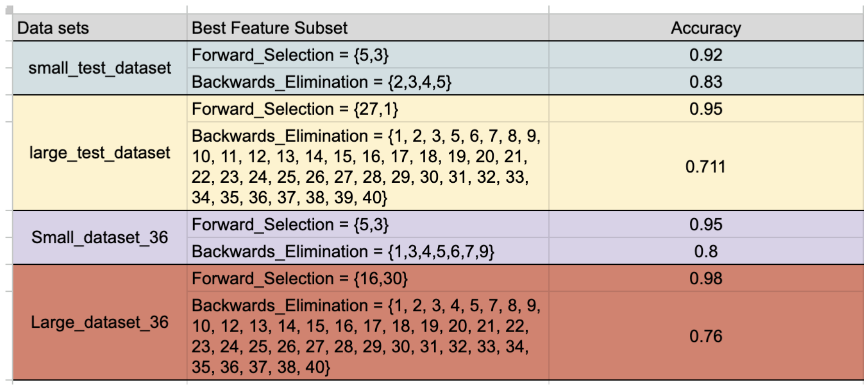
Name: Nicholas Martinez SID: nmart130 Dataset number: 36

SOLUTION



Data sets	Best Feature Subset	Accuracy
small_test_dataset	Forward_Selection = {5,3}	0.92
	Backwards_Elimination = {2,3,4,5}	0.83
large_test_dataset	Forward_Selection = {27,1}	0.95
	Backwards_Elimination = {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}	0.711
Small_dataset_36	Forward_Selection = {5,3}	0.95
	Backwards_Elimination = {1,3,4,5,6,7,9}	0.8
Large_dataset_36	Forward_Selection = {16,30}	0.98
	Backwards_Elimination = {1, 2, 3, 4, 5, 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, 40}	0.76

I. INTRODUCTION

Using feature selection algorithms; Forward Selection and Backwards Elimination, we attempt to search for the subset of features that produce the best accuracy when Nearest Neighbor classification is applied.

II. CHALLENGES

The design and minimization of runtime were the most challenging components of the project. The Design of the program had a great influence on the overall difficulty of the project itself. In particular there was a difficulty in the interactions between objects, and how the data was moved between them. With the idea to minimize unnecessary computations of distances, loading of data, etc. I attempted to take a dynamic programming approach, which led to only a mess of bugs, and as the deadline neared I abandoned that approach.

While implementing the algorithms to function correctly was the first obstacle to overcome,implementation of feature selections to function efficiently was the one to follow. When applying the feature searches to small data sets containing 100 instances, inefficiencies were easily overlooked. This was not the case for the larger datasets containing 1000 instances, where the inefficient implementations were emphasized. As my earlier approaches to dynamic programming only showed failures one would assume that it would not be attempted again. So after I tried dynamic programming again and failed, I looked for a different solution to minimization of runtime. The solution I found was to minimize the amount times the sort function is called to sort the neighbors of the current node. At first I would call the sort function anytime a neighbor was added to the nodes neighbor list, afterwards I made this to only be called when classifying a node drastically improving my runtime.

III. CODE DESIGN

A. Data and Search Module

The Data module contains the functions used to load, convert, normalize[1], and to find the default rate of the given dataset. The Search module contains the implementations of Forward Selection and Backwards Elimination.

B. Class Validator

Contains two class methods; Both the methods are used during feature selection. When `validator.leave_one_out_cross_validation(args)` is called we compute distances for the current node and then based on those distances we find the nearest neighbor.

C. Class NN_Classifier

Holds methods that pertain specifically to the nearest neighbor classification, such as computing the euclidean distances and such.

D. Class Node

The Node object stores the data of each instance/row of data. It is used within the majority of the functions, to easily retrieve features and assign data to the attributes. The neighbors of each node are stored within the object. When classifying the node, the neighbors are sorted in ascending order and the top neighbor is retrieved.

IV. DATASET DETAILS

A. Small Test Dataset

Number of Features: 10
Number of Instances: 100

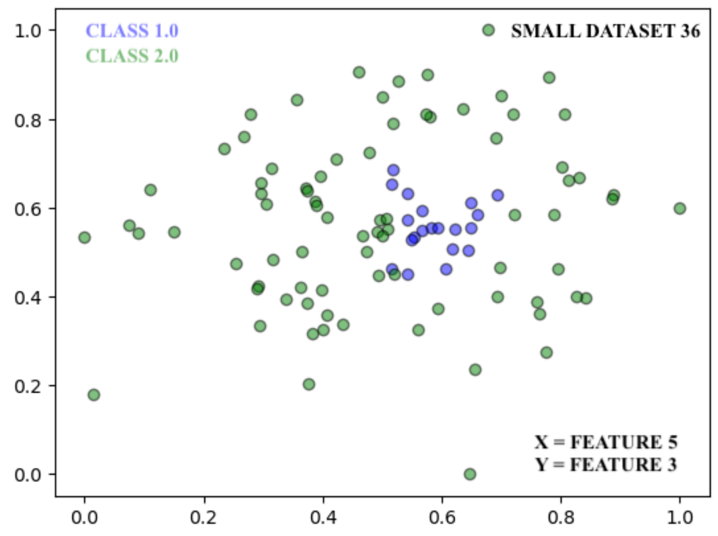
B. Large Test Dataset

Number of Features: 40
Number of Instances: 1000

C. Small Dataset 36

Number of Features: 10
Number of Instances: 100
Default Rate: 0.8
Accuracy using all Features: 0.66

[1] The Min Max Normalization method was used on the data points.



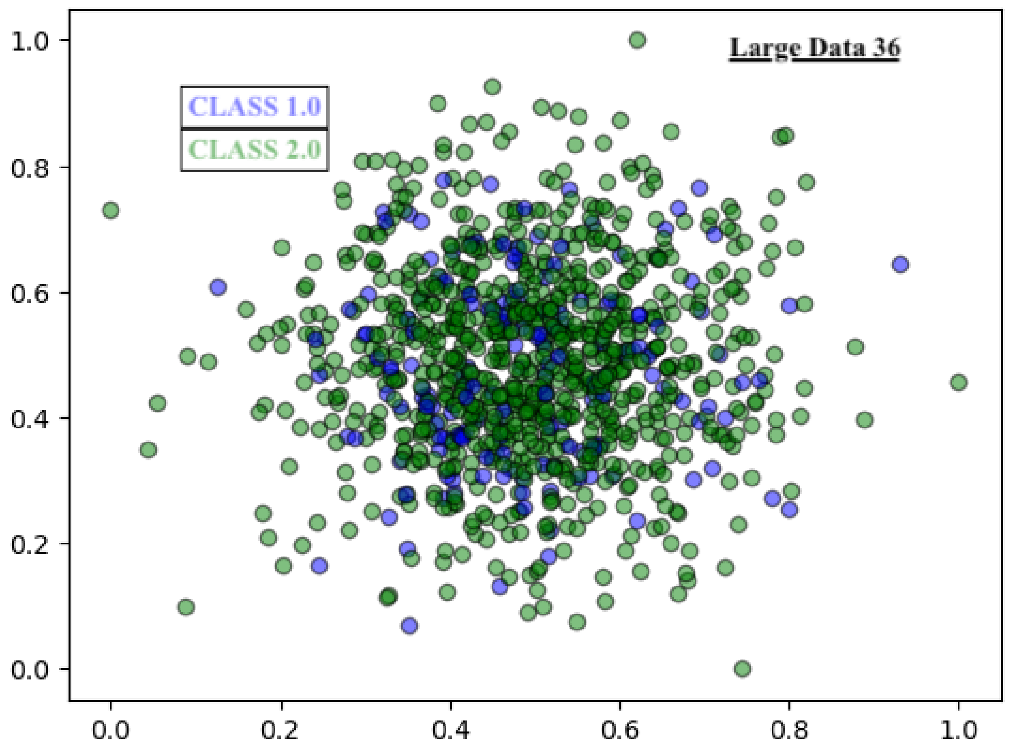
D. Large Dataset 36

Number of Features: 40

Number of Instances: 1000

Default Rate: 0.83

Accuracy using all Features: 0.72



V. ALGORITHMS

A. **Forward Selection**

Starting with an empty set of features, we iterate over the potential features and choose the feature with the maximum accuracy. We then iterate over the features again computing the accuracy of the features in combination with the already selected features. If no combination of features are found to be greater than the greatest current accuracy the current subset of features is returned.

Forward selection is a greedy search algorithm, with the objective to optimize a classification algorithm, by maximizing the classification accuracy using some feature subset, and minimizing the amount of features.

B. **Backwards Elimination**

Backwards Elimination starts off with all features being used to compute the accuracy of the classification. After the accuracy has been determined a feature is dropped, and the accuracy is computed again. If the accuracy of the subset is greater than or equal to the previously calculated accuracy then this implies the dropped feature is irrelevant, and can be discarded.

VI. ANALYSIS

Forward Selection vs. Backward Elimination

A. Forward Selection produces a much smaller subset than Backwards Elimination. This is probably due to the fact that Forward selection starts off with no features selected, while Backwards Elimination begins with all features selected. This means Forward Selection goes through fewer iterations to find a small subset of features with a relatively high accuracy, than Backwards Elimination to reach that same subset, if it ever does. Which leads me to believe that if we know that a current feature set does not contain any irrelevant features then Backwards Elimination would be safe to use, since it will more than likely return a feature subset with more features than Forward Selection. However, if features contribution to the accuracy of a classification is unknown it forward selection seems like a better choice.

B. The accuracy retrieved from Forward Selection is higher than the accuracy retrieved from Backwards Elimination for all datasets. However the subsets for Backwards Elimination are much larger than those retrieved by Forward Selection.

C. Pros and Cons
   - a. Forward Selection
     - i. Pros
       1. Higher Accuracy
       2. Returns a smaller feature subset which allows the focus of the data to be narrowed

            3. Implementation is intuitive
            4. Faster convergence

    ii. Cons
        1. A smaller subset of features also means each feature has a stronger contribution to the accuracy. Which implies that errors made during collection of data may go unnoticed
        2. High Memory allocation (depending on implementation)
        3. Accuracy needs to be computed for a combination of subsets.
        4. The depth of Tree is greater than that of Backwarwards Elimination
        5. Greater potential that classification is over-fitted.

b. Backwards Elimination
    i. Pros
        1. Depth of tree is more shallow than that of Forward Selection
        2. Larger feature subset allows more flexibility for future computations
        3. Less chance of overfitting data

    ii. Cons
        1. Larger feature subset, means a focus on the subject is broader
        2. High Memory allocation
        3. Does not always produce a better accuracy.
        4. Needs to recompute the accuracy of a greater amount of feature subsets
        5. Counter intuitive

VII. Conclusion

Feature Selection greatly improves the accuracy of classifiers. When comparing the accuracy results of a classifier using all features, with a classifier using only a subset of features we notice an increase in accuracy for the classifier when used with a subset of features. However, this is not the case for the default rates of the datasets. The Default Rates seem to be greater than the accuracies produced by Backwards Elimination, whereas Forward Selection seems to produce a greater accuracy than the default rate.

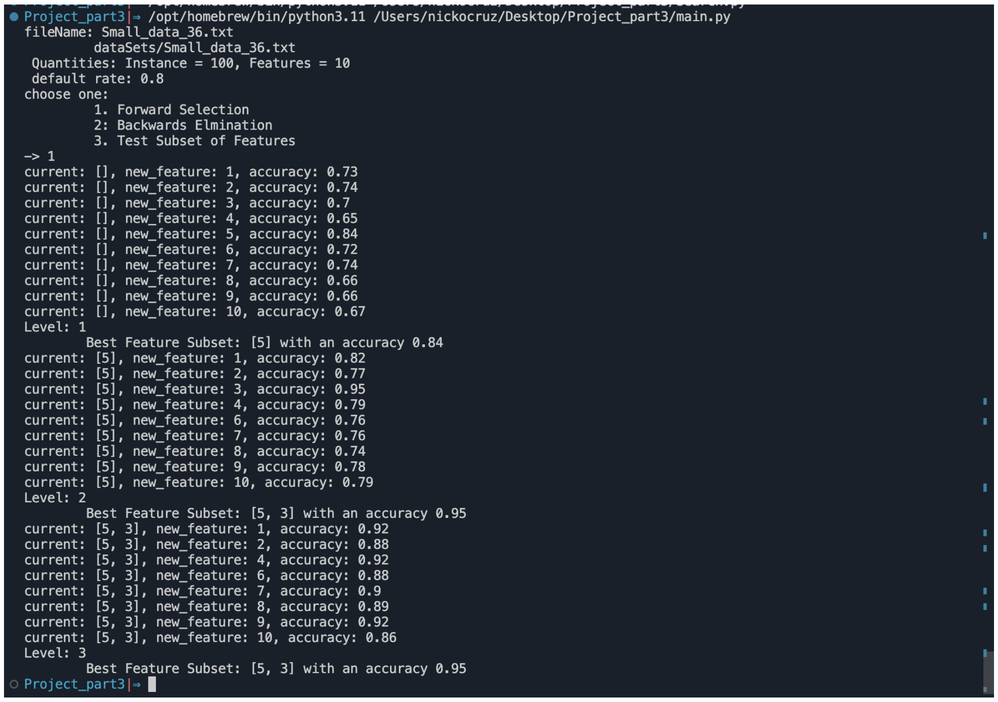
After applying feature selection, the features used during classification are narrowed, as a fortunate consequence this also allows faster computation in the future when classifying new instances. Using Forward Search narrows the features down more so than Backwards elimination, which decreases future runtimes more so. However the great decrease in features after applying Forward Search may indicate that the subset of features has been overfitted.

The cost of applying feature selection to a dataset can be justified by the increase in accuracy. However, there are instances where a feature selection is too expensive such as when the increase in accuracy is negligible. To justify the cost of computing a subset of features certain factors must be determined such as; the desired accuracy compared with the current, if the resulting feature subset be used on future data sets or do the values of features greatly varying indicating the subset of features to be overfitted to the current data set.

While Forward Selection provides a simple model with high accuracy, the potential of the subset features being overfitted is great. Whereas Backwards elimination does not greatly narrow down the subset of features, it does decrease the potential of overfitting which may lead to more accurate future classifications.

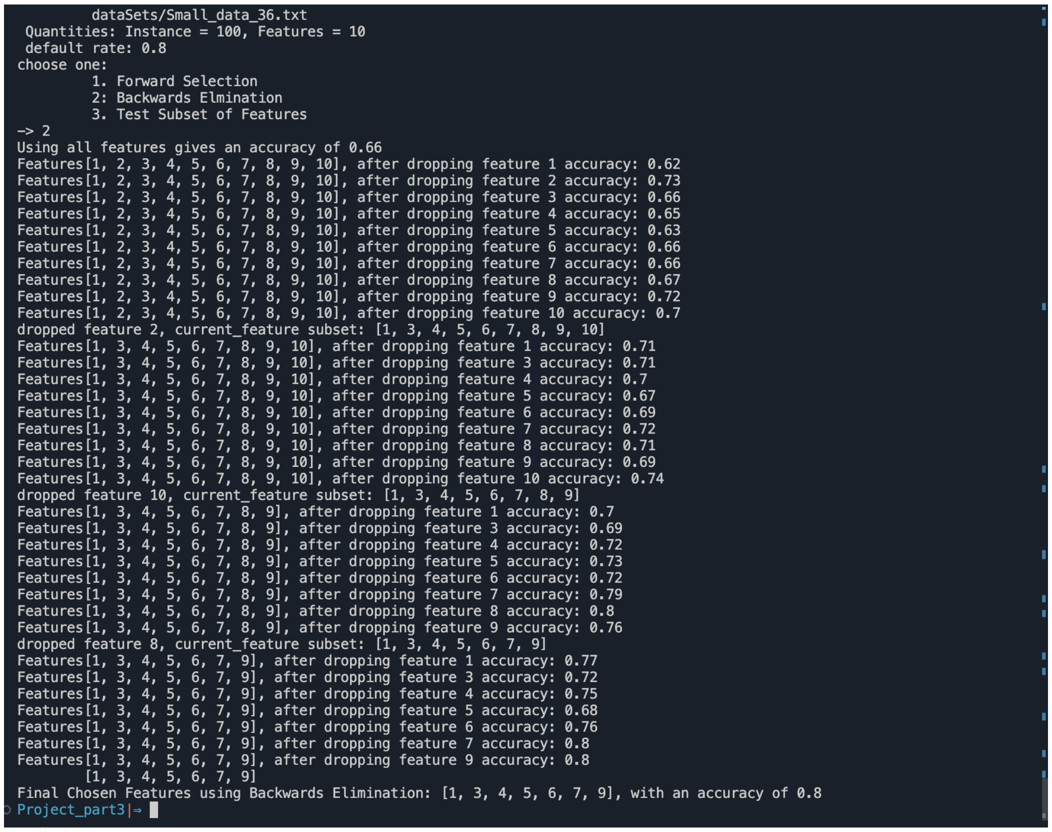
VIII. TRACE

A. Forward Selection



```
Project_part3|⇒ /opt/homebrew/bin/python3.11 /Users/nickocruz/Desktop/Project_part3/main.py
fileName: Small_data_36.txt
         dataSets/Small_data_36.txt
 Quantities: Instance = 100, Features = 10
 default rate: 0.8
choose one:
         1. Forward Selection
         2: Backwards Elmination
         3. Test Subset of Features
-> 1
current: [], new_feature: 1, accuracy: 0.73
current: [], new_feature: 2, accuracy: 0.74
current: [], new_feature: 3, accuracy: 0.7
current: [], new_feature: 4, accuracy: 0.65
current: [], new_feature: 5, accuracy: 0.84
current: [], new_feature: 6, accuracy: 0.72
current: [], new_feature: 7, accuracy: 0.74
current: [], new_feature: 8, accuracy: 0.66
current: [], new_feature: 9, accuracy: 0.66
current: [], new_feature: 10, accuracy: 0.67
Level: 1
        Best Feature Subset: [5] with an accuracy 0.84
current: [5], new_feature: 1, accuracy: 0.82
current: [5], new_feature: 2, accuracy: 0.77
current: [5], new_feature: 3, accuracy: 0.95
current: [5], new_feature: 4, accuracy: 0.79
current: [5], new_feature: 6, accuracy: 0.76
current: [5], new_feature: 7, accuracy: 0.76
current: [5], new_feature: 8, accuracy: 0.74
current: [5], new_feature: 9, accuracy: 0.78
current: [5], new_feature: 10, accuracy: 0.79
Level: 2
        Best Feature Subset: [5, 3] with an accuracy 0.95
current: [5, 3], new_feature: 1, accuracy: 0.92
current: [5, 3], new_feature: 2, accuracy: 0.88
current: [5, 3], new_feature: 4, accuracy: 0.92
current: [5, 3], new_feature: 6, accuracy: 0.88
current: [5, 3], new_feature: 7, accuracy: 0.9
current: [5, 3], new_feature: 8, accuracy: 0.89
current: [5, 3], new_feature: 9, accuracy: 0.92
current: [5, 3], new_feature: 10, accuracy: 0.86
Level: 3
        Best Feature Subset: [5, 3] with an accuracy 0.95
Project_part3|⇒
```

B. Backwards Elimination



```
         dataSets/Small_data_36.txt
 Quantities: Instance = 100, Features = 10
 default rate: 0.8
choose one:
         1. Forward Selection
         2: Backwards Elmination
         3. Test Subset of Features
-> 2
Using all features gives an accuracy of 0.66
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 1 accuracy: 0.62
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 2 accuracy: 0.73
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 3 accuracy: 0.66
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 4 accuracy: 0.65
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 5 accuracy: 0.63
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 6 accuracy: 0.66
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 7 accuracy: 0.66
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 8 accuracy: 0.67
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 9 accuracy: 0.72
Features[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 10 accuracy: 0.7
dropped feature 2, current_feature subset: [1, 3, 4, 5, 6, 7, 8, 9, 10]
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 1 accuracy: 0.71
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 3 accuracy: 0.71
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 4 accuracy: 0.7
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 5 accuracy: 0.67
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 6 accuracy: 0.69
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 7 accuracy: 0.72
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 8 accuracy: 0.71
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 9 accuracy: 0.69
Features[1, 3, 4, 5, 6, 7, 8, 9, 10], after dropping feature 10 accuracy: 0.74
dropped feature 10, current_feature subset: [1, 3, 4, 5, 6, 7, 8, 9]
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 1 accuracy: 0.7
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 3 accuracy: 0.69
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 4 accuracy: 0.72
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 5 accuracy: 0.73
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 6 accuracy: 0.72
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 7 accuracy: 0.79
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 8 accuracy: 0.8
Features[1, 3, 4, 5, 6, 7, 8, 9], after dropping feature 9 accuracy: 0.76
dropped feature 8, current_feature subset: [1, 3, 4, 5, 6, 7, 9]
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 1 accuracy: 0.77
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 3 accuracy: 0.72
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 4 accuracy: 0.75
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 5 accuracy: 0.68
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 6 accuracy: 0.76
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 7 accuracy: 0.8
Features[1, 3, 4, 5, 6, 7, 9], after dropping feature 9 accuracy: 0.8
        [1, 3, 4, 5, 6, 7, 9]
Final Chosen Features using Backwards Elimination: [1, 3, 4, 5, 6, 7, 9], with an accuracy of 0.8
Project_part3|⇒
```