.txt
Type the number of the algorithm you want to run.
1. Forward Selection
2. Backwards Elimination
Using Features(s) {} while considering 1, the accuracy is 11.5%
Using Features(s) {} while considering 2, the accuracy is 16.25%
Using Features(s) {} while considering 3, the accuracy is 14.88%
Using Features(s) {} while considering 4, the accuracy is 19.75%
Using Features(s) {} while considering 5, the accuracy is 18.88%
Using Features(s) {} while considering 6, the accuracy is 24.38%
Using Features(s) {} while considering 7, the accuracy is 30.12%
Using Features(s) {} while considering 8, the accuracy is 20.5%
Using Features(s) {} while considering 9, the accuracy is 32%
Using Features(s) {} while considering 10, the accuracy is 29%
Using Features(s) {} while considering 11, the accuracy is 13%
Using Features(s) {} while considering 12, the accuracy is 15.75%
Using Features(s) {} while considering 13, the accuracy is 18.25%
Using Features(s) {} while considering 14, the accuracy is 7.625%
Using Features(s) {} while considering 15, the accuracy is 18.5%
Using Features(s) {} while considering 16, the accuracy is 16.75%
Using Features(s) {} while considering 17, the accuracy is 17.88%
Using Features(s) {} while considering 18, the accuracy is 24%
Using Features(s) {} while considering 19, the accuracy is 10.5%
Using Features(s) {} while considering 20, the accuracy is 19.38%
Using Features(s) {} while considering 21, the accuracy is 11.75%
Using Features(s) {} while considering 22, the accuracy is 10.62%
Using Features(s) {} while considering 23, the accuracy is 9.25%
Using Features(s) {} while considering 24, the accuracy is 13.25%
Using Features(s) {} while considering 25, the accuracy is 10.12%
Using Features(s) {} while considering 26, the accuracy is 9.5%
Using Features(s) {} while considering 27, the accuracy is 7.875%
Using Features(s) {} while considering 28, the accuracy is 11.88%
Using Features(s) {} while considering 29, the accuracy is 8.75%
For level 1, the index chosen is 9 with an accuracy of 32%
The current feature list is: {9}

Using Features(s) {9} while considering 1, the accuracy is 36.75%
Using Features(s) {9} while considering 2, the accuracy is 33.25%
Using Features(s) {9} while considering 3, the accuracy is 38%
Using Features(s) {9} while considering 4, the accuracy is 41.5%
Using Features(s) {9} while considering 5, the accuracy is 41.12%
Using Features(s) {9} while considering 6, the accuracy is 41.5%

Using Features(s) {9} while considering 7, the accuracy is 42.62%
Using Features(s) {9} while considering 8, the accuracy is 42.25%
Using Features(s) {9} while considering 10, the accuracy is 46%
Using Features(s) {9} while considering 11, the accuracy is 36.75%
Using Features(s) {9} while considering 12, the accuracy is 37.25%
Using Features(s) {9} while considering 13, the accuracy is 40%
Using Features(s) {9} while considering 14, the accuracy is 33.5%
Using Features(s) {9} while considering 15, the accuracy is 34.88%
Using Features(s) {9} while considering 16, the accuracy is 36.75%
Using Features(s) {9} while considering 17, the accuracy is 39.88%
Using Features(s) {9} while considering 18, the accuracy is 46.25%
Using Features(s) {9} while considering 19, the accuracy is 25.12%
Using Features(s) {9} while considering 20, the accuracy is 39.12%
Using Features(s) {9} while considering 21, the accuracy is 26.12%
Using Features(s) {9} while considering 22, the accuracy is 27.12%
Using Features(s) {9} while considering 23, the accuracy is 24.75%
Using Features(s) {9} while considering 24, the accuracy is 31.25%
Using Features(s) {9} while considering 25, the accuracy is 23.12%
Using Features(s) {9} while considering 26, the accuracy is 23.75%
Using Features(s) {9} while considering 27, the accuracy is 21.88%
Using Features(s) {9} while considering 28, the accuracy is 34.38%
Using Features(s) {9} while considering 29, the accuracy is 20.62%
For level 2, the index chosen is 18 with an accuracy of 46.25%
The current feature list is: {9, 18}

//Deleted lines to save space

numLevelsDecreasing = 3
For level 14, the index chosen is 15 with an accuracy of 95.12%
The current feature list is: {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28}

Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 4, the accuracy is 94.88%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 5, the accuracy is 95%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 6, the accuracy is 95.12%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 10, the accuracy is 95%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 11, the accuracy is 94.88%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 19, the accuracy is 41.5%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 20, the accuracy is 86%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 21, the accuracy is 46.75%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 22, the accuracy is 93%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 23, the accuracy is 52.5%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 24, the accuracy is 88.62%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 25, the accuracy is 44%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 26, the accuracy is 47.5%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 27, the accuracy is 45%
Using Features(s) {1, 2, 3, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 28} while considering 29, the accuracy is 46.62%
The highest accuracy features are: {1, 7, 8, 9, 12, 14, 16, 17, 18, 28, 3} with an accuracy of: 95.12%

Time: 1083 with early abandoning after three consecutive decreases.
Time: 1916 ms without early abandoning.

Code

```
#include <bits/stdc++.h>
#include <typeinfo>
using namespace std;
using namespace chrono;

double highestAccuracy = 0;
vector<int> highestAccuracyFeatures;
int maxLevelDec = 500;

void printFile(vector<vector<double>> &data){
   for(int i = 0; i < data.size(); i++){
      for(int j = 0; j < data[i].size(); j++){
         cout << data[i][j] << " ";
      }
      cout << endl;
   }
}

void readFile(vector<vector<double>> &data, string &fileName){
   ios_base::sync_with_stdio(false);
   cin.tie(0);
   ifstream file (fileName);
   string curRow;
   double curValue;

   while(getline(file, curRow)){
      stringstream row(curRow);
      vector<double> temp;
      while(row >> curValue){
         temp.push_back(curValue);
      }
      data.push_back(temp);
   }
   file.close();
}

bool isDupe(vector<int> curFeatures,  int index){
   for(int i = 0; i < curFeatures.size(); i++){
      if(curFeatures[i] == index) return true;
   }
   return false;
   // int originalSize = curFeatures.size();
   // curFeatures.insert(index);
   // return originalSize == curFeatures.size();
}

void printFeatures(vector<int> features){
   cout << "{";
   if(features.size() > 0){
      cout << features[0];
   }
```

```
    for(int i = 1; i < features.size(); i++){
        cout << ", " << features[i];
    }
    cout << "}";
}

int nearestNeighbor(vector<vector<double>> &data, int rowNum, vector<int> curSet){
    //cout << "Data Size: " << data.size() << endl;
    //cout << "features: " << curSet.size() << endl;
    double closestDistance = DBL_MAX;
    int indexOfClosest = -1;
    double sum = 0;
    double curDist = 0;
    //int len = data[0].size();
    for(int i = 0; i < data.size(); i++){
        double sum = 0;
        if(i == rowNum) continue;
        //cout << i << endl;
        for(int j  = 0; j < curSet.size(); j++){
            sum += pow(data[i][curSet[j]] - data[rowNum][curSet[j]], 2);
            //cout << "sum " << sum << endl;
        }

        curDist = sqrt(sum);
        //cout << "Distance between rowNum: " << rowNum << " and i " << i << " is " << curDist  << "    " <<
closestDistance << endl;
        if(curDist < closestDistance){
            closestDistance = curDist;
            indexOfClosest = i;
        }
        //cout << " index: " << i << " distance: " << curDist << endl;
    }
    return indexOfClosest;
}

double leaveOneOut(vector<vector<double>> &data, vector<int> curSet){
    //cout << "leave one out" << endl;
    int numCorrectlyClassified = 0;
    for(int i = 0; i < data.size(); i++){
        //cout << i << endl;
        int neighborIndex = nearestNeighbor(data, i, curSet);
        //cout << "i "  << i << " neighbor " << neighborIndex << endl;
        if(data[neighborIndex][0] == data[i][0]){
            //cout << "i "  << i << " neighbor " << neighborIndex << endl;
            //cout << "if data[neighborIndex][0] "  << data[10][0] << endl;    // << "  data[i][0] " << data[i][0] << endl;
            numCorrectlyClassified++;
        }
    }
    //cout << "Correctly guessed: " << (double)numCorrectlyClassified/(data.size()-1) * 100 << endl;
    //cout << "numClassified: " << (double)numCorrectlyClassified << endl;
    //cout << "data size: " << data.size() << endl;
    return (double)numCorrectlyClassified/(data.size()-1) * 100;
}
```

```
void forwardSelection(vector<vector<double>> &data){
    vector<int> indexOfFeaturesPicked = {};
    highestAccuracy = 0;
    double curBestAccuracy = 0;
    double prevBestAccuracy = 0;
    int numLevelsDecreasing = 0;

    for(int i = 1; i < data[0].size(); i++){
        //cout << i << " " << data[0].size() << endl;
        vector<int> indexOfFeaturesConsidering;
        prevBestAccuracy = curBestAccuracy;     //save this accuracy for next level
        //cout << "\nprevBestAccuracy = " << prevBestAccuracy << "\n";
        curBestAccuracy = 0;
        double curBestAccuracyFeatureIndex;

        for(int j = 1; j < data[0].size(); j++){
            if(isDupe(indexOfFeaturesPicked, j)) continue;

            indexOfFeaturesConsidering = indexOfFeaturesPicked;
            indexOfFeaturesConsidering.push_back(j);
            // printFeatures(indexOfFeaturesConsidering);
            // cout << endl;

            double accuracy = leaveOneOut(data, indexOfFeaturesConsidering);
            cout << "Using Features(s) ";
            printFeatures(indexOfFeaturesPicked);
            cout << " while considering " << j << ", the accuracy is " << accuracy << "%" << endl;
            if(accuracy > curBestAccuracy){
                curBestAccuracy = accuracy;
                curBestAccuracyFeatureIndex = j;
                if(curBestAccuracy > highestAccuracy){
                    highestAccuracy = curBestAccuracy;
                    highestAccuracyFeatures = indexOfFeaturesConsidering;
                }
            }
        }
        if(curBestAccuracy > prevBestAccuracy){
            numLevelsDecreasing = 0;
        }
        else{                           //if accuracy decreased or stayed the same
            if(numLevelsDecreasing == maxLevelDec){
                break;
            }
            numLevelsDecreasing++;
            cout << "\nnumLevelsDecreasing = " << numLevelsDecreasing << "\n";
        }
        indexOfFeaturesPicked.push_back(curBestAccuracyFeatureIndex);
        sort(indexOfFeaturesPicked.begin(), indexOfFeaturesPicked.end());
        cout << "For level " << i << ", the index chosen is " << curBestAccuracyFeatureIndex << " with an accuracy of "
<< curBestAccuracy << "%" << endl;
        cout << "The current feature list is: ";
        printFeatures(indexOfFeaturesPicked);
```

```
        cout << endl << endl;
    }
}

void backwardElimination(vector<vector<double>> &data){
    vector<int> indexOfAllFeaturesPicked;
    int len = data[0].size();
    highestAccuracy = 0;
    vector<int> indexOfFeaturesToBeKept;
    double curBestAccuracy = 0;
    double prevBestAccuracy = 0;
    int numLevelsDecreasing = 0;
    double accuracy = 0;

    for(int i = 1; i < len; i++){
        indexOfAllFeaturesPicked.push_back(i);
    }

    for(int i = 1; i < data[0].size(); i++){
        // cout << "i: " << i << endl;
        indexOfFeaturesToBeKept.clear();
        prevBestAccuracy = curBestAccuracy;     //save this accuracy for next level
        cout << "\nprevBestAccuracy = " << prevBestAccuracy << "\n";
        curBestAccuracy = 0;
        double curFeatureIndexLoweringAccuracy;

        for(int j = 1; j < data[0].size(); j++){    //represents a level
            // cout << "j " << j << endl;
            //go through the features columns and choose the subset from that is in indexOfFeaturesPicked that gives
you the best accuracy.
            if(!isDupe(indexOfAllFeaturesPicked, j)) continue;

            indexOfFeaturesToBeKept = indexOfAllFeaturesPicked; //start of with the same vector
            indexOfFeaturesToBeKept.erase(remove(indexOfFeaturesToBeKept.begin(), indexOfFeaturesToBeKept.end(),
j), indexOfFeaturesToBeKept.end());
            //printFeatures(indexOfFeaturesToBeKept);
            // cout << endl;

            accuracy = leaveOneOut(data, indexOfFeaturesToBeKept);          //seg fault is because of this line
            //accuracy = j;
            cout << "Using Features(s) ";
            printFeatures(indexOfAllFeaturesPicked);
            cout << " while considering removing " << j << ", the accuracy is " << accuracy << "%" << endl;
            if(accuracy > curBestAccuracy){     //update accuracy
                curBestAccuracy = accuracy;
                curFeatureIndexLoweringAccuracy = j;
                if(curBestAccuracy > highestAccuracy){
                    highestAccuracy = curBestAccuracy;
                    highestAccuracyFeatures = indexOfFeaturesToBeKept;
                }
            }
        }
        if(curBestAccuracy > prevBestAccuracy){
```

```
            numLevelsDecreasing = 0;
        }
        else{                           //if accuracy decreased or stayed the same
            if(numLevelsDecreasing == maxLevelDec){
                break;
            }
            numLevelsDecreasing++;
            cout << "\nnumLevelsDecreasing = " << numLevelsDecreasing << "\n";
        }
        indexOfAllFeaturesPicked.erase(remove(indexOfAllFeaturesPicked.begin(), indexOfAllFeaturesPicked.end(),
curFeatureIndexLoweringAccuracy), indexOfAllFeaturesPicked.end()); ;
        sort(indexOfAllFeaturesPicked.begin(), indexOfAllFeaturesPicked.end());
        cout << "For level " << i << ", the index removed is " << curFeatureIndexLoweringAccuracy << " with an new
accuracy of " << curBestAccuracy << "%" << endl;
        cout << "The current feature list is: ";
        printFeatures(indexOfAllFeaturesPicked);
        cout << endl << endl;
    }
}

bool ifSameClassLabel(double predclass, vector<vector<double>> &data, double instnum){      //check if predicted
class is the same as the actual class stored in the first column of data
    return fabs(predclass - data[instnum][0]) < 0.01;
}

int main(int argc, char* argv[]){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout << setprecision(4);
    int dataSize = 0;
    int dataFileNumber = 0;
    string fileName;
    vector<vector<double>> data;
    vector<string> fileSizeName = {"small", "large", "XXXlarge"};

    //Get data file size + data validation
    cout << "What size dataset are you planning to use?" << endl;
    cout << "1: small" << endl;
    cout << "2: large" << endl;
    cout << "3: XXXlarge" << endl;
    cout << "4: Custom Test Data" << endl;
    cout << "5: Pokemon Dataset" << endl;

    cin >> dataSize;
    while(dataSize < 1 || dataSize > 5){
        cout << "That is not a valid input, please try again." << endl;
        cin >> dataSize;
    }

    if(dataSize != 4){
        //Get File number + data validation
        switch(dataSize){
            case 1:
```

```
            do{
                cout << "What dataset file do you want to use?" << endl;
                cout << "Enter a valid number from 1-33: " << endl;
                cin >> dataFileNumber;
            } while(dataFileNumber < 1 || dataFileNumber > 33);
            break;
        case 2:
            do{
                cout << "What dataset file do you want to use?" << endl;
                cout << "Enter a valid number from 1-33: " << endl;
                cin >> dataFileNumber;
            } while(dataFileNumber < 1 || dataFileNumber > 33);
            break;
        case 3:
            do{
                cout << "What dataset file do you want to use?" << endl;
                cout << "Enter a valid number from 1-24: " << endl;
                cin >> dataFileNumber;
            } while(dataFileNumber < 1 || dataFileNumber > 24);
            break;
        case 5:
            cout << "Using Pokemon Dataset..." << endl;
            break;
        default:
            break;
    }

    //Testing File Name
    if(dataSize != 5){
        fileName = "Datasets/CS170_" + fileSizeName[dataSize-1] + "_Data__" + to_string(dataFileNumber) + ".txt";
    }
    else{
        fileName = "pokemon.txt";
    }
    cout << fileName << endl;

}
else{
    fileName = "Datasets/TestData.txt";
}

//Testing readFile
readFile(data, fileName);
//printFile(data);

//Testing NearestNeighbor
// vector<double> testDataPoint = {2, 1, 2, 3, 4};
// int nearestNeighborindex = nearestNeighbor(data, testDataPoint);
// cout << "Nearest Neighbor Testing Index: " << nearestNeighborindex << endl;

cout << "Type the number of the algorithm you want to run." << endl;
cout << "1. Forward Selection" << endl;
cout << "2. Backwards Elimination" << endl;
```

```
    int algo = 0;
    cin >> algo;
    while(algo != 1 && algo != 2){
        cout << "That was not a valid input, please try again: " << endl;
        cin >> algo;
    }

    auto start = high_resolution_clock::now();
    switch(algo){
        case 1:
            forwardSelection(data);
            break;
        case 2:
            backwardElimination(data);
            break;
        default:
            break;
    }
    auto stop = high_resolution_clock::now();

    cout << "The highest accuracy features are: ";
    printFeatures(highestAccuracyFeatures);
    cout << " with an accuracy of: " << highestAccuracy << "%" << endl;
    cout << "Time: " << duration_cast<milliseconds>(stop - start).count() << " milliseconds. " << endl;
    return 0;

    //vector<int> featSubset;
    //featSubset.push_back(1);
    //featSubset.push_back(2);
    //int lOO = leaveOneOutValidator(data, featSubset);
    //cout << "\nifSameClassLabel = " << ifSameClassLabel(1, data, 0) << '\n';
    //cout << "\nlOO accuracy = " << lOO << '\n';
    //return 0;
}
```