**Milan Maragiri**

SID# 862548940

Email: mmara031@ucr.edu

June 7th, 2025

Project 2 for CS 205 Spring 2025, with Dr. Eamonn Keogh

All important code is original. Unimportant subroutines and library calls that are not completely novel include:

- STL containers & algorithms (from <vector>, <unordered_map>, <array>, <string>, <sstream>, <algorithm>): used for data storage, parsing, counting label frequencies, and common operations like std::find, std::sort, and std::copy.
- Math utilities (from <cmath> and <limits>): std::sqrt for distance calculations, and std::numeric_limits<double>::infinity() / quiet_NaN() for initialization.
- I/O and formatting (from <fstream>, <iostream>, <iomanip>): basic file and console input/output, plus std::fixed and std::setprecision for readable accuracy/time prints.
- #include <chrono>: uses std::chrono::high_resolution_clock to measure total runtime duration.

# Feature Selection With Nearest Neighbor

Milan Maragiri, SID 862548940, May-07-2025

## Introduction

In this project, we were asked by Dr. Keogh to explore feature selection for a nearest-neighbor classifier. I used C++ to build a program that implements two greedy search methods, forward selection and backward elimination, using leave-one-out cross-validation. The code was run on three datasets (small, large and breast-cancer) and the resulting accuracy curves are included below; I've followed the sample report structure from the project handout [1].

All charts were plotted using "Plotly Chart Studio" [2]

## Small Dataset

### 1) Forward Selection

In Figure 1 we see the result of running forward selection on CS205_small_Data__48.txt, which was the small file assigned to me.

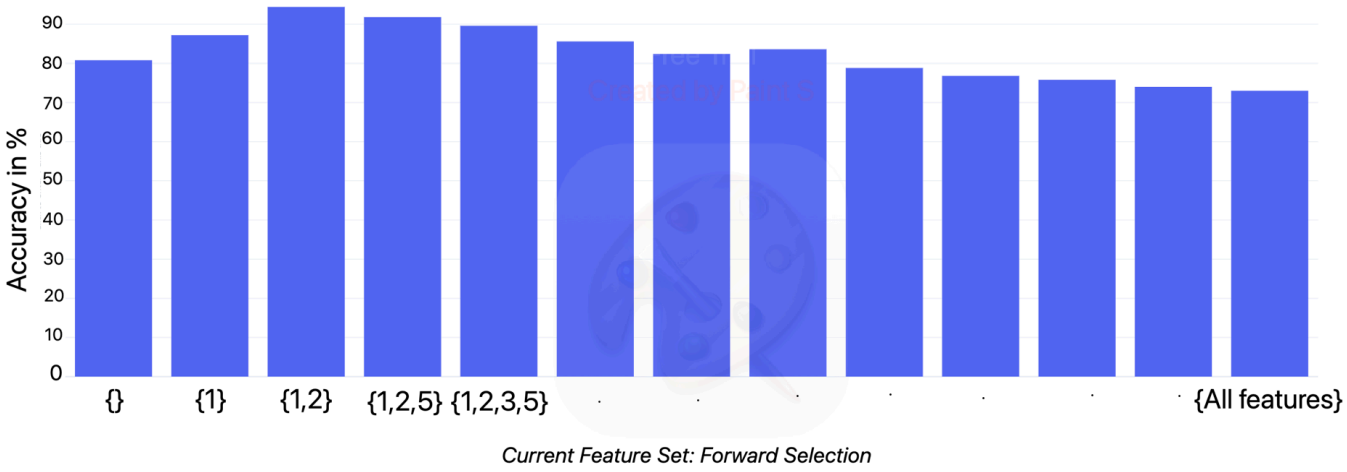


*Figure 1: Accuracy of increasingly large subsets of features discovered by forward selection*

At the beginning of the search, we have no features (denoted by {}), so I reported the default rate, which was 80.8%. Adding feature "1" dramatically improved the accuracy to 87.2%, and then adding feature "2" gave us an accuracy of 94.4% which was the best accuracy using forward selection.

After that, each additional feature actually caused accuracy to drop with a slight increase in adding feature 11, potentially indicating that 11 could be a contributing feature. Using the full set of 12 features yielded just 73%

## 2) Backward Elimination

In Figure 2, we see the result of running backward elimination on CS205_small_Data__48.txt, which was the small file assigned to me.

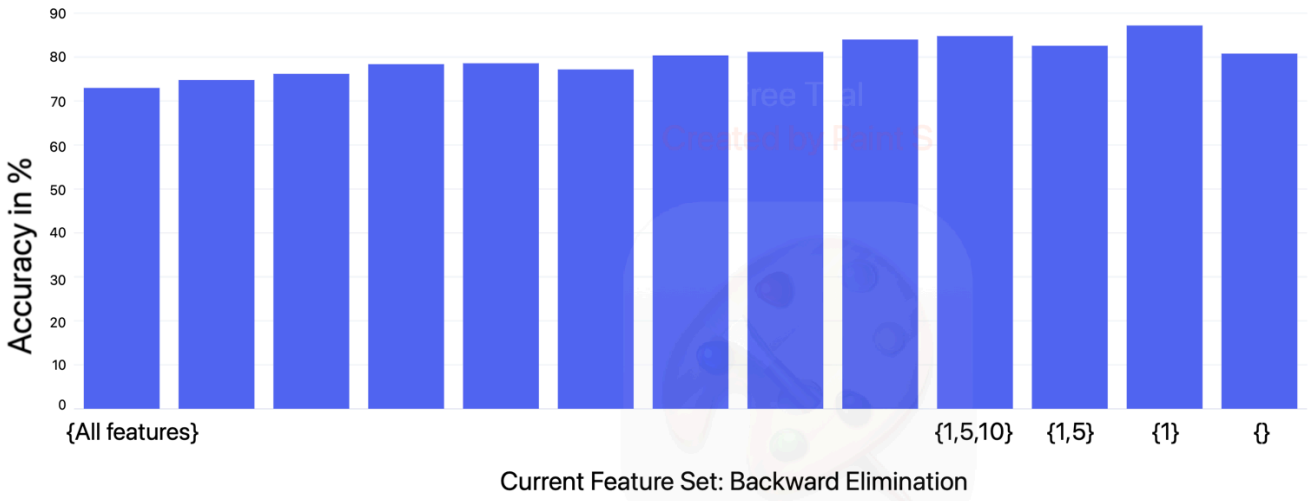


*Figure 2: Accuracy of increasingly small subsets of features discovered by backward elimination*

First, a sanity check. The first tested subset, that of all features, should have the same accuracy as the last tested subset in forward selection. With a score of 73%, that is the case.

Overall, backward elimination gave feature set {1} as the best set with an accuracy of 87.2% compared to 94.4% for forward selection. For this problem, backward elimination is not as accurate as forward selection. However, backward elimination's optimal subset includes feature "1", which confirms that feature 1 is a good feature.

### Conclusion For Small Dataset

I believe that features '1' and '2' are the best features for this problem. There is very weak evidence that feature '11' might be useful, but that needs further investigation. If we deploy this {1,2} model, I believe the accuracy will be about 94.4%.

# Large Dataset

## 1) Forward Selection

We now turn our attention to the more challenging dataset I was tasked with investigating. In Figure 3, we see the result of running forward selection on CS205_large_Data__43.txt, which was the large file assigned to me.

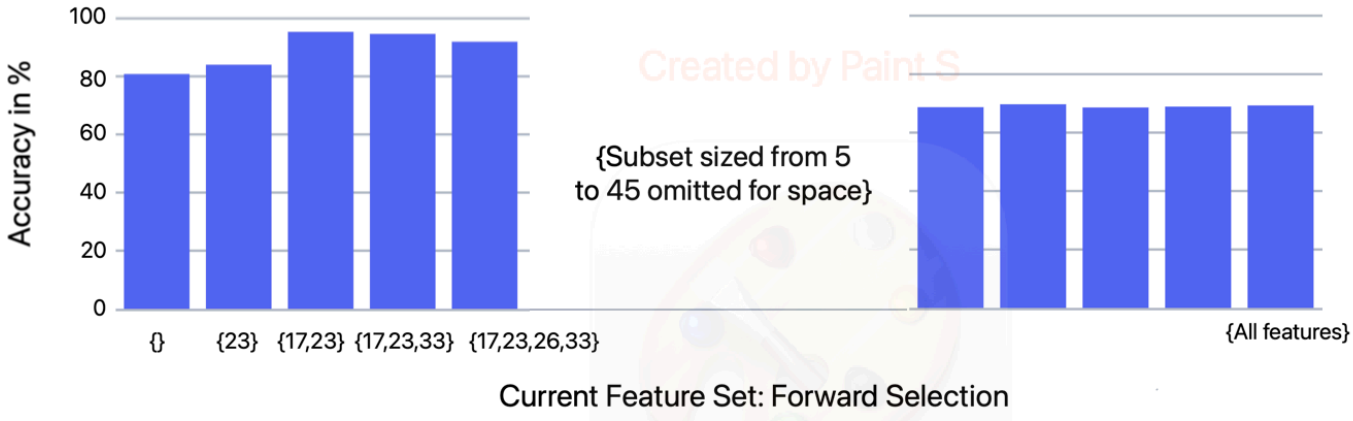


*Figure 3: Accuracy of increasingly large subsets of features discovered by forward selection*

At the beginning of the search, we have no features (denoted by {}), so I reported the default rate, which was 80.8%. Adding feature “23” dramatically improved the accuracy up to 95.4%. The forward selection algorithm identified {17,23} as the best two-feature subset, with 95.4% accuracy. After that, every extra feature we tried actually dragged accuracy down, by the time we used the full feature set it had fallen into the 67–69% range.

## 2) Backward Elimination

In Figure 4, we see the result of running backward elimination on CS205_large_Data__43.txt, which was the large file assigned to me

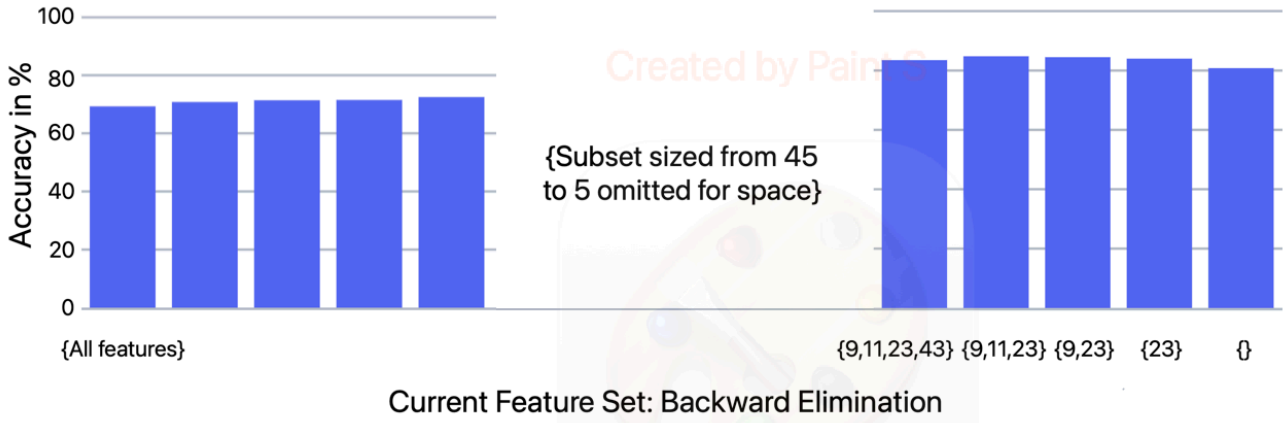


*Figure 4: Accuracy of increasingly small subsets of features discovered by backward elimination*

The backward-elimination search begins with the full feature set, which gives an accuracy of 69.3%, exactly the same as the final accuracy from forward selection on all features.

As features are removed one by one, the best subset it finds is {9,11,23}, which achieves an accuracy of 84.8%. Although this is still well below the 95.4% peak from forward selection, the fact that feature “23” is picked in both searches tells that “23” is a good feature.

### Conclusion For Large Dataset

I believe that features ‘17’ and ‘23’ are the best features for this problem. There is weak evidence that features ‘9’ and ‘11’ might be useful, but that needs further investigation. If we deploy this {17,23} model, I believe the accuracy will be about 95.4%.

## Computational Effort For Search

I implemented the search in C++, and ran all experiments on a MacBook with Apple M3 Silicon and 8 GB of main memory. In Table 1, I report the running time for the four searches I conducted.

| **Algorithm** | **Small Dataset** **(12 Features, 500 instances)** | **Large Dataset** **(50 Features, 1000 instances)** |
|---|---|---|
| Forward Selection | 1s | 3 mins 16 s |
| Backward Elimination | 1.3s | 5 mins 37 s |

*Table 1: Execution time for the four searches*

## Breast-Cancer Dataset

For Part 2 of the project, I selected the Wisconsin Breast Cancer Dataset [3]. This dataset contains 683 instances and eleven columns: a sample code number, nine features, and a diagnosis label. Since the sample code number is not important, it needs to be removed. I wrote a C++ preprocessing program to:

- Delete the sample code number column.
- Reformat the remaining data into the layout required by our feature-selection code,
- Apply Z-score normalization to each feature.
- Exclude any records with missing values.

In Figure 5, we see the result of running forward selection on the preprocessed dataset on breast-cancer diagnosis.

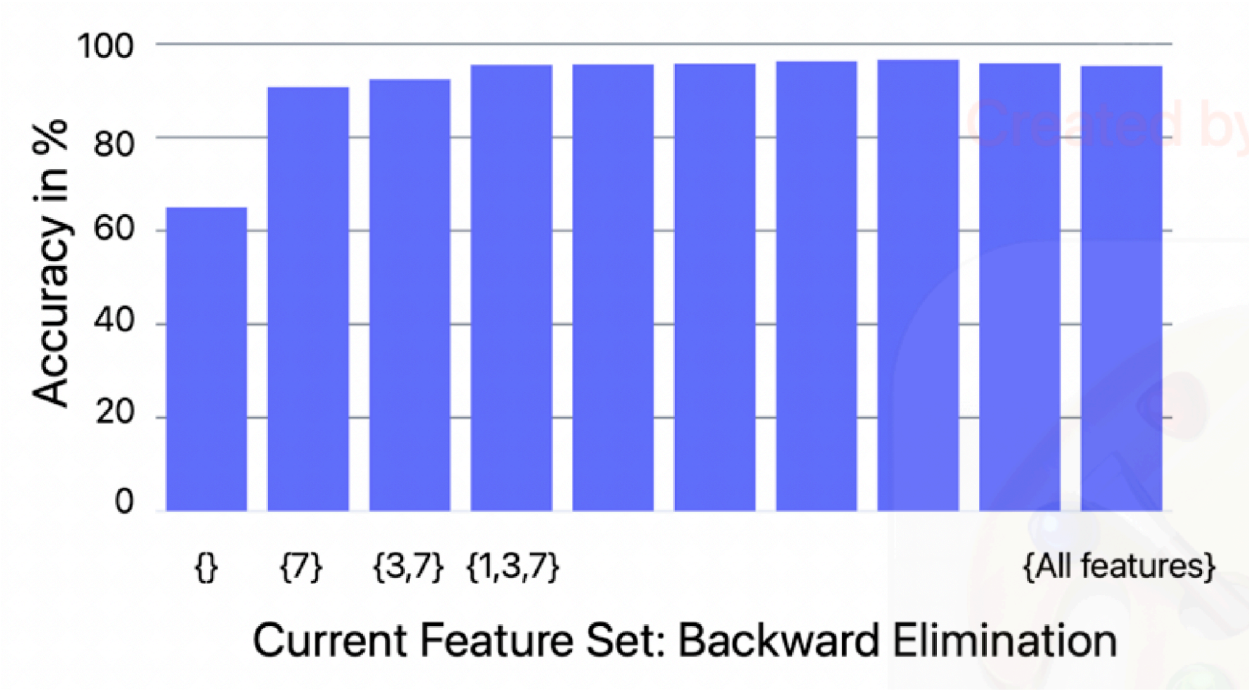


*Figure 5: Accuracy of increasingly large subsets of features discovered by forward selection*

## Order Of Feature Selection

1) Feature 7 - Bland Chromatin
2) Feature 3 - Uniformity of Cell Shape
3) Feature 1 - Clump Thickness
4) Feature 6 - Bare Nuclei
5) Feature 4 - Marginal Adhesion
6) Feature 9 - Mitoses
7) Feature 5 - Single Epithelial Cell Size
8) Feature 2 - Uniformity of Cell Size
9) Feature 8 - Normal Nucleoli

Feature 7, Bland Chromatin, was the first feature added. Bland Chromatin measures the texture of chromatin in the cell nucleus on a scale from 1 (very smooth) to 10 (highly clumped) [3]. This is consistent with a study [4] that used decision trees, partial least squares, regression, and clustering to rank Bland Chromatin among the top predictors of malignancy. In fact, logistic-regression based feature selection [5] singled out Bland Chromatin as one of the most informative variables for distinguishing benign from malignant samples.

Overall, the feature set {1,3,4,5,6,7,9} was the best, having an accuracy of 96.6%.

Features 2 and 8 were dropped because they added little real value: Cell Size (feature 2) is nearly redundant with Cell Shape (feature 3). Further investigation needs to be done to see why Normal Nucleoli (feature 8) is not included in the final feature set.

## Trace Of Forward Selection On Small Dataset

Below I show a trace of my algorithm. I am only showing Forward Selection on the small dataset

```
milanmaragiri@Mac feature_selection_with_nearest_neighbor % g++ main.cpp
milanmaragiri@Mac feature_selection_with_nearest_neighbor % ./a.out
Welcome to Bertie Woosters Feature Selection Algorithm
Type in the name of the file to test: CS205_small_Data__48.txt

This dataset has 12 features (Not including the class attribute), with 500 instances

Type the number of the algorithm you want to run

1) Forward Selection
2) Backward Elimination

1
Default rate using no features is: 80.8%

Running nearest neighbor with all 12 features, using "leave-one-out" evaluation, I get an accuracy of 73.0%

Beginning search.
Using feature(s) {1} accuracy is 87.2%
Using feature(s) {2} accuracy is 71.8%
Using feature(s) {3} accuracy is 68.6%
Using feature(s) {4} accuracy is 73.8%
Using feature(s) {5} accuracy is 67.4%
Using feature(s) {6} accuracy is 69.0%
Using feature(s) {7} accuracy is 69.6%
Using feature(s) {8} accuracy is 67.6%
Using feature(s) {9} accuracy is 71.0%
Using feature(s) {10} accuracy is 67.4%
Using feature(s) {11} accuracy is 68.4%
Using feature(s) {12} accuracy is 69.0%
Feature set {1} gave an accuracy of: 87.2%
Using feature(s) {1,2} accuracy is 94.4%
Using feature(s) {1,3} accuracy is 83.0%
Using feature(s) {1,4} accuracy is 83.2%
Using feature(s) {1,5} accuracy is 82.6%
Using feature(s) {1,6} accuracy is 82.8%
Using feature(s) {1,7} accuracy is 81.4%
```

Dear Professor: Here I deleted most of the trace to save space. However, I have retain at least the full first level of the search, and the last level of the search}

```
Feature set {1,2,3,4,5,8,10,11} gave an accuracy of: 78.8%
Warning, accuracy has decreased! Continuing search in case of local maxima
Using feature(s) {1,2,3,4,5,6,8,10,11} accuracy is 75.0%
Using feature(s) {1,2,3,4,5,7,8,10,11} accuracy is 76.8%
Using feature(s) {1,2,3,4,5,8,9,10,11} accuracy is 73.6%
Using feature(s) {1,2,3,4,5,8,10,11,12} accuracy is 76.6%
Feature set {1,2,3,4,5,7,8,10,11} gave an accuracy of: 76.8%
Warning, accuracy has decreased! Continuing search in case of local maxima
Using feature(s) {1,2,3,4,5,6,7,8,10,11} accuracy is 75.8%
Using feature(s) {1,2,3,4,5,7,8,9,10,11} accuracy is 74.6%
Using feature(s) {1,2,3,4,5,7,8,10,11,12} accuracy is 75.2%
Feature set {1,2,3,4,5,6,7,8,10,11} gave an accuracy of: 75.8%
Warning, accuracy has decreased! Continuing search in case of local maxima
Using feature(s) {1,2,3,4,5,6,7,8,9,10,11} accuracy is 73.6%
Using feature(s) {1,2,3,4,5,6,7,8,10,11,12} accuracy is 74.0%
Feature set {1,2,3,4,5,6,7,8,10,11,12} gave an accuracy of: 74.0%
Warning, accuracy has decreased! Continuing search in case of local maxima
Using feature(s) {1,2,3,4,5,6,7,8,9,10,11,12} accuracy is 73.0%
Feature set {1,2,3,4,5,6,7,8,9,10,11,12} gave an accuracy of: 73.0%
Warning, accuracy has decreased! Continuing search in case of local maxima

Finished search!! The best feature subset is {1,2}, which has an accuracy of 94.4%

Total execution time: 934 ms
milanmaragiri@Mac feature_selection_with_nearest_neighbor %
```

## Github Link

https://github.com/3rst/feature_selection_with_nearest_neighbor

## References

[1] Dr Keogh Project Handout - https://www.dropbox.com/scl/fo/ydr7o2fo4ljbv0l5mmo7x/AAHQO7k2yBpVejVcljN4mQk?rlkey=tikonndbdlen603v3ln51hsa1&e=1&dl=0
[2] Plotly Chart Studio - https://chart-studio.plotly.com/
[3] Dataset Chosen - https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original
[4] Breast-Cancer Diagnosis using Data Mining - https://support.sas.com/resources/papers/proceedings16/9420-2016.pdf
[5] Breast-Cancer Diagnosis using Machine Learning - https://pdfs.semanticscholar.org/2475/6d2b55622118d57120c68d864e947eac8609.pdf