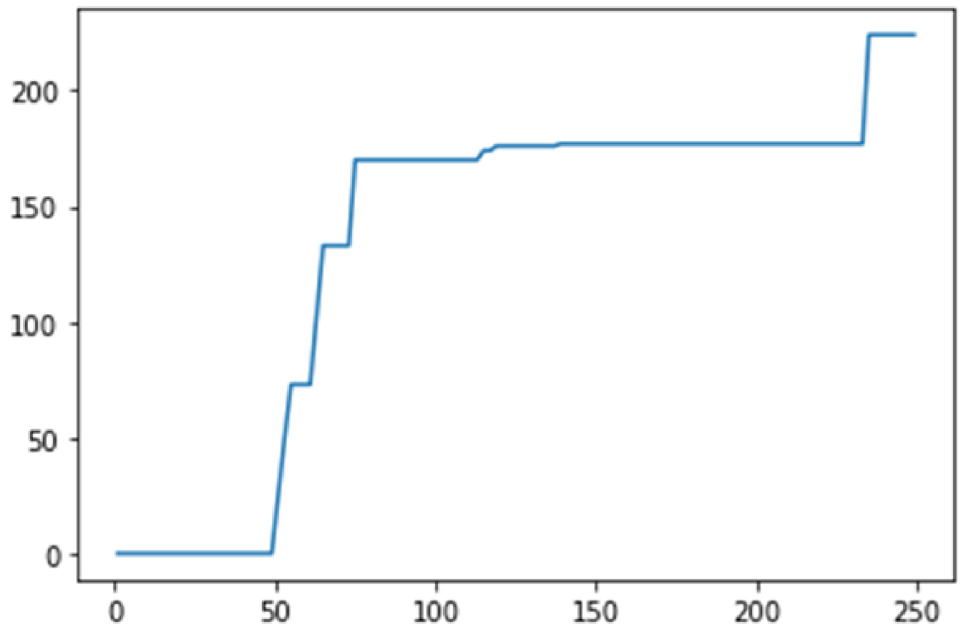
Michael Ryvin

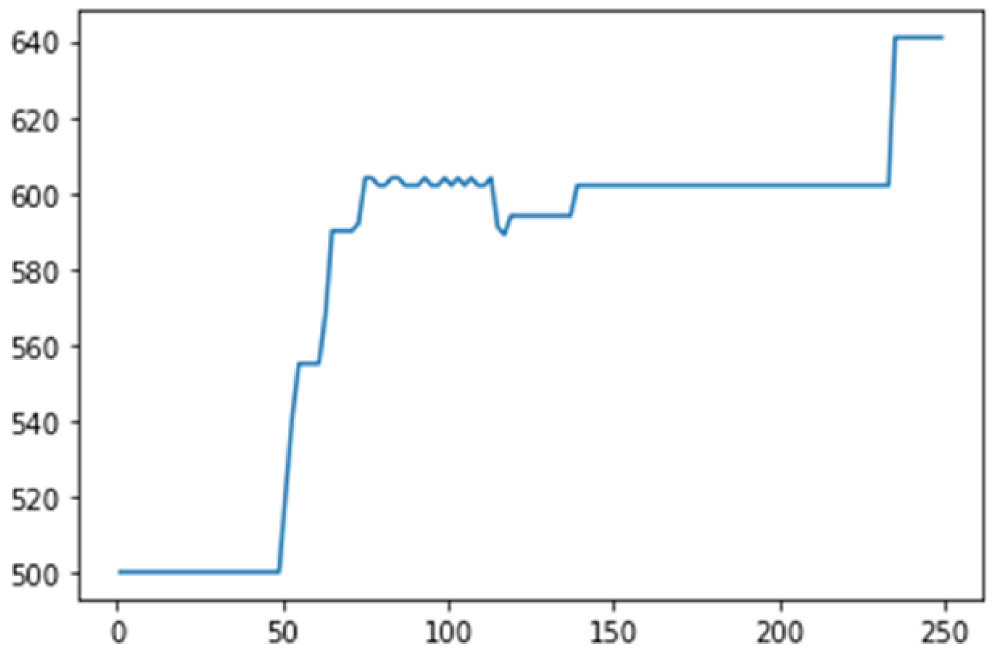
Problem 1:

To preface: these results came from using a training data of size 1000 and testing data of size 500

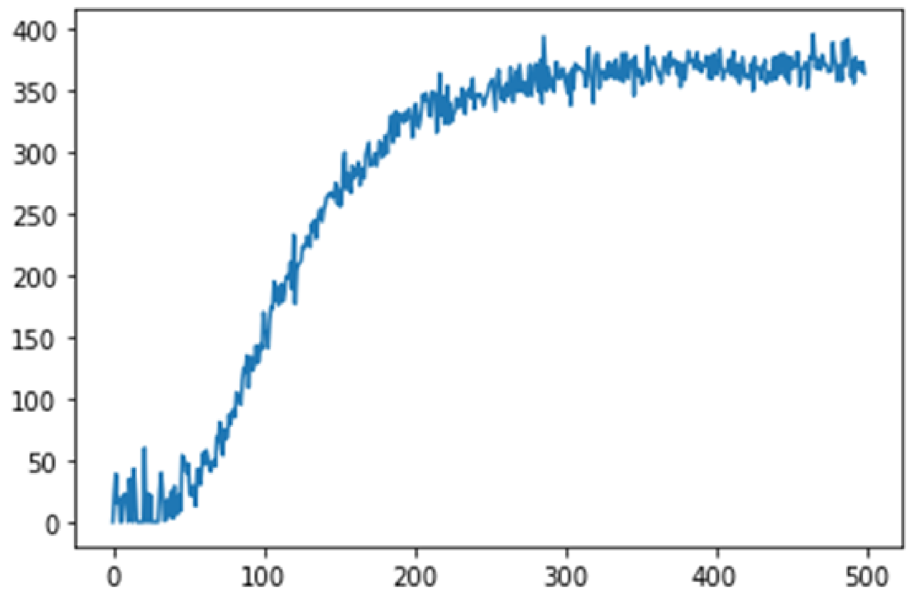
- The tree did not seem to work for me at larger data sizes, though I do not know why. The results seem to work at these lower levels.

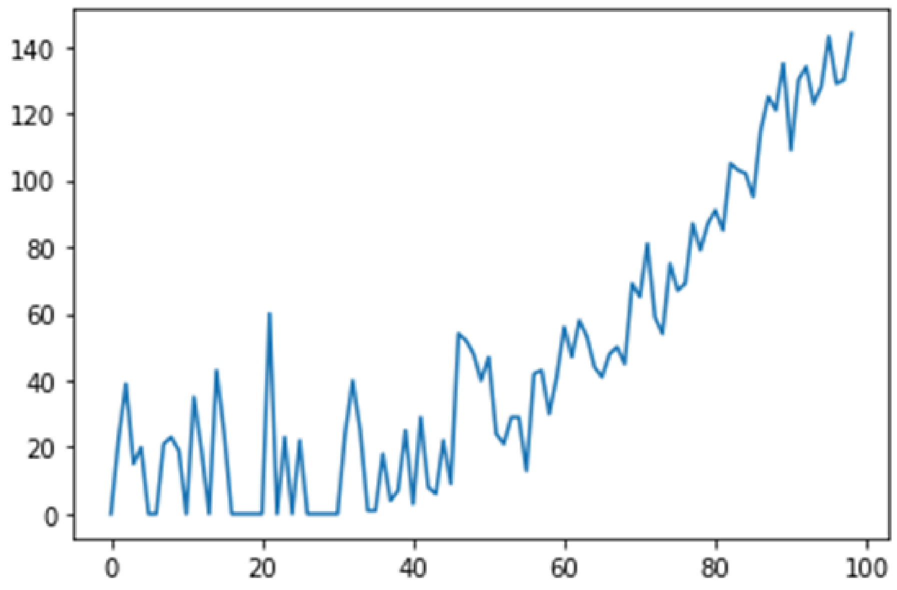
1. According to my data, the optimal size to grow the data to is 49
    a. Plot of training errors:
        i. 

    b. Plot of testing errors:

i. 

2. After several attempts, the optimal size always stays around 50
    a. Because we do not want an even number for the sample size, I will keep the 49 from the previous question as s
3. At lower standard deviations, it appears that the noise is relatively random and has no real pattern regardless of the value of the standard deviation. However, starting from about a value of 0.4 and up, the randomness rapidly increases. This trend ends up plateauing at around a value of 2 and up. This can all be seen in the following two graphs

a. 

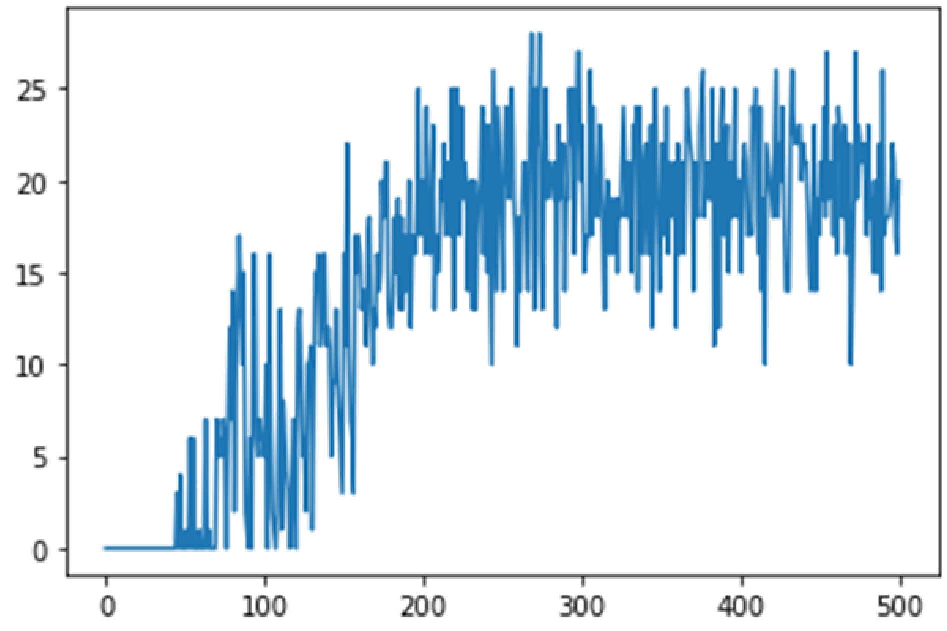
b. 

c. All of this means that once you get to standard deviations above 0.4, the noise have a negative effect on the effectiveness of the decision tree

4. Just like with the last question, at very small standard deviations there is no difference in the likelihood of having unnecessary features. However, once we approach a standard deviation of about 0.5, the number of unnecessary features begins to rise. This eventually ends up plateauing once we get to a standard deviation of about 2. Once it

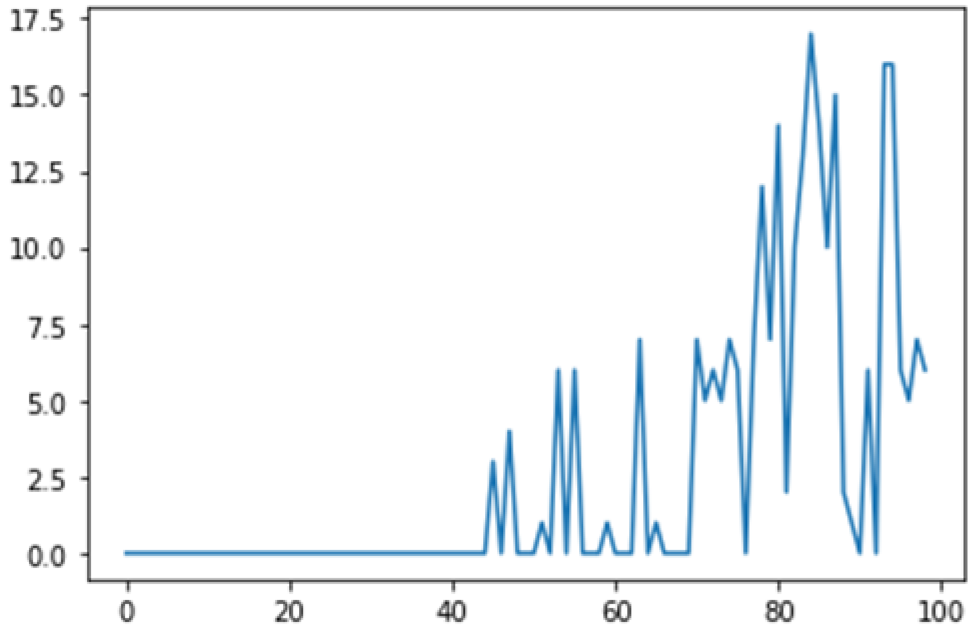
reaches that point, it appears that the average number of unnecessary features is about 20, though the amount can differ drastically from one simulation to the next.

```
[<matplotlib.lines.Line2D at 0x17c9753adf0>]
```



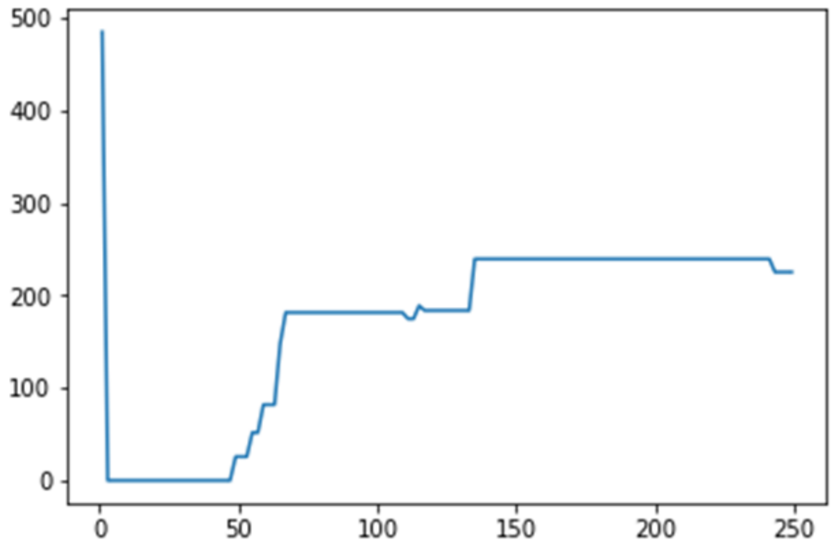
a.

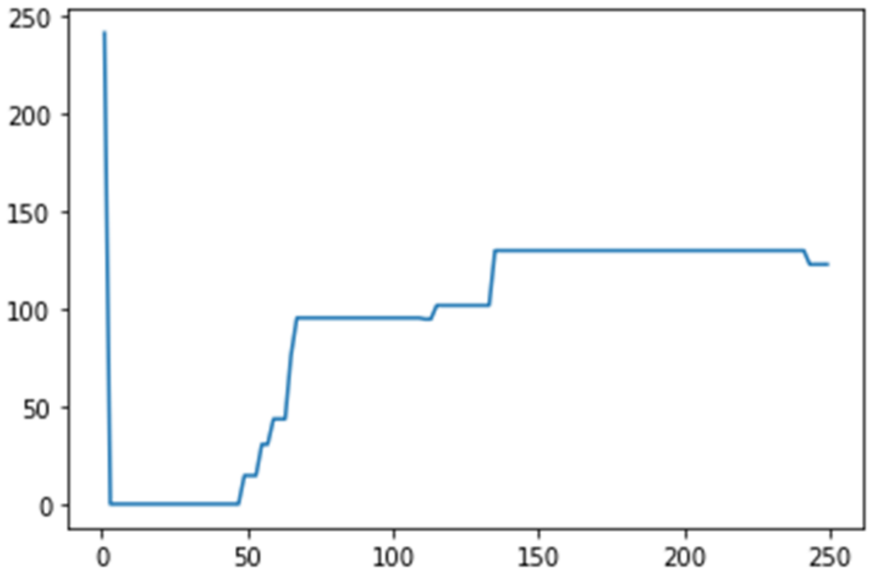
```
[<matplotlib.lines.Line2D at 0x17c9885c5e0>]
```



b.

Problem 2

1. The graphs and data seem to suggest that the optimal sample size to grow the tree to is 47. This is fairly close to the one from the previous section, albeit slightly smaller.
    a. This graph shows the errors for the training data
        i. 

    b. This graph shows the errors for the testing data
        i. 

2. Could not figure out how to do

  a. In theory if i had completed it, overfitting would be an issue. This is because the logistic model would almost certainly be using every data point in the model
    i. This would mean that data points that theoretically should have little to no effect on the value of y would be part of the model
    ii. meaning that if the data were generated again, the model would produce different weights for those variables, as the weights with the original data would not necessarily work with other data

3. Due to not completing question 2, I cannot accurately answer this question. However…
  a. To determine the influence of irrelevant features, I would compare how many data points have their value of Y changed depending on the value they have for x_6 to x_{15}.
    i. For logistic regression, check how many y values change if the weights for x_6 to x_{15} are set to zero compared to how they are by default
    ii. For decision trees, create a new decision tree only involving x_1 to x_5 and compare this with the original decision tree. Make note of how many y values changed.
    iii. Compare the amount of y values that changed. The model with fewer changes has a smaller influence from irrelevant features, while the model with more changes has more influence from irrelevant features

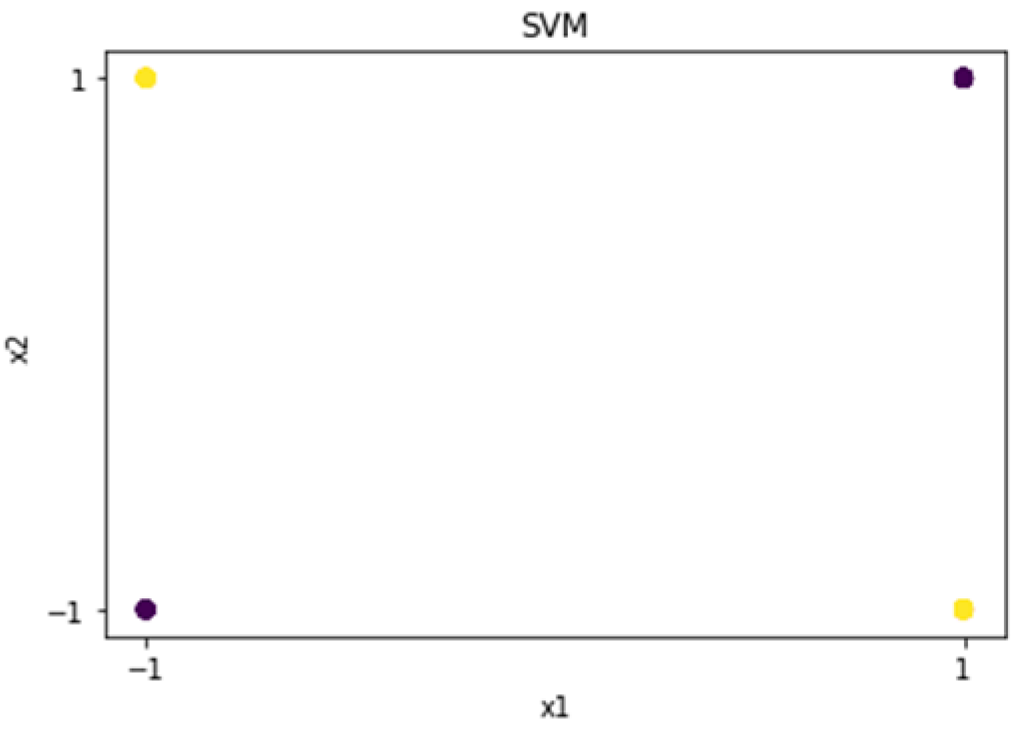
Problem 3

The following Kernel is used:

sign(x_1 * -x_2)

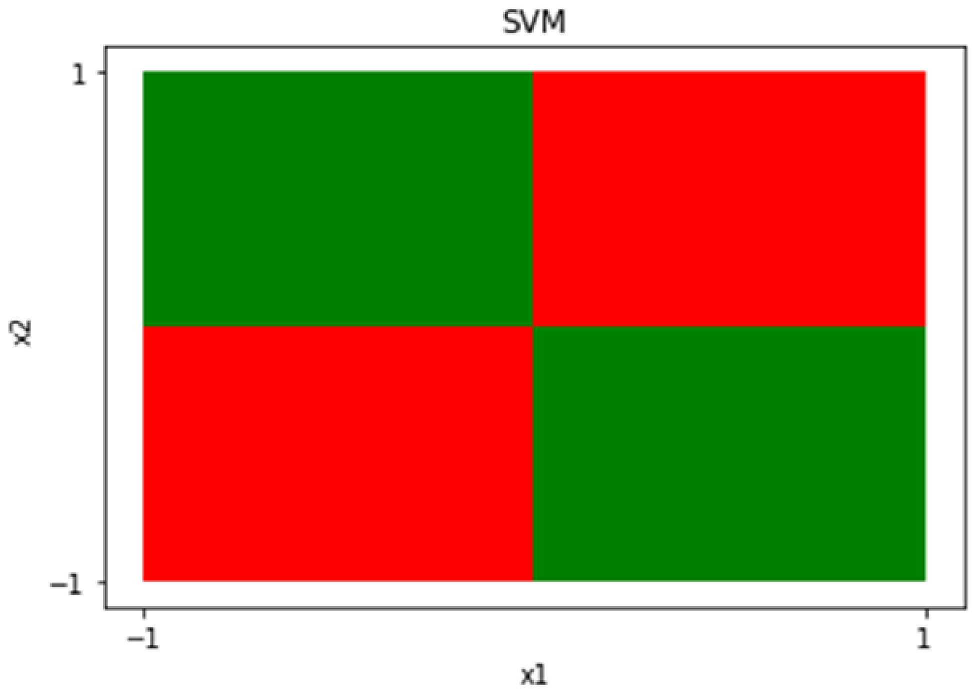
- This means the following
  - If both values are positive, then y will be negative
  - If x_1 is positive and x_2 is negative, then y will be positive
  - If x_1 is negative and x_2 is positive, then y will be positive
  - if both values are negative, then y will be negative

This is how the Dual SVM looks plotted



-
  - Where yellow indicates a positive Y and maroon indicates a negative Y
  - As long as alpha is positive, these are the results

This is a more clear display of the classification regions:



-
    - Where Green indicates a positive value for Y, and red indicates a negative value for y