Project 2: Feature Selection with Nearest Neighbor

Student Name: Zheming Kang SID:X708079

Solution: <the datasets are your uniquely assigned datasets>

Dataset	Best Feature Set	Accuracy
Small Number: 38	Forward Selection = {3, 7, 9}	0.9
	Backward Elimination = {1, 2, 7}	0.91
	Custom Algorithm = {Ramdom features}	Around 0.6
Large Number: 38	Forward Selection = {9, 30}	0.95
	Backward Elimination = { 2, 4, 7, 8, 10, 11, 14, 15, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 31, 32, 33, 36, 38, 39, 40 }	0.803
	Custom Algorithm = {Random features}	Around 0.54

----------<Begin Report>----------

In completing this project, I consulted the following resources:

<mention your resources here>

Python Random - random() Function - GeeksforGeeks

numpy.delete — NumPy v1.24 Manual

Python Time Module - GeeksforGeeks

Additional materials and slides in our course folder

numpy.loadtxt() in Python - GeeksforGeeks

I. Introduction

This project followed the direction of Project Briefing Slides, with tested classifier, validation functions, and feature selection functions. In this project, there are two parts, which are the feature selection part, and the classification part. In the first part, I conducted a forward selection function and a backward elimination function. In the second part, I wrote two validation functions and a random return function. Then, I put them together and get the whole project.

II. Challenges

- How to get specific cols in the data.

In the validation functions, we need to select specific columns to get a new data table. Since I used Numpy, I can use “data = [:, feature_set]” to get the specific columns.

- How to get the label in the data
  In the text files, all labels are float numbers but we need a single integer to label the rows in our program. Since python will read the text file and store them as a string, I can get the “data[i][0]” as the first string, which is “float label”. Then, I read the first element in the string in “data[i][0][0]” and turn it into an integer by int() function. Then, I got an integer label.
- How to read the data files.
  I used numpy.loadtxt function but found the delimiter of double spaces is not work correctly. The reason is that if there is a negative number in the data file, the “-” will take up a space. As a solution, I wrote a wash data function in the NN file, open the file, and read the number of the space before the “ -”(space -). If there is only one space ahead of the ‘-’, we add one space. If not, we continue. Then, we write the file back to “cleaned_file.txt”, and use this cleaned file as input.
- How to prevent the change of feature sets in validation
  When I do the validation with the feature set, I found if I remove the feature in the validation function, it will reflect on the global feature set. That is because the parameter is the reference so we need to import the deep copy. Make a deep copy, and use the copied set to do the calculation.

III. Code Design

In the folder, you can see a main.py, an NN.py, an RDM.py, a Forward selection file, and a Backward elimination file. You can also see two datasets.

Let’s start with the main file. In the main file, there is the menu. It will output the instruction for my project. You will need to input a number to choose the dataset, and a number to choose the method. Then, it will wash the data, and do the calculation.

Then, let’s see the ForwardSelection file. In this file, you can see the current feature set, and global variables immediately. Then, we come into a for loop to select all the features. Each time we will consider one feature, we consider all other features to do the leave-one-out cross-validation. We keep the best feature for this level, and in the last level, we will see if the best score in this level is greater than the global best score. If the current one is smaller than the global one, we will stop here and output the answer. Actually, in the beginning, I’ll test all possibilities which means I won’t stop when the accuracy decreased. Then I went to see the professor and she told me I could stop here or do one more iteration to see if we could escape from the trap.

The Backward one is very similar to the forward one. The RDM is also a very similar file, with the difference of changing the accuracy function.

In the NN file, there are 3 major functions here. The first one is live one out cross validation, the second is anti-loocv function, and the third one is randomly GUESS

function. In the loocv func, we will firstly make a deep copy to prevent the unexpected change, then, we will add the label to “data” variable. After that, we will do the live-one-out-cross-validation with Euclidean distance and return the accuracy. In the anti-function, you can see the difference is that at the beginning, we calculate the accuracy of the whole current set. I do this in order to preserve the feature that will be removed so that it won’t lose accuracy. The GUESS function is totally randomly guessing. The wash data function will add a space in front of the “-” as I mentioned in part II. There is also a main function to test whether the function is correct.

IV. Dataset details

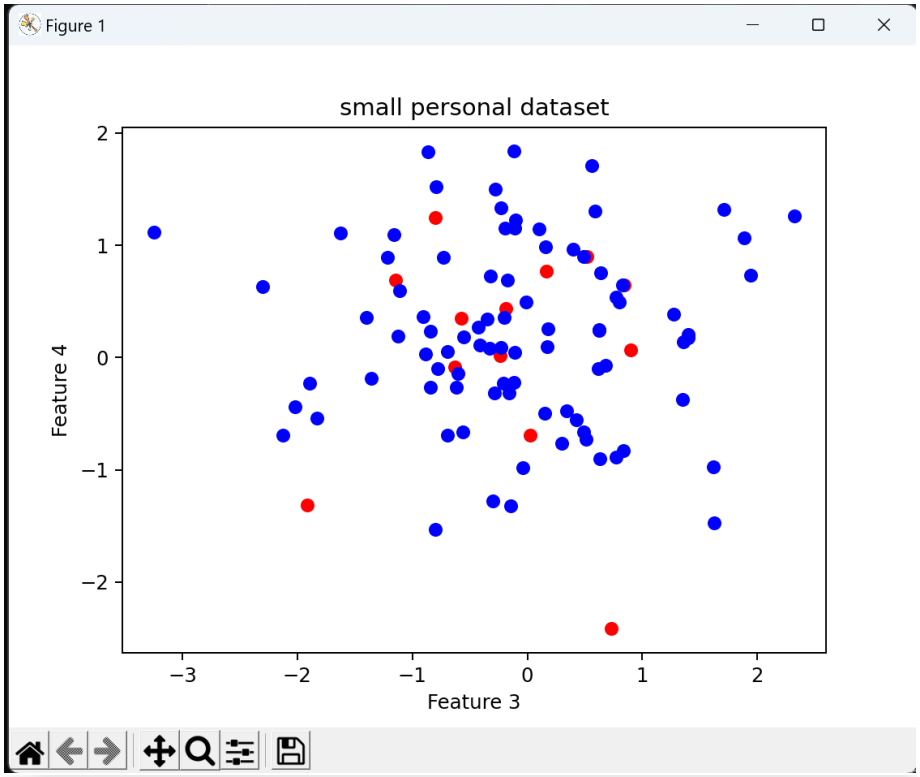
The General Small Dataset: 10 features, 100 instances

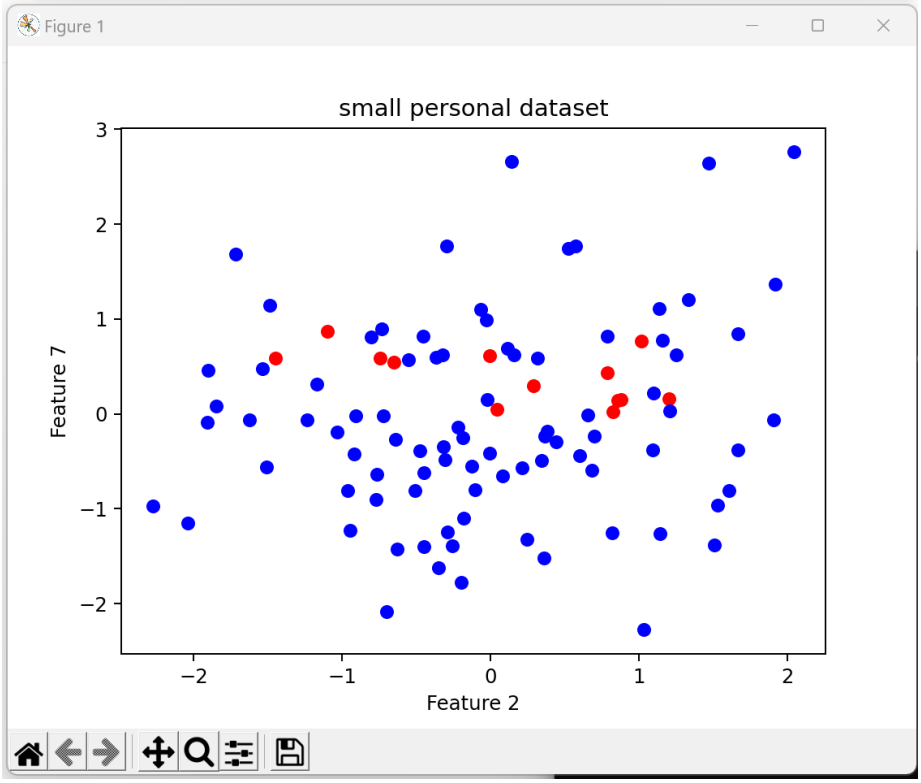
The General Large Dataset: 40 features, 1000 instances

Your Small Dataset: Number of features: 10, number of instances: 100

Your Large Dataset: Number of features: 40, number of instances: 1000

Plot some features and color code them by class and explore your dataset.





V. Algorithms

1. Forward Selection

Start with an empty feature set. Each time we add one feature and calculate the Euclidean distance to find the nearest number to judge the label of nodes. When the accuracy decreased, we stopped.

2. Backward Elimination
   Start with a whole feature set. Each time we remove a feature and do the same things as the Forward Selection. When the accuracy decreased, we stopped.
3. Zheming's random guess algorithm (optional)

   Each time we return a random label. If the frequency of the two labels is equal, the result should be around 50%. If the frequency is not equal, the result might around another number.

VI. Analysis

Experiment 1:

- Comparing Forward Selection vs Backward Elimination.

In forward selection, we add features one by one. When accuracy decreases, we stop. In this case, we will probably choose the most relevant features. However, in backward elimination, we remove features one by one. In that cases, we will probably remove the most irrelevant features.

Since we will stop when accuracy decreases, the forward selection will perform better if there exist some essential features when the data is super large. However, it might not give us a better combination of features than backward elimination when the data is small.

In my experiment, when I use the large dataset, the forward selection stopped in the second round with 0.95 accuracies yet the backward stopped in the 14^{th} round with an accuracy of 0.803. That shows the relevant features will be quickly preserved by forward selection. However, when I change to the small dataset, the forward provide me the feature set of [3, 7, 9] and the backward provide me the feature set of [1, 2, 7]. You can see 7 in both sets which means it is a very essential relevant feature. The answer is [3, 7, 9]: 0.90 < [1, 2, 7]: 0.91. When the first time we insert the 3 in the forward function, we will lose the combination of [1, 2, 7] because 3 is the current best score provider.

- Compare accuracy with no feature selection vs with feature selection.

When we didn't choose a feature, we have two choices: randomly return an answer, or use the full feature set.

If we randomly return an answer, we will get the frequency of the dataset. If we assume the probability of the answer of "validation" is 50%, the accuracy will reflect the frequency of the labels in the raw dataset.

If we choose the whole feature set, we will use many irrelevant features. These irrelevant features might have a bad effect on our judgment. As a result, the accuracy is

lower than the feature selection. (That is the reason why backward elimination will not stop at the beginning)

- Compare feature set and accuracy for forward selection vs backward elimination.

  When the dataset is small, these two functions will provide us with feature sets of similar size. When the dataset is large, forward selection will provide a much smaller set, while the backward elimination will provide a much larger one.

  Talk about pros and cons of both algorithms.

  The forward selection will be faster and can quickly give us the major features. However, it will lose the combination of the global best answer. The backward elimination will be slower but might sometimes give us a better combination of the answer.

Experiment 2: Effect of normalization

Compare accuracy when using normalized vs unnormalized data.

The normalized data will provide higher accuracy. When the data is unnormalized, we will get a noisy answer in Euclidean distance computing. If we normalize them into the same range, this will have few effects.

Experiment 3: Effect of number neighbors (k)

Plot accuracy vs increasing values of k and examine the trend.

If we increase the k, we will get a higher accuracy since there will be a better classifier with a higher k than the lower k. For example, if k is 1, we will be misled by the noise yet when k is 3, it will be more likely to give us the correct answer.

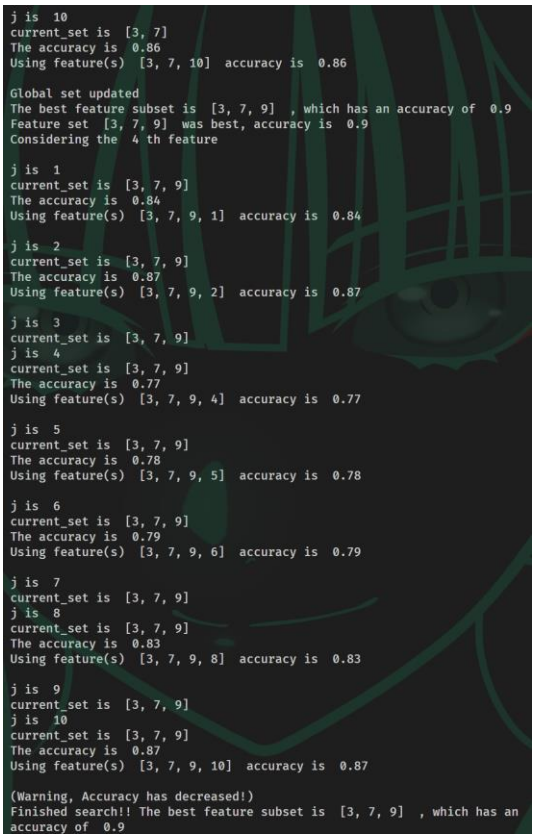
VII. Conclusion

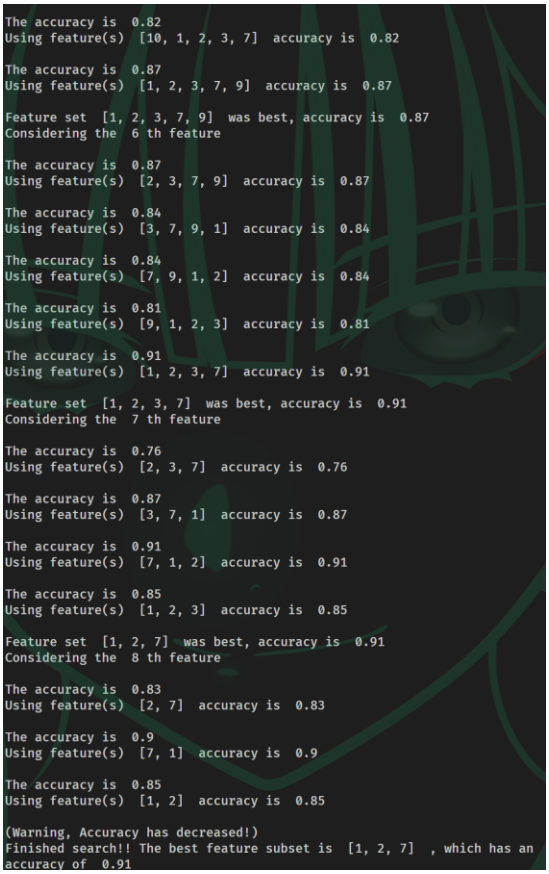
General summary of your findings from the Analysis. Potential improvements to this approach of doing feature selection.

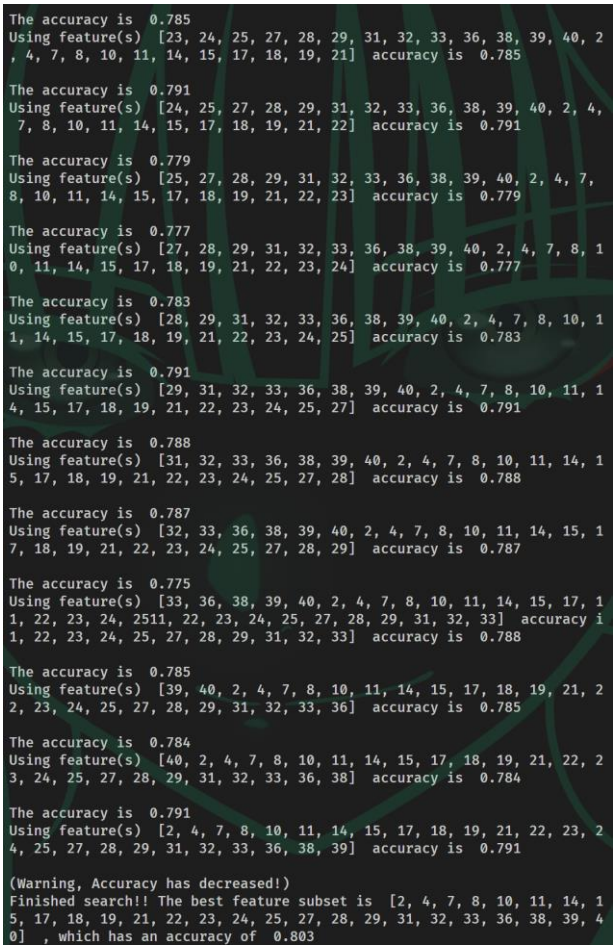
a. Both backward elimination and forward selection will give us locally optimized answers.
b. We need to use both of these functions to get the global optimized feature set.
c. Cleaning the dataset is of vital importance.
d. We can train more times to get higher accuracy but it might be useless when the number of iteration is too many.

VIII. Trace of your small dataset

See next page.




The accuracy is 0.82
Using feature(s) [10, 1, 2, 3, 7] accuracy is 0.82
The accuracy is 0.87
Using feature(s) [1, 2, 3, 7, 9] accuracy is 0.87
Feature set [1, 2, 3, 7, 9] was best, accuracy is 0.87
Considering the 6 th feature
The accuracy is 0.87
Using feature(s) [2, 3, 7, 9] accuracy is 0.87
The accuracy is 0.84
Using feature(s) [3, 7, 9, 1] accuracy is 0.84
The accuracy is 0.84
Using feature(s) [7, 9, 1, 2] accuracy is 0.84
The accuracy is 0.81
Using feature(s) [9, 1, 2, 3] accuracy is 0.81
The accuracy is 0.91
Using feature(s) [1, 2, 3, 7] accuracy is 0.91
Feature set [1, 2, 3, 7] was best, accuracy is 0.91
Considering the 7 th feature
The accuracy is 0.76
Using feature(s) [2, 3, 7] accuracy is 0.76
The accuracy is 0.87
Using feature(s) [3, 7, 1] accuracy is 0.87
The accuracy is 0.91
Using feature(s) [7, 1, 2] accuracy is 0.91
The accuracy is 0.85
Using feature(s) [1, 2, 3] accuracy is 0.85
Feature set [1, 2, 7] was best, accuracy is 0.91
Considering the 8 th feature
The accuracy is 0.83
Using feature(s) [2, 7] accuracy is 0.83
The accuracy is 0.9
Using feature(s) [7, 1] accuracy is 0.9
The accuracy is 0.85
Using feature(s) [1, 2] accuracy is 0.85
(Warning, Accuracy has decreased!)
Finished search!! The best feature subset is [1, 2, 7] , which has an accuracy of 0.91



<- This is the trace of large dataset.