Design

Tree Class

A Tree stores all the different feature combinations, held in Nodes.

Node Class

Nodes hold the classification, features, parents, and children.

Search Algorithms

Forwards search starts at the empty combination and searches its children. From the children, the combination with the highest accuracy is picked, and the search continues from that combination. If the accuracy is lower than the previous highest accuracy, this does not terminate the search (eg: previous highest may be a local maxima). The search stops only when the algorithm hits the end of the Tree.

Backwards search works similarly to Forwards search, except it starts at the final combination and searches the parents.

Justin's Special uses a similar approach to Forward search. It uses epsilon(a percent: ex. 5%, 20%, etc.) to allow for forward search to explore features that are within epsilon accuracy of the current best feature set. An epsilon of 0 is a pure Forward Search while an epsilon of 100 would be a breadth first search.

Classifier

Utilizes the dataframe in Panda's library to organize which features to explore(when provided) and to assist with doing euclidean distance over all instances. Returns a list of Classifications based on 1-NN Euclidean.

Validator

Validates the accuracy of the feature set given by comparing the Classification list from the Classifier class. The accuracy returned is:

# of correct classifications / total instances

Optimizations

Using pandas data frames, the math became much simpler and could be done quicker. Furthermore, the tree would be built according to the path taken by the

search algorithm. This means that only nodes that are seen by the search algorithm are created, essentially pruning our tree during run time.

Dataset Findings Comparison

Best Feature Set w/ Accuracy for all Data Sets and Algorithms

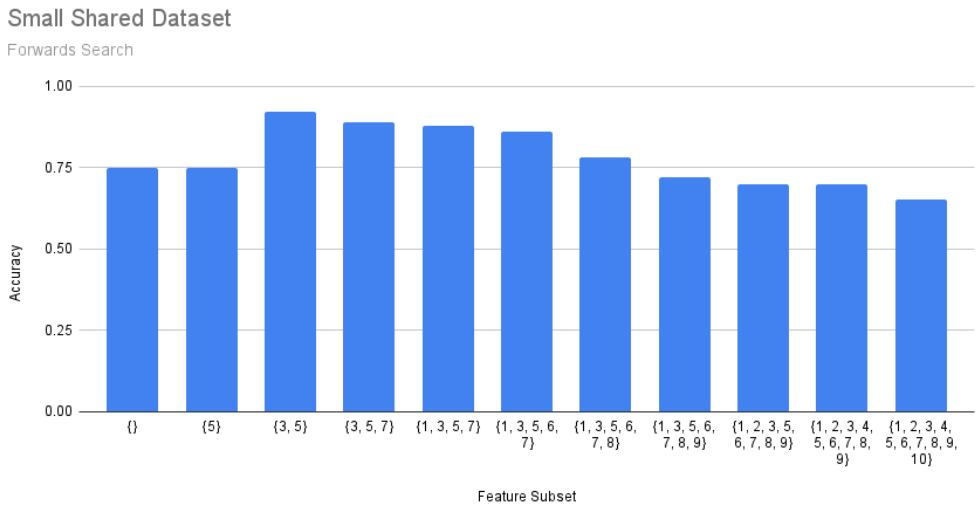
Dataset	Forwards Search	Backwards Search	Justin's Special
Small Dataset	{3, 5} 92%	{2, 4, 5, 7, 10}* 82%	{3, 5} 92%
Large Dataset	{1, 27} 95.5%	{27} 84.7%	{1, 27} 95.5%
Small Personal Dataset (ID 29)	{4, 9} 93%	{4, 9} 93%	{4, 9} 93%
Large Personal Dataset (ID 29)	{40, 9} 96.1%	{40} 84.8%	{40, 9} 96.1%

F1. Best feature set and accuracy on all data sets for each search algorithm

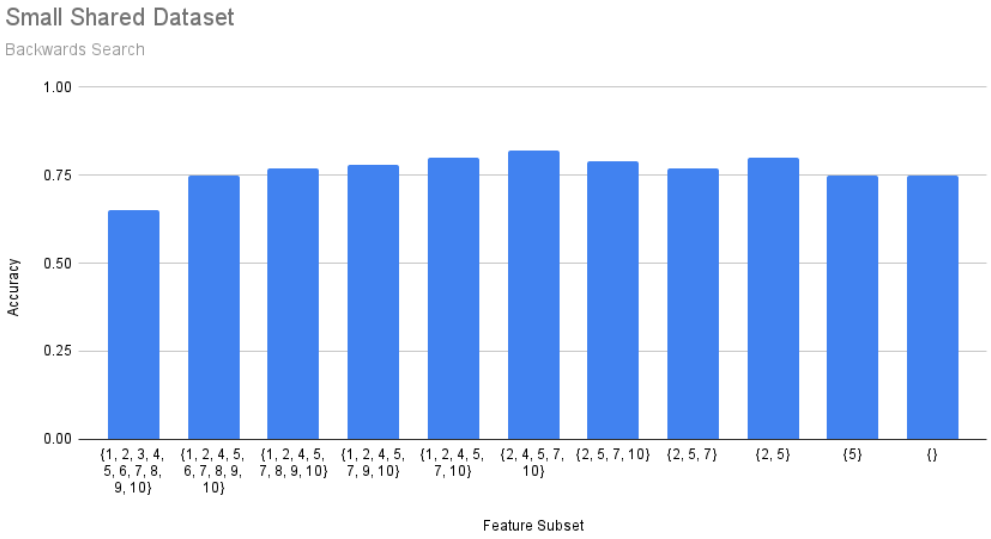
small datasets avg. runtime: ~10s
large datasets avg. runtime: 2h:15m

The running time of forward search vs backward search is substantially more noticeable. This is due to forward search validating many small feature sets early and a few large sets later; opposite to what backward search would do. Justin's special performed just as fast, with a small epsilon of range not limited to (0,3]. This idea of Backward Search taking longer also ties into memory usage and storage, tree nodes were large and many large feature sets were validated. While backward search performed poorly compared to the two forward search algorithms, its usage in this project shows that there may be errors in the algorithm's logic. This is because backward search should in theory return the same feature set as our forward search algorithms.

Plots/Graphs



F2. Bar Chart(Accuracy for best feature at each tree layer) for Forward Search on Shared Data set



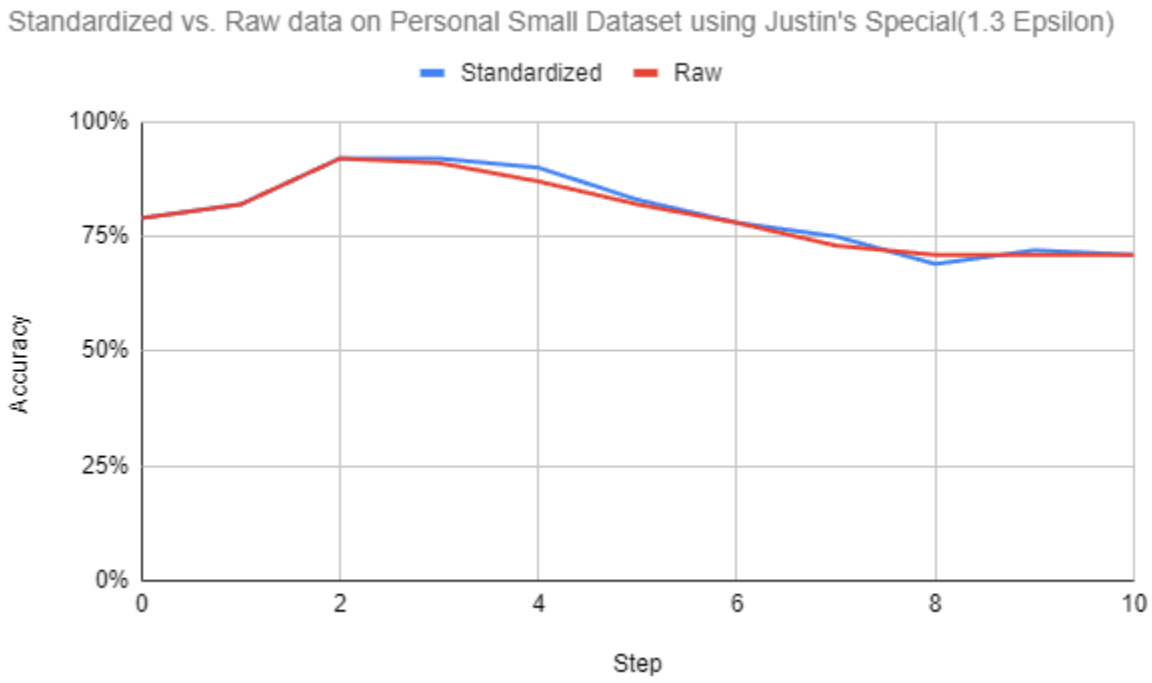
F3. Bar Chart(Accuracy for best feature at each tree layer) for Backward Search on Shared Data set

Chart Analysis

F2 and F3 show the two algorithm taking different paths and possibly some error in the code. Regardless, this outcome is possible with a greedy search. This outcome does not match with what was to be expected from testing on the shared data set, where the outcome should have been {3,5,7} with an accuracy of 89%. While we did come across {3,5,7} and calculated the accuracy correctly, this raises concerns of whether the data used/processed was not correct or whether the validation is incorrect.

It is also seen that feature 3 has good information for classification. Features 8 and 9 are not good for classification. Both algorithms also show that feature 5 has no impact on our data set, thus meaning we can remove it.

Data, Normalized vs Raw



F4. Comparison of Standardized vs Raw data on Personal Small Data set w/ Justin's Special(1.3 epsilon)

F4 shows that using the small Personal data set, standardizing data seems to provide more accuracy. This could be further proven in theory by using a larger data set, but due to the lack of time, this will be set aside for future plans. This chart does not mean that raw data performs, but instead shows how that data normalization performs just as good as raw data. This lines up with why normalized data for NN classification is good overall.

Tools/Libraries

---

1) Google Collab(IDE)
2) Numpy(Library)
3) Pandas(Library)
4) Misc Documentation for the above and for Python

Trace (Personal Small Dataset: ID 29)

Trace provided from forwards search

```
Test small or large dataset? (1 or 2): 1
Which algo?
1 - Forward Search
2 - Backward Search
3 - Justin's Special
1
Small Dataset (has 100 instances and 10 features)
Enter the desired features (1 - 10), separated by spaces. Enter nothing to
use all features:
Please wait while I normalize data...  Done!
Running nearest neighbor with no features (default rate), using
"leaving-one-out" evaluation, I get an accuracy of : 79.0%

Start search

      Using feature(s) {1} accuracy is 64.0%
      Using feature(s) {2} accuracy is 60.0%
      Using feature(s) {3} accuracy is 71.0%
      Using feature(s) {4} accuracy is 82.0%
      Using feature(s) {5} accuracy is 71.0%
      Using feature(s) {6} accuracy is 61.0%
      Using feature(s) {7} accuracy is 67.0%
      Using feature(s) {8} accuracy is 72.0%
      Using feature(s) {9} accuracy is 73.0%
      Using feature(s) {10} accuracy is 75.0%


Feature set {4} was best, accuracy is 82.0%

      Using feature(s) {3, 4} accuracy is 77.0%
      Using feature(s) {4, 6} accuracy is 74.0%
      Using feature(s) {4, 5} accuracy is 83.0%
      Using feature(s) {9, 4} accuracy is 93.0%
```

```
      Using feature(s) {2, 4} accuracy is 83.0%
      Using feature(s) {4, 7} accuracy is 76.0%
      Using feature(s) {8, 4} accuracy is 83.0%
      Using feature(s) {10, 4} accuracy is 81.0%
      Using feature(s) {1, 4} accuracy is 71.0%


Feature set {9, 4} was best, accuracy is 93.0%

      Using feature(s) {9, 4, 1} accuracy is 87.0%
      Using feature(s) {9, 2, 4} accuracy is 91.0%
      Using feature(s) {9, 3, 4} accuracy is 86.0%
      Using feature(s) {8, 9, 4} accuracy is 92.0%
      Using feature(s) {1, 4, 9} accuracy is 87.0%
      Using feature(s) {9, 4, 5} accuracy is 85.0%
      Using feature(s) {9, 4, 7} accuracy is 88.0%
      Using feature(s) {9, 10, 4} accuracy is 89.0%
      Using feature(s) {9, 4, 6} accuracy is 88.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)


Feature set {8, 9, 4} was best, accuracy is 92.0%

      Using feature(s) {8, 9, 4, 7} accuracy is 86.0%
      Using feature(s) {8, 9, 10, 4} accuracy is 85.0%
      Using feature(s) {8, 9, 2, 4} accuracy is 90.0%
      Using feature(s) {8, 9, 4, 6} accuracy is 87.0%
      Using feature(s) {8, 1, 4, 9} accuracy is 82.0%
      Using feature(s) {8, 9, 4, 1} accuracy is 82.0%
      Using feature(s) {8, 9, 3, 4} accuracy is 84.0%
      Using feature(s) {8, 9, 4, 5} accuracy is 84.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)


Feature set {8, 9, 2, 4} was best, accuracy is 90.0%

      Using feature(s) {1, 2, 4, 8, 9} accuracy is 80.0%
      Using feature(s) {2, 3, 4, 8, 9} accuracy is 77.0%
      Using feature(s) {2, 4, 6, 8, 9} accuracy is 83.0%
      Using feature(s) {2, 4, 7, 8, 9} accuracy is 80.0%
      Using feature(s) {2, 4, 5, 8, 9} accuracy is 78.0%
      Using feature(s) {2, 4, 8, 9, 10} accuracy is 80.0%
```

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {2, 4, 6, 8, 9} was best, accuracy is 83.0%

Using feature(s) {2, 4, 6, 8, 9, 10} accuracy is 77.0%
Using feature(s) {2, 4, 6, 7, 8, 9} accuracy is 77.0%
Using feature(s) {1, 2, 4, 6, 8, 9} accuracy is 76.0%
Using feature(s) {2, 4, 5, 6, 8, 9} accuracy is 77.0%
Using feature(s) {2, 3, 4, 6, 8, 9} accuracy is 78.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {2, 3, 4, 6, 8, 9} was best, accuracy is 78.0%

Using feature(s) {1, 2, 3, 4, 6, 8, 9} accuracy is 71.0%
Using feature(s) {2, 3, 4, 6, 7, 8, 9} accuracy is 70.0%
Using feature(s) {2, 3, 4, 6, 8, 9, 10} accuracy is 71.0%
Using feature(s) {2, 3, 4, 5, 6, 8, 9} accuracy is 75.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {2, 3, 4, 5, 6, 8, 9} was best, accuracy is 75.0%

Using feature(s) {1, 2, 3, 4, 5, 6, 8, 9} accuracy is 65.0%
Using feature(s) {2, 3, 4, 5, 6, 7, 8, 9} accuracy is 69.0%
Using feature(s) {2, 3, 4, 5, 6, 8, 9, 10} accuracy is 66.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {2, 3, 4, 5, 6, 7, 8, 9} was best, accuracy is 69.0%

Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9} accuracy is 65.0%
Using feature(s) {2, 3, 4, 5, 6, 7, 8, 9, 10} accuracy is 72.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {2, 3, 4, 5, 6, 7, 8, 9, 10} was best, accuracy is 72.0%

Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} accuracy is 71.0%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Feature set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} was best, accuracy is 71.0%

Finished search, best feature subset is {9, 4}, which has an accuracy of 93.0%