Project: 2 CS 205. Introduction to Artificial Intelligence

**Due Date: June 7th at 11:59pm 2025**

**Feature Selection with Nearest Neighbor**

Important: I am leaving the country for two weeks shortly after this deadline. I will be offline the whole time. Therefore, I will not offer any extensions. If you are late, you will have to take a low/failing grade, or ask for an incomplete (which is a huge pain for both of us).

Note that there are two parts to the project, finish Part 1 completely before attempting Part 1

**Part 1:** As we have seen in class this quarter, the nearest neighbor algorithm is a very simple, yet very competitive classification algorithm. It does have one major drawback however; it is very sensitive to irrelevant features. With this in mind you will code up the nearest neighbor classifier, and then use it inside a “wrapper” which does two kinds of searches (listed below)

1. Forward Selection
2. Backward Elimination

Don’t be scared by the phrase “search algorithm”, in this case it is really simply nested loops, nothing else.

To make life simple, you can assume the following. I will only give you datasets that have two classes. I will only give you datasets that have continuous features.

Think carefully before you start coding this. Students in the past seem to have made this more complicated than it needed to be. In particular, in Matlab I was able to write the nearest neighbor algorithm in 8 lines of code, and the two search algorithms in another 17 lines of code.

C++ and Java programs tend to be longer, but even so, I would be surprised if this took more than 100 lines of code (although I will not penalize you for this).

**Very important**: Make sure your nearest neighbor algorithm is working correctly before you attempt the search algorithms. We will provide some test datasets for this purpose.

You may use some predefined utility routines, for example sorting routines. However, I expect all the major code to be original. You must document any book, webpage, person or other resources you consult in doing this project (see the first day’s handout).

You may consult colleagues at a high level, discussing ways to implement the tree data structure for example. But you may **not** share code. At most, you might illustrate something to a colleague with pseudocode.

You will hand in a report, see Project\_2\_sample\_report.pdf

You must keep the evolving versions of your code, so that, if necessary you can demonstrate to the course staff how you went about solving this problem (in other words, we may ask you to prove that you did the work, rather than copy it from somewhere).

You can use a simple text line interface or a more sophisticated GUI (but don’t waste time making it pretty unless you are sure it works and you have lots free time). However, your program should have a trace like the one below, so that it can be tested.

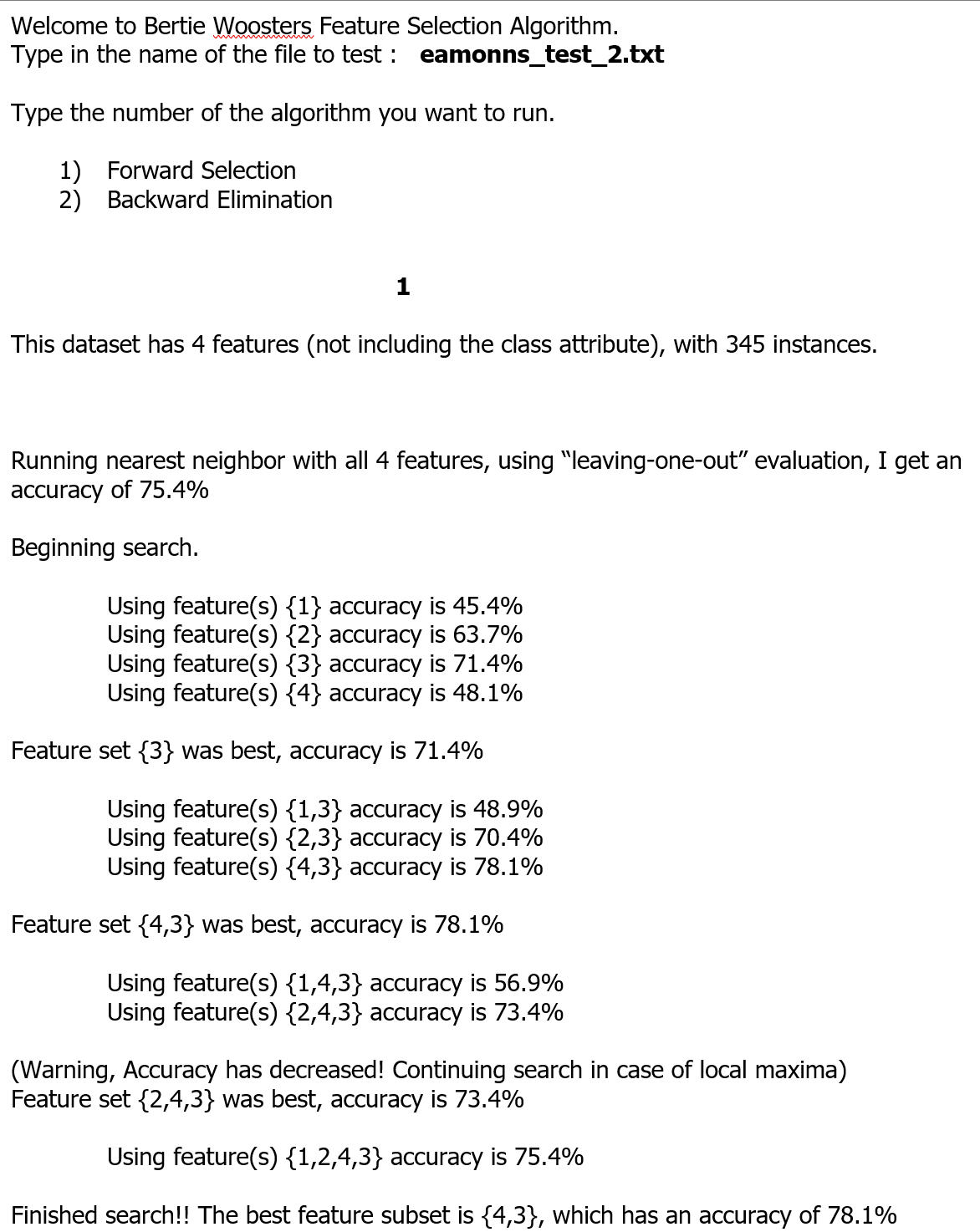
The data files will be in the following format. ASCII Text, IEEE standard for 8 place floating numbers. This is a common format; you should be able to find some code to load the data into your program, rather that writing it from scratch (as always, document borrowed code). The first column is the class, these values will always be either “1”s or “2”s. The other columns contain the features, which are **not** normalized. There may be an arbitrary number of features (for simplicity I will cap the maximum number at 64). There may an arbitrary number of instances (rows), for simplicity I will cap the maximum number at 2,048. Below is a trivial sample dataset. The first record is class “2”, the second is class “1” etc. This example has just two features.

2.0000000e+000 1.2340000e+010 2.3440000e+000

1.0000000e+000 6.0668000e+000 5.0770000e+000

2.0000000e+000 2.3400000e+010 3.6460000e+000

1.0000000e+000 4.5645400e+010 3.0045000e+000



Below is a key to the datasets you must use. For example, if you are the first student listed, you should use CS205\_small\_Data\_\_22.txt and CS205\_larger\_Data\_\_45.txt

|  |  |
| --- | --- |
|  |  |

**Part 2**

As this URL [a] are about 400 real-world machine learning datasets.

There are also 100s of datasets on Kaggle [g]

Not all of them are in a format that suits your code, but at least half are.

There may be some very slight massaging of data format you have to do.

For example, we have the class label in the first column, some may have the class label in the last column.

When you look at the “Attribute Types” (which is another word for “feature types”) some are

* Categorical
* Integer
* Real
* Some combinations of the above.

Avoid “Categorical” datasets.

If you like a dataset that has a mixture of Categorical and real/integer, you can use it, after you delete the Categorical features.

---

You need to pick one of these datasets. For concreteness I am going to talk about the glass dataset, but this is the only one you *cannot* pick. The glass dataset has 9 features, they are

1. RI: refractive index
2. Na: Sodium
3. Mg: Magnesium
4. Al: Aluminum
5. Si: Silicon
6. K: Potassium
7. Ca: Calcium
8. Ba: Barium
9. Fe: Iron

This is just like your small dataset, In this dataset, the first column is just the data entry number, it has no meaning, so I deleted it).

While your small dataset had two class, this one has more classes, they are:

-- 1 building\_windows\_float\_processed  
-- 2 building\_windows\_non\_float\_processed  
-- 3 vehicle\_windows\_float\_processed  
-- 4 vehicle\_windows\_non\_float\_processed (none in this database)  
-- 5 containers  
-- 6 tableware  
-- 7 headlamps

I would prefer it if everyone had a unique dataset. If two or more people use the same dataset, I will carefully examine the reports to look for evidence of collusion. If someone else happens to pick the same dataset as you, you should be OK, so long as you work 100% independently of them.

--

Now here is the problem.

Run forward search (only forward) on your chosen dataset, and ..

1. Make a plot, like you did for your SMALL and LARGE, that shows the order the features were added and the accuracy.
2. Write a short report, that explains why you think the first features added are the best features. For example (below I am just making stuff up). The report should be 6 to 20 sentences.

I found that forward search first added the feature “RI: refractive index”. This makes sense, because according to [d] “refractive index is typically highly predictive of class type”. The next feature added was Silicon, this makes sense, as [e] says there is a lot of Silicon in tableware and container glass, but not in building\_windows, so this feature can help discriminate these types of glass. Finally, I noted that search seems to suggest that both Potassium and Barium are useless features for this problem. Not only do they not help, but they actually seem to hurt accuracy.

---

That is it.

There is one extra step that you might need to take for the features. You will probably have to normalize them [f]. This is a one-time step, you can even do it in excel.

For every feature (not including the class label) subtract its mean, then divide by the standard deviation.

In Matlab, that is one line of code. For column X, I would do

>> X= (X-mean(X))/std(X);

Ot use matlab’s built-in function

>> X = zscore(X);

Finally, there is one other trick you can do, if the size of the dataset is small, and you have lots of classes.

You can combine similar classes.

For example, suppose you had three classes {lion,tiger, hippo}

You could combine the two similar classes, to make this {BigCat, hippo}

If you do that, mention it in your report.

[a] <https://archive.ics.uci.edu/ml/datasets.php>

[b] <https://archive.ics.uci.edu/ml/datasets/Glass+Identification>

[c] <https://archive.ics.uci.edu/ml/machine-learning-databases/glass/glass.data>

[d] J Forensic Sci. 2009 Jan;54(1):49-59. doi: 10.1111/j.1556-4029.2008.00905.x. Epub 2008 Nov 6.

Classification of glass fragments based on elemental composition and refractive index. Grzegorz Zadora

[e] <https://en.wikipedia.org/wiki/Glass>

[f] https://www.statology.org/z-score-normalization/

[g] https://www.kaggle.com/datasets?search=classification