CS205: Artificial Intelligence

Dr. Eamonn Keogh

Project 2: Feature Selection with Nearest Neighbor

Huy Dinh Tran
SID: 862325308
Email: htran197@ucr.edu

Source code on GitHub: https://github.com/huydinhtran/CS205Project2-Feature-Selection-NN

In completing this assignment I consulted:

- All of the lecture slides and undergraduate video lectures especially pseudo code from the Project_2_Briefing slides from Dr. Eamonn Keogh
- Python documentation from https://www.python.org/, programming Q&A forum https://stackoverflow.com/, and as well as programming tutorial websites such as https://www.geeksforgeeks.org/ and https://www.w3schools.com/

All important core code is written by myself originally and with reference from Dr. Eamonn Keogh's MATLAB pseudo code. I'm using Jupyter Notebook instead of standard Python code for this project. Here is a list of additional libraries which have functions that were used to assist me in implementing the algorithm.

- import numpy as np
- import math
- import time

Outline of the report:

CS205 Project 2
Feature Selection w/ Nearest Neighbor

Huy Dinh Tran, SID: 862325308

Introduction

When given a dataset for classification, one of the most important considerations to get the best accuracy results is determining which features to use. Using more features doesn't result in a better model. It can cause lower accuracy and also more execution time because of complexity. Feature Selection is an algorithm that determines the best set of features to use for classification training. It mixes and matches each feature together to the most optimal set of features as shown in figure 1. In this CS205 Project 2, we are tasked to implement our version of Feature Selection using Forwarding Selection and Backward Elimination. Our classification method of choice is using Nearest Neighbor using the Euclidean formula shown in figure 2. We would be using small and large datasets for this implementation.

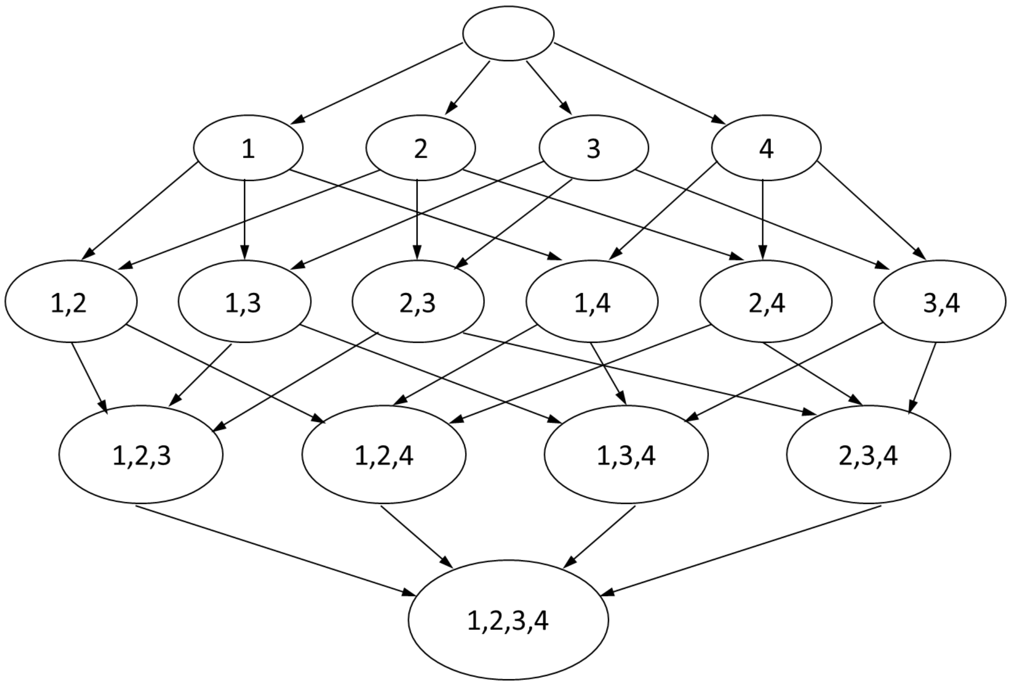


Figure 1. Feature selection search tree with 4 features

$$d(x,\ y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Figure 2. Euclidean distance formula

Description of Datasets, Computer Specifications, and Algorithms

Specification of Datasets

SMALLtestdata__66:
No. of features: 10
No. of instances: 300

LARGEtestdata__48:
No. of features: 40
No. of instances: 1000

Computer Specifications

CPU: AMD Ryzen 7 5800X 8-Core Processor 3.80GHz
RAM: 16GB DDR4 3200MHZ

Forward Selection

The Forward Selection algorithm is basically searching from the top to the bottom of the search tree which is shown in figure 1. Firstly, we have an empty set. We started with testing one feature and adding more features to the set to determine the best set of features. In the end, we will test all of the features together.

Backward Elimination

The Backward Elimination algorithm has the opposite flow of the Forward Selection algorithm. It searches from the bottom to the top of the search tree which is shown in figure 1. We will have a set of all of the features at the start. We will work backward with testing all of the features together first and eliminate a feature for each iteration. In the end, we will be left with no features at the top of the search tree.

Results of Small Dataset

Figure 3 shows the results for running forward selection of the CS205_SP_2022_SMALLtestdata__66.txt. We started off by adding feature 3 which gives us 82% accuracy. After the next two iterations, we added feature 7 and 9 which both gives us 97%. For the rest of the run, we got worse and worse performance. The algorithm presents features 3 and 7 as the final set of features since having only 2 features makes the model much simpler which can decrease execution time.

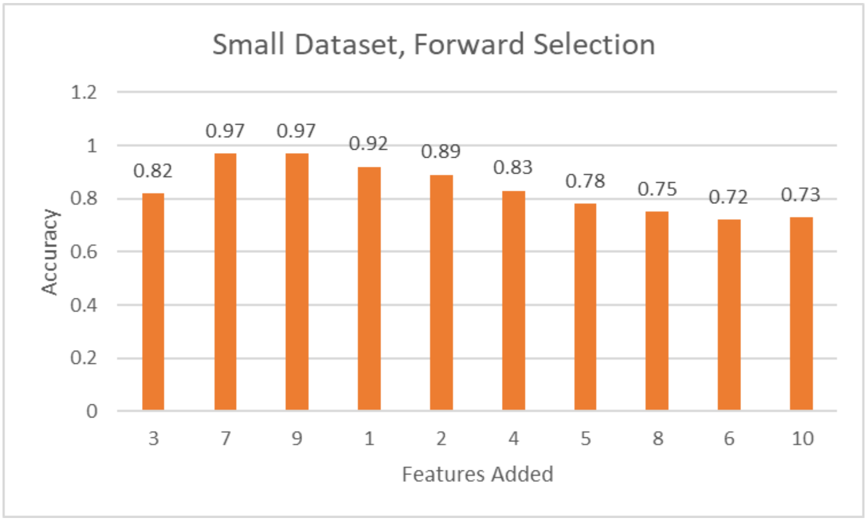


Figure 3. Results for small dataset with Forward Selection

Figure 4 shows the results for running backward elimination of the same small dataset. We can see that the last iteration of the backward elimination has the same result as the first iteration of the forward selection with an accuracy of 81%-82%. When removing more features, we see an increase in accuracies. After removing feature 10, we left with features 7 and 3 which are the best set of features with an accuracy of 97%. For the small dataset, we proved that both forward selection and backward elimination give the same outputs.

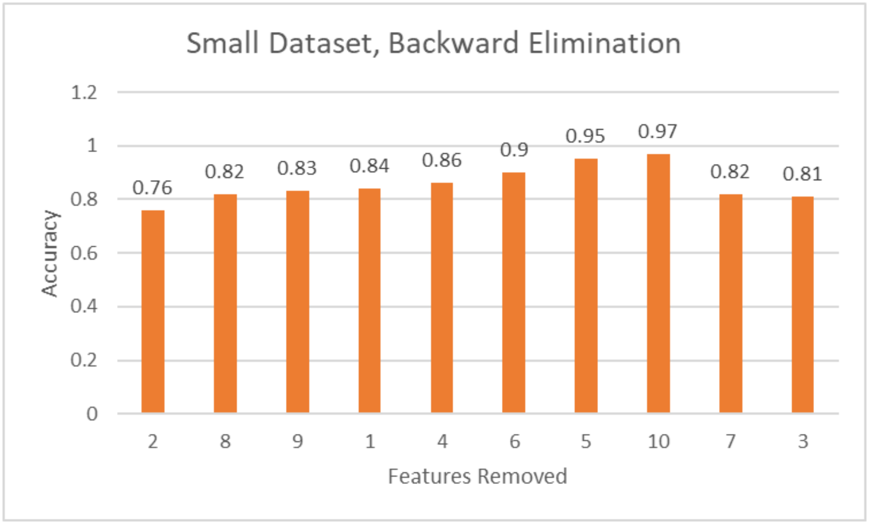


Figure 4. Results for small dataset with Backward Elimination

Results of Large Dataset

Figure 5 shows the results for running forward selection of the CS205_SP_2022_LARGEtestdata__48.txt. We started off by adding feature 21 which gives us 82.8% accuracy. After the next iteration, we added feature 12 which gives us 97.5%. For the rest of the run, we got worse and worse performance. After the run, we concluded that our best set of features for the large dataset when running forward selection is features 21 and 12 with an accuracy of 97.5%.

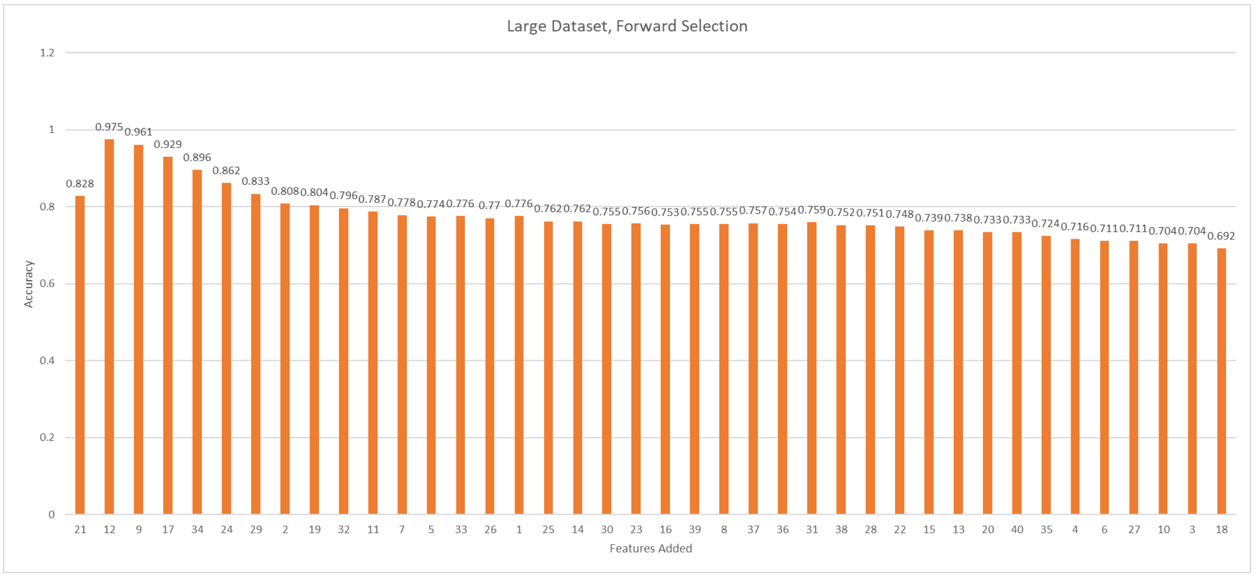


Figure 5. Results for large dataset with Forward Selection

However, running backward elimination on the large data set gave us a different story which is shown in figure 6. We started off with a quite low accuracy of 70.4% when removing feature 18. After that, it just fluctuated when removing more features. The graph looks quite curved if we fit a quadratic line across it. The most surprising thing is we got our best results at the very end when all of the features are removed. We got an accuracy of 80.7% when just purely predicting the classes.

In my opinion, one of the reasons for this difference in results between forward selection and backward elimination is the order of selecting of features. There is no guarantee of a strict rule of which feature is going to get added or removed. Each algorithm has its own approach to iterating through the search tree and choosing the right features. Especially in the very large dataset where there are lots of randomness and variations. Some algorithms are better at handling bad features than others. The choice of distance formula night also is one of the factors that dictate the outcome of the implementation.

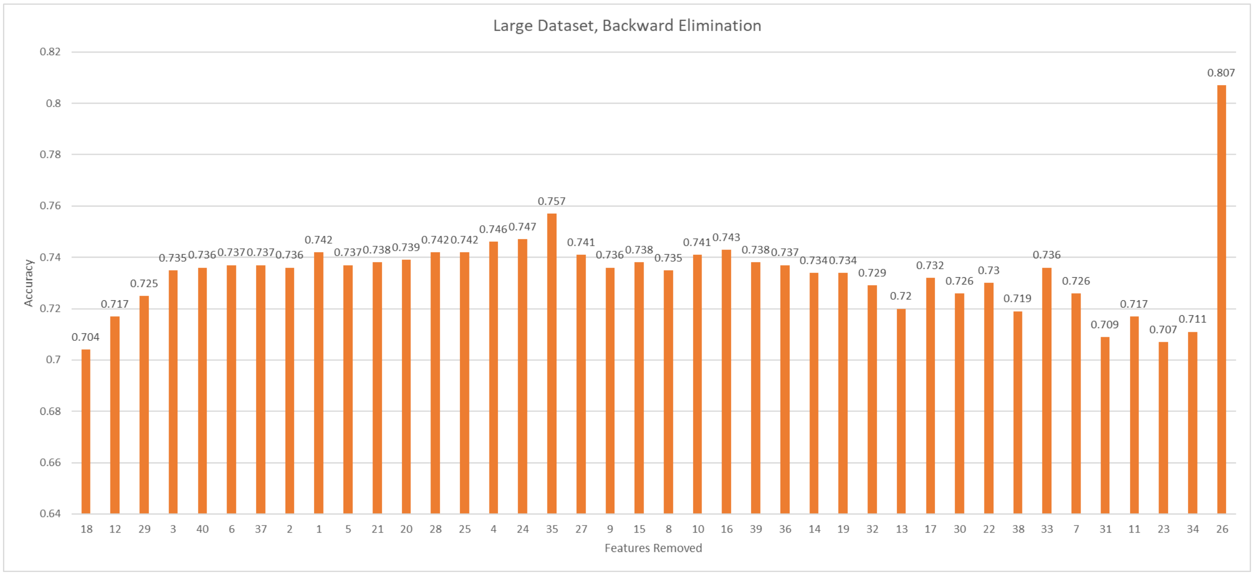


Figure 6. Results for large dataset with Forward Selection

Computation Time

I implemented the project using an AMD Ryzen 7 5800X 8-Core Processor 3.80GHz with 16GB DDR4 3200MHZ on Visual Studio Code with Jupyter Notebook Extension. Table 1 shows the comparison of the execution time between both approaches and on both datasets.

	Small Dataset (10 features, 300 instances)	Large Dataset (40 features, 1000 instances)
Forward Selection	11.09 seconds	1.49 hrs
Backward Elimination	15.89 seconds	2.72 hrs

Table 1. Execution time of both algorithms on both datasets

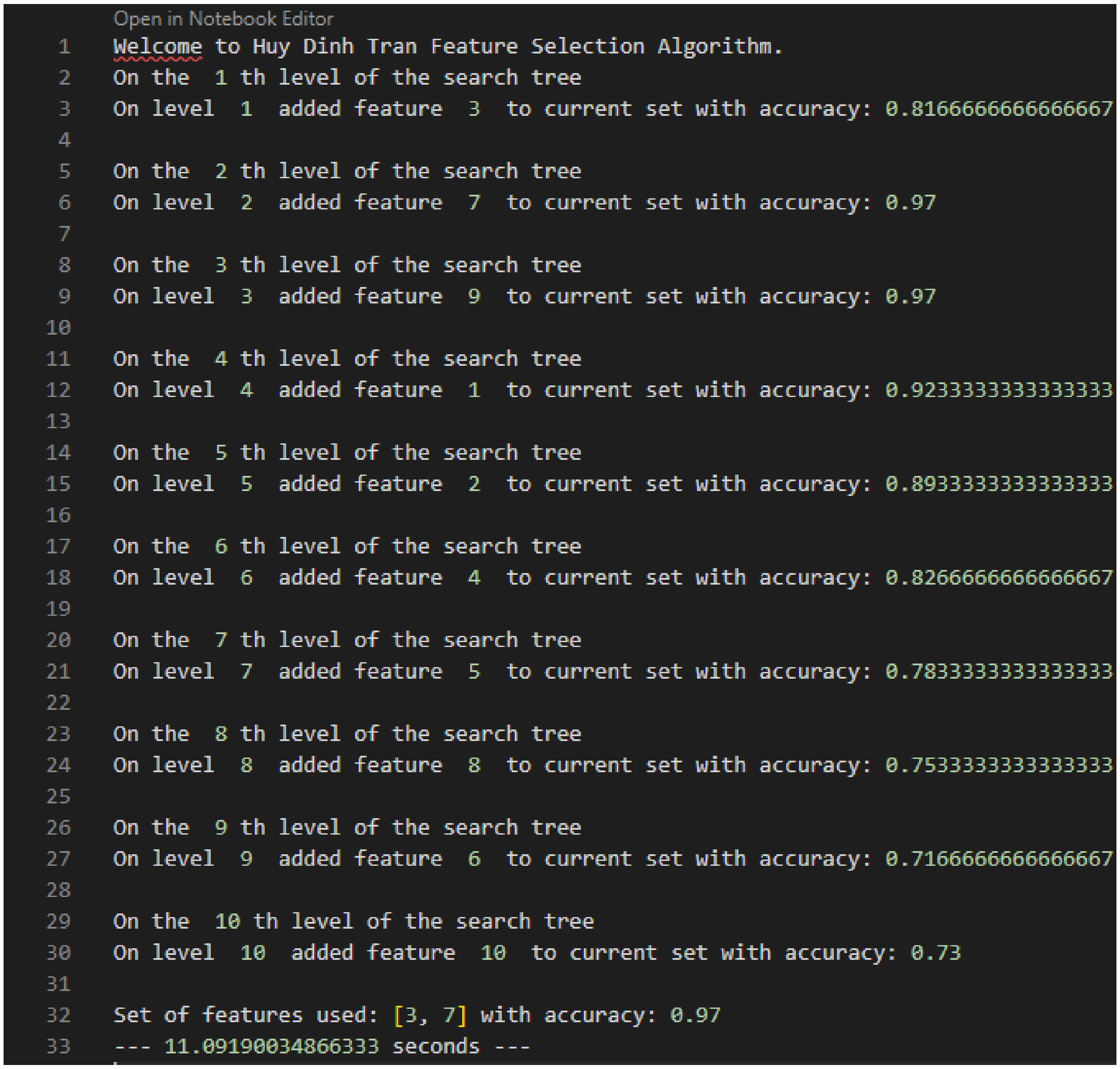
One of the reasons behind the high execution time is that I wrote the calculation for the distance value in the leave_one_out_cross_validation function differently compared to the pseudo-code provided. I created another for loop to run through each data point for summing and square root outside of the loop to calculate the Euclidean distance. I attempted to follow the method proposed from the pseudo-code but didn't have success implementing it. With this major modification, we, unfortunately, increased the execution time by a big degree.

It is also interestingly found that my backward elimination also runs much slower compared to my forward selection. I'm not too sure what is the reason behind this difference since it should perform the same in theory. My best guess might be because of the method of calculating distance I mentioned above or it might be because of Visual Studio Code or Jupyter Notebook.

Conclusion

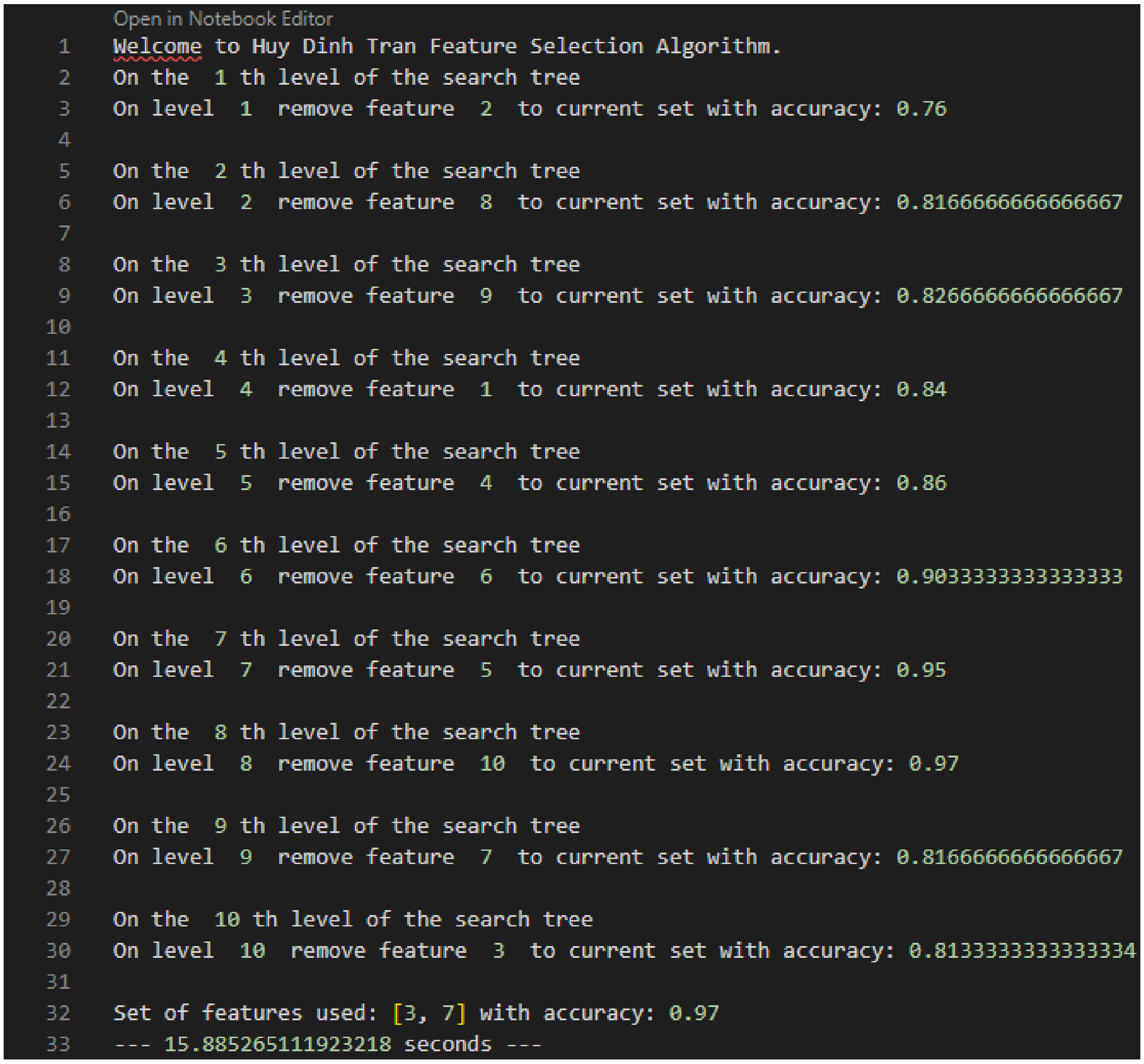
For the small dataset, we can clearly see that both forward Selection and backward Elimination give the same output of the best set of features and also the accuracy. It is true that both algorithms can perform the same in some situations since one is looping forward and the other one just doing the opposite. However, the large dataset proved that not all approaches are similar. Both forward selection and backward elimination give us different sets of features and also accuracies. It is our task to test each approach and choose the best result for our case. In this case, forward selection gives us not only better results but also took lesser time to compute.

Trace of Small Data with Forward Selection


Open in Notebook Editor

```
Welcome to Huy Dinh Tran Feature Selection Algorithm.
On the  1 th level of the search tree
On level  1  added feature  3  to current set with accuracy: 0.8166666666666667

On the  2 th level of the search tree
On level  2  added feature  7  to current set with accuracy: 0.97

On the  3 th level of the search tree
On level  3  added feature  9  to current set with accuracy: 0.97

On the  4 th level of the search tree
On level  4  added feature  1  to current set with accuracy: 0.9233333333333333

On the  5 th level of the search tree
On level  5  added feature  2  to current set with accuracy: 0.8933333333333333

On the  6 th level of the search tree
On level  6  added feature  4  to current set with accuracy: 0.8266666666666667

On the  7 th level of the search tree
On level  7  added feature  5  to current set with accuracy: 0.7833333333333333

On the  8 th level of the search tree
On level  8  added feature  8  to current set with accuracy: 0.7533333333333333

On the  9 th level of the search tree
On level  9  added feature  6  to current set with accuracy: 0.7166666666666667

On the  10 th level of the search tree
On level  10  added feature  10  to current set with accuracy: 0.73

Set of features used: [3, 7] with accuracy: 0.97
--- 11.09190034866333 seconds ---
```

Trace of Small Data with Backward Elimination



Open in Notebook Editor

```
Welcome to Huy Dinh Tran Feature Selection Algorithm.
On the  1 th level of the search tree
On level  1  remove feature  2  to current set with accuracy: 0.76

On the  2 th level of the search tree
On level  2  remove feature  8  to current set with accuracy: 0.8166666666666667

On the  3 th level of the search tree
On level  3  remove feature  9  to current set with accuracy: 0.8266666666666667

On the  4 th level of the search tree
On level  4  remove feature  1  to current set with accuracy: 0.84

On the  5 th level of the search tree
On level  5  remove feature  4  to current set with accuracy: 0.86

On the  6 th level of the search tree
On level  6  remove feature  6  to current set with accuracy: 0.9033333333333333

On the  7 th level of the search tree
On level  7  remove feature  5  to current set with accuracy: 0.95

On the  8 th level of the search tree
On level  8  remove feature  10  to current set with accuracy: 0.97

On the  9 th level of the search tree
On level  9  remove feature  7  to current set with accuracy: 0.8166666666666667

On the  10 th level of the search tree
On level  10  remove feature  3  to current set with accuracy: 0.8133333333333334

Set of features used: [3, 7] with accuracy: 0.97
--- 15.885265111923218 seconds ---
```