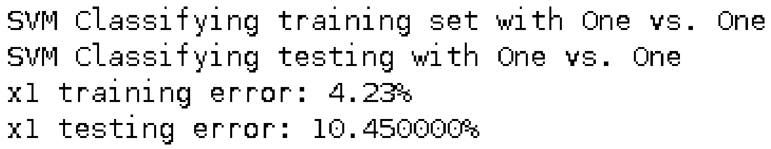
Tristan Pitts
Intro to Machine Learning
Final Project

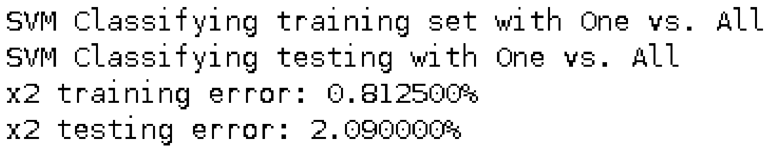
1c)



```
SVM Classifying training set with One vs. One
SVM Classifying testing with One vs. One
x1 training error: 4.23%
x1 testing error: 10.450000%
```

Figure 1: output/fp-1-c.png

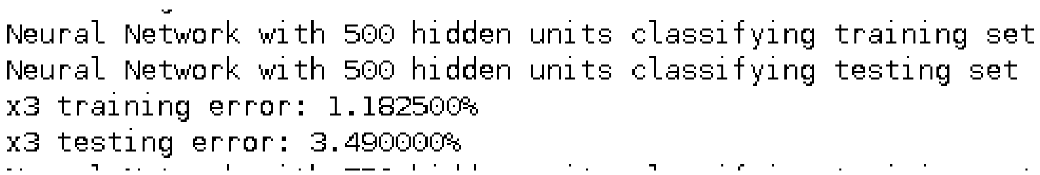
1d)



```
SVM Classifying training set with One vs. All
SVM Classifying testing with One vs. All
x2 training error: 0.812500%
x2 testing error: 2.090000%
```

Figure 2: output/fp-1-d.png

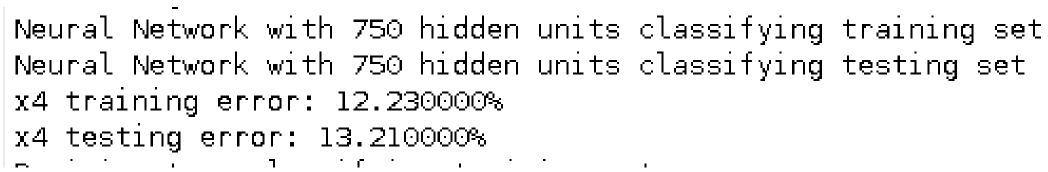
1e)



```
Neural Network with 500 hidden units classifying training set
Neural Network with 500 hidden units classifying testing set
x3 training error: 1.182500%
x3 testing error: 3.490000%
```

Figure 3: output/fp-1-e.png

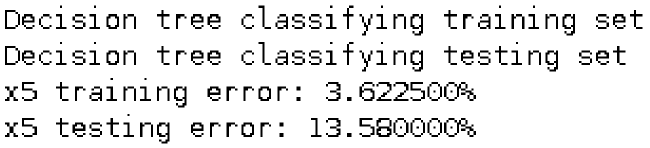
1f)



```
Neural Network with 750 hidden units classifying training set
Neural Network with 750 hidden units classifying testing set
x4 training error: 12.230000%
x4 testing error: 13.210000%
```

Figure 4: output/fp-1-f.png

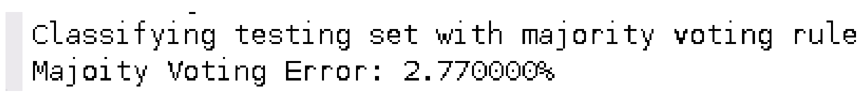
1g)



```
Decision tree classifying training set
Decision tree classifying testing set
x5 training error: 3.622500%
x5 testing error: 13.580000%
```

Figure 5: output/fp-1-g.png

1h)



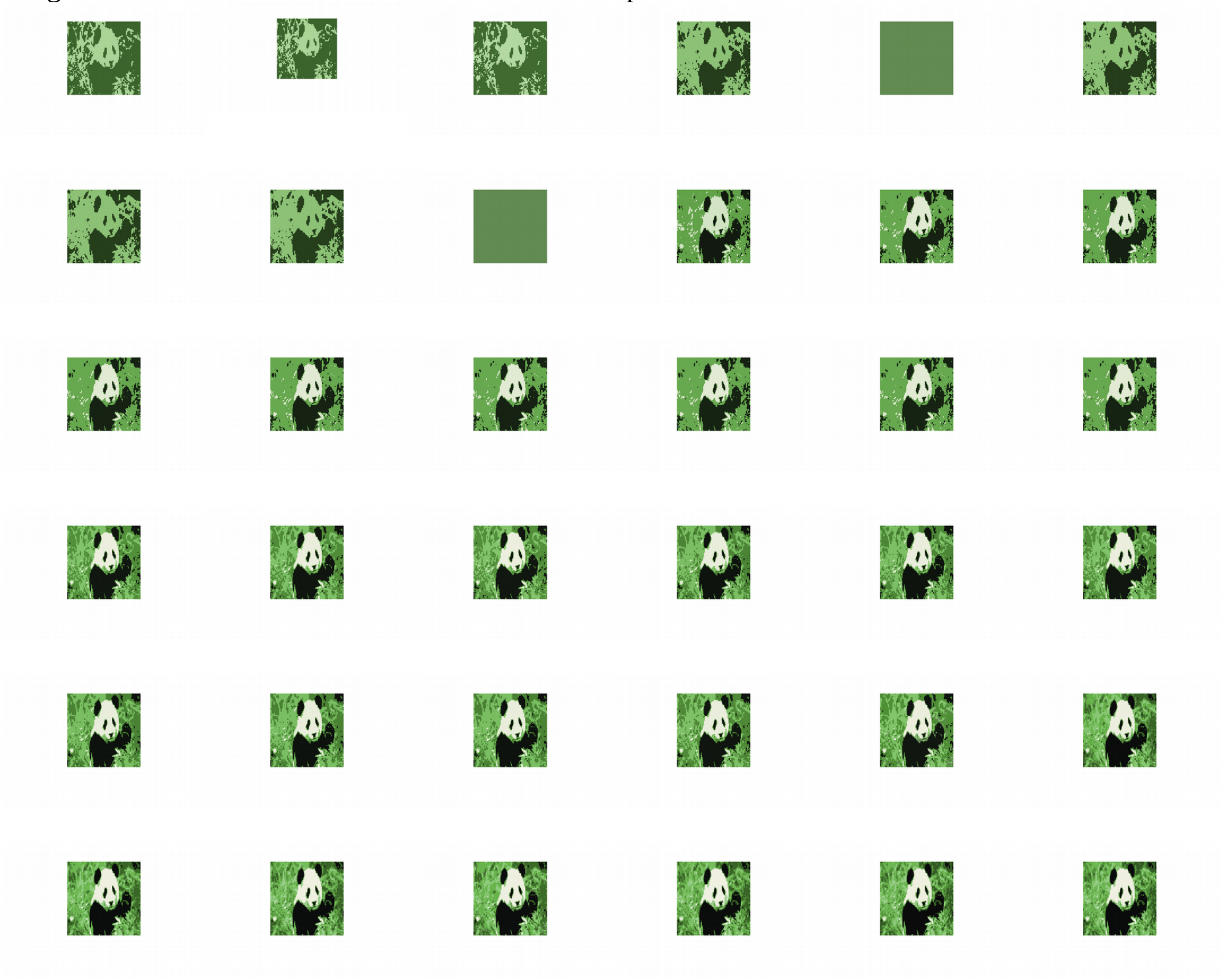
```
Classifying testing set with majority voting rule
Majoity Voting Error: 2.770000%
```

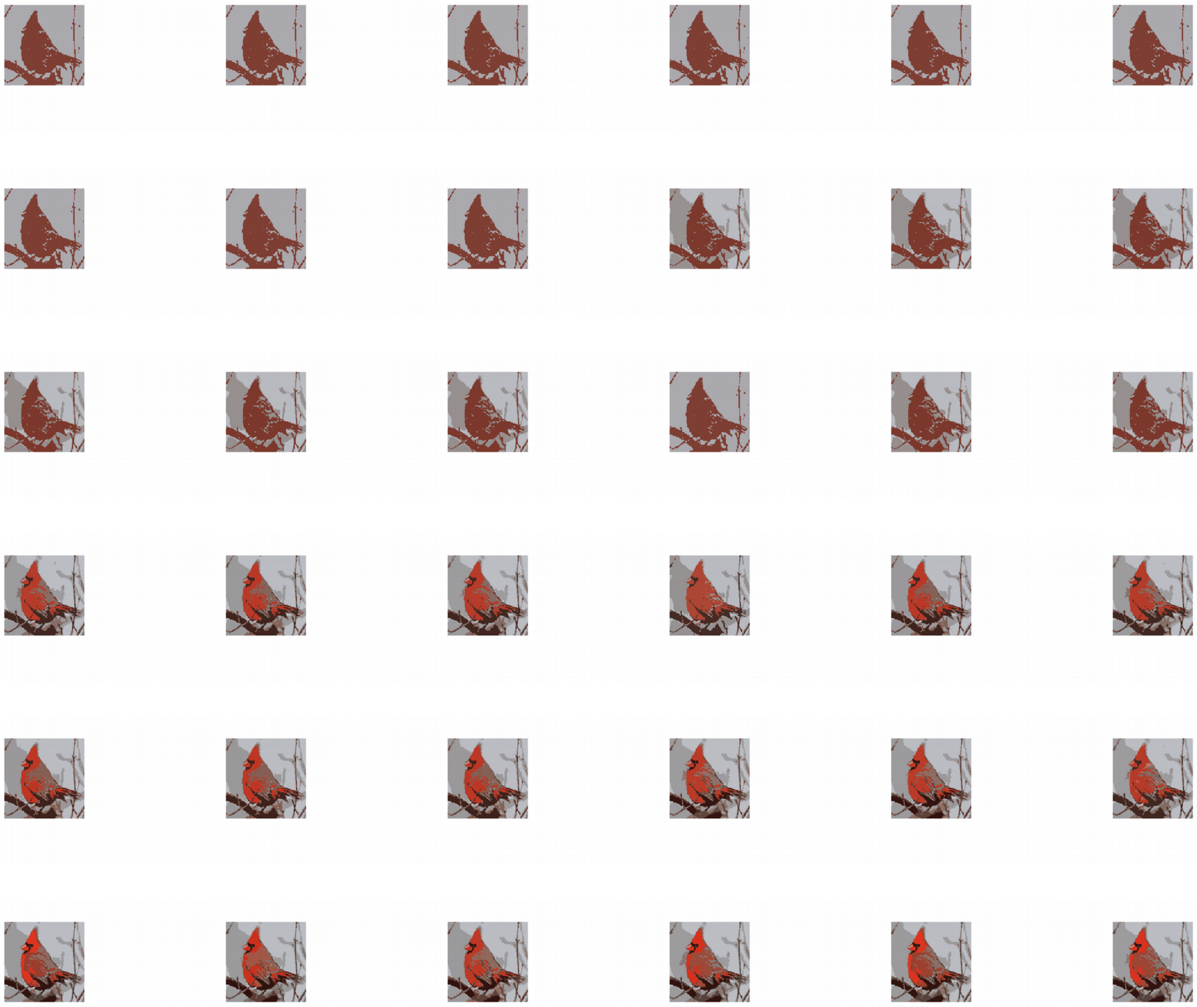
Figure 6: output/fp-1-h.png

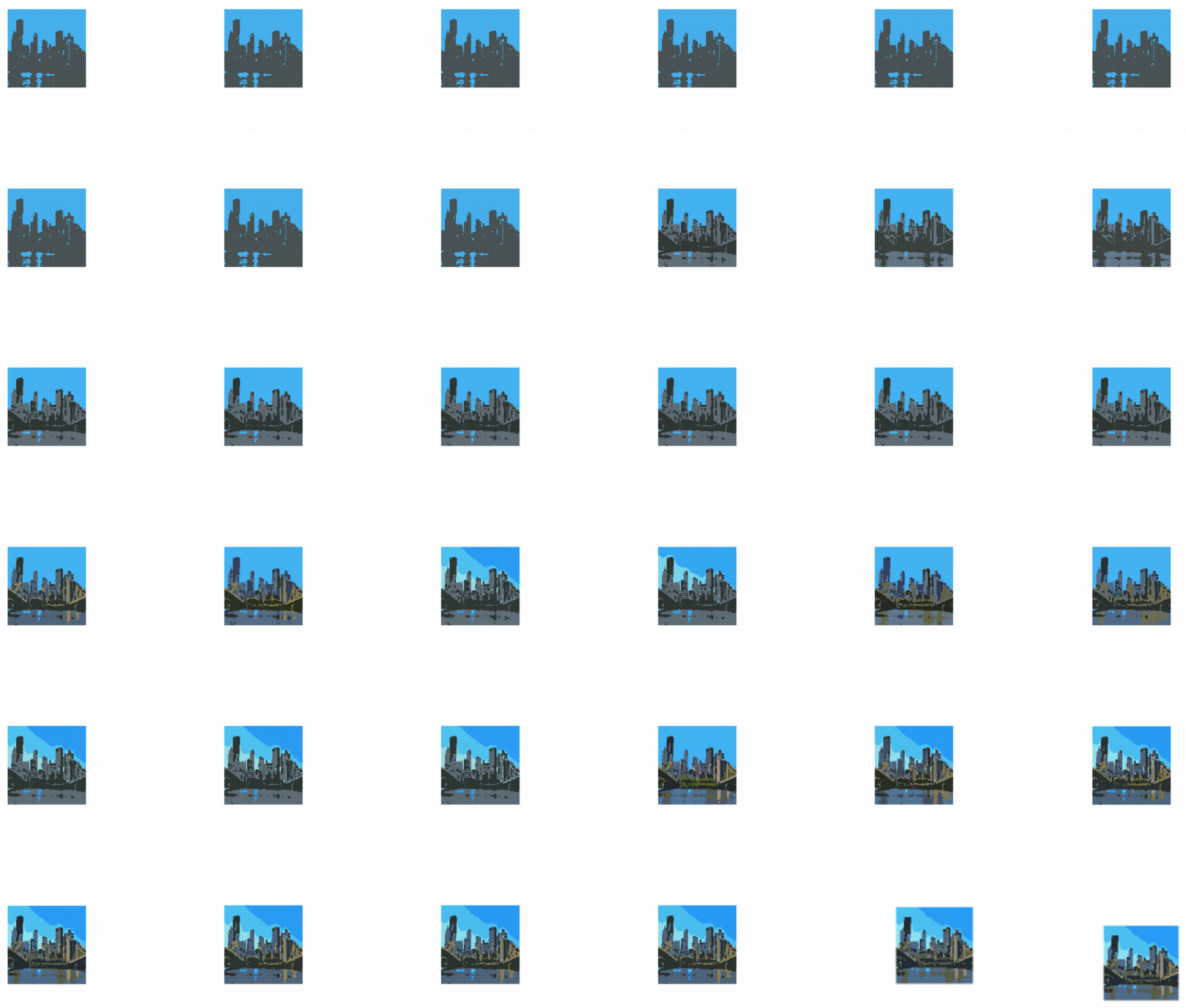
1i) i)In general, each classifier performed fairly well on its own training subset. A few also did well on the testing set alone specifically the neural net with 500 hidden units and the one vs all svm classifier.

ii) I was surprised by the performance of the 750 hidden unit neural net. I expected it to be on par with or better than the 500 hidden unit neural net. Overall, the use of multiple types of classifiers may be a good approach if you're willing to sacrifice time/memory for accuracy. Bagging might help depending how you split up your indecies into subsets to begin with.

2d)
All files in the outputs folder from problem 2 are titled as follows:
fp-2-c-img#-K-iters-R.png
images in the table have the K value increase from top to bottom







With a low K value, each image sort of looks like someone tried to draw the original image in Microsoft Paint. As the K value increases, the images become more detailed. If we think of one cluster as one major color of the image, this makes sense. We can observe this by looking at how the panda image starts out green, then as K increases we get some black then some white then some other greens and by the end we have something that looks pretty similar to our first picture in a much lower quality.

In total, this program took over 12 hours to run in its entirety. Was not able to figure out why two of the img1 images returned were solid green.