Group 10: Project 2 Report

Group 10 Contributors

Contributor	SID	SECTION
Riya Patel	862172362	001
Leo Yoo	862166955	002
Ynah Niyel Novilla	862236890	002
Mrinisha Adhikari	862309931	002

Solution

DataSet	Best Feature Set	Accuracy

Small Number: 10	Forward Selection = {5,2}
	Backward Elimination = {5,2}
	Recursive Feature Elimination = {2,5}
		0.94
		0.94
		0.94

Large Number: 10	Forward Selection = {12,29}
	Backward Elimination = {12,36}
	Recursive Feature Elimination = {12,36}
		0.976
		0.864
		0.864

In completing this project, We consulted following resources:

- Lecture Slides
- Discussion for Project Part 2
- ML | Multiple Linear Regression (Backward Elimination Technique) - GeeksforGeeks
- REF | Recursive Elimination Technique (for custom algorithm) - GeeksforGeeks
- TA
- How does the Recursive Feature elimination(RFE) works and how it is different from Backward elimination? - techniques - Data Science, Analytics and Big Data discussions (analyticsvidhya.com) (then another hyperlink within this)

- https://www.analyticsvidhya.com/blog/2021/04/backward-feature-elimination-and-its-implementation/
- Reading a ASCII text file in C++ -TutorialsPoint

Contribution of each student in the group:

- Riya Patel: worked on the backward elimination, output debugging, report analysis, conclusion, algorithm.
- Leo Yoo: worked on getting the program set up (overall code along with forward selection and backward elimination), worked with Ynah to implement the time feature for each step and debugging. Editing report
- Ynah Niyel Novilla: Worked on plots for report analysis and dataset details, implemented time spent measurement for each step with Leo, and algorithm and output debugging.
- Mrinisha Adhikari: for code- implemented functions (ZNormalization, Accuracy, NN-Classifier, leave-one-out validator), custom algorithm 3, and UI menu debugging. for report- parts of analysis for experiment 1,2 and the screenshots of each dataset

I. Introduction

This project is a feature selection program that trains and tests a given dataset with a set class and features using Greedy Forward Selection and Backwards Elimination feature selection, Nearest Neighbor Classification, and Leave-One-Out-Validation. The objective is to choose characteristics that will produce the best accurate classification.

II. Challenges

The first stage was not too tough to think through, but the implementation of the validator and the evaluator were really challenging. Each level was challenging to integrate into the current application. It was particularly challenging to figure out how to turn the existing data instances into training and testing data. Thinking about how everything was going to fit together was the hardest obstacle of this project. It was challenging and complex to determine which classes and files should contain which functions, the general data flow, and the number of functions and variables to code and keep track of.

III. Code Design

The code begins by including necessary libraries and defining a `Data` structure, which holds the class label and a vector of features. The `Classifier` class contains a vector of `Data` instances for training. The `Train` function sets the training instances for the classifier, and the `Test` function performs the classification on a given test instance using the nearest neighbor algorithm based on Euclidean distance.

The code also includes utility functions such as `loadData`, which reads data from a file into a vector of `Data` instances, and `SelectFeatures`, which selects specific features from the dataset based on given indices. Additionally, there are functions for data normalization (`ZnormalizeData`) and for converting features to a string representation (`featuresToString`).

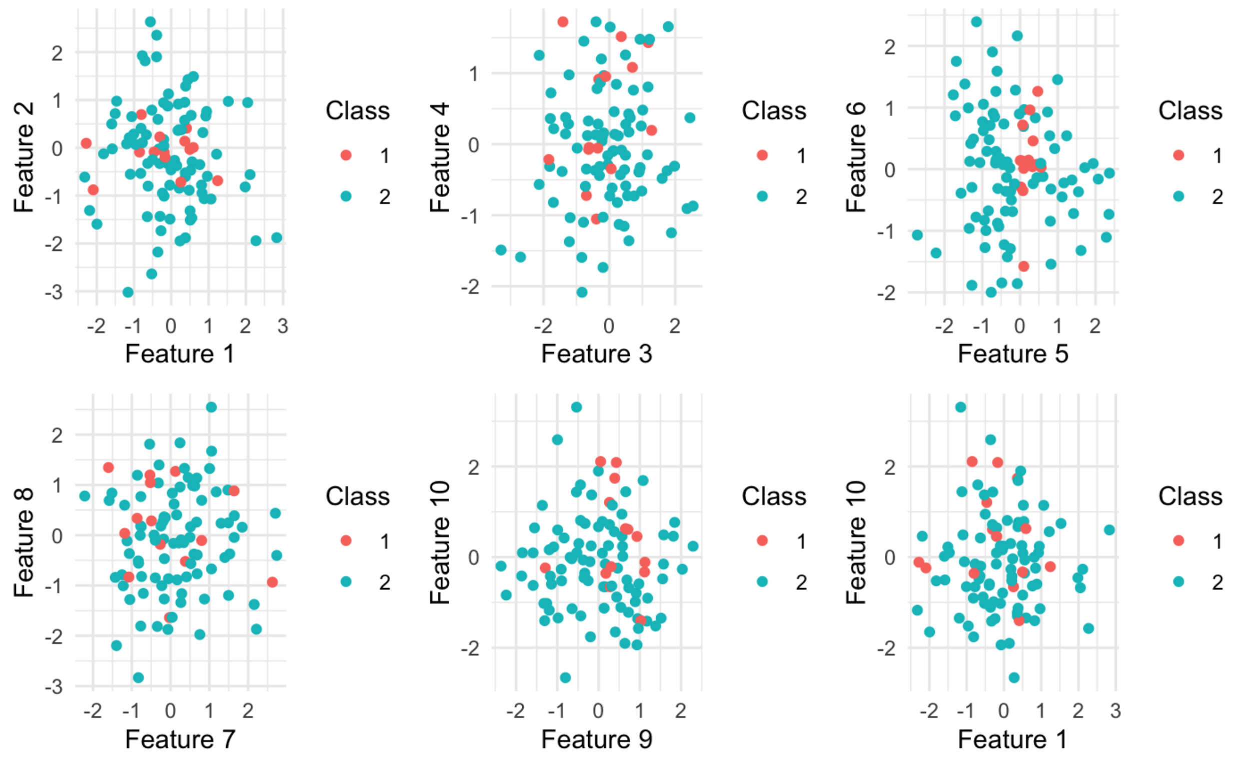
The main part of the code involves feature selection algorithms: `ForwardSelection` and `BackwardElimination`. These algorithms iterate through the features of the dataset, adding or removing them one by one, and calculate the accuracy using leave-one-out cross-validation. The selected features are stored in a vector, and the algorithm outputs the feature set with the highest accuracy.

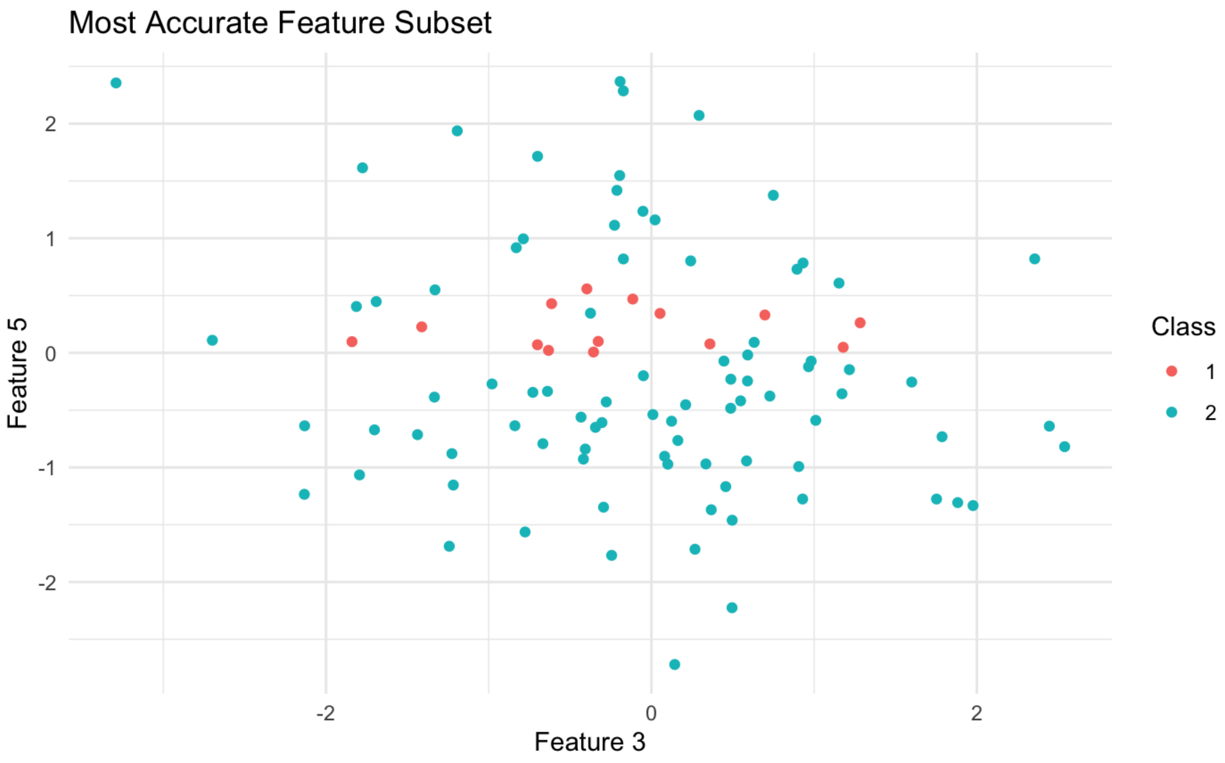
The `main` function serves as the entry point of the program. It prompts the user to input the name of the file containing the dataset, loads the data, normalizes it, and allows the user to choose between forward selection, backward elimination, or a custom algorithm. The selected algorithm is executed, and the resulting feature set and accuracy are displayed.

IV. Dataset Details

General Small Dataset: 10 features and 100 instances
General Large Dataset: 40 features and 1000 instances
Small Dataset #10: 10 features and 100 instances
Large Dataset #10: 40 features and 1000 instances

General Small Dataset Plotted





We have plotted seven different scatter plots with each X and Y axis being a different feature. The last plot is the most accurate feature-set output for the general small dataset using forward selection ({5, 3}). When comparing the most accurate feature-set plot to the six others, we can see that it is the only plot that has the most distinguishable clusters of class one and two, which would make sense considering that is the most accurate subset that was found by our algorithm.

V. Algorithms

1. Forward Selection

Given a set of features, assess each feature's accuracy and select the one that is the most accurate to expand (greedy). Using the existing features and the most accurate one, generate feature subsets and expand the most accurate subset. Continue doing this with bigger subsets until you find the feature subset with the highest accuracy (either the parent subset has greater accuracy than all the kids or all the features in the subset have the highest accuracy).

2. Backward Elimination

Examine the accuracy of each feature in a big feature subset given a collection of features. Remove one feature from the large subset and divide it into smaller subsets of the current feature subset size, which is 1, each with one feature removed. Continue doing this with smaller subsets until you find the feature subset with the highest accuracy (either the parent subset has a greater accuracy than all the kids, or all the features in the subset have the highest accuracy).

3. Recursive Feature Elimination

RFE is a feature selection that eliminates features from an initial feature subset iteratively in order to discover the optimal feature subset. It works by iteratively training the model on the current feature subset, ranking the relevance of each feature, and deleting the least important features. There may not be a specific term for a recursive forward selection algorithm.

VI. Analysis

Experiment 1: Comparing Forward Selection vs Backward Elimination

- *No Feature Selection vs. with Feature Selection*

No feature selection deals with the use of a “default rate”. A loop is used to count the number of instances and is divided by the most common class in the dataset. Essentially, this works by calculating a given portion of the most common class in the dataset, regardless of features. This leads to the inclusion of potential “useless data/noise”, and a low accuracy value. By including a feature selection, the machine/program learns only through relevant features. This makes it easier/faster to train and results in a much higher accuracy. This was observed in both our small and large datasets.

- *Comparisons of Feature Sets and Accuracies*

The feature set for the forward selection is based on adding features that result in the highest accuracy. It starts the search with each individual feature, and creates appropriate subsets by building on the highest calculated accuracy. This is known as the greedy search method, and tends to guarantee a high accuracy. This was observed in our large data set, asforward search took a greedy approach of first selecting feature 12, then 29, for a high accuracy of 97.6%

The feature set for the backward elimination is based on removing the feature that contributes the lowest accuracy. It starts the search with all initial features, and then tests through each level by leaving out a feature. A subset with the highest accuracy continues the search. This was observed in both our datasets. Taking our large dataset for reference, the first search level involves a search of all features. Then continues by a leave-out method, eventually recording a subset with features 12 and 36, for a accuracy of 86.4%

We noticed that since the backward begins with all features first, all data in a given dataset is considered. So, the features it selects tend to be more varied. We also noticed that the decrease in accuracy in the backward, is potentially due to informative features being removed first, as the search commences and more less informative features are removed, the accuracy slowly improves.

- *Pros and Cons of Algorithms*

<u>Backwards Elimination:</u>

Pros: Backwards elimination begins with a model that includes all accessible features and then removes the least significant features iteratively. When working with a large number of characteristics, this strategy can be computationally efficient. Backwards elimination is a simple and intuitive method for selecting features. It helps to simplify the model and reduce complexity by gradually removing features. It enables a systematic assessment of the significance of each feature by observing the impact on the model's performance when features are removed.

Cons: Backward elimination may remove elements that have a minor impact on the overall model but could contribute significantly when paired with other features. This may result in underutilization of valuable information and an increased risk of overfitting. The technique presupposes that the features are independent of one another, which may not always be true in real-world circumstances. If two or more aspects are connected, the elimination procedure may eliminate one of the correlated traits without taking the whole relationship into account. Time complexity also plays an important part in the backwards algorithm.

<u>Forward Selection:</u>
Pros: Forward selection provides for more flexible feature space exploration by slowly adding features to the model. It begins with an empty model and selects the best-performing feature at each stage, making it appropriate for scenarios in which the optimal subset of features is unknown in advance. Forward selection has the potential to determine the optimal subset of characteristics that enhances the model's performance by iteratively adding features. catch of feature interactions: Because forward selection adds features one at a time, it might catch potential interactions between features that might not be obvious when looking at them individually.

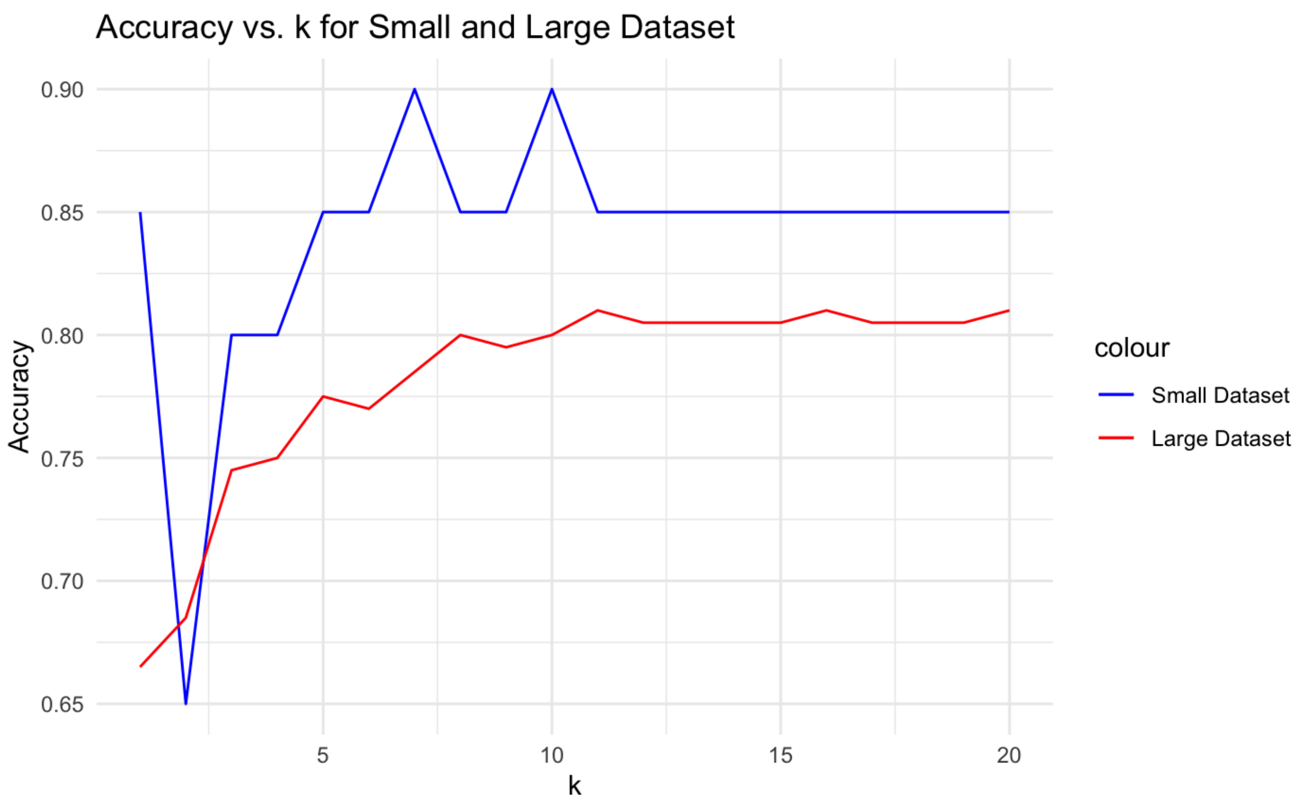
Cons: Computational complexity: Because forward selection includes evaluating many combinations of features, it can be computationally expensive, especially when working with a large number of features. Forward selection is susceptible to local optima due to its stepwise structure, which might result in a suboptimal feature subset. Sensitivity to initial feature: In forward selection, the initial feature selection can have a substantial impact on the end result. Starting with a different initial feature may result in the selection of different subsets.

Experiment 2: The Effect of Normalization

We used Z-Score normalization for our program. Normalization deals with making sure no given the value of a certain feature dominates the entire model at hand. Z-score in particular is used by calculating the mean and standard deviation for each feature in the dataset. Then, the mean is subtracted from each feature, so that the overall distribution has a mean of 0. The standard deviation is used to scale the distribution to a value of 1.

The accuracy with unnormalized data before Z-normalization was low while the features in the subsets were extremely high. The features selected were skewed together, and unvaried when each search explored a level. However, after normalizing the data we have been able to increase our accuracies significantly while maintaining a low amount of features in our subset.

Experiment 3: The Effect of the Number of Neighbors (k)



Here we have plotted the accuracies against each value of K for both the small and large dataset. The line that corresponds to the small dataset fluctuates heavily at K < 10, which is as to be expected for a dataset that does not have too many instances. On the other hand, the large dataset's line gradually increases in accuracy as K increases, and similar to the small dataset, we can see that at K = 11, the accuracy of both datasets level out. This would lead us to conclude that using K = 11 would be the best choice, since it leads to the best accuracy for various sizes of datasets, and it would thus guarantee us that we will not overfit our dataset.

VII. Conclusion

Overall, forward selection should be used to choose characteristics in simpler everyday scenarios rather than backward elimination. Running forward selection is much faster, in our code it runs at a time complexity of 1 minute maximum. The backward algorithm takes 3-10 minutes depending on the processor and IDE the code is run on. As the dataset becomes bigger and the number of features becomes large, using the backward algorithm would require the processor to compute the accuracy of a large number of features at the very start which means that it is incredibly slow in comparison to running forward selection on the same dataset. Additionally, forward selection typically gave us the largest accuracy, which would lead us to assume that this algorithm would be the best for any dataset given.

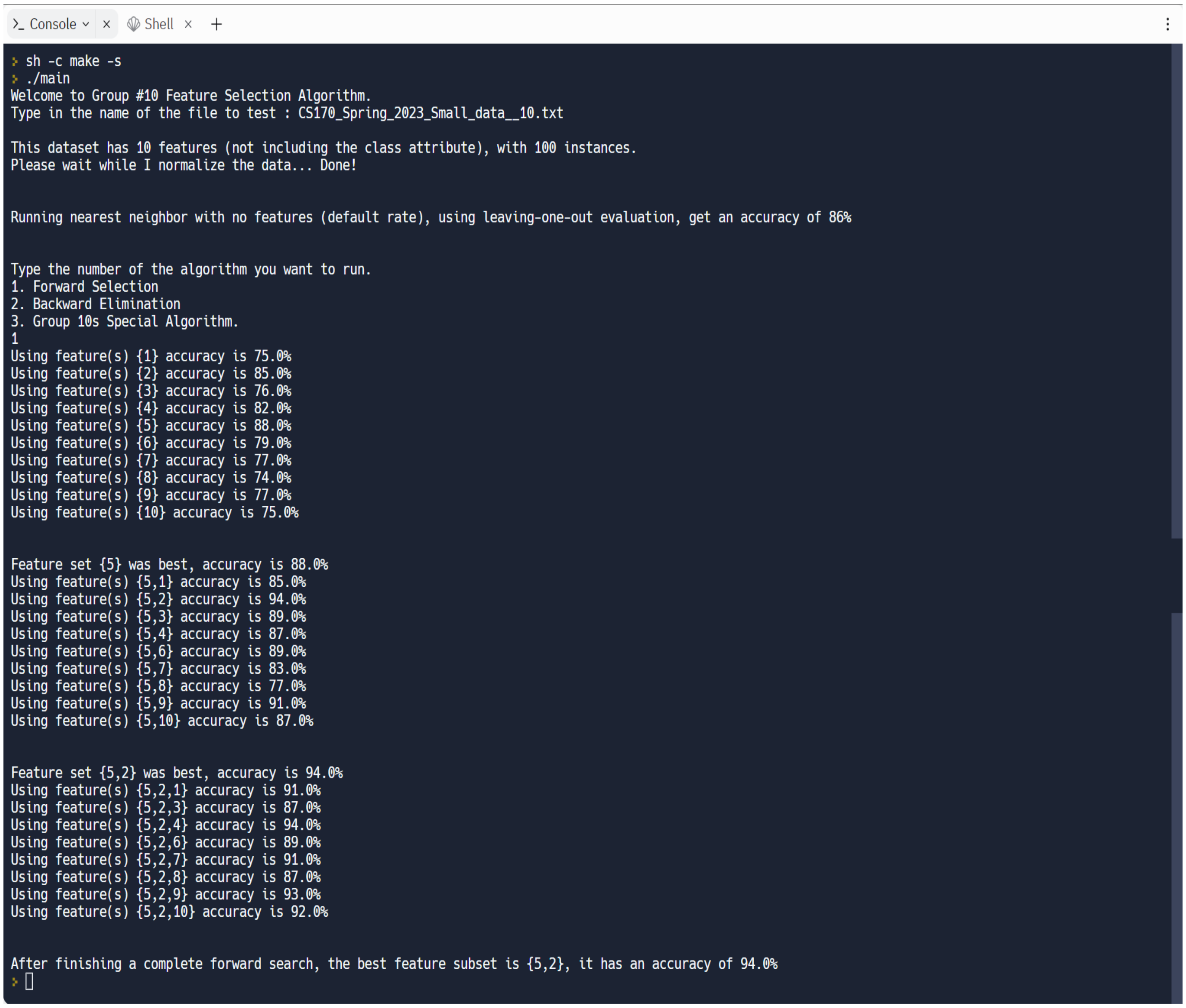
Based on the analysis of our data, the forward selection algorithm gives very high accuracy and can be seen to be quite consistent and wouldn't really need much adjustments. But in the case of backwards elimination and recursive feature elimination there can definitely be more done to improve that accuracy so that it can reach the accuracy ceiling similar to that of the forward selection. Possible improvements can be made by things such as: by checking for overfitting and optimizing the algorithm.

The nearest neighbor algorithm could be improved by using more neighbors, or by looking into other data validation techniques to leave-one-out validation. When it comes to the nearest neighbor method, two factors can be improved. First, increasing the algorithm's number of neighbors to 11 can improve its performance. By taking into account a larger number of neighbors, the algorithm can improve its predictability and resilience. This method reduces the risk of primarily depending on a single nearest neighbor, which could bring noise or mistakes into the results.

In summary, due to its efficiency and ability to reliably identify essential qualities, forward selection is a preferred strategy for selecting characteristics in simpler everyday circumstances. Furthermore, by taking into account a larger number of neighbors and investigating different data validation methodologies, the nearest neighbor algorithm can be improved in terms of prediction power and dependability.

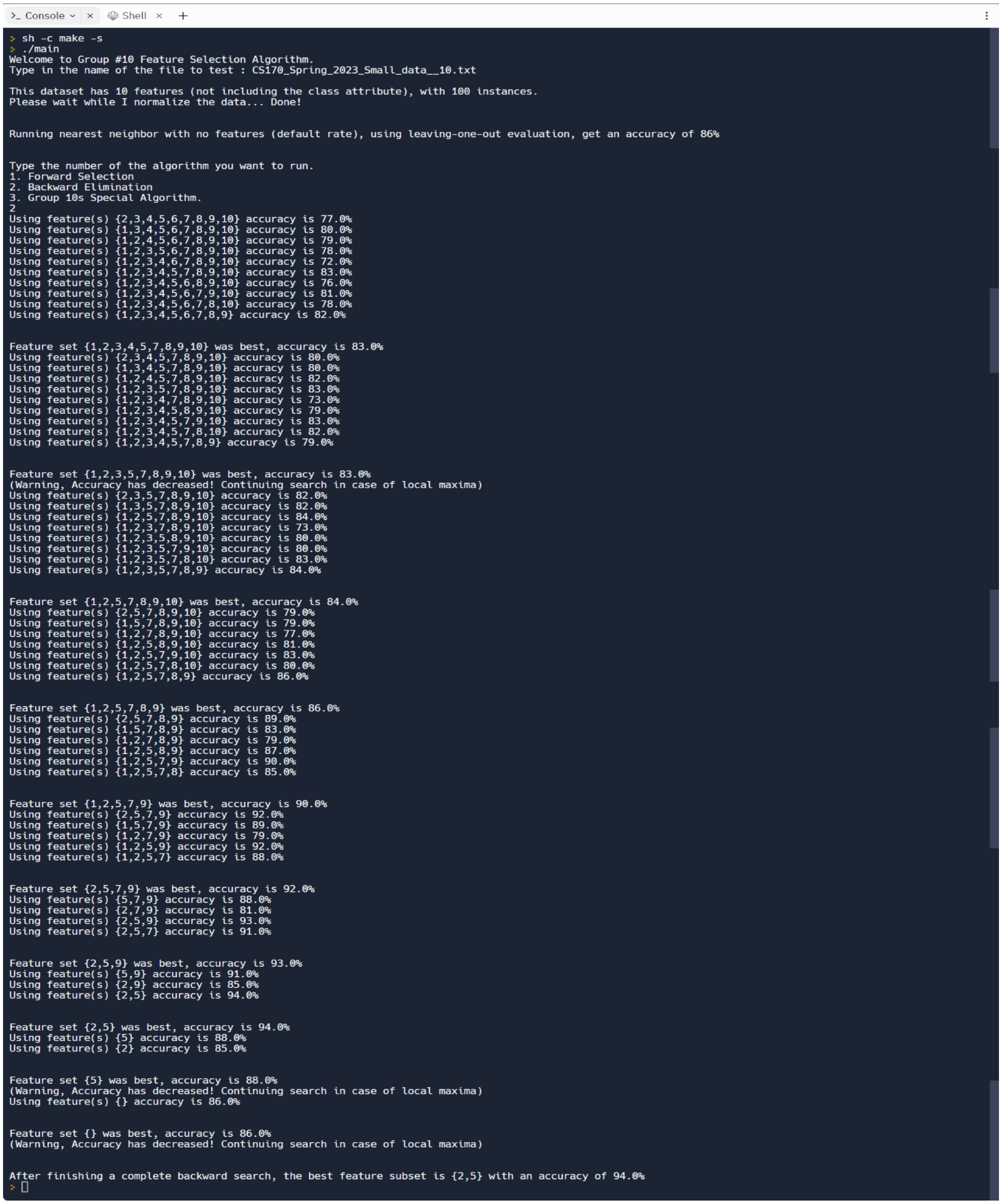
VIII. Trace of small dataset

Forward Selection:



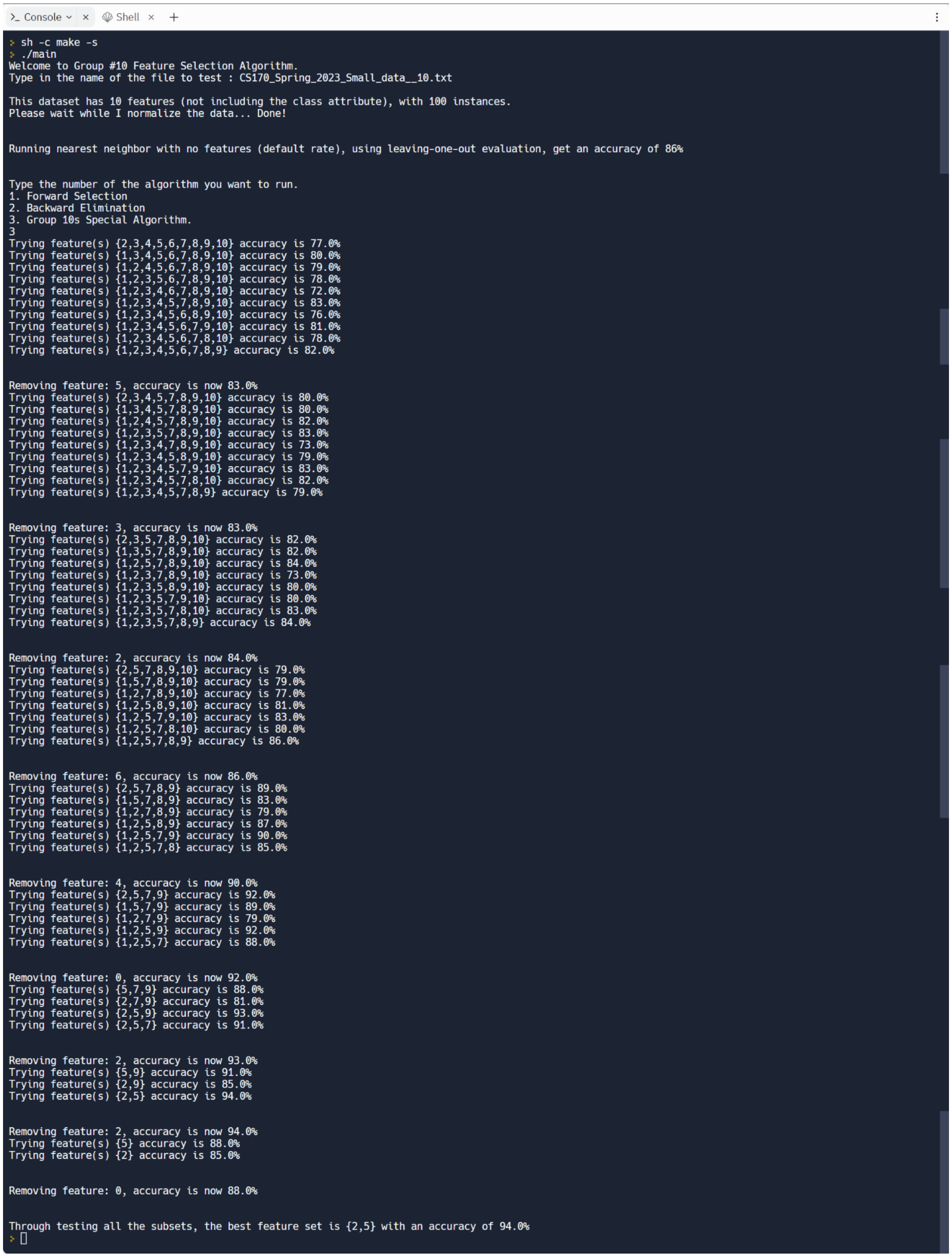
```
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Small_data__10.txt

This dataset has 10 features (not including the class attribute), with 100 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 86%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
1
Using feature(s) {1} accuracy is 75.0%
Using feature(s) {2} accuracy is 85.0%
Using feature(s) {3} accuracy is 76.0%
Using feature(s) {4} accuracy is 82.0%
Using feature(s) {5} accuracy is 88.0%
Using feature(s) {6} accuracy is 79.0%
Using feature(s) {7} accuracy is 77.0%
Using feature(s) {8} accuracy is 74.0%
Using feature(s) {9} accuracy is 77.0%
Using feature(s) {10} accuracy is 75.0%


Feature set {5} was best, accuracy is 88.0%
Using feature(s) {5,1} accuracy is 85.0%
Using feature(s) {5,2} accuracy is 94.0%
Using feature(s) {5,3} accuracy is 89.0%
Using feature(s) {5,4} accuracy is 87.0%
Using feature(s) {5,6} accuracy is 89.0%
Using feature(s) {5,7} accuracy is 83.0%
Using feature(s) {5,8} accuracy is 77.0%
Using feature(s) {5,9} accuracy is 91.0%
Using feature(s) {5,10} accuracy is 87.0%


Feature set {5,2} was best, accuracy is 94.0%
Using feature(s) {5,2,1} accuracy is 91.0%
Using feature(s) {5,2,3} accuracy is 87.0%
Using feature(s) {5,2,4} accuracy is 94.0%
Using feature(s) {5,2,6} accuracy is 89.0%
Using feature(s) {5,2,7} accuracy is 91.0%
Using feature(s) {5,2,8} accuracy is 87.0%
Using feature(s) {5,2,9} accuracy is 93.0%
Using feature(s) {5,2,10} accuracy is 92.0%


After finishing a complete forward search, the best feature subset is {5,2}, it has an accuracy of 94.0%
>
```

Backward Elimination:


>_ Console Shell

```
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Small_data__10.txt

This dataset has 10 features (not including the class attribute), with 100 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 86%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
2
Using feature(s) {2,3,4,5,6,7,8,9,10} accuracy is 77.0%
Using feature(s) {1,3,4,5,6,7,8,9,10} accuracy is 80.0%
Using feature(s) {1,2,4,5,6,7,8,9,10} accuracy is 79.0%
Using feature(s) {1,2,3,5,6,7,8,9,10} accuracy is 78.0%
Using feature(s) {1,2,3,4,6,7,8,9,10} accuracy is 72.0%
Using feature(s) {1,2,3,4,5,7,8,9,10} accuracy is 83.0%
Using feature(s) {1,2,3,4,5,6,8,9,10} accuracy is 76.0%
Using feature(s) {1,2,3,4,5,6,7,9,10} accuracy is 81.0%
Using feature(s) {1,2,3,4,5,6,7,8,10} accuracy is 78.0%
Using feature(s) {1,2,3,4,5,6,7,8,9} accuracy is 82.0%


Feature set {1,2,3,4,5,7,8,9,10} was best, accuracy is 83.0%
Using feature(s) {2,3,4,5,7,8,9,10} accuracy is 80.0%
Using feature(s) {1,3,4,5,7,8,9,10} accuracy is 80.0%
Using feature(s) {1,2,4,5,7,8,9,10} accuracy is 82.0%
Using feature(s) {1,2,3,5,7,8,9,10} accuracy is 83.0%
Using feature(s) {1,2,3,4,7,8,9,10} accuracy is 73.0%
Using feature(s) {1,2,3,4,5,8,9,10} accuracy is 79.0%
Using feature(s) {1,2,3,4,5,7,9,10} accuracy is 83.0%
Using feature(s) {1,2,3,4,5,7,8,10} accuracy is 82.0%
Using feature(s) {1,2,3,4,5,7,8,9} accuracy is 79.0%


Feature set {1,2,3,5,7,8,9,10} was best, accuracy is 83.0%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Using feature(s) {2,3,5,7,8,9,10} accuracy is 82.0%
Using feature(s) {1,3,5,7,8,9,10} accuracy is 82.0%
Using feature(s) {1,2,5,7,8,9,10} accuracy is 84.0%
Using feature(s) {1,2,3,7,8,9,10} accuracy is 73.0%
Using feature(s) {1,2,3,5,8,9,10} accuracy is 80.0%
Using feature(s) {1,2,3,5,7,9,10} accuracy is 80.0%
Using feature(s) {1,2,3,5,7,8,10} accuracy is 83.0%
Using feature(s) {1,2,3,5,7,8,9} accuracy is 84.0%


Feature set {1,2,5,7,8,9,10} was best, accuracy is 84.0%
Using feature(s) {2,5,7,8,9,10} accuracy is 79.0%
Using feature(s) {1,5,7,8,9,10} accuracy is 79.0%
Using feature(s) {1,2,7,8,9,10} accuracy is 77.0%
Using feature(s) {1,2,5,8,9,10} accuracy is 81.0%
Using feature(s) {1,2,5,7,9,10} accuracy is 83.0%
Using feature(s) {1,2,5,7,8,10} accuracy is 80.0%
Using feature(s) {1,2,5,7,8,9} accuracy is 86.0%


Feature set {1,2,5,7,8,9} was best, accuracy is 86.0%
Using feature(s) {2,5,7,8,9} accuracy is 89.0%
Using feature(s) {1,5,7,8,9} accuracy is 83.0%
Using feature(s) {1,2,7,8,9} accuracy is 79.0%
Using feature(s) {1,2,5,8,9} accuracy is 87.0%
Using feature(s) {1,2,5,7,9} accuracy is 90.0%
Using feature(s) {1,2,5,7,8} accuracy is 85.0%


Feature set {1,2,5,7,9} was best, accuracy is 90.0%
Using feature(s) {2,5,7,9} accuracy is 92.0%
Using feature(s) {1,5,7,9} accuracy is 89.0%
Using feature(s) {1,2,7,9} accuracy is 79.0%
Using feature(s) {1,2,5,9} accuracy is 92.0%
Using feature(s) {1,2,5,7} accuracy is 88.0%


Feature set {2,5,7,9} was best, accuracy is 92.0%
Using feature(s) {5,7,9} accuracy is 88.0%
Using feature(s) {2,7,9} accuracy is 81.0%
Using feature(s) {2,5,9} accuracy is 93.0%
Using feature(s) {2,5,7} accuracy is 91.0%


Feature set {2,5,9} was best, accuracy is 93.0%
Using feature(s) {5,9} accuracy is 91.0%
Using feature(s) {2,9} accuracy is 85.0%
Using feature(s) {2,5} accuracy is 94.0%


Feature set {2,5} was best, accuracy is 94.0%
Using feature(s) {5} accuracy is 88.0%
Using feature(s) {2} accuracy is 85.0%


Feature set {5} was best, accuracy is 88.0%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Using feature(s) {} accuracy is 86.0%


Feature set {} was best, accuracy is 86.0%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)


After finishing a complete backward search, the best feature subset is {2,5} with an accuracy of 94.0%
>
```

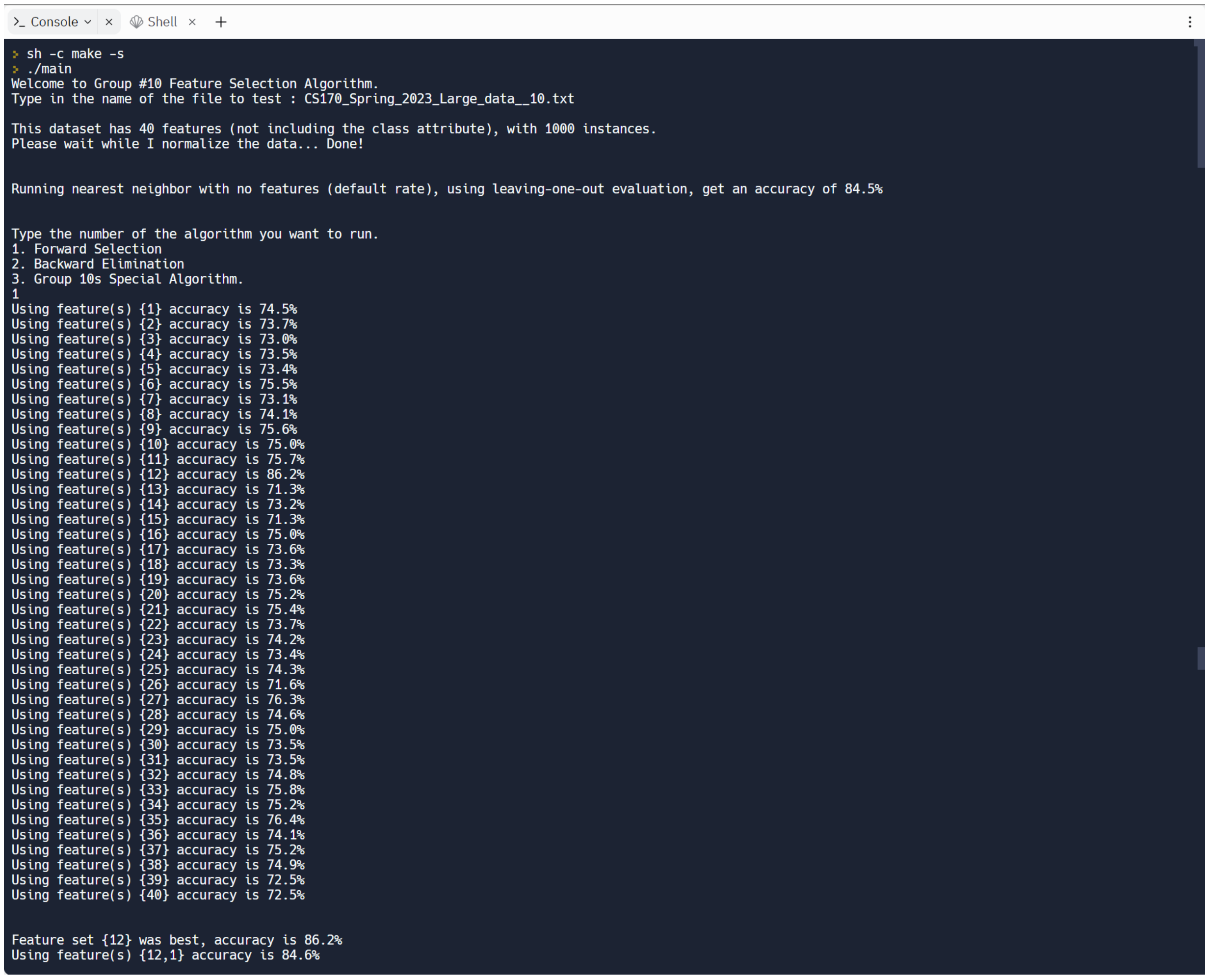
Custom Algorithm 3:



```
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Small_data__10.txt

This dataset has 10 features (not including the class attribute), with 100 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 86%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
3
Trying feature(s) {2,3,4,5,6,7,8,9,10} accuracy is 77.0%
Trying feature(s) {1,3,4,5,6,7,8,9,10} accuracy is 80.0%
Trying feature(s) {1,2,4,5,6,7,8,9,10} accuracy is 79.0%
Trying feature(s) {1,2,3,5,6,7,8,9,10} accuracy is 78.0%
Trying feature(s) {1,2,3,4,6,7,8,9,10} accuracy is 72.0%
Trying feature(s) {1,2,3,4,5,7,8,9,10} accuracy is 83.0%
Trying feature(s) {1,2,3,4,5,6,8,9,10} accuracy is 76.0%
Trying feature(s) {1,2,3,4,5,6,7,9,10} accuracy is 81.0%
Trying feature(s) {1,2,3,4,5,6,7,8,10} accuracy is 78.0%
Trying feature(s) {1,2,3,4,5,6,7,8,9} accuracy is 82.0%


Removing feature: 5, accuracy is now 83.0%
Trying feature(s) {2,3,4,5,7,8,9,10} accuracy is 80.0%
Trying feature(s) {1,3,4,5,7,8,9,10} accuracy is 80.0%
Trying feature(s) {1,2,4,5,7,8,9,10} accuracy is 82.0%
Trying feature(s) {1,2,3,5,7,8,9,10} accuracy is 83.0%
Trying feature(s) {1,2,3,4,7,8,9,10} accuracy is 73.0%
Trying feature(s) {1,2,3,4,5,8,9,10} accuracy is 79.0%
Trying feature(s) {1,2,3,4,5,7,9,10} accuracy is 83.0%
Trying feature(s) {1,2,3,4,5,7,8,10} accuracy is 82.0%
Trying feature(s) {1,2,3,4,5,7,8,9} accuracy is 79.0%


Removing feature: 3, accuracy is now 83.0%
Trying feature(s) {2,3,5,7,8,9,10} accuracy is 82.0%
Trying feature(s) {1,3,5,7,8,9,10} accuracy is 82.0%
Trying feature(s) {1,2,5,7,8,9,10} accuracy is 84.0%
Trying feature(s) {1,2,3,7,8,9,10} accuracy is 73.0%
Trying feature(s) {1,2,3,5,8,9,10} accuracy is 80.0%
Trying feature(s) {1,2,3,5,7,9,10} accuracy is 80.0%
Trying feature(s) {1,2,3,5,7,8,10} accuracy is 83.0%
Trying feature(s) {1,2,3,5,7,8,9} accuracy is 84.0%


Removing feature: 2, accuracy is now 84.0%
Trying feature(s) {2,5,7,8,9,10} accuracy is 79.0%
Trying feature(s) {1,5,7,8,9,10} accuracy is 79.0%
Trying feature(s) {1,2,7,8,9,10} accuracy is 77.0%
Trying feature(s) {1,2,5,8,9,10} accuracy is 81.0%
Trying feature(s) {1,2,5,7,9,10} accuracy is 83.0%
Trying feature(s) {1,2,5,7,8,10} accuracy is 80.0%
Trying feature(s) {1,2,5,7,8,9} accuracy is 86.0%


Removing feature: 6, accuracy is now 86.0%
Trying feature(s) {2,5,7,8,9} accuracy is 89.0%
Trying feature(s) {1,5,7,8,9} accuracy is 83.0%
Trying feature(s) {1,2,7,8,9} accuracy is 79.0%
Trying feature(s) {1,2,5,8,9} accuracy is 87.0%
Trying feature(s) {1,2,5,7,9} accuracy is 90.0%
Trying feature(s) {1,2,5,7,8} accuracy is 85.0%


Removing feature: 4, accuracy is now 90.0%
Trying feature(s) {2,5,7,9} accuracy is 92.0%
Trying feature(s) {1,5,7,9} accuracy is 89.0%
Trying feature(s) {1,2,7,9} accuracy is 79.0%
Trying feature(s) {1,2,5,9} accuracy is 92.0%
Trying feature(s) {1,2,5,7} accuracy is 88.0%


Removing feature: 0, accuracy is now 92.0%
Trying feature(s) {5,7,9} accuracy is 88.0%
Trying feature(s) {2,7,9} accuracy is 81.0%
Trying feature(s) {2,5,9} accuracy is 93.0%
Trying feature(s) {2,5,7} accuracy is 91.0%


Removing feature: 2, accuracy is now 93.0%
Trying feature(s) {5,9} accuracy is 91.0%
Trying feature(s) {2,9} accuracy is 85.0%
Trying feature(s) {2,5} accuracy is 94.0%


Removing feature: 2, accuracy is now 94.0%
Trying feature(s) {5} accuracy is 88.0%
Trying feature(s) {2} accuracy is 85.0%


Removing feature: 0, accuracy is now 88.0%


Through testing all the subsets, the best feature set is {2,5} with an accuracy of 94.0%
> 
```

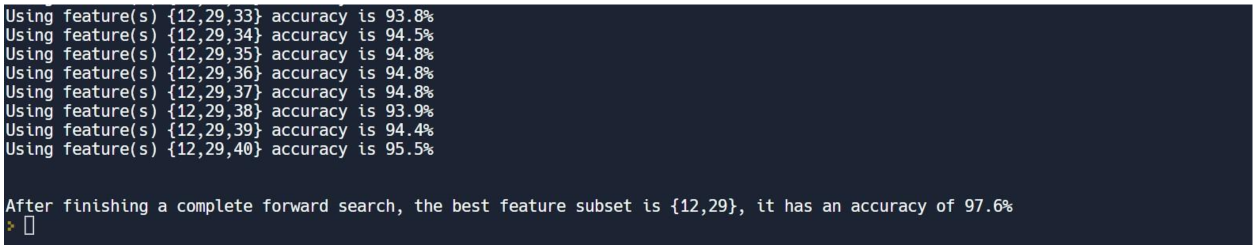
IX. Trace of largest dataset

Forward:



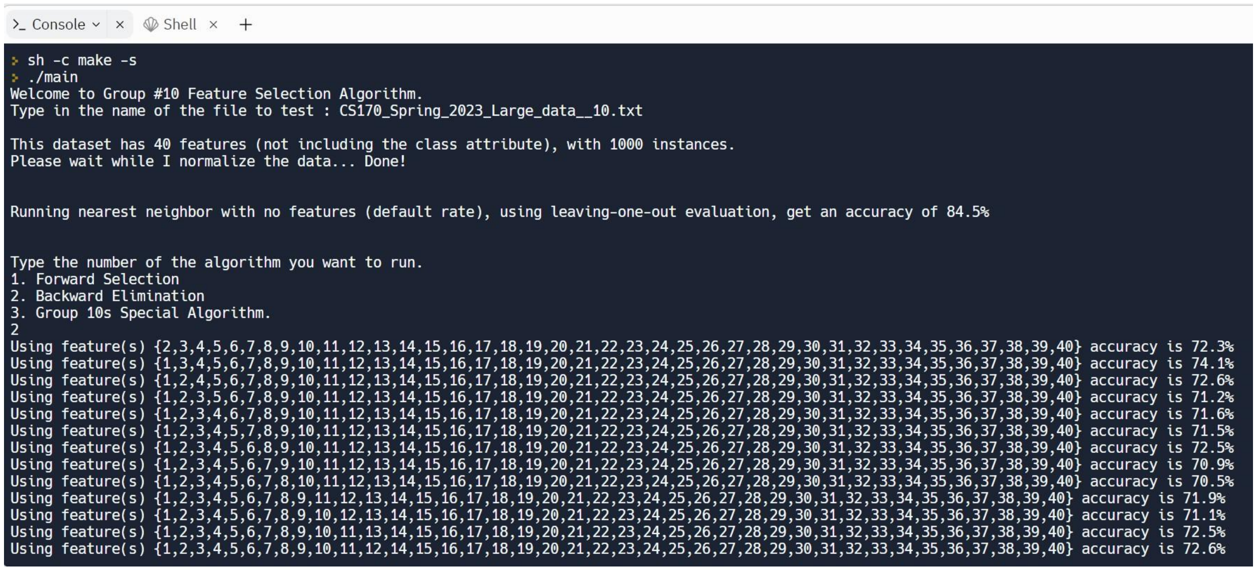
```
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Large_data__10.txt

This dataset has 40 features (not including the class attribute), with 1000 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 84.5%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
1
Using feature(s) {1} accuracy is 74.5%
Using feature(s) {2} accuracy is 73.7%
Using feature(s) {3} accuracy is 73.0%
Using feature(s) {4} accuracy is 73.5%
Using feature(s) {5} accuracy is 73.4%
Using feature(s) {6} accuracy is 75.5%
Using feature(s) {7} accuracy is 73.1%
Using feature(s) {8} accuracy is 74.1%
Using feature(s) {9} accuracy is 75.6%
Using feature(s) {10} accuracy is 75.0%
Using feature(s) {11} accuracy is 75.7%
Using feature(s) {12} accuracy is 86.2%
Using feature(s) {13} accuracy is 71.3%
Using feature(s) {14} accuracy is 73.2%
Using feature(s) {15} accuracy is 71.3%
Using feature(s) {16} accuracy is 75.0%
Using feature(s) {17} accuracy is 73.6%
Using feature(s) {18} accuracy is 73.3%
Using feature(s) {19} accuracy is 73.6%
Using feature(s) {20} accuracy is 75.2%
Using feature(s) {21} accuracy is 75.4%
Using feature(s) {22} accuracy is 73.7%
Using feature(s) {23} accuracy is 74.2%
Using feature(s) {24} accuracy is 73.4%
Using feature(s) {25} accuracy is 74.3%
Using feature(s) {26} accuracy is 71.6%
Using feature(s) {27} accuracy is 76.3%
Using feature(s) {28} accuracy is 74.6%
Using feature(s) {29} accuracy is 75.0%
Using feature(s) {30} accuracy is 73.5%
Using feature(s) {31} accuracy is 73.5%
Using feature(s) {32} accuracy is 74.8%
Using feature(s) {33} accuracy is 75.8%
Using feature(s) {34} accuracy is 75.2%
Using feature(s) {35} accuracy is 76.4%
Using feature(s) {36} accuracy is 74.1%
Using feature(s) {37} accuracy is 75.2%
Using feature(s) {38} accuracy is 74.9%
Using feature(s) {39} accuracy is 72.5%
Using feature(s) {40} accuracy is 72.5%


Feature set {12} was best, accuracy is 86.2%
Using feature(s) {12,1} accuracy is 84.6%
```

Keeps going, for a result of……………



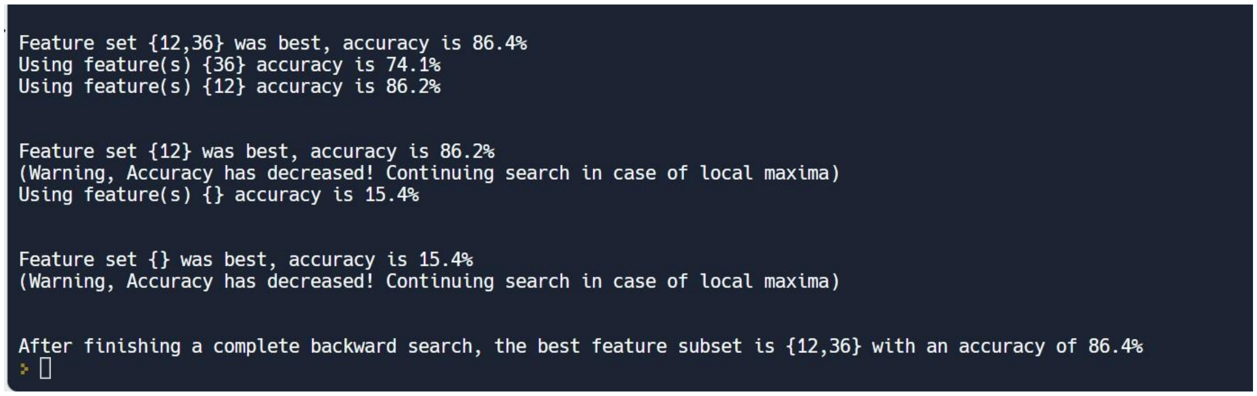
```
Using feature(s) {12,29,33} accuracy is 93.8%
Using feature(s) {12,29,34} accuracy is 94.5%
Using feature(s) {12,29,35} accuracy is 94.8%
Using feature(s) {12,29,36} accuracy is 94.8%
Using feature(s) {12,29,37} accuracy is 94.8%
Using feature(s) {12,29,38} accuracy is 93.9%
Using feature(s) {12,29,39} accuracy is 94.4%
Using feature(s) {12,29,40} accuracy is 95.5%


After finishing a complete forward search, the best feature subset is {12,29}, it has an accuracy of 97.6%
>
```

Backward:



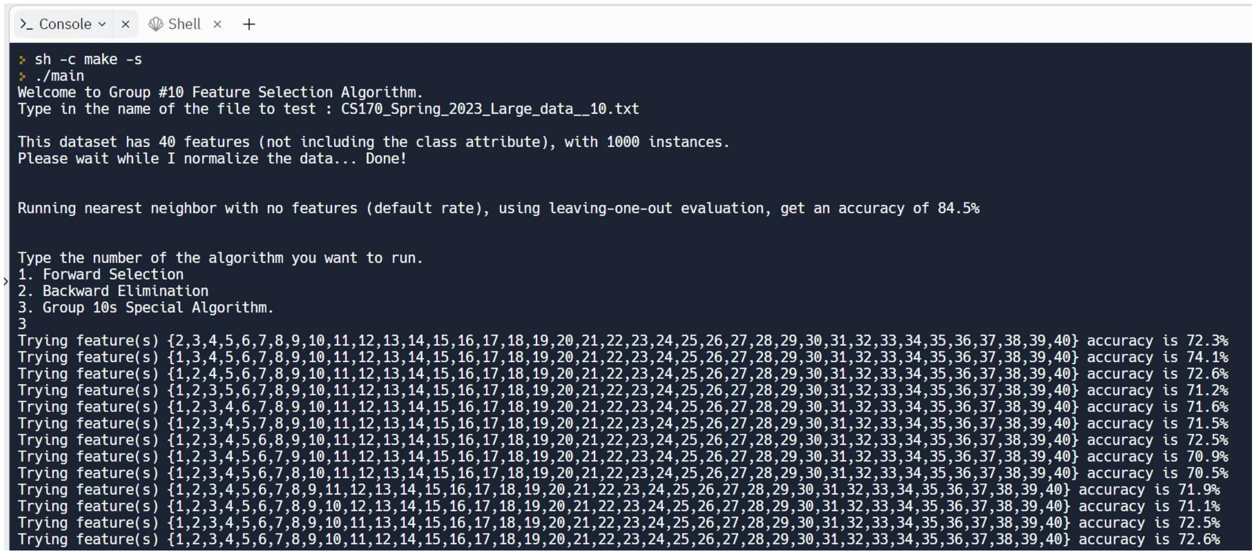
```
>_ Console   Shell  +
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Large_data__10.txt

This dataset has 40 features (not including the class attribute), with 1000 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 84.5%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
2
Using feature(s) {2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.3%
Using feature(s) {1,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 74.1%
Using feature(s) {1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.6%
Using feature(s) {1,2,3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.2%
Using feature(s) {1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.6%
Using feature(s) {1,2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.5%
Using feature(s) {1,2,3,4,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.5%
Using feature(s) {1,2,3,4,5,6,7,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 70.9%
Using feature(s) {1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 70.5%
Using feature(s) {1,2,3,4,5,6,7,8,9,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.9%
Using feature(s) {1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.1%
Using feature(s) {1,2,3,4,5,6,7,8,9,10,11,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.5%
Using feature(s) {1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.6%
```

Keeps going, for a result of………



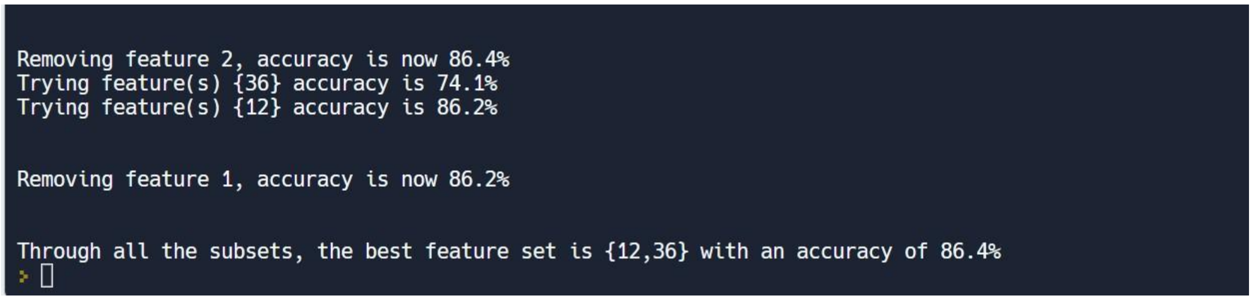
```
Feature set {12,36} was best, accuracy is 86.4%
Using feature(s) {36} accuracy is 74.1%
Using feature(s) {12} accuracy is 86.2%


Feature set {12} was best, accuracy is 86.2%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Using feature(s) {} accuracy is 15.4%


Feature set {} was best, accuracy is 15.4%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)


After finishing a complete backward search, the best feature subset is {12,36} with an accuracy of 86.4%
>
```

Custom Algorithm 3:



```
>_ Console   Shell  +
> sh -c make -s
> ./main
Welcome to Group #10 Feature Selection Algorithm.
Type in the name of the file to test : CS170_Spring_2023_Large_data__10.txt

This dataset has 40 features (not including the class attribute), with 1000 instances.
Please wait while I normalize the data... Done!


Running nearest neighbor with no features (default rate), using leaving-one-out evaluation, get an accuracy of 84.5%


Type the number of the algorithm you want to run.
1. Forward Selection
2. Backward Elimination
3. Group 10s Special Algorithm.
3
Trying feature(s) {2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.3%
Trying feature(s) {1,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 74.1%
Trying feature(s) {1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.6%
Trying feature(s) {1,2,3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.2%
Trying feature(s) {1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.6%
Trying feature(s) {1,2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.5%
Trying feature(s) {1,2,3,4,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.5%
Trying feature(s) {1,2,3,4,5,6,7,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 70.9%
Trying feature(s) {1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 70.5%
Trying feature(s) {1,2,3,4,5,6,7,8,9,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.9%
Trying feature(s) {1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 71.1%
Trying feature(s) {1,2,3,4,5,6,7,8,9,10,11,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.5%
Trying feature(s) {1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40} accuracy is 72.6%
```

Keeps going, for a result of…………



```
Removing feature 2, accuracy is now 86.4%
Trying feature(s) {36} accuracy is 74.1%
Trying feature(s) {12} accuracy is 86.2%


Removing feature 1, accuracy is now 86.2%


Through all the subsets, the best feature set is {12,36} with an accuracy of 86.4%
>
```