Project 2: Nearest Neighbor

Student Name 1: Ainaz Estiri SID: 862192636 Lecture Session: 5PM

Student Name 2: Jacob Cunningham SID: 862243371 Lecture Session: 11AM

Student Name 3: Binh Le SID: 862183050 Lecture Session: 11AM

Student Name 4: Billy Chau SID: 862179537 Lecture Session: 11AM

Solution: 50

Dataset	Best Feature Set	Accuracy
Small Number 50	Forward Selection = {0, 6, 7}	0.90
	Backward Elimination = {0, 6, 7}	0.90
	Custom Algorithm = Not Implemented	N/A
Large Number 50	Forward Selection = {1,5}	0.965
	Backward Elimination = {1, 28, 22}	0.841
	Custom Algorithm = Not Implemented	N/A

In completing this project, I consulted the following resources:

https://www.w3schools.com/python/matplotlib_scatter.asp

Contribution of each student in the group:

Ainaz Estiri: Implemented Classifier Class and Validator Class.

Jacob Cunningham: Implemented Forward Selection and Backwards Elimination search algorithms.

Binh Le: Implemented the plot programming and contributed to the merging of object oriented code.

Billy Chau: Implemented dataset input formatting and contributed to the merging of object oriented code.

I. Introduction

For project 2, Ainaz, Jacob, Binh, and Billy have completed a Feature Selection Algorithm. For this project, our program takes in a txt dataset and develops an algorithm using either Forward Selection or Backwards Elimination to determine which features are necessary for us to consider.

II. Challenges

One of our main challenges when developing this algorithm was integrating the classifier and validator class and getting them to work with our search functions. We solved this issue by peer coding each of the functions and have a revision of how our data is being interacted with. We also added type annotations to the future classes we implemented so it would be easier to understand how each function is called.

III. Code Design

We preserved separation of concerns by splitting our project into three classes.

- Instance represents a single point in the dataset. The dataset is a list of instances.
- Classifier holds the train and test functions for applying the nearest neighbor algorithm.
- Validator holds the evaluate function for applying leave-one-out evaluation.
- The forward_selection and backwards_elimination functions control the validator by giving it different feature sets to analyze.

This structure allows us to switch out search, evaluation, and training algorithms without changing the logic of the other pieces.

IV. Dataset details

-The General Small Dataset: Number of features: **10**, number of instances: **100**

Forward Selection = Features Selected **{4, 6}** with an accuracy of **92%**

Backwards Elimination = Features Selected **{1, 3, 4, 6, 9}** with an accuracy of **83%**

-The General Large Dataset: Number of features: **40**, number of instances: **1000**

Forward Selection = Features Selected **{0, 26}** with an accuracy of **95.5%**

Backwards Elimination = Features Selected **{26}** with an accuracy of **84.4%**

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

Forward Selection = Features Selected **{0, 6, 7}** with an accuracy of **90%**

Backwards Elimination = Features Selected **{0, 6, 7}** with an accuracy of **90%**

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

Forward Selection = Features Selected **{1, 5}** with an accuracy of **96.5%**

Backwards Elimination = Features Selected **{1, 28, 22}** with an accuracy of **84.1%**

V. Algorithms

The number of feature combinations grows exponentially, so we cannot practically check every combination of features to find the best set. Instead we perform greedy searches on the feature space to estimate the best feature set. We start with one set of features and incrementally change the set to move towards the set with better accuracy. We keep track of the best feature set for each number of features to make sure we don't get stuck at a local optimum. At the end of the search we return the set of features with the overall highest accuracy. There are two ways to perform this search.

Each search takes three arguments: the number of features in the dataset, a validator containing the evaluation algorithm we want to use, and a list of items in the dataset.

The runtime of this algorithm is dependent on the time complexity of the evaluate function. Evaluate has to iterate over the dataset once for each item to perform leave-one-out validation. During each validation it calls test, which iterates over the dataset again to find the nearest neighbor. Therefore the time complexity of this algorithm is O(n^2) with the size of the dataset.

1. Forward Selection
Forward Selection starts with an empty set, which always predicts the class with the highest frequency in the dataset. It then tries to add each feature not yet in the feature set and saves the set with the highest accuracy as the new best option. It then repeats this process until the maximum number of features are in the set. At the end it checks each of the best options and returns the one with the highest accuracy.

2. Backward Elimination
Backwards Eliminations works the opposite in that it starts with the entire feature set and contains all of the possible actions. It will then try to eliminate features that it sees as not as accurate as others, and removes a feature for each iteration. At the end, it will go through each iteration with the best accuracy and return it.

VI. Analysis

Experiment Forward Search vs Backwards Search:

No feature selection is not as accurate as searching with feature selection. When running forward selection on the 4 datasets, we got that the accuracy for each was around 80% and one of them was 75% accurate. Whereas when running forward selection with feature selection, the accuracy was around 90%.

We have found that forward search discovered a higher accuracy with the small dataset of 90% utilizing two features. Backwards search however, found an accuracy of 83% with its best accuracy utilizing five features.

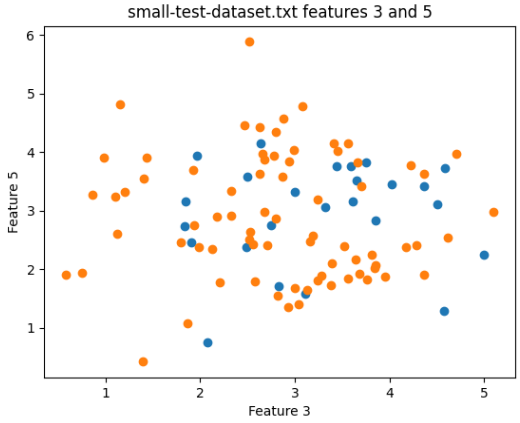
The accuracy of forward and backward search is dependent on the individual dataset. For example, in our individual small dataset we see that both searches return the same feature set. This is likely because the short list of features decreases the chance that each search gets stuck at a local optimum. However it does not eliminate the possibility of a local optimum as we see in the shared small dataset where the searches have different accuracy values. This demonstrates that greedy search is not optimal for this problem, but is good enough.

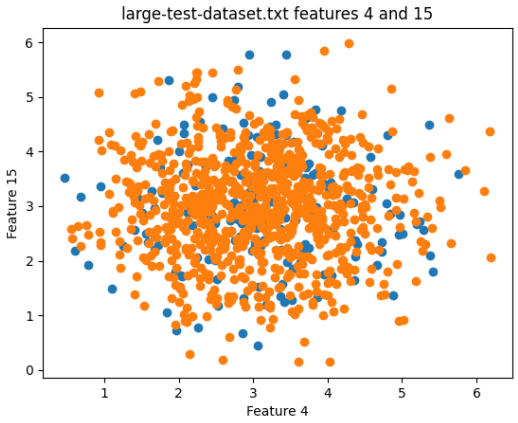
VII. Conclusion

From our findings, we can confirm that having a feature selection will help with accuracy. In our case, accuracy received a boost of 10-15%. From our testing and experiments, we find that forward select discovered a higher accuracy with the small datasets as opposed to backwards elimination. Backwards elimination also tended to have a dramatically higher run time as compared to forward selects since it has to process a lot of features at the very beginning.

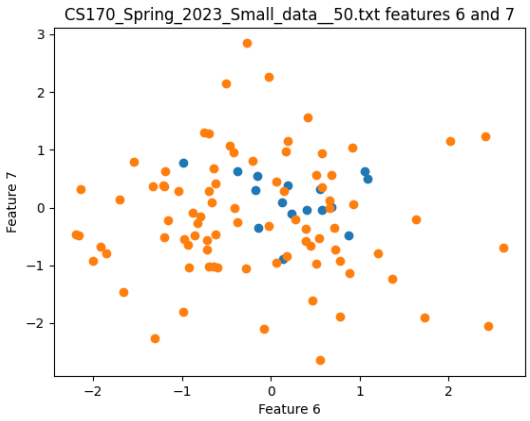
Potential improvements can be made by cleaning the data within the program such as removing irrelevant or redundant features which can reduce overfitting and efficiency. We can also optimize the code more by implementing more efficient computations and data structures.

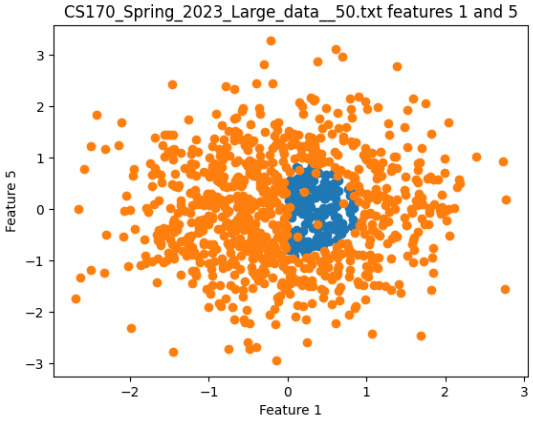
Plots of our graphs of the **general** datasets.





Plots of our **personal** datasets (dataset 50)





VIII. Trace of your small dataset

Forward Select Trace of our **personal** small dataset 50

Adding features to (0.84, set())
Features {0} have accuracy 0.87
Features {1} have accuracy 0.73
Features {2} have accuracy 0.69
Features {3} have accuracy 0.67
Features {4} have accuracy 0.77
Features {5} have accuracy 0.76
Features {6} have accuracy 0.77
Features {7} have accuracy 0.76
Features {8} have accuracy 0.74
Features {9} have accuracy 0.73
(0.87, {0}) is best choice

Adding features to (0.87, {0})
Features {0, 1} have accuracy 0.75
Features {0, 2} have accuracy 0.78
Features {0, 3} have accuracy 0.77
Features {0, 4} have accuracy 0.77
Features {0, 5} have accuracy 0.8
Features {0, 6} have accuracy 0.82
Features {0, 7} have accuracy 0.89
Features {0, 8} have accuracy 0.79
Features {0, 9} have accuracy 0.75
(0.89, {0, 7}) is best choice

Adding features to (0.89, {0, 7})
Features {0, 1, 7} have accuracy 0.84
Features {0, 2, 7} have accuracy 0.86
Features {0, 3, 7} have accuracy 0.88
Features {0, 4, 7} have accuracy 0.82
Features {0, 5, 7} have accuracy 0.85
Features {0, 6, 7} have accuracy 0.9
Features {0, 8, 7} have accuracy 0.81
Features {0, 9, 7} have accuracy 0.83
(0.9, {0, 6, 7}) is best choice

Adding features to (0.9, {0, 6, 7})
Features {0, 1, 6, 7} have accuracy 0.83
Features {0, 2, 6, 7} have accuracy 0.89
Features {0, 3, 6, 7} have accuracy 0.86
Features {0, 4, 6, 7} have accuracy 0.84
Features {0, 5, 6, 7} have accuracy 0.86
Features {0, 8, 6, 7} have accuracy 0.81
Features {0, 9, 6, 7} have accuracy 0.87
(0.89, {0, 2, 6, 7}) is best choice

Adding features to (0.89, {0, 2, 6, 7})
Features {0, 1, 2, 6, 7} have accuracy 0.82
Features {0, 2, 3, 6, 7} have accuracy 0.81
Features {0, 2, 4, 6, 7} have accuracy 0.78

Features {0, 2, 5, 6, 7} have accuracy 0.77
Features {0, 2, 6, 7, 8} have accuracy 0.8
Features {0, 2, 6, 7, 9} have accuracy 0.81
(0.82, {0, 1, 2, 6, 7}) is best choice

Adding features to (0.82, {0, 1, 2, 6, 7})
Features {0, 1, 2, 3, 6, 7} have accuracy 0.76
Features {0, 1, 2, 4, 6, 7} have accuracy 0.76
Features {0, 1, 2, 5, 6, 7} have accuracy 0.79
Features {0, 1, 2, 6, 7, 8} have accuracy 0.77
Features {0, 1, 2, 6, 7, 9} have accuracy 0.81
(0.81, {0, 1, 2, 6, 7, 9}) is best choice

Adding features to (0.81, {0, 1, 2, 6, 7, 9})
Features {0, 1, 2, 3, 6, 7, 9} have accuracy 0.73
Features {0, 1, 2, 4, 6, 7, 9} have accuracy 0.74
Features {0, 1, 2, 5, 6, 7, 9} have accuracy 0.75
Features {0, 1, 2, 6, 7, 8, 9} have accuracy 0.76
(0.76, {0, 1, 2, 6, 7, 8, 9}) is best choice

Adding features to (0.76, {0, 1, 2, 6, 7, 8, 9})
Features {0, 1, 2, 3, 6, 7, 8, 9} have accuracy 0.71
Features {0, 1, 2, 4, 6, 7, 8, 9} have accuracy 0.78
Features {0, 1, 2, 5, 6, 7, 8, 9} have accuracy 0.78
(0.78, {0, 1, 2, 4, 6, 7, 8, 9}) is best choice

Adding features to (0.78, {0, 1, 2, 4, 6, 7, 8, 9})
Features {0, 1, 2, 3, 4, 6, 7, 8, 9} have accuracy 0.73
Features {0, 1, 2, 4, 5, 6, 7, 8, 9} have accuracy 0.78
(0.78, {0, 1, 2, 4, 5, 6, 7, 8, 9}) is best choice

(0.77, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}) is best choice

Best option for each depth:
(0.84, set())
(0.87, {0})
(0.89, {0, 7})
(0.9, {0, 6, 7})
(0.89, {0, 2, 6, 7})
(0.82, {0, 1, 2, 6, 7})
(0.81, {0, 1, 2, 6, 7, 9})
(0.76, {0, 1, 2, 6, 7, 8, 9})
(0.78, {0, 1, 2, 4, 6, 7, 8, 9})
(0.78, {0, 1, 2, 4, 5, 6, 7, 8, 9})
(0.77, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})

Backwards Elimination Trace of our **personal** small dataset

Eliminating features from (0.77, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
Features {1, 2, 3, 4, 5, 6, 7, 8, 9} have accuracy 74.0%
Features {0, 2, 3, 4, 5, 6, 7, 8, 9} have accuracy 83.0%
Features {0, 1, 3, 4, 5, 6, 7, 8, 9} have accuracy 73.0%

Features {0, 1, 2, 4, 5, 6, 7, 8, 9} have accuracy 78.0%
Features {0, 1, 2, 3, 5, 6, 7, 8, 9} have accuracy 77.0%
Features {0, 1, 2, 3, 4, 6, 7, 8, 9} have accuracy 73.0%
Features {0, 1, 2, 3, 4, 5, 7, 8, 9} have accuracy 75.0%
Features {0, 1, 2, 3, 4, 5, 6, 8, 9} have accuracy 74.0%
Features {0, 1, 2, 3, 4, 5, 6, 7, 9} have accuracy 70.0%
Features {0, 1, 2, 3, 4, 5, 6, 7, 8} have accuracy 82.0%
(0.83, {0, 2, 3, 4, 5, 6, 7, 8, 9}) is best choice

Eliminating features from (0.83, {0, 2, 3, 4, 5, 6, 7, 8, 9})
Features {2, 3, 4, 5, 6, 7, 8, 9} have accuracy 72.0%
Features {0, 3, 4, 5, 6, 7, 8, 9} have accuracy 78.0%
Features {0, 2, 4, 5, 6, 7, 8, 9} have accuracy 82.0%
Features {0, 2, 3, 5, 6, 7, 8, 9} have accuracy 80.0%
Features {0, 2, 3, 4, 6, 7, 8, 9} have accuracy 80.0%
Features {0, 2, 3, 4, 5, 7, 8, 9} have accuracy 79.0%
Features {0, 2, 3, 4, 5, 6, 8, 9} have accuracy 79.0%
Features {0, 2, 3, 4, 5, 6, 7, 9} have accuracy 74.0%
Features {0, 2, 3, 4, 5, 6, 7, 8} have accuracy 78.0%
(0.82, {0, 2, 4, 5, 6, 7, 8, 9}) is best choice

Eliminating features from (0.82, {0, 2, 4, 5, 6, 7, 8, 9})
Features {2, 4, 5, 6, 7, 8, 9} have accuracy 80.0%
Features {0, 4, 5, 6, 7, 8, 9} have accuracy 85.0%
Features {0, 2, 5, 6, 7, 8, 9} have accuracy 84.0%
Features {0, 2, 4, 6, 7, 8, 9} have accuracy 80.0%
Features {0, 2, 4, 5, 7, 8, 9} have accuracy 78.0%
Features {0, 2, 4, 5, 6, 8, 9} have accuracy 76.0%
Features {0, 2, 4, 5, 6, 7, 9} have accuracy 77.0%
Features {0, 2, 4, 5, 6, 7, 8} have accuracy 76.0%
(0.85, {0, 4, 5, 6, 7, 8, 9}) is best choice

Eliminating features from (0.85, {0, 4, 5, 6, 7, 8, 9})
Features {4, 5, 6, 7, 8, 9} have accuracy 81.0%
Features {0, 5, 6, 7, 8, 9} have accuracy 82.0%
Features {0, 4, 6, 7, 8, 9} have accuracy 78.0%
Features {0, 4, 5, 7, 8, 9} have accuracy 79.0%
Features {0, 4, 5, 6, 8, 9} have accuracy 82.0%
Features {0, 4, 5, 6, 7, 9} have accuracy 81.0%
Features {0, 4, 5, 6, 7, 8} have accuracy 77.0%
(0.82, {0, 5, 6, 7, 8, 9}) is best choice

Eliminating features from (0.82, {0, 5, 6, 7, 8, 9})
Features {5, 6, 7, 8, 9} have accuracy 79.0%
Features {0, 6, 7, 8, 9} have accuracy 80.0%
Features {0, 5, 7, 8, 9} have accuracy 82.0%
Features {0, 5, 6, 8, 9} have accuracy 81.0%
Features {0, 5, 6, 7, 9} have accuracy 85.0%

Features {0, 5, 6, 7, 8} have accuracy 82.0%
(0.85, {0, 5, 6, 7, 9}) is best choice

Eliminating features from (0.85, {0, 5, 6, 7, 9})
Features {9, 5, 6, 7} have accuracy 83.0%
Features {0, 9, 6, 7} have accuracy 87.0%
Features {0, 9, 5, 7} have accuracy 78.0%
Features {0, 9, 5, 6} have accuracy 77.0%
Features {0, 5, 6, 7} have accuracy 86.0%
(0.87, {0, 9, 6, 7}) is best choice

Eliminating features from (0.87, {0, 9, 6, 7})
Features {9, 6, 7} have accuracy 79.0%
Features {0, 9, 7} have accuracy 83.0%
Features {0, 9, 6} have accuracy 79.0%
Features {0, 6, 7} have accuracy 90.0%
(0.9, {0, 6, 7}) is best choice

Eliminating features from (0.9, {0, 6, 7})
Features {6, 7} have accuracy 76.0%
Features {0, 7} have accuracy 89.0%
Features {0, 6} have accuracy 82.0%
(0.89, {0, 7}) is best choice

Eliminating features from (0.89, {0, 7})
Features {7} have accuracy 76.0%
Features {0} have accuracy 87.0%
(0.87, {0}) is best choice

Eliminating features from (0.87, {0})
Features set() have accuracy 84.0%
(0.84, set()) is best choice

Best option for each depth:
(0.77, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
(0.83, {0, 2, 3, 4, 5, 6, 7, 8, 9})
(0.82, {0, 2, 4, 5, 6, 7, 8, 9})
(0.85, {0, 4, 5, 6, 7, 8, 9})
(0.82, {0, 5, 6, 7, 8, 9})
(0.85, {0, 5, 6, 7, 9})
(0.87, {0, 9, 6, 7})
(0.9, {0, 6, 7})
(0.89, {0, 7})
(0.87, {0})
(0.84, set())

Forward Select Trace of our **general** small dataset 50

Adding features to (0.25, set())
Features {0} have accuracy 0.0
Features {1} have accuracy 0.5
Features {2} have accuracy 1.0
Features {3} have accuracy 0.0
Features {4} have accuracy 0.0
Features {5} have accuracy 1.0
Features {6} have accuracy 0.0
Features {7} have accuracy 0.0
Features {8} have accuracy 0.0
Features {9} have accuracy 0.0
(1.0, {2}) is best choice

Adding features to (1.0, {2})
Features {0, 2} have accuracy 1.0
Features {1, 2} have accuracy 1.0
Features {2, 3} have accuracy 1.0
Features {2, 4} have accuracy 1.0
Features {2, 5} have accuracy 1.0
Features {2, 6} have accuracy 1.0
Features {2, 7} have accuracy 1.0
Features {8, 2} have accuracy 1.0
Features {9, 2} have accuracy 1.0
(1.0, {0, 2}) is best choice

Adding features to (1.0, {0, 2})
Features {0, 1, 2} have accuracy 1.0
Features {0, 2, 3} have accuracy 1.0
Features {0, 2, 4} have accuracy 1.0
Features {0, 2, 5} have accuracy 1.0
Features {0, 2, 6} have accuracy 1.0
Features {0, 2, 7} have accuracy 1.0
Features {0, 8, 2} have accuracy 1.0
Features {0, 9, 2} have accuracy 1.0
(1.0, {0, 1, 2}) is best choice

Adding features to (1.0, {0, 1, 2})
Features {0, 1, 2, 3} have accuracy 1.0
Features {0, 1, 2, 4} have accuracy 1.0
Features {0, 1, 2, 5} have accuracy 1.0
Features {0, 1, 2, 6} have accuracy 1.0
Features {0, 1, 2, 7} have accuracy 1.0
Features {0, 1, 2, 8} have accuracy 1.0
Features {0, 1, 2, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3}) is best choice

Adding features to (1.0, {0, 1, 2, 3})
Features {0, 1, 2, 3, 4} have accuracy 0.75
Features {0, 1, 2, 3, 5} have accuracy 1.0
Features {0, 1, 2, 3, 6} have accuracy 0.75
Features {0, 1, 2, 3, 7} have accuracy 1.0
Features {0, 1, 2, 3, 8} have accuracy 0.75
Features {0, 1, 2, 3, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 5}) is best choice

Adding features to (1.0, {0, 1, 2, 3, 5})
Features {0, 1, 2, 3, 4, 5} have accuracy 1.0
Features {0, 1, 2, 3, 5, 6} have accuracy 1.0
Features {0, 1, 2, 3, 5, 7} have accuracy 1.0
Features {0, 1, 2, 3, 5, 8} have accuracy 1.0
Features {0, 1, 2, 3, 5, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 4, 5}) is best choice

Adding features to (1.0, {0, 1, 2, 3, 4, 5})
Features {0, 1, 2, 3, 4, 5, 6} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 7} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 8} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 4, 5, 6}) is best choice

Adding features to (1.0, {0, 1, 2, 3, 4, 5, 6})
Features {0, 1, 2, 3, 4, 5, 6, 7} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 6, 8} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 6, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 4, 5, 6, 7}) is best choice

Adding features to (1.0, {0, 1, 2, 3, 4, 5, 6, 7})
Features {0, 1, 2, 3, 4, 5, 6, 7, 8} have accuracy 1.0
Features {0, 1, 2, 3, 4, 5, 6, 7, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 4, 5, 6, 7, 8}) is best choice

Adding features to (1.0, {0, 1, 2, 3, 4, 5, 6, 7, 8})
Features {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} have accuracy 1.0
(1.0, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}) is best choice

Best option for each depth:
(0.25, set())
(1.0, {2})
(1.0, {0, 2})
(1.0, {0, 1, 2})
(1.0, {0, 1, 2, 3})
(1.0, {0, 1, 2, 3, 5})
(1.0, {0, 1, 2, 3, 4, 5})
(1.0, {0, 1, 2, 3, 4, 5, 6})

(1.0, {0, 1, 2, 3, 4, 5, 6, 7})
(1.0, {0, 1, 2, 3, 4, 5, 6, 7, 8})
(1.0, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})

Backwards Elimination Trace of our **general** small dataset

Eliminating features from (0.68, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
Features {1, 2, 3, 4, 5, 6, 7, 8, 9} have accuracy 71.0%
Features {0, 2, 3, 4, 5, 6, 7, 8, 9} have accuracy 62.0%
Features {0, 1, 3, 4, 5, 6, 7, 8, 9} have accuracy 73.0%
Features {0, 1, 2, 4, 5, 6, 7, 8, 9} have accuracy 70.0%
Features {0, 1, 2, 3, 5, 6, 7, 8, 9} have accuracy 69.0%
Features {0, 1, 2, 3, 4, 6, 7, 8, 9} have accuracy 71.0%
Features {0, 1, 2, 3, 4, 5, 7, 8, 9} have accuracy 62.0%
Features {0, 1, 2, 3, 4, 5, 6, 8, 9} have accuracy 72.0%
Features {0, 1, 2, 3, 4, 5, 6, 7, 9} have accuracy 67.0%
Features {0, 1, 2, 3, 4, 5, 6, 7, 8} have accuracy 72.0%
(0.73, {0, 1, 3, 4, 5, 6, 7, 8, 9}) is best choice

Eliminating features from (0.73, {0, 1, 3, 4, 5, 6, 7, 8, 9})
Features {1, 3, 4, 5, 6, 7, 8, 9} have accuracy 72.0%
Features {0, 3, 4, 5, 6, 7, 8, 9} have accuracy 73.0%
Features {0, 1, 4, 5, 6, 7, 8, 9} have accuracy 69.0%
Features {0, 1, 3, 5, 6, 7, 8, 9} have accuracy 64.0%
Features {0, 1, 3, 4, 6, 7, 8, 9} have accuracy 75.0%
Features {0, 1, 3, 4, 5, 7, 8, 9} have accuracy 68.0%
Features {0, 1, 3, 4, 5, 6, 8, 9} have accuracy 68.0%
Features {0, 1, 3, 4, 5, 6, 7, 9} have accuracy 73.0%
Features {0, 1, 3, 4, 5, 6, 7, 8} have accuracy 67.0%
(0.75, {0, 1, 3, 4, 6, 7, 8, 9}) is best choice

Eliminating features from (0.75, {0, 1, 3, 4, 6, 7, 8, 9})
Features {1, 3, 4, 6, 7, 8, 9} have accuracy 75.0%
Features {0, 3, 4, 6, 7, 8, 9} have accuracy 73.0%
Features {0, 1, 4, 6, 7, 8, 9} have accuracy 64.0%
Features {0, 1, 3, 6, 7, 8, 9} have accuracy 58.0%
Features {0, 1, 3, 4, 7, 8, 9} have accuracy 71.0%
Features {0, 1, 3, 4, 6, 8, 9} have accuracy 78.0%
Features {0, 1, 3, 4, 6, 7, 9} have accuracy 77.0%
Features {0, 1, 3, 4, 6, 7, 8} have accuracy 71.0%
(0.78, {0, 1, 3, 4, 6, 8, 9}) is best choice

Eliminating features from (0.78, {0, 1, 3, 4, 6, 8, 9})
Features {1, 3, 4, 6, 8, 9} have accuracy 76.0%
Features {0, 3, 4, 6, 8, 9} have accuracy 73.0%
Features {0, 1, 4, 6, 8, 9} have accuracy 67.0%
Features {0, 1, 3, 6, 8, 9} have accuracy 61.0%

Features {0, 1, 3, 4, 8, 9} have accuracy 75.0%
Features {0, 1, 3, 4, 6, 9} have accuracy 79.0%
Features {0, 1, 3, 4, 6, 8} have accuracy 76.0%
(0.79, {0, 1, 3, 4, 6, 9}) is best choice

Eliminating features from (0.79, {0, 1, 3, 4, 6, 9})
Features {1, 3, 4, 6, 9} have accuracy 83.0%
Features {0, 3, 4, 6, 9} have accuracy 71.0%
Features {0, 1, 4, 6, 9} have accuracy 75.0%
Features {0, 1, 3, 6, 9} have accuracy 60.0%
Features {0, 1, 3, 4, 9} have accuracy 75.0%
Features {0, 1, 3, 4, 6} have accuracy 77.0%
(0.83, {1, 3, 4, 6, 9}) is best choice

Eliminating features from (0.83, {1, 3, 4, 6, 9})
Features {9, 3, 4, 6} have accuracy 79.0%
Features {1, 4, 6, 9} have accuracy 81.0%
Features {1, 3, 6, 9} have accuracy 71.0%
Features {1, 3, 4, 9} have accuracy 76.0%
Features {1, 3, 4, 6} have accuracy 76.0%
(0.81, {1, 4, 6, 9}) is best choice

Eliminating features from (0.81, {1, 4, 6, 9})
Features {9, 4, 6} have accuracy 75.0%
Features {1, 6, 9} have accuracy 68.0%
Features {1, 4, 9} have accuracy 72.0%
Features {1, 4, 6} have accuracy 77.0%
(0.77, {1, 4, 6}) is best choice

Eliminating features from (0.77, {1, 4, 6})
Features {4, 6} have accuracy 80.0%
Features {1, 6} have accuracy 54.0%
Features {1, 4} have accuracy 80.0%
(0.8, {4, 6}) is best choice

Eliminating features from (0.8, {4, 6})
Features {6} have accuracy 62.0%
Features {4} have accuracy 75.0%
(0.75, {4}) is best choice

Eliminating features from (0.75, {4})
Features set() have accuracy 75.0%
(0.75, set()) is best choice

Best option for each depth:
(0.68, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
(0.73, {0, 1, 3, 4, 5, 6, 7, 8, 9})
(0.75, {0, 1, 3, 4, 6, 7, 8, 9})

(0.78, {0, 1, 3, 4, 6, 8, 9})
(0.79, {0, 1, 3, 4, 6, 9})
(0.83, {1, 3, 4, 6, 9})
(0.81, {1, 4, 6, 9})
(0.77, {1, 4, 6})
(0.8, {4, 6})
(0.75, {4})
(0.75, set())