Assignment 2 CS205: Introduction to Artificial Intelligence, Dr. Eamonn Keogh

Marcus Martin

SID 862026991

Email mmart149@ucr.edu

Date June-10-2021

In completing this assignment I consulted:

- Project_2_Briefing slides and videos from lecture
- Nearest Neighbor slides from lecture
- Feature Selection Project Directions and Example Assignment
- Code of feature selection with nearest neighbor and accuracy slides and videos from lecture

All important code is original. Unimportant subroutines that are not completely original are…

- Copy (deepcopy) for copying data
- Numpy and Pandas libraries for manipulating data
- Time and Math for keeping track of the run time of the program
- The truncate() function used to truncate the run time of the program to four decimal places, which was from: https://www.delftstack.com/howto/python/python-truncate-float-python/
- Conversion of lists of lists to list of strings for graph data, which was from: https://www.geeksforgeeks.org/python-convert-list-of-lists-to-list-of-strings/
- Matplotlib.pyplot for charting data
- Code of feature selection with nearest neighbor and accuracy slides and videos from lecture

Outline of this report:

CS205 Assignment 2 Feature Selection Project

Marcus Martin, SID 862026991 June-10-2021

Introduction

This project implements the Nearest Neighbor algorithm with Forward Selection and Backward Elimination search. To calculate the accuracy, we used K-fold cross validation as it allows us to deal with outliers. K-fold cross validation takes into consideration the distance from a node to its surrounding nodes and helps the algorithm to not overfit to the data. The process of k-fold cross validation is to create K subsets of the dataset and train the model on the subsets by using a subset to test. We are provided with two datasets, a small and large dataset, that contain a class label consisting of two classes and several features. Using Forward Selection, we start with no features and gradually add features with the highest classification accuracy to the group and keep track of the classification accuracy of the group of features. Using Backward Elimination, we start with all the features and gradually delete features with the highest classification accuracy from the group and keep track of the accuracy of the group of features. We then graph the results and analyze the graphs to determine the important features. Important features are the features that produce the best classification accuracy.

Forward Selection and Backward Elimination on Small Dataset

Figure 1 shows the results of running Forward Selection on the small dataset CS205_small_testdata__36.txt.

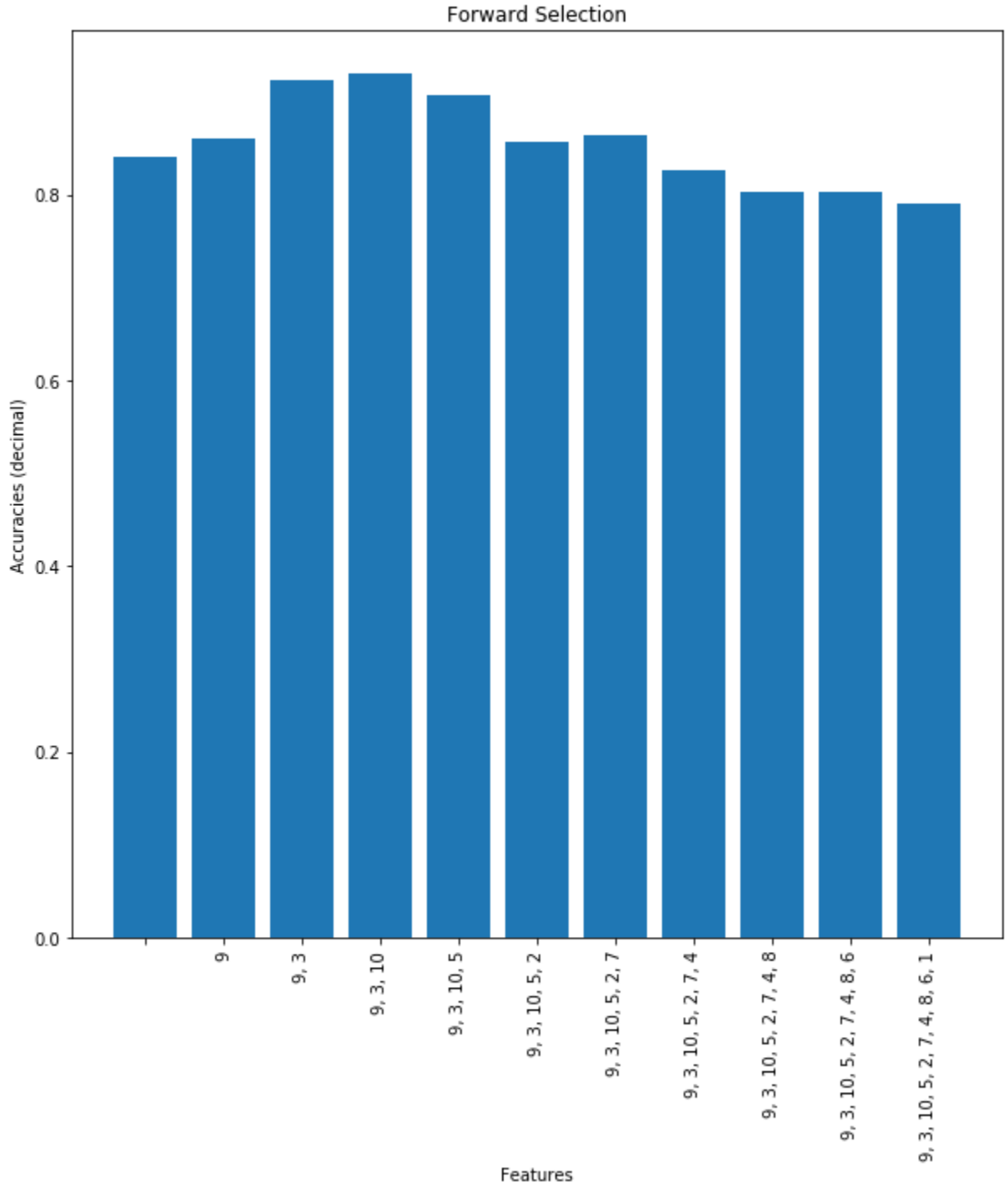

Figure 1: Accuracy vs Increasing Subsets of Features with Forward Selection on Small Dataset

Viewing the graph from left to right, we start with no features and calculate the accuracy to be 84%. Then, after determining that the addition of feature 9 has the greatest accuracy compared to the addition of the other features using forward selection, we add feature 9 to the feature subset which increases the accuracy to 86%. Then, after determining that the addition of feature 3 to the feature subset of 9 has the greatest accuracy, we add feature 3 to the feature subset which increases the accuracy to 93%. Then, adding feature 10 results in an accuracy of 93%. We see that the following addition of features after feature 10 results in the accuracy gradually decreasing, with the full set of features having an accuracy of 79%.

As we can see, the addition of feature 9 marks a modest increase in accuracy, the addition of feature 3 results in a substantial increase in accuracy, and the addition of feature 10 results in a small increase in accuracy. The substantial increase in accuracy from adding feature 3 suggests that the feature is important. The modest increase in accuracy from adding feature 9 suggests that the feature is not as important as feature 3. The small increase of accuracy with feature 10 suggests that the feature is not as important as feature 9.

Figure 2 shows the results of running Backward Elimination on the small dataset CS205_small_testdata__36.txt.

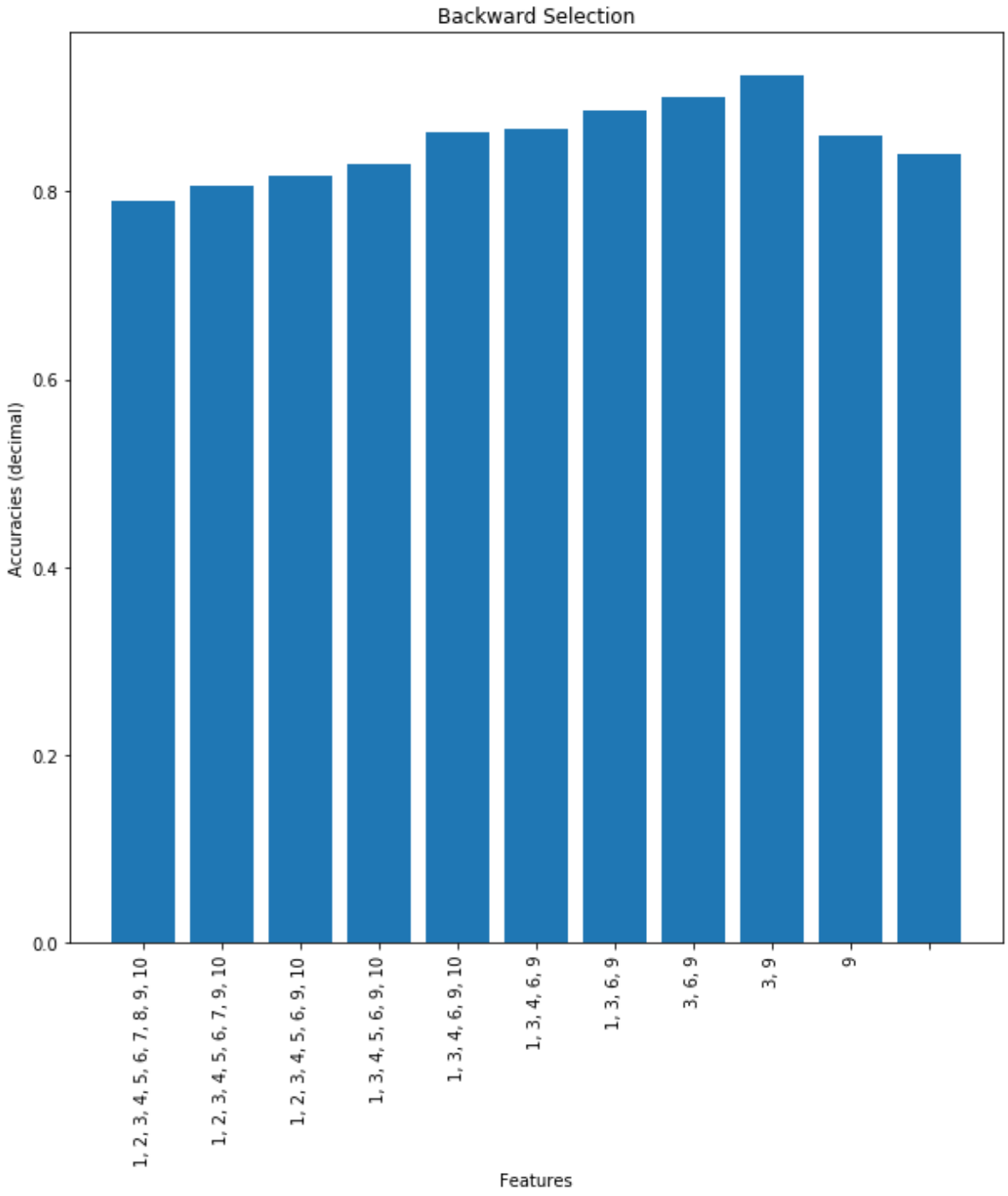

Figure 2: Accuracy vs Decreasing Subsets of Features with Backward Elimination on Small Dataset

Viewing the graph from left to right, we start with all the features and calculate the accuracy to be 79%, which is the same accuracy that we had with all the features when using forward selection. Then, after determining that the deletion of feature 8 has the greatest accuracy compared to the deletion of the other features using backward elimination, we delete feature 8 from the feature subset which increases the accuracy to 80.66%. Then, we can see that for the rest of the features up to the deletion of feature 3 (reading the graph from left to right), they all contribute to increasing the accuracy. In particular, the deletion of feature 5 increases the accuracy from 83% to 86.33% and the deletion of feature 10 increases the accuracy from 86.33% to 86.66%. The deletion of feature 6 results in an accuracy of 93%. We see that the following deletion of features after feature 6 results in the accuracy gradually decreasing, with the full set of features having an accuracy of 84%, which is the same accuracy that we had with no features when using forward selection.

As we can see, the deletion of feature 10 marks a small increase in accuracy, and the deletion of the other features up to feature 3 results in a modest increase in accuracy with feature 5 resulting in a substantial increase in accuracy. The substantial increase in accuracy from the deletion of feature 5 suggests that the feature is important, while the most increase in accuracy by the other features suggests that they are not as important as feature 5. The small increase of accuracy with feature 10 suggests that the feature is not as important as the features that modestly increased the accuracy.

Conclusion for Small Dataset

From our observations, we saw that the addition of feature 3 and the deletion of feature 5 resulted in a substantial increase in accuracy. We can presume that these features are important features for the dataset. The other features that contribute to increasing the accuracy may also be important. However, it is not clear that they are important and further testing is needed.

Forward Selection and Backward Elimination on Large Dataset

For Forward Selection on the large dataset, I only explored up to 10 features because accuracy generally decreases after 10 features for most problems. Although this method does not hold true for Backward Elimination, I also only explored up to 10 features due to time constraint.

Figure 3 shows the results of running Forward Selection on the large dataset CS205_large_testdata__23.txt.

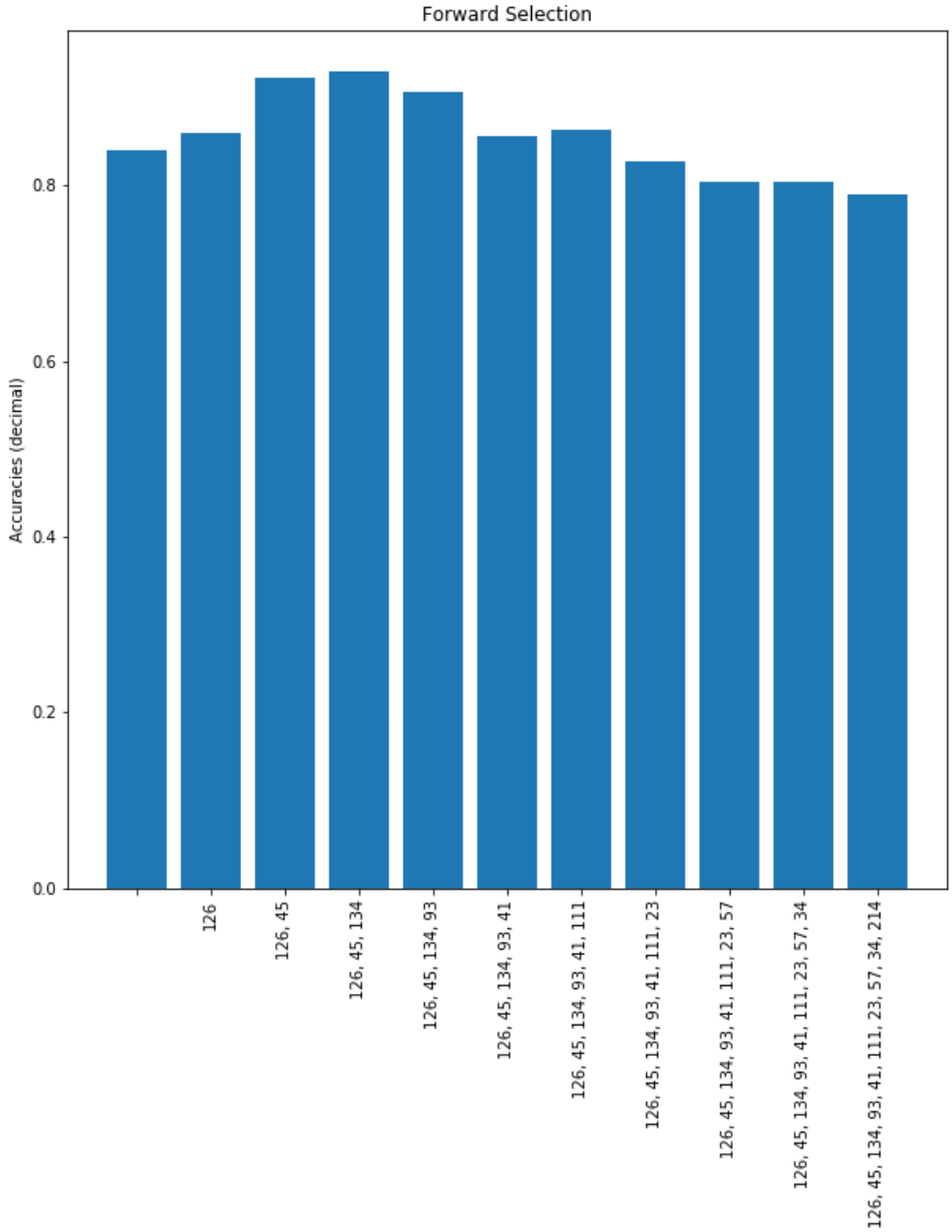

Figure 3: Accuracy vs Increasing Subsets of Features with Forward Selection on Large Dataset

Viewing the graph from left to right, we start with no features and calculate the accuracy to be 80.33%. Then, after determining that the addition of feature 126 has the greatest accuracy compared to the addition of the other features using forward selection, we add feature 126 to the feature subset which increases the accuracy to 83.5%. Then, after determining that the addition of feature 45 to the feature subset of 126 has the greatest accuracy, we add feature 45 to the feature subset which increases the accuracy to 95.22%. Then, adding feature 134 results in the accuracy remaining at 95.22%. We see that the following addition of features after feature 134 results in the accuracy gradually decreasing.

As we can see, the addition of feature 126 marks a modest increase in accuracy, the addition of feature 45 results in a substantial increase in accuracy, and the addition of feature 134 results in no increase in accuracy. The substantial increase in accuracy from feature 45 suggests that the feature is important. The modest increase in accuracy of feature 126 suggests that the feature is not as important as feature 45. The fact that feature 134 did not increase the accuracy suggests that the feature is not as important as feature 126. I presume that the full set of features will have a lower accuracy than 95.22%. Later, as we can see from Figure 4, the full set of features will have an accuracy of 71.22%.

Figure 4 shows the results of running Backward Elimination on the large dataset CS205_large_testdata__23.txt.

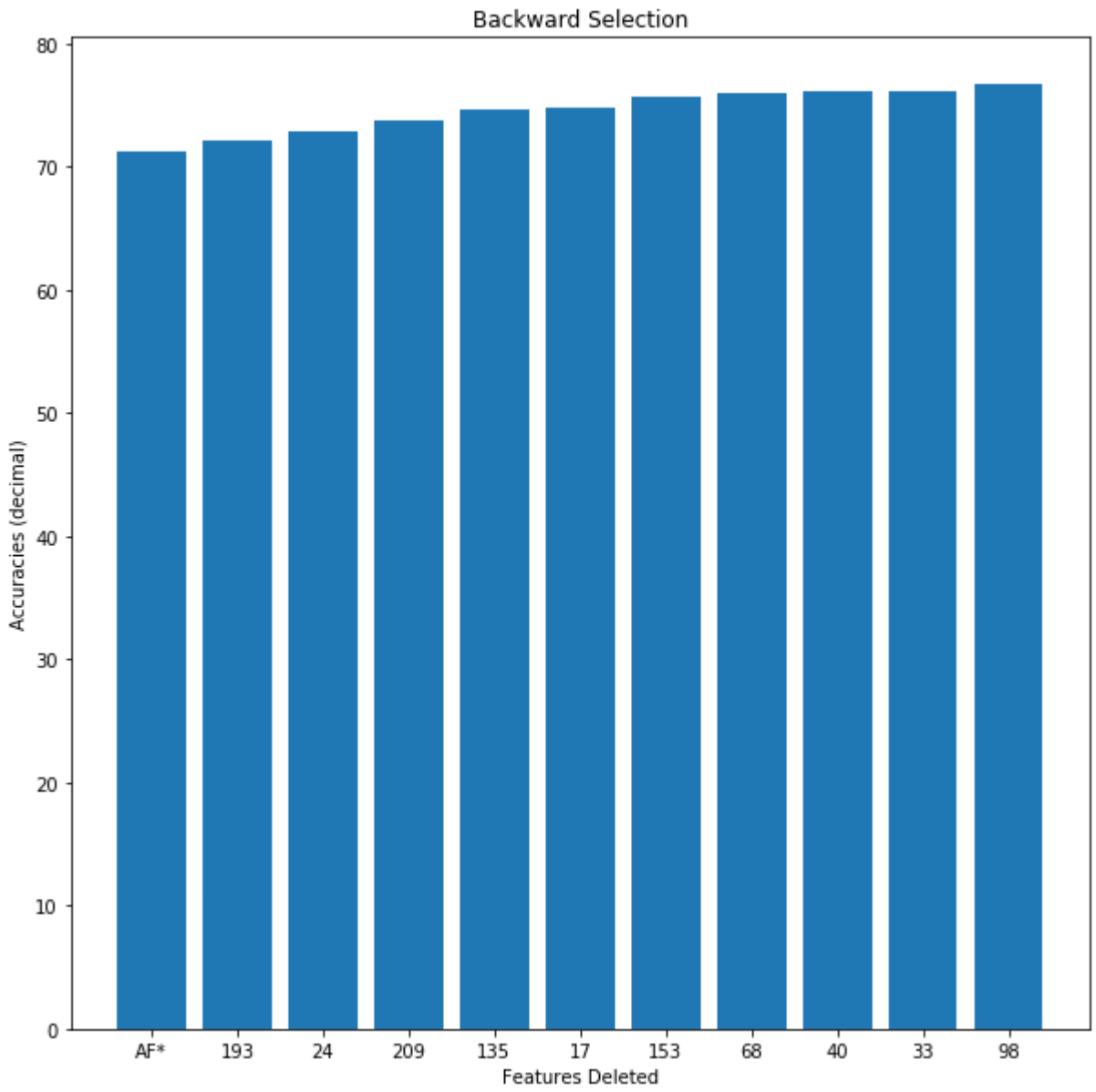

Figure 4: Accuracy vs Decreasing Subsets of Features with Backward Elimination on Large Dataset

*AF = All Features (no features deleted)

Due to space, I decided to change the representation of the x-axis values for Backward Elimination graph on the large dataset. Viewing the graph from left to right, we start with the full set of features, and then each value on the x-axis represents the value that was deleted from the current set of features.

Viewing the graph from left to right, we start with all the features and calculate the accuracy to be 71.22%, which is presumably the same accuracy that we would have calculated with all the features when using forward selection. Then, after determining that the deletion of feature 193 has the greatest accuracy compared to the deletion of the other features using backward elimination, we delete feature 193 to the feature subset which increases the accuracy to 72.11%. Deleting feature 17 increases the accuracy from 74.55% to 74.77%. Deleting feature 33 keeps the accuracy the same at 76.11%. Features 193, 24, 209, and 135 all increase the accuracy by a similar amount due to the almost linear line shown by the graph. We can see that the deletion of the other features (from left to right) gradually increases the accuracy. We can presume that the graph could keep increasing in accuracy with each deletion of a feature, peak at a certain point, and then end at 80.33% since that is the accuracy that Forward Selection started at with an empty feature set.

As we can see, the deletion of features results in a gradual increase in accuracy. It is hard to determine if these features are important or not since we do not have the full graph available. The similar and substantial increase in accuracy from features 193, 24, 209, and 135 suggests that these features are important. The comparably smaller increase in accuracy from the other features (excluding 17 and 33) suggests that they are not as important as the previous listed features. The fact that feature 17 only slightly increased the accuracy and feature 33 did not increase the accuracy suggests that those features are not as important as these other features.

Conclusion for Large Dataset

From our observations, we saw that the addition of feature 45 resulted in a substantial increase in accuracy, and the deletion of features 193, 24, 209, and 135 resulted in an increase in accuracy that was greater than the other feature's increases in accuracy. We can presume that feature 45 and features 193, 24, 209, and 135 are important features for the dataset. The rest of the features may also be important. However, it is not clear that they are important and further testing is needed. We can also note that since the deletion of several features shows that the accuracy may still be increasing, more features need to be deleted and shown on the graph to be able to make a convincing analysis.

Computational Effort for Search

I implemented the search in Python 3.0 and used my laptop with Intel Core i7-7700HQ and 16 gigs of main memory. On the small dataset, Forward Selection took 40.6003 seconds and Backward Elimination took 39.0859 seconds. On the large dataset, Forward Selection took 5.04 hours (18148.6071 seconds) and Backward Elimination took 5.80 hours (20893.1775 seconds). Note that both Forward Selection and Backward Elimination only ran up to 10 features on the large dataset.

Traces

Trace (on Small file for Forward and Backward, only included some output for brevity)

```
Welcome to my Feature Selection Algorithm.
Type in the name of the file to test: small.txt
Type in the number of the algorithm you want to run:
1) Forward Selection
2) Backward Elimination
1
Beginning search.
Initial accuracy: 84.0% with current set []
On the 1th level of the search tree
        Considering adding the 1 feature
        Using feature(s) [1] accuracy is 72.0%
        Considering adding the 2 feature
        Using feature(s) [2] accuracy is 74.66%
        Considering adding the 3 feature
        Using feature(s) [3] accuracy is 77.0%
        Considering adding the 4 feature
        Using feature(s) [4] accuracy is 70.66%
        Considering adding the 5 feature
        Using feature(s) [5] accuracy is 69.66%
        Considering adding the 6 feature
        Using feature(s) [6] accuracy is 77.33%
        Considering adding the 7 feature
        Using feature(s) [7] accuracy is 74.0%
        Considering adding the 8 feature
        Using feature(s) [8] accuracy is 71.66%
        Considering adding the 9 feature
        Using feature(s) [9] accuracy is 86.0%
        Considering adding the 10 feature
        Using feature(s) [10] accuracy is 73.33%
Features set 9 was best, accuracy is 86.0%
On level 1, I added feature 9 with accuracy 86.0% to previous set [] to ha
ve current set [9]
On the 2th level of the search tree
…
Features set 1 was best, accuracy is 79.0%
On level 10, I added feature 1 with accuracy 79.0% to previous set [9, 3,
10, 5, 2, 7, 4, 8, 6] to have current set [9, 3, 10, 5, 2, 7, 4, 8, 6, 1]
Warning, Accuracy has decreased! Continuing search in case of local maxima
```

```
Finished search in 40.6003 seconds!! The best feature subset is [9, 3, 10]
, which has an accuracy of 93.0%



Beginning search.
Initial accuracy: 79.0% with current set [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
On the 1th level of the search tree
Considering deleting the 1 feature
        Using feature(s) [2, 3, 4, 5, 6, 7, 8, 9, 10] accuracy is 80.33%
        Considering deleting the 2 feature
        Using feature(s) [1, 3, 4, 5, 6, 7, 8, 9, 10] accuracy is 79.0%
        Considering deleting the 3 feature
        Using feature(s) [1, 2, 4, 5, 6, 7, 8, 9, 10] accuracy is 77.33%
        Considering deleting the 4 feature
        Using feature(s) [1, 2, 3, 5, 6, 7, 8, 9, 10] accuracy is 77.66%
        Considering deleting the 5 feature
        Using feature(s) [1, 2, 3, 4, 6, 7, 8, 9, 10] accuracy is 78.66%
        Considering deleting the 6 feature
        Using feature(s) [1, 2, 3, 4, 5, 7, 8, 9, 10] accuracy is 76.66%
        Considering deleting the 7 feature
        Using feature(s) [1, 2, 3, 4, 5, 6, 8, 9, 10] accuracy is 80.0%
        Considering deleting the 8 feature
        Using feature(s) [1, 2, 3, 4, 5, 6, 7, 9, 10] accuracy is 80.66%
        Considering deleting the 9 feature
        Using feature(s) [1, 2, 3, 4, 5, 6, 7, 8, 10] accuracy is 73.66%
        Considering deleting the 10 feature
        Using feature(s) [1, 2, 3, 4, 5, 6, 7, 8, 9] accuracy is 75.0%
Features set 8 was best, accuracy is 80.66%
On level 1, I deleted feature 8 with accuracy 80.66% from previous set [1,
2, 3, 4, 5, 6, 7, 8, 9, 10] to have current set [1, 2, 3, 4, 5, 6, 7, 9, 1
0]
On the 2th level of the search tree
…
Features set 9 was best, accuracy is 84.0%
On level 10, I deleted feature 9 with accuracy 84.0% from previous set [9]
to have current set []
Warning, Accuracy has decreased! Continuing search in case of local maxima
Finished search in 39.0859 seconds!! The best feature subset is [3, 9], wh
ich has an accuracy of 92.33%
Forward Elapsed Time: 40.6003 seconds
Backward Elapsed Time: 39.0859 seconds
```

Trace (on Large file for Forward and Backward, only included some output for brevity)

```
Welcome to my Feature Selection Algorithm.
Type in the name of the file to test: large.txt
Type in the number of the algorithm you want to run:
1) Forward Selection
2) Backward Elimination
1
Beginning search.
Initial accuracy: 80.33% with current set []
On the 1th level of the search tree
…

Features set 126 was best, accuracy is 83.55%
On level 1, I added feature 126 with accuracy 83.55% to previous set [] to
have current set [126]
On the 2th level of the search tree
…

Features set 214 was best, accuracy is 81.0%
On level 10, I added feature 214 with accuracy 81.0% to previous set [126,
45, 134, 93, 41, 111, 23, 57, 34] to have current set [126, 45, 134, 93, 4
1, 111, 23, 57, 34, 214]
Warning, Accuracy has decreased! Continuing search in case of local maxima
Finished search in 18148.6071 seconds!! The best feature subset is [126, 4
5], which has an accuracy of 95.22%



Beginning search.
Initial accuracy: 71.22% with current set [1, 2, 3, 4, 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, 41, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66
, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102,
103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132,
133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147,
148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162,
163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250]
On the 1th level of the search tree
…

Features set 193 was best, accuracy is 72.11%
On level 1, I deleted feature 193 with accuracy 72.11% from previous set [
1, 2, 3, 4, 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
, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
```

59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77
, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111
, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 12
6, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 1
41, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170,
171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185,
186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200,
201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215,
216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230,
231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
246, 247, 248, 249, 250] to have current set [1, 2, 3, 4, 5, 6, 7, 8, 9, 1
0, 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, 41, 42, 43, 44, 45, 46, 47
, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84
, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 10
2, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 1
17, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131,
132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176,
177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191,
192, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250]
On the 2th level of the search tree
…

Features set 98 was best, accuracy is 76.66%
On level 10, I deleted feature 98 with accuracy 76.66% from previous set [
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22,
23, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44
, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82
, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,
101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 132, 133, 134, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
147, 148, 149, 150, 151, 152, 154, 155, 156, 157, 158, 159, 160, 161, 162,
163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224,
225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239,
240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250] to have current set
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22
, 23, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 39, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 99, 100, 1
01, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,

116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 132, 133, 134, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
147, 148, 149, 150, 151, 152, 154, 155, 156, 157, 158, 159, 160, 161, 162,
163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224,
225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239,
240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250]
Finished search in 20893.1775 seconds!! The best feature subset is [1, 2,
3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 2
5, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64
, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 99, 100, 101, 102,
103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132,
133, 134, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148,
149, 150, 151, 152, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164,
165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179,
180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195,
196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 210, 211,
212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226,
227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241,
242, 243, 244, 245, 246, 247, 248, 249, 250], which has an accuracy of 76.
66%
Forward Elapsed Time: 18148.6071 seconds
Backward Elapsed Time: 20893.1775 seconds

Code

Project Code Description

I used Python 3 and used the copy, numpy, pandas, time, and math libraries in the project. I first prompt the user to enter in a file and the algorithm to run first, and then I used panda and numpy to get the data from the provided text files. I then run either feature search Forward Selection or Backward Elimination which is heavily based off the provided project pseudocode from the slides. The calculation of the accuracy is also heavily based off the provided project pseudocode from the slides. Then, I also run the remaining feature search that the user did not choose so that I can graph both the Forward Selection and Backward Elimination data. I then graph the Forward Selection and Backward Elimination data.

Project Code

The GitHub repository URL for this project is:
https://github.com/marcusamartin/CS205_FeatureSelection

Below is my code for the project:

```
import copy
from copy import deepcopy
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
import math

# used code for truncating from
https://www.delftstack.com/howto/python/python-truncate-float-python/
def truncate(number, decimal):
 integer = int(number * (10 ** decimal)) / (10 ** decimal)
 return float(integer)

def init():
    print("Welcome to my Feature Selection Algorithm.")
    fileName = input("Type in the name of the file to test: ")
    algoNum = int(input("Type in the number of the algorithm you want to
run:\n1) Forward Selection\n2) Backward Elimination\n"))

    # https://www.geeksforgeeks.org/python-read-csv-using-pandas-read_csv/
    return (pd.read_csv(fileName, delim_whitespace=True,
header=None).to_numpy(), algoNum)

def forwardGraph(features, accuracies):
    fig, ax = plt.subplots(figsize=(15, 15))
    ax.set_title('Forward Selection')
```

```
    ax.bar(features, accuracies, tick_label=features)
    ax.set_xlabel('Features')
    plt.xticks(rotation="vertical")
    ax.set_ylabel('Accuracies (decimal)')
def backwardGraph(features, accuracies):
#     featuresLarge = ['AF*', '193', '24', '209', '135', '17', '153',
'68', '40', '33', '98']
#     accuraciesLarge = [71.22, 72.11, 72.77, 73.66, 74.55, 74.77, 75.66,
75.88, 76.11, 76.11, 76.66]
#     features = featuresLarge
#     accuracies = accuraciesLarge

    fig, ax = plt.subplots(figsize=(15, 15))
    ax.set_title('Backward Selection')
    ax.bar(features, accuracies, tick_label=features)
    ax.set_xlabel('Features')
#     ax.set_xlabel('Features Deleted')
    plt.xticks(rotation="vertical")
#     plt.xticks(rotation="vertical")
    ax.set_ylabel('Accuracies (decimal)')


# need to convert to string so that we can use them for graph (can't use
ints for graph axis labeling)
# https://www.geeksforgeeks.org/python-convert-list-of-lists-to-list-of-
strings/
def convert(features):
    features = [', '.join(map(str, ele)) for ele in features]
    return features

def leaveOneOutCrossValidation(rows):
    numberCorrectlyClassified = 0

    for i in range(len(rows)):
        objectToClassify = rows[i, 1:]
        labelObjectToClassify = rows[i, 0]
        nearestNeighborDistance = float("inf")
        nearestNeighborLocation = float("inf")

        for j in range(len(rows)):
            if (j != i):
                distance = np.sqrt(np.sum(np.square(objectToClassify -
rows[j, 1:])))
                if (distance < nearestNeighborDistance):
                    nearestNeighborDistance = distance
```

```
                    nearestNeighborLocation = j
                    nearestNeighborLabel = rows[nearestNeighborLocation,
0]

        if (labelObjectToClassify == nearestNeighborLabel):
            numberCorrectlyClassified += 1

    accuracy = numberCorrectlyClassified / len(rows)

    return accuracy

def forwardFeatureSearch(rows):
    print("Beginning search.")
    start = time.time()

    features = []
    accuracies = []
    currentSetOfFeatures = [0]   # "empty" set
    bestAccuracy = 0

    # empty set accuracy
    currentSetAdded1 = copy.deepcopy(currentSetOfFeatures)
    rowsAdded2 = rows[:, currentSetAdded1]
    accuracy2 = leaveOneOutCrossValidation(rowsAdded2)
    currentSetAdded2 = copy.deepcopy(currentSetAdded1)
    currentSetAdded2.pop(0)   # remove class label
    print("Initial accuracy: " + str(truncate(accuracy2 * 100, 2)) + "%
with current set " + str(currentSetAdded2))
    features.append(currentSetAdded1)
    accuracies.append(accuracy2)

    for i in range(1, len(rows[0])):
        print("On the " + str(i) + "th level of the search tree")
        currentSet = copy.deepcopy(currentSetOfFeatures)
        featureToAdd = []
        bestSoFarAccuracy = 0

        for j in range(1, len(rows[0])):
            if (j not in currentSetOfFeatures):
                currentSet.append(j)
                rowsAdded = rows[:, currentSet]

                accuracy = leaveOneOutCrossValidation(rowsAdded)
                print("\tConsidering adding the " + str(j) + " feature")
                selectedCurrentSetAdded = copy.deepcopy(currentSet)
```

```
                selectedCurrentSetAdded.pop(0)
                resetSet = copy.deepcopy(currentSetOfFeatures)
                currentSet = resetSet
                print("\tUsing feature(s) " + str(selectedCurrentSetAdded)
+ " accuracy is " + str(truncate(accuracy * 100, 2)) +"%")

                if (accuracy > bestSoFarAccuracy):
                    bestSoFarAccuracy = accuracy
                    featureToAdd = j

        prevCurrentSetOfFeatures = copy.deepcopy(currentSetOfFeatures)
        prevCurrentSetOfFeatures.pop(0)
        currentSetOfFeatures.append(featureToAdd)
        features.append(copy.deepcopy(currentSetOfFeatures))
        accuracies.append(bestSoFarAccuracy)

        selectedCurrentSetOfFeatures = copy.deepcopy(currentSetOfFeatures)
        selectedCurrentSetOfFeatures.pop(0)

        print("Features set " + str(featureToAdd) + " was best, accuracy
is " + str(truncate(bestSoFarAccuracy * 100, 2)) + "%")
        print("On level " + str(i) + ", I added feature " +
str(featureToAdd) + " with accuracy " + str(truncate(bestSoFarAccuracy *
100, 2)) + "% to previous set " + str(prevCurrentSetOfFeatures) + " to
have current set " + str(selectedCurrentSetOfFeatures))

        if (bestAccuracy > bestSoFarAccuracy):
            print("Warning, Accuracy has decreased! Continuing search in
case of local maxima")
        elif (bestSoFarAccuracy > bestAccuracy):
            bestAccuracy = bestSoFarAccuracy

    end = time.time()
    elapsedTime = str(truncate(end - start, 4))

    # find best features
    count = 0
    max2 = 0
    index = 0
    for i in accuracies:
        if i > max2:
            max2 = i
            index = count
        count = count + 1
```

```
    temp = copy.deepcopy(features)
    bestFeatures = temp[index]
    bestFeatures.pop(0)

    print("Finished search in " + str(elapsedTime) + " seconds!! The best
feature subset is " + str(bestFeatures) + ", which has an accuracy of " +
str(truncate(bestAccuracy * 100, 2)) + "%")

    return (features, accuracies, elapsedTime)

def backwardFeatureSearch(rows):
    print("Beginning search.")
    start = time.time()

    features = []
    accuracies = []
    currentSetOfFeatures = [0]   # "empty" set
    bestAccuracy = 0

    count = 1
    for i in range(1, len(rows[0])):
        currentSetOfFeatures.append(count)
        count += 1

    # full set accuracy
    currentSetAdded1 = copy.deepcopy(currentSetOfFeatures)
    rowsAdded2 = rows[:, currentSetAdded1]
    accuracy2 = leaveOneOutCrossValidation(rowsAdded2)
    currentSetAdded2 = copy.deepcopy(currentSetAdded1)
    currentSetAdded2.pop(0)   # remove class label
    print("Initial accuracy: " + str(truncate(accuracy2 * 100, 2)) + "%
with current set " + str(currentSetAdded2))
    features.append(currentSetAdded1)
    accuracies.append(accuracy2)

    for i in range(1, len(rows[0])):
        print("On the " + str(i) + "th level of the search tree")
        currentSet = copy.deepcopy(currentSetOfFeatures)
        featureToDelete = []
        bestSoFarAccuracy = 0

        for j in range(1, len(rows[0])):
            if (j in currentSetOfFeatures):
                currentSet.remove(j)
                rowsDeleted = rows[:, currentSet]
```

```
                accuracy = leaveOneOutCrossValidation(rowsDeleted)
                print("\tConsidering deleting the " + str(j) + " feature")
                selectedCurrentSetDeleted = copy.deepcopy(currentSet)
                selectedCurrentSetDeleted.pop(0)
                resetSet = copy.deepcopy(currentSetOfFeatures)
                currentSet = resetSet
                print("\tUsing feature(s) " +
str(selectedCurrentSetDeleted) + " accuracy is " + str(truncate(accuracy *
100, 2)) +"%")

                if accuracy > bestSoFarAccuracy:
                    bestSoFarAccuracy = accuracy
                    featureToDelete = j

        prevCurrentSetOfFeatures = copy.deepcopy(currentSetOfFeatures)
        prevCurrentSetOfFeatures.pop(0)
        currentSetOfFeatures.remove(featureToDelete)
        features.append(copy.deepcopy(currentSetOfFeatures))
        accuracies.append(bestSoFarAccuracy)

        selectedCurrentSetOfFeatures = copy.deepcopy(currentSetOfFeatures)
        selectedCurrentSetOfFeatures.pop(0)

        print("Features set " + str(featureToDelete) + " was best,
accuracy is " + str(truncate(bestSoFarAccuracy * 100, 2)) + "%")
        print("On level " + str(i) + ", I deleted feature " +
str(featureToDelete) + " with accuracy " + str(truncate(bestSoFarAccuracy
* 100, 2)) + "% from previous set " + str(prevCurrentSetOfFeatures) + " to
have current set " + str(selectedCurrentSetOfFeatures))

        if (bestAccuracy > bestSoFarAccuracy):
            print("Warning, Accuracy has decreased! Continuing search in
case of local maxima")
        elif (bestSoFarAccuracy > bestAccuracy):
            bestAccuracy = bestSoFarAccuracy

    end = time.time()
    elapsedTime = str(truncate(end - start, 4))

    # find best features
    count = 0
    max2 = 0
    index = 0
    for i in accuracies:
```

```
        if i > max2:
            max2 = i
            index = count
        count = count + 1
    temp = copy.deepcopy(features)
    bestFeatures = temp[index]
    bestFeatures.pop(0)

    print("Finished search in " + str(elapsedTime) + " seconds!! The best
feature subset is " + str(bestFeatures)  + ", which has an accuracy of " +
str(truncate(bestAccuracy * 100, 2)) + "%")

    return (features, accuracies, elapsedTime)

def main():
    rows, specifiedFeatureSearch = init()

    forwardFeatures = []
    forwardAccuracies = 0
    backwardFeatures = []
    backwardAccuracies = 0
    forwardElapsedTime = 0
    backwardElapsedTime = 0


    if (specifiedFeatureSearch == 1):
        forwardFeatures, forwardAccuracies, forwardElapsedTime =
forwardFeatureSearch(rows)
    elif (specifiedFeatureSearch == 2):
        backwardFeatures, backwardAccuracies, backwardElapsedTime =
backwardFeatureSearch(rows)

    print("\n\n\n\n\n")   # separation between forward and backward
outputs

    # run opposite feature search for graph
    if (specifiedFeatureSearch == 1):
        backwardFeatures, backwardAccuracies, backwardElapsedTime =
backwardFeatureSearch(rows)
    elif (specifiedFeatureSearch == 2):
        forwardFeatures, forwardAccuracies, forwardElapsedTime =
forwardFeatureSearch(rows)

    print("Forward Elapsed Time: " + str(forwardElapsedTime) + " seconds")
```

```
    print("Backward Elapsed Time: " + str(backwardElapsedTime) + "
seconds")

    return (forwardFeatures, forwardAccuracies, forwardElapsedTime,
backwardFeatures, backwardAccuracies, backwardElapsedTime)

forwardFeatures, forwardAccuracies, forwardElapsedTime, backwardFeatures,
backwardAccuracies, backwardElapsedTime = main()

# remove class labels
forwardFeatures2 = []
temp = copy.deepcopy(forwardFeatures)
for item in temp:
    item.pop(0)
    forwardFeatures2.append(copy.deepcopy(item))
backwardFeatures2 = []
temp2 = copy.deepcopy(backwardFeatures)
for item in temp2:
    item.pop(0)
    backwardFeatures2.append(copy.deepcopy(item))

print(forwardFeatures2)
print(len(forwardFeatures2))
print()
print(forwardAccuracies)
print(len(forwardAccuracies))
print()
print(backwardFeatures2)
print(len(backwardFeatures2))
print()
print(backwardAccuracies)
print(len(backwardAccuracies))
print()

tempForward = copy.deepcopy(forwardFeatures2)
tempBackward = copy.deepcopy(backwardFeatures2)
print(tempForward)
print("\n")
resForward = convert(tempForward)
resBackward = convert(tempBackward)
print(resForward)

forwardGraph(resForward, forwardAccuracies)

backwardGraph(resBackward, backwardAccuracies)
```