Katie Hughes

Tim Dufala

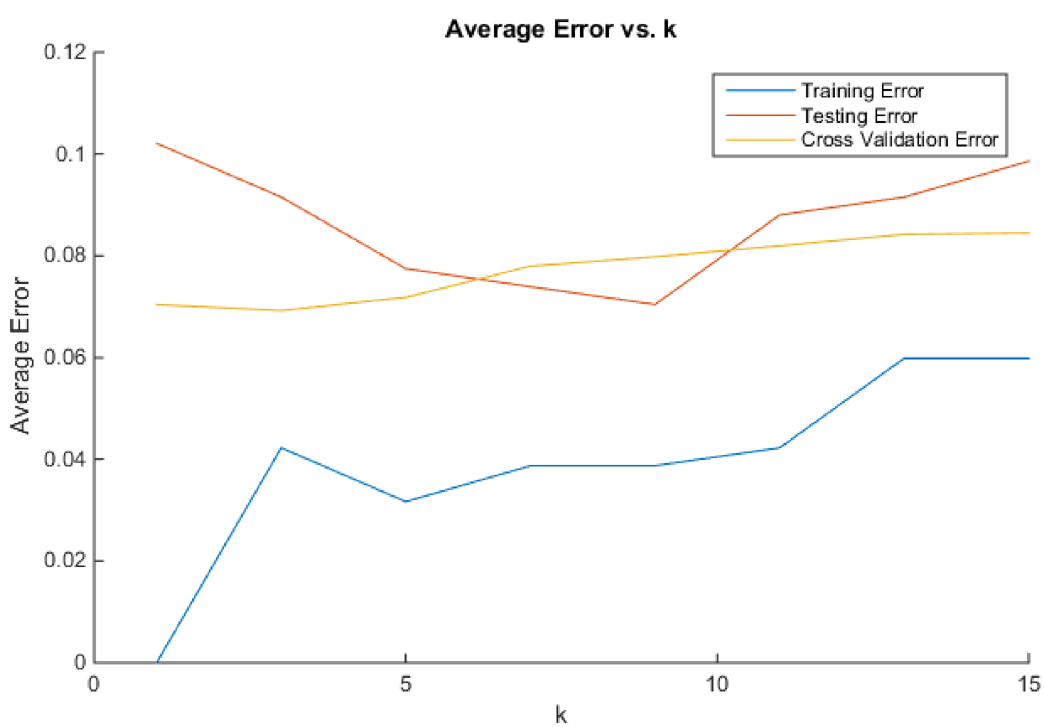
Jasper LaFortune

Assignment 3

Part I

We implemented a K-Nearest Neighbors algorithm to predict breast cancer based on 30 features. The training set and testing set each consisted of 284 examples. We performed leave-one-out cross-validation on the training set for k values of odd numbers from 1 to 15. Based on cross-validation our choice of k is 3, where the average error is lowest at 6.9%.

We also computed the average error for each of the above k values on the training and testing sets. Our results are below:



All three types of error have a minimum somewhere between 3 and 9 (excluding the trivial minimum at k = 1 for the training error). For k values larger than 9, average error increases for all error types due to underfitting. All error types show that a k value of 1 overfits the data.

Part II

For this project we learned a decision tree on the synthetic MONKS dataset. We used information gain to measure the benefit of each decision and halted tree growth below a certain information gain threshold.

Our Decision Stump:

```
x5 == 1 benefit = 0.286200761221 +62 -62
```

Our Decision Tree:

```
x5 == 1 benefit = 0.286200761221 +62 -62
|x1 == 1 benefit = 0.0465802044855 +33 -62
||x2 == 1 benefit = 0.420861388684 +25 -31
|||x5 == 3 benefit = 0.0205013176587 +25 -11
||||x4 == 1 benefit = 0.0486213526342 +18 -6
|||||x6 == 1 benefit = 0.0597731301493 +10 -5
||||||x5 == 4 benefit = 0.152007283806 +7 -2
|||||||x1 == 1 benefit = 0 +3 -0
|||||||x3 == 1 benefit = 0.109170338676 +4 -2
||||||||x1 == 1 benefit = 0 +1 -0
||||||||x1 == 2 benefit = 0.019973094022 +3 -2
|||||||||x2 == 2 benefit = 0.918295834054 +2 -1
||||||||||x1 == 1 benefit = 0 +2 -0
||||||||||x1 == 1 benefit = 0 +0 -1
|||||||||x2 == 2 benefit = 1.0 +1 -1
||||||||||x1 == 1 benefit = 0 +0 -1
||||||||||x1 == 1 benefit = 0 +1 -0
||||||x3 == 1 benefit = 0.190874504621 +3 -3
|||||||x1 == 2 benefit = 0.419973094022 +2 -3
||||||||x1 == 1 benefit = 0 +0 -2
||||||||x2 == 2 benefit = 0.918295834054 +2 -1
|||||||||x1 == 1 benefit = 0 +0 -1
|||||||||x1 == 1 benefit = 0 +2 -0
|||||||x1 == 1 benefit = 0 +1 -0
```

```
|||||x1 == 2 benefit = 0.102187170949 +8 -1
||||||x2 == 2 benefit = 0.721928094887 +4 -1
|||||||x1 == 1 benefit = 0 +4 -0
|||||||x1 == 1 benefit = 0 +0 -1
||||||x1 == 1 benefit = 0 +4 -0
||||x2 == 2 benefit = 0.104348971095 +7 -5
|||||x1 == 2 benefit = 0.985228136034 +3 -4
||||||x1 == 1 benefit = 0 +3 -0
||||||x1 == 1 benefit = 0 +0 -4
|||||x1 == 2 benefit = 0.721928094887 +4 -1
||||||x1 == 1 benefit = 0 +0 -1
||||||x1 == 1 benefit = 0 +4 -0
|||x1 == 1 benefit = 0 +0 -20
||x2 == 1 benefit = 0.732066690093 +8 -31
|||x1 == 1 benefit = 0 +0 -31
|||x1 == 1 benefit = 0 +8 -0
|x1 == 1 benefit = 0.0 +29 -0
```

Error Rates:

```
Training Stump Error: 0.266129032258
Testing Stump Error: 0.25
Training Tree Error: 0.0
Testing Tree Error: 0.0740740740741
```

Given the formula used to generate the classifications, the optimal decision tree would look as such:

```
has_tie
|head_shape = round
||body_shape = round
|||if true -> Classify as positive
|||if false -> Classify as negative
||head_shape = square
|||body_shape = square
||||if true -> Classify as positive
||||if false -> Classify as negative
|||body_shape = octagon
||||if true -> Classify as positive
||||if false -> Classify as negative
|if false -> Classify as negative
```

The greedy algorithm would not necessarily learn this optimal tree. The decision of whether head_shape = body_shape must be broken down into each possibility for head_shape and body_shape. Consequently, the first decision made does not necessarily gain any information immediately. It may happen that whether a monk is holding a sword happens to gain more information than which body type the monk has, in which case the greedy algorithm will take that step first.