Big Data and Machine Learning Final Report

Thomas Pedraza

29 April 2018

1 Generating Data Sets

1.1 My Cats

I created 2 data sets consisting of 4 images each. One set includes a picture of my Russian blue cat Luna, and my calico cat Kitty. These two have very distinct features in color and body type as shown below.

Figure 1: Kitty

Figure 2: Luna

Figure 3: Kitty

Figure 4: Luna

The difference in features between these two cats are distinct, but they're not classifiably seperable as they're both cats. I used an alpha value of 0.5 to mean shift this data set since grayscale values aren't close together. The results are shown below. Figure 6 represents the mean shifted data, and figure 5 represents the raw data of each 8x8 feature in a 256x256 image.

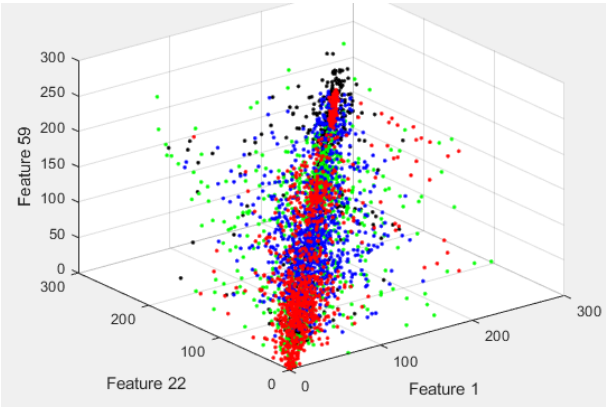

Figure 5: No mean cats

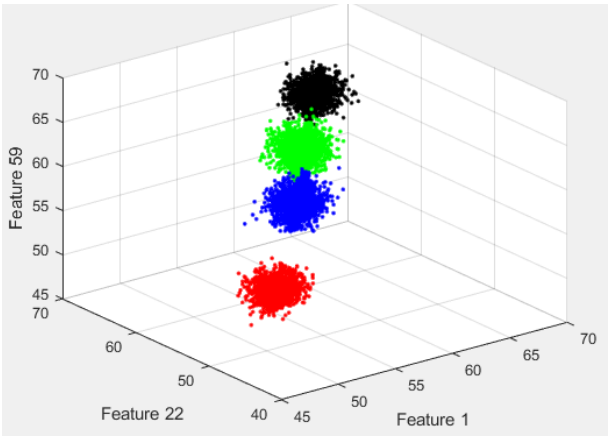

Figure 6: Mean shifted cats

1.2 Yogurt Lids

This data set was a little tricky. The different between the lids was minimal. Color between 2 of the lids were very similar, and the shape of all of the lids were the same. These lids are shown below.

Figure 7: Keylime

Figure 8: Berry

Figure 9: Lemon

Figure 10: Orange

I had to use an alpha value of 5.00 with this data set, because the grayscale values of lemon and orange were very insignificantly different, even the image of the fruit on the front of the lid was nearly exact. left figure below is no mean shift, and right is mean shifted with a value of 5.00

1.3 Conclusion

Making the distinction between yogurt lids is very difficult, they're the same shape and too similar in features. However, with a high enough mean shift it's possible to distinguish the two.

My cats on the other hand are very different in shape and color, making it easier to tell which cat is which. This data set used a low alpha value for mean shifts meaning it would not have a hard time during the training period.

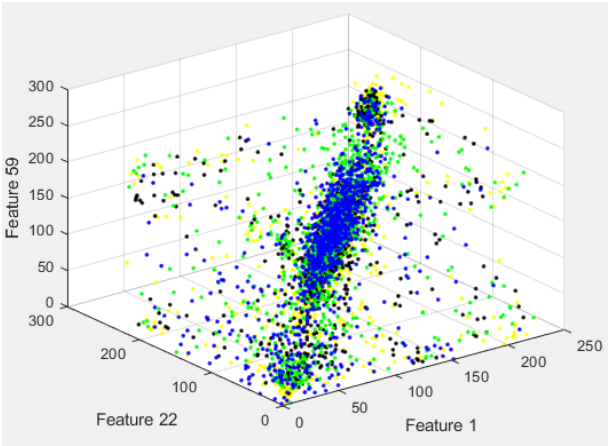

Figure 11: No mean Yogurt

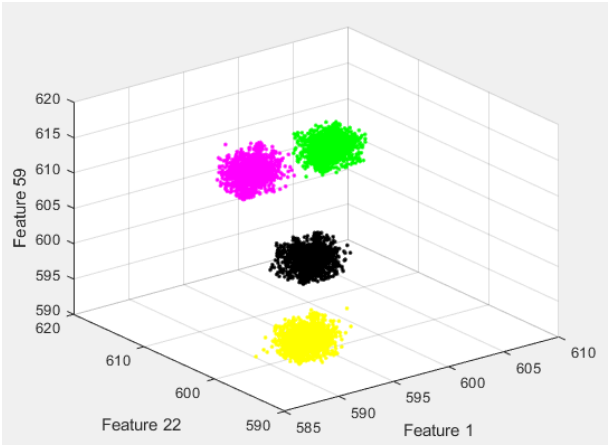

Figure 12: Mean shifted Yogurt

2 Selecting Data Sets

The Cifar-10 dataset is a labeled subset of an 80 million image dataset. The batches contain 50,000 training images and 10,000 test images. The classes are mutually exclusive so theres minimal potential for poor training with this dataset. The mutual exclusivity is the main point of my choice in this set. The training phase is the most important part for machine learning, and this

extensive selection of tiny images will do the job very well.

3 Introduction of the RHadoop System

For this assignment I was tasked with implementing a distributed file system on linux for the purpose of big data processing in the final assignment. I used Hadoop 2.6 on Ubuntu 14.04.

3.1 Installing Java

The Hadoop framework is written in java, so my first step was installing JDK version 1.7.0 on Ubuntu in my virtual machine.

```
tom@VirtualBox:~$ cd ~

# Update the source list
tom@VirtualBox:~$ sudo apt-get update

# The OpenJDK project is the default version of Java
# that is provided from a supported Ubuntu repository.
tom@VirtualBox:~$ sudo apt-get install default-jdk

tom@VirtualBox:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Figure 13: Installing Java

3.2 Adding a Dedicated Hadoop User

```
tom@VirtualBox:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1002) ...
Done.

tom@VirtualBox:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] Y
```

Figure 14: Adding a dedicated Hadoop user.

3.3 Installing SSH

```
tom@VirtualBox:~$ sudo apt-get install ssh

tom@VirtualBox:~$ which ssh
/usr/bin/ssh

tom@VirtualBox:~$ which sshd
/usr/sbin/sshd
```

Figure 15: Installing SSH.

3.4 Create and Setup SSH Certificates

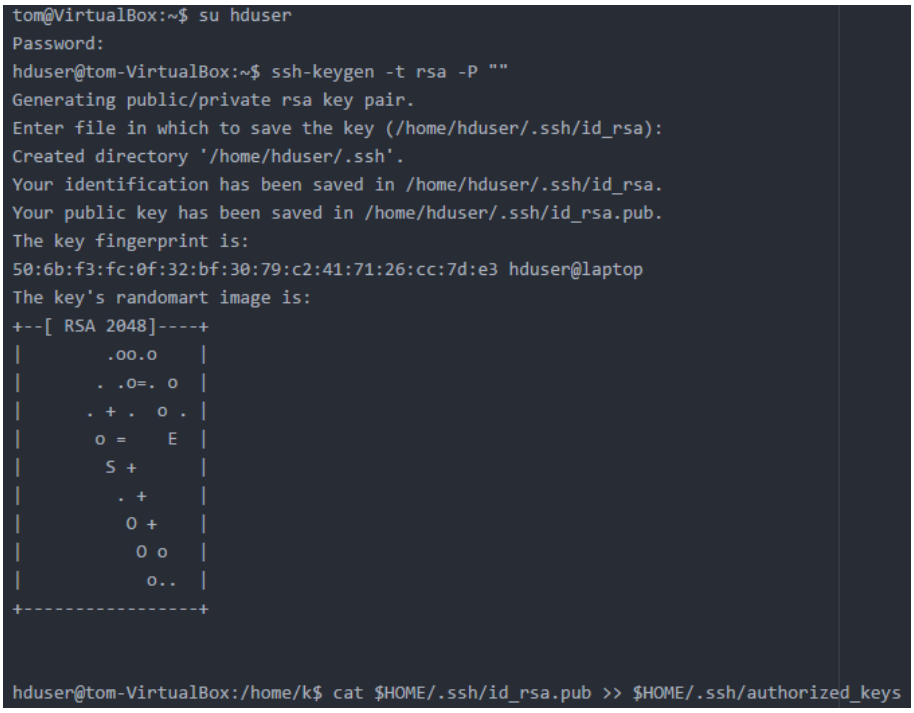

Figure 16: Creating and setting up SSH Certificates.

3.5 Installing Hadoop

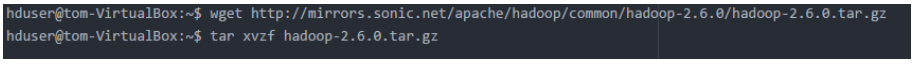

Figure 17: Installing Hadoop.

```
hduser@tom-VirtualBox:~/hadoop-2.6.0$ su tom
Password:

tom@VirtualBox:/home/hduser$ sudo adduser hduser sudo
[sudo] password for k:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Figure 18: Logging in as root user then adding hduser to sudo.

The hduser now has root privilege. We can now move the Hadoop installation to the /usr/local/hadoop directory.

```
tom@VirtualBox:/home/hduser$ sudo su hduser

hduser@tom-VirtualBox:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
hduser@tom-VirtualBox:~/hadoop-2.6.0$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

Figure 19: Moving Hadoop installation.

4 Configuring Hadoop Files

4.1 /.bashrc

I needed to set some hadoop variables inside the /.bashrc file. Such as the hadoop install path and Java's home.

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

Figure 20: Appended lines

4.2 /usr/local/hadoop/etc/hadoop/hadoop-env.sh

```
hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Figure 21: Setting JAVA_HOME by modifying hadoop-env.sh

4.3 /usr/local/hadoop/etc/hadoop/core-site.xml:

```
hduser@laptop:~$ sudo mkdir -p /app/hadoop/tmp
hduser@laptop:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

```
hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

Figure 22: Changing Hadoop config that it uses when starting up.

4.4 /usr/local/hadoop/etc/hadoop/mapred-site.xml

The mapred-site.xml file is used to specify which framework is being used for MapReduce. I needed to enter Figure 23's content in between the ¡configuration¿¡/configuration¿ tag

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

Figure 23: Changing mapred-site.xml.

4.5 /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The /usr/local/hadoop/etc/hadoop/hdfs-site.xml file needs to be configured for each host in the cluster that is being used. It's used to specify the di-

rectories which will be used as the namenode and the datanode on that host. I needed to enter Figure 24's content in between the ¡configuration¿¡/configuration¿ tag

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

```
hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml

<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Figure 24: Changing hdfs-site.xml.

5 Formatting the Hadoop Filesystem

The Hadoop file system needs to be formatted so that I could start using it. The format command was issued with write permission since it creates current directory under /usr/local/hadoop_store/hdfs/namenode folder. This command was executed before using Hadoop.

```
hduser@laptop:~$ hadoop namenode -format
```

Figure 25: Format Hadoop command.

6 Starting Hadoop

Figure 26: Netstat command.

:::8088 in this cluster is the host node. The other nodes are all slaves. Since :::8088 is the host of the node cluster, once Hadoop is started the web interface can be accessed at localhost:8088. This web interface details the current state of the Hadoop Distributed File System. You can view current working jobs that the file system is running, and manage the nodes individually. This interface is shown below in figure 2. My Node Cluster has a total of 8GB memory.

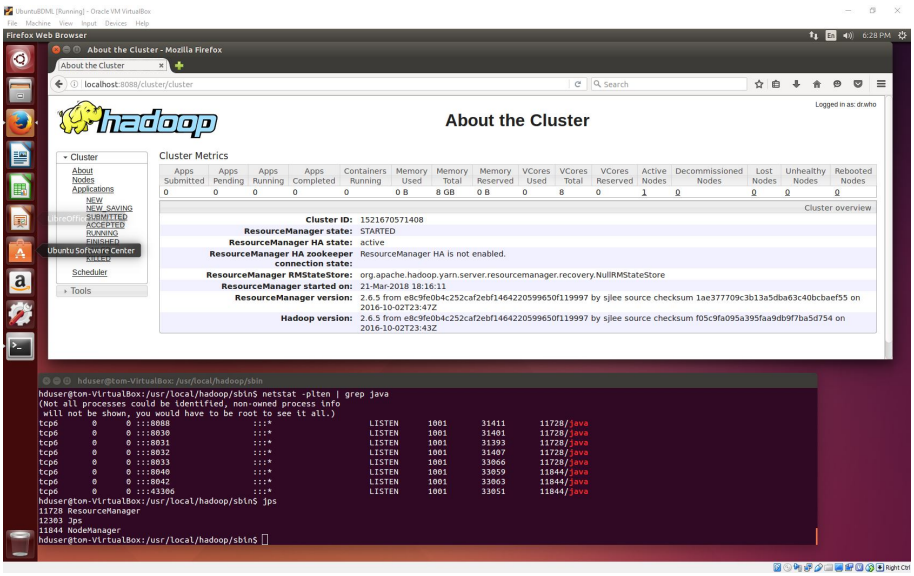

Figure 27: Hadoop web interface.

7 Machine Learning

7.1 Introduction

The last step in this class is to introduce a machine learning algorithm and use the generated data on it. This machine learning algorithm will be fed with my two datasets of four images each and test it's Out-Of-Bucket (OOB) estimation to determine how well the machine has learned these images.

7.2 Choosing a Machine Learning Model

With a variety of machine learning models to choose from. In this assignment we will pursue a supervised learning model that uses inferences from the dataset to predict future knowledge. Choosing a supervised learning models means that the data is labeled, and the desired output is to match those labels. We will also be using decision trees and random forest models for our supervised learning techniques. Random forest is a good choice for my datasets because my data contains several classes.

7.3 Labeling and Shuffling

Recall Figure's 1 through 4, two pictures of Kitty and two pictures of Luna. We expressed these images as data in csv files, an 8x8 grid of pixels is an observation, and every pixel in that grid is a feature. One image can be defined as a 256x256 image converted into 1024 observations containing 64 features. This data was labeled by me, a new 65th column was added to each observation. These labels were expressed in the numbers one to four, one and two represented the two pictures of Luna, three and four represented the two pictures of Kitty. The same method was then applied to the four different yogurt lids. Since we're not comparing cats to yogurt lids, the yogurt labels were also one through four. Now the data needs to be consolidated into one data file instead of four different files. The data also needs to be shuffled, to confuse the machine learning algorithm during testing phase. These were done in MatLab with a very simple script shown in figure 15.

```
a = csvread('berry.csv');
b = csvread('keylime.csv');
c = csvread('orange.csv');
d = csvread('lemon.csv');

appendAll = [a;b;c;d;];

A = appendAll(randperm(size(appendAll,1)),:);

csvwrite('shuffledYogurt.csv', A);
```

Figure 28: MatLab Script.

7.4 Cat Comparisons: Kitty or Luna?

The data has been labelled, appended, and shuffled at this point. Now we can start feeding in these two datasets into our machine learning algorithm. First we'll take a look at the machine's evaluation of the data in decision tree form. In this next figure, figure 16, the diagram will look like an upside-down tree. At the root of this tree is a feature that contributed the most in determining the location of the decision tree's split.

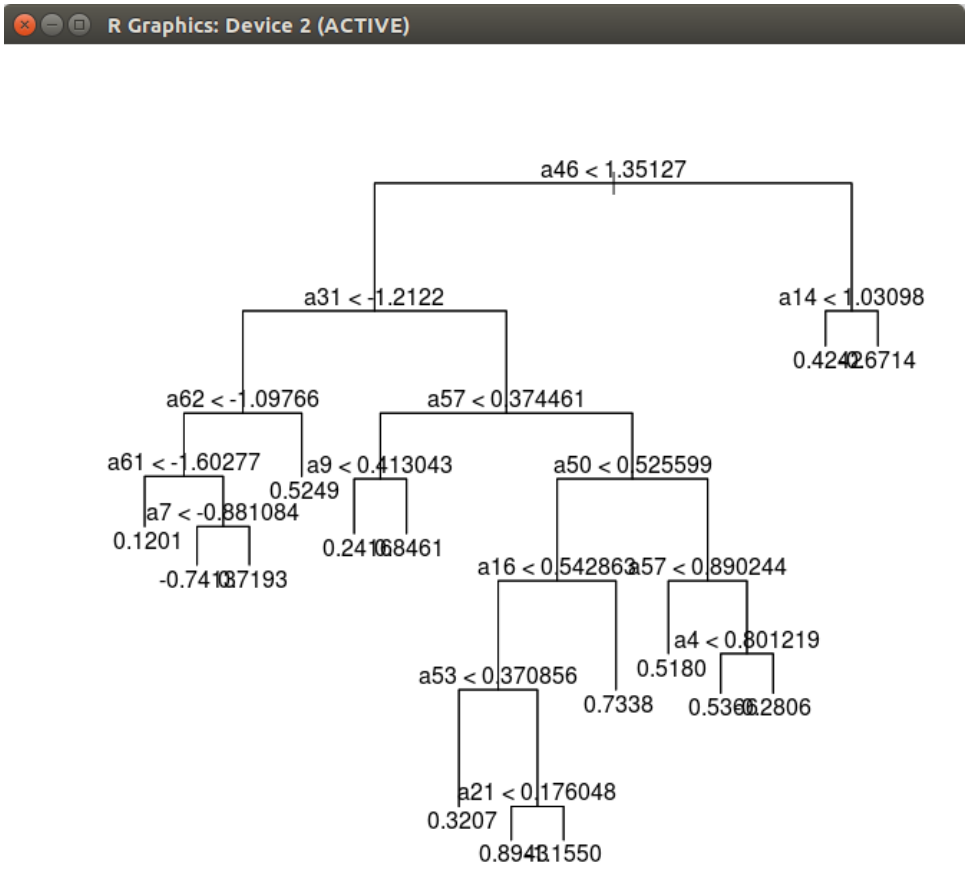

Figure 29: Decision tree for the cat dataset Script.

We can see that feature 46 contributed the most to the location of the split. Any values in the data that are less than 1.35127 are split to the left, else it's split to the right. So on and so forth until the algorithm stops splitting and we're left with the leaves. The next five figures are results of the decision tree represented in other forms.

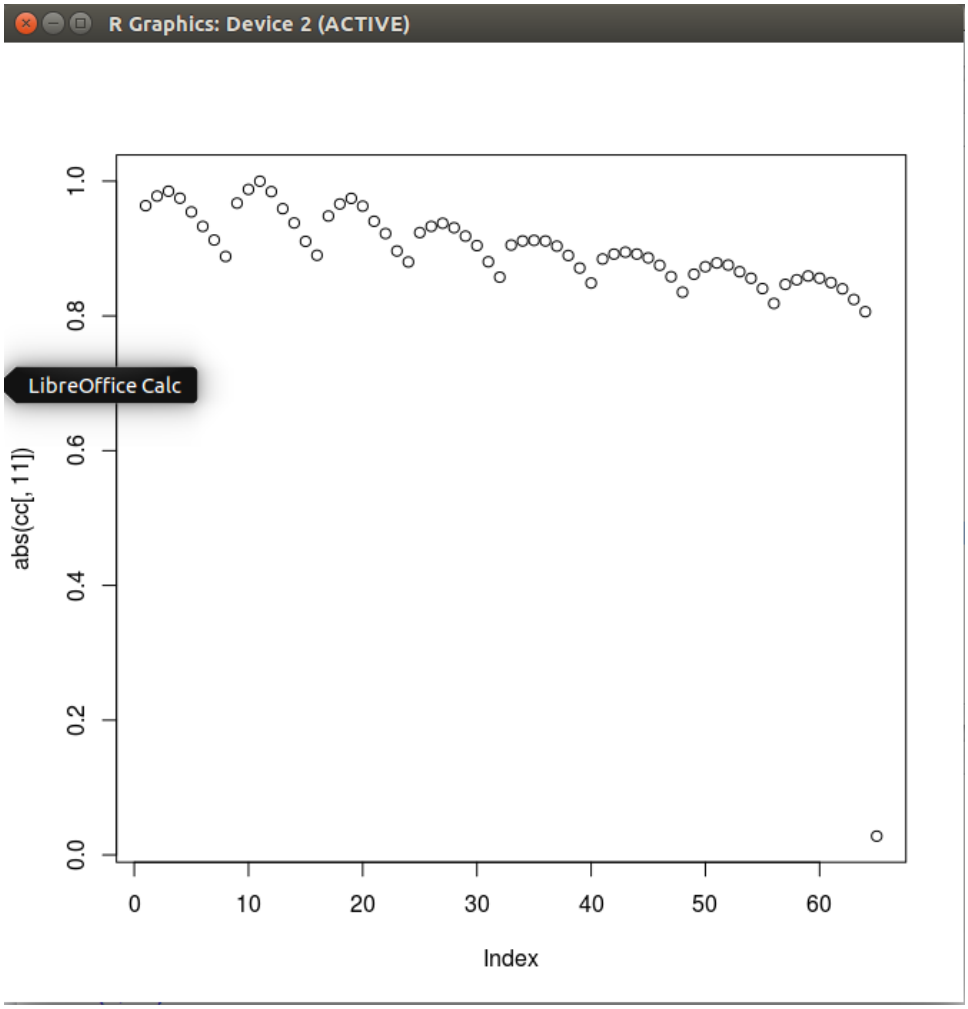

Figure 30: Decision tree for the cat dataset Script.

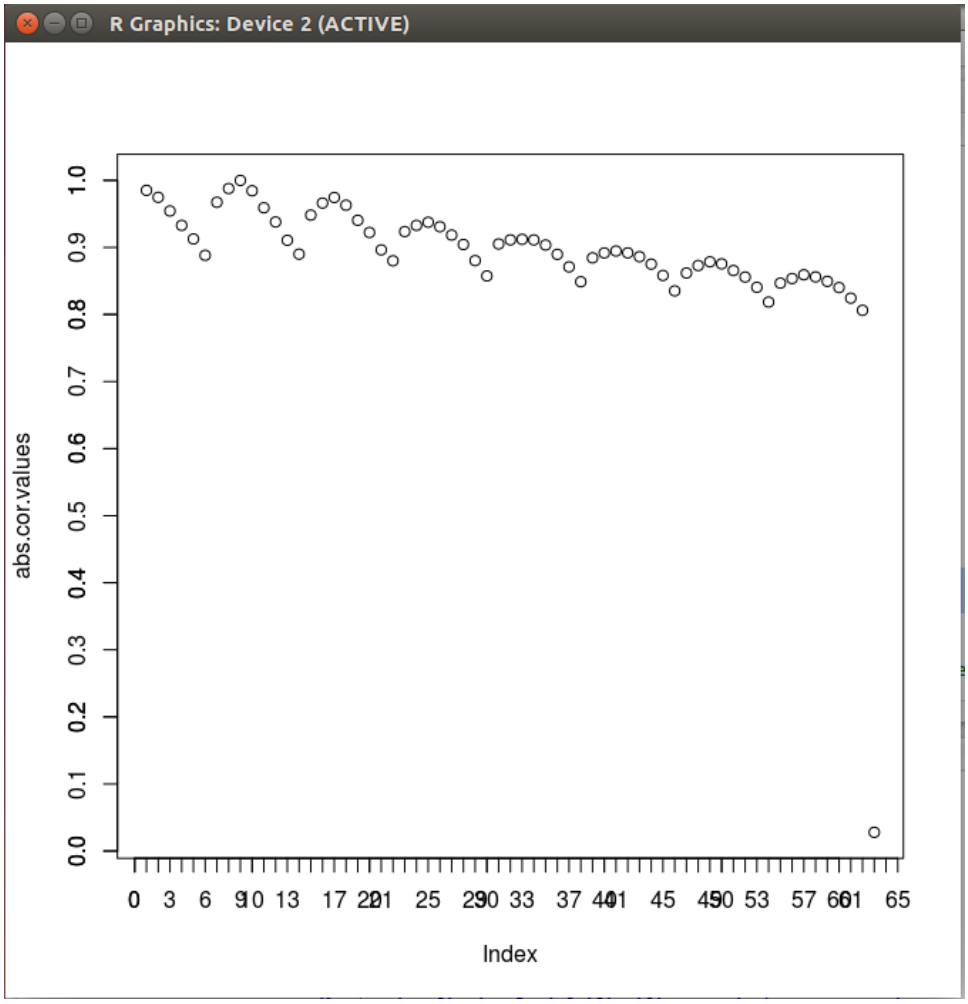

Figure 31: Decision tree for the cat dataset Script.

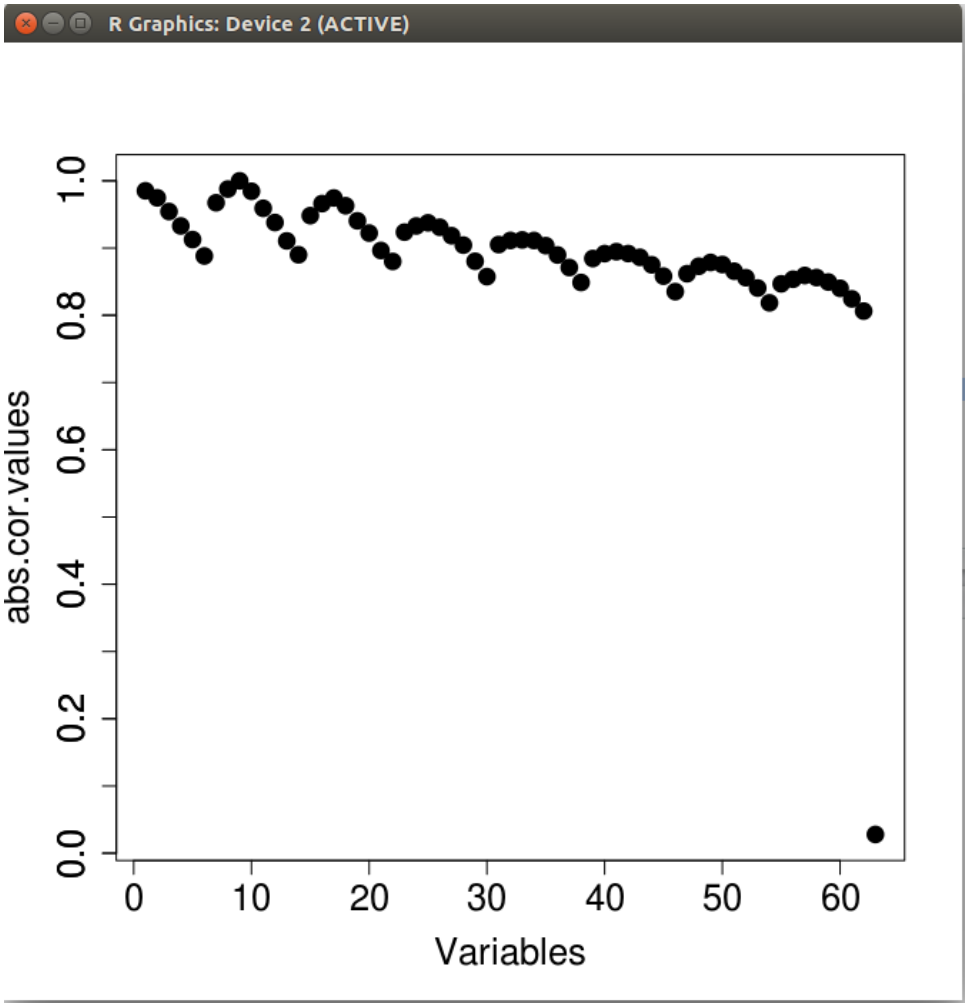

Figure 32: Decision tree for the cat dataset Script.

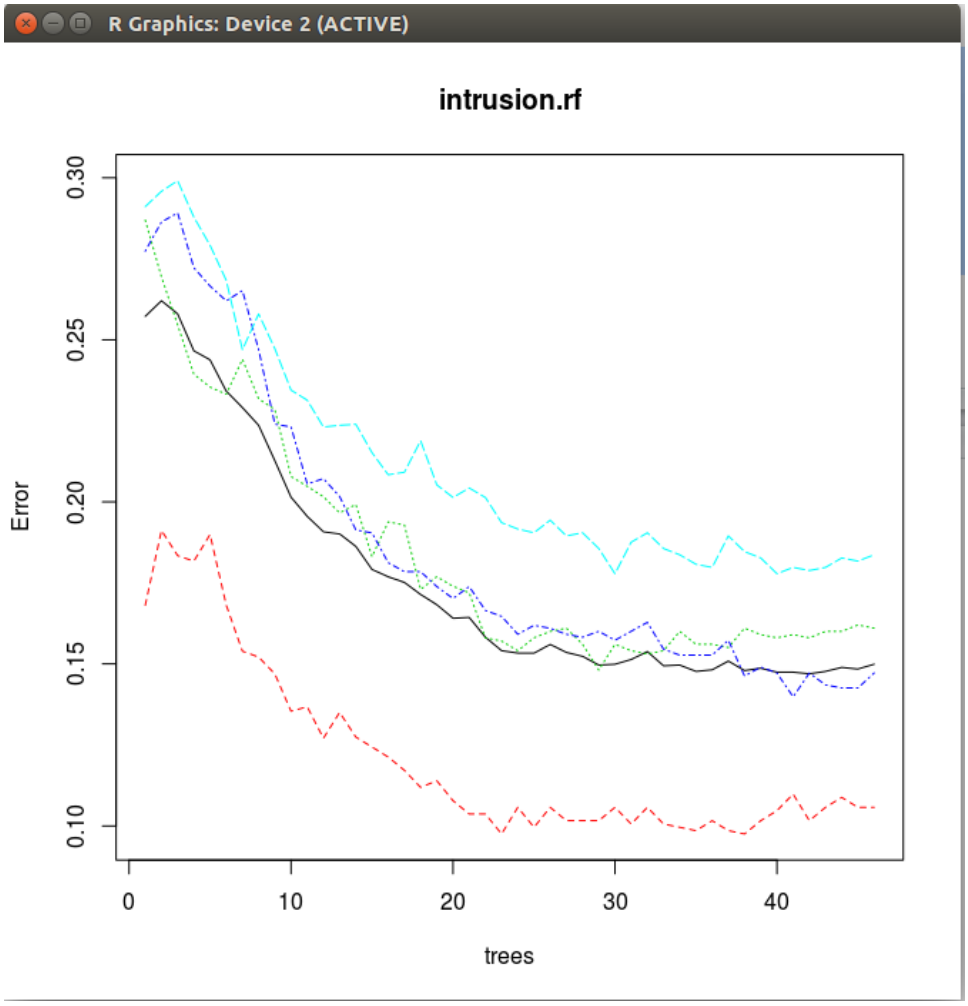

Figure 33: Decision tree for the cat dataset Script.

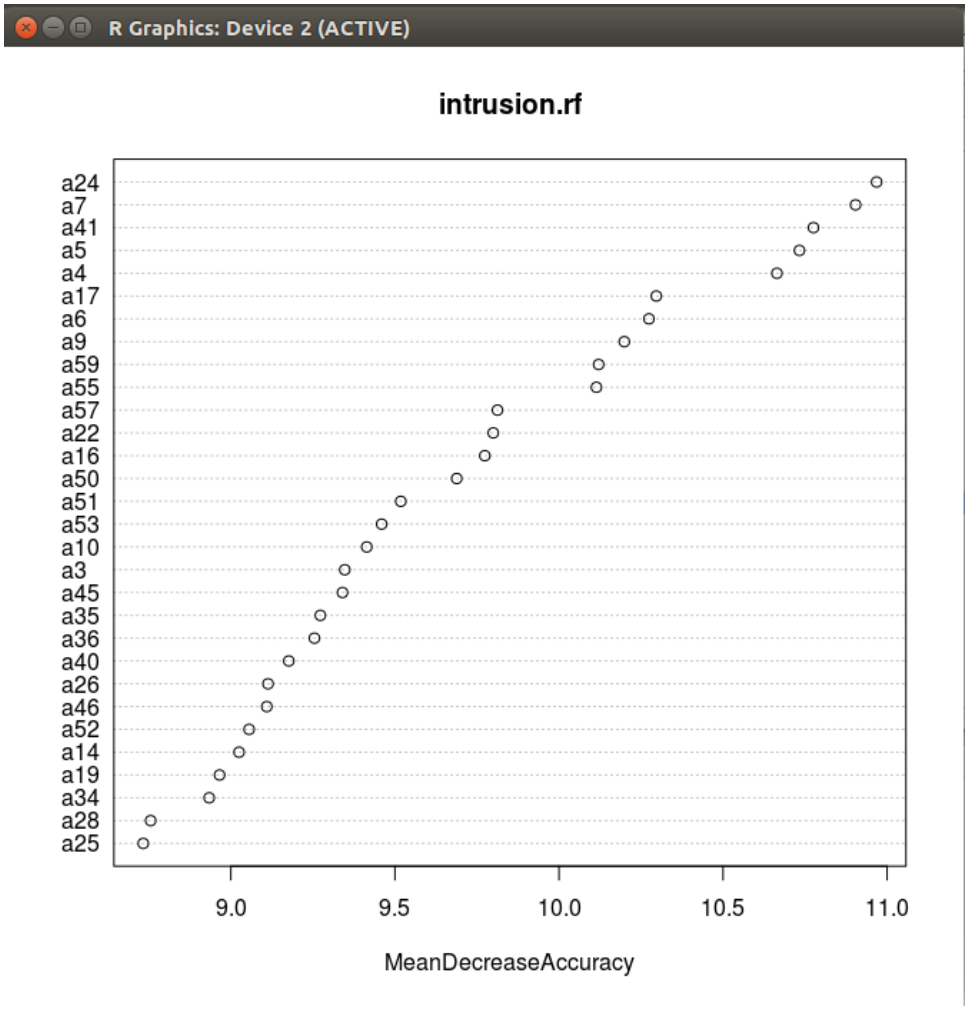

Figure 34: Decision tree for the cat dataset Script.

In the next three figure I'll show the random forest model's results of the cat dataset. Each of these figures have a different amount of trees, one is 250, one is 500, and the last is 1000. The differences between these trees should be substantial, however they're all within 1% OOB estimate range of eachother.

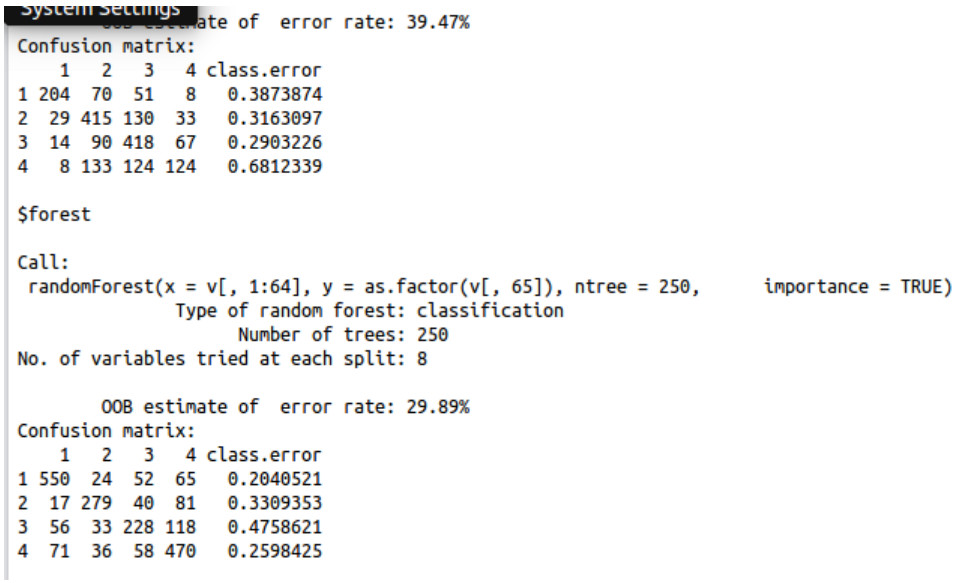

Figure 35: OOB where ntree = 250.

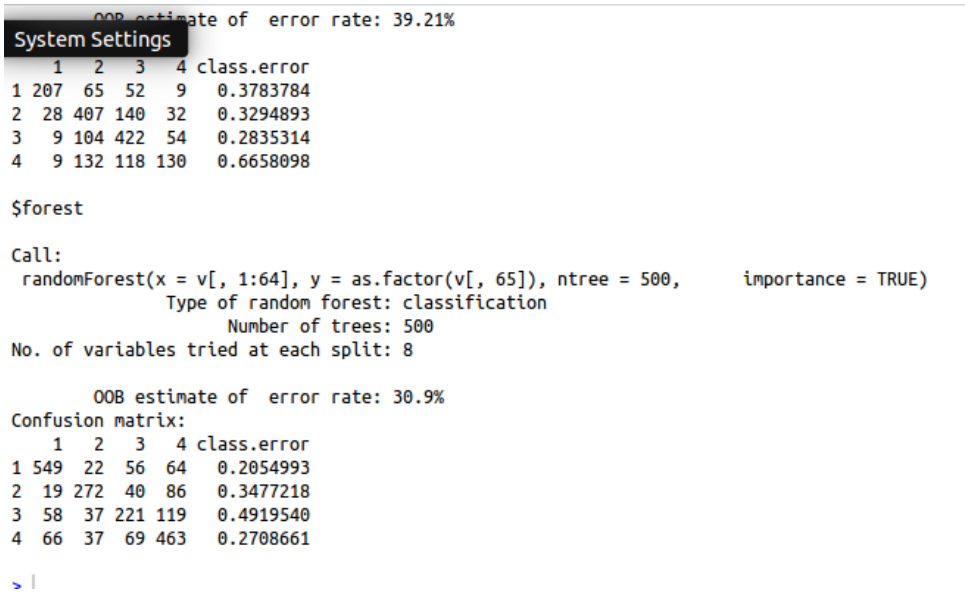

Figure 36: OOB where ntree = 500.

```
    1   2   3   4 class.error
1 205  67  52   9   0.3843844
2  25 424 129  29   0.3014827
3   8 102 424  55   0.2801358
4   6 140 116 127   0.6735219

$forest

Call:
 randomForest(x = v[, 1:64], y = as.factor(v[, 65]), ntree = 1000,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 8

        OOB estimate of  error rate: 30.07%
Confusion matrix:
    1   2   3   4 class.error
1 553  24  52  62   0.1997106
2  18 278  36  85   0.3333333
3  62  31 222 120   0.4896552
4  65  35  65 470   0.2598425

> |
```

Figure 37: OOB where ntree = 1000.

Of the above three random forests 250 trees seems to be the best with a 70.11% in accuracy.

7.5 Which Flavor of Yogurt Should I Eat?

After labelling, appending, and shuffling the data of the 4 different yogurt lids we're ready to pass the data into the random forests and decision trees as we have with the cat dataset.

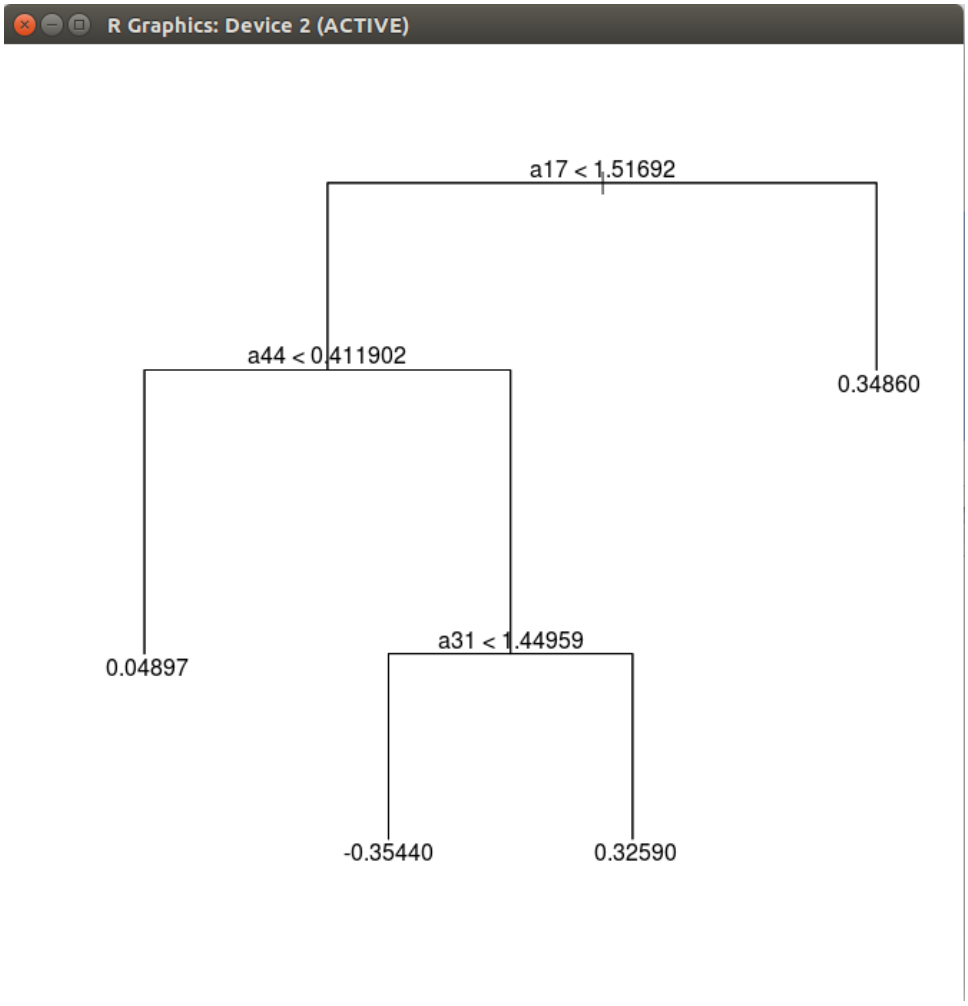

Figure 38: Decision tree for the yogurt dataset Script.

You may be thinking that this decision tree has much less branching than the cat dataset. This can be explained as less levels of decisions. If you can recall figures seven through eleven, the yogurt lids have far less differences between each other than the differences between the four cat images. In this decision tree, feature a17 has the most influence on the location of the splitting of the tree. Again, values less than 1.51692 are branched to the left, while the greater values are branched to the right. The next five figures are results of the decision tree represented in other forms.

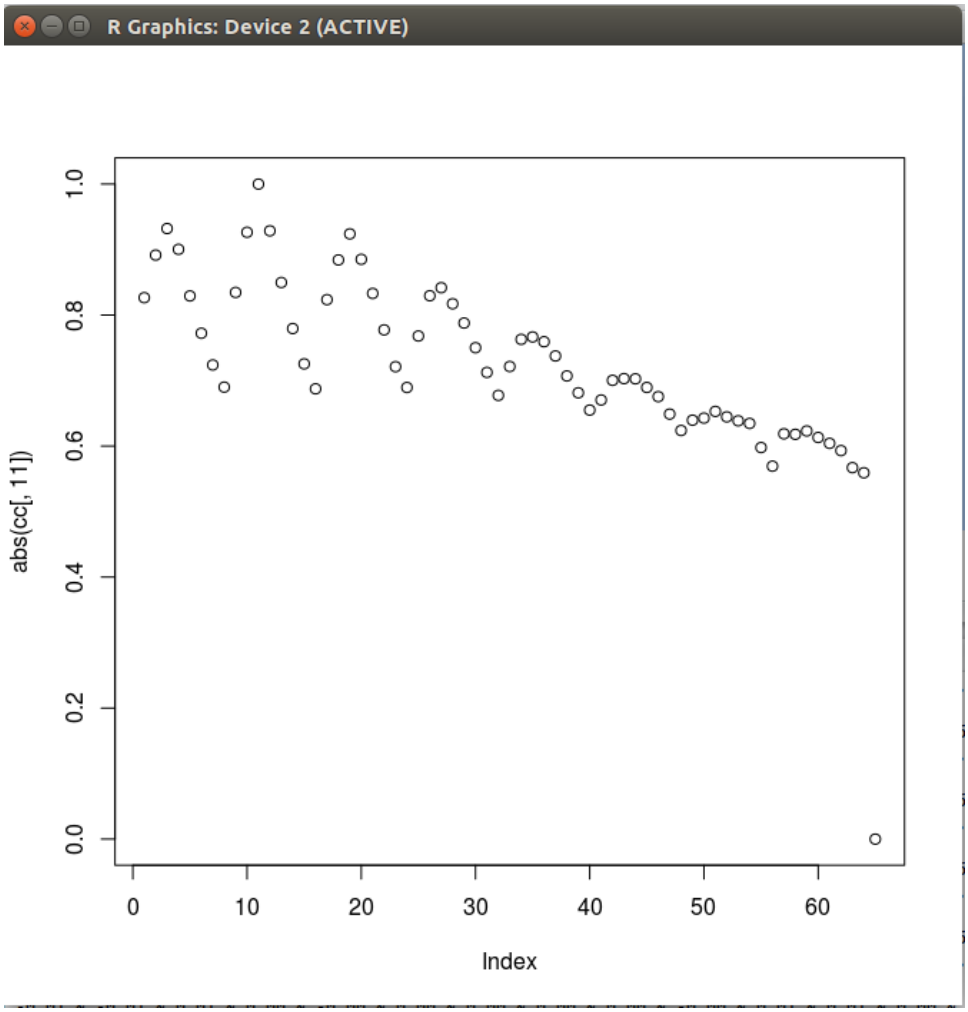

Figure 39: Decision tree for the yogurt dataset Script.

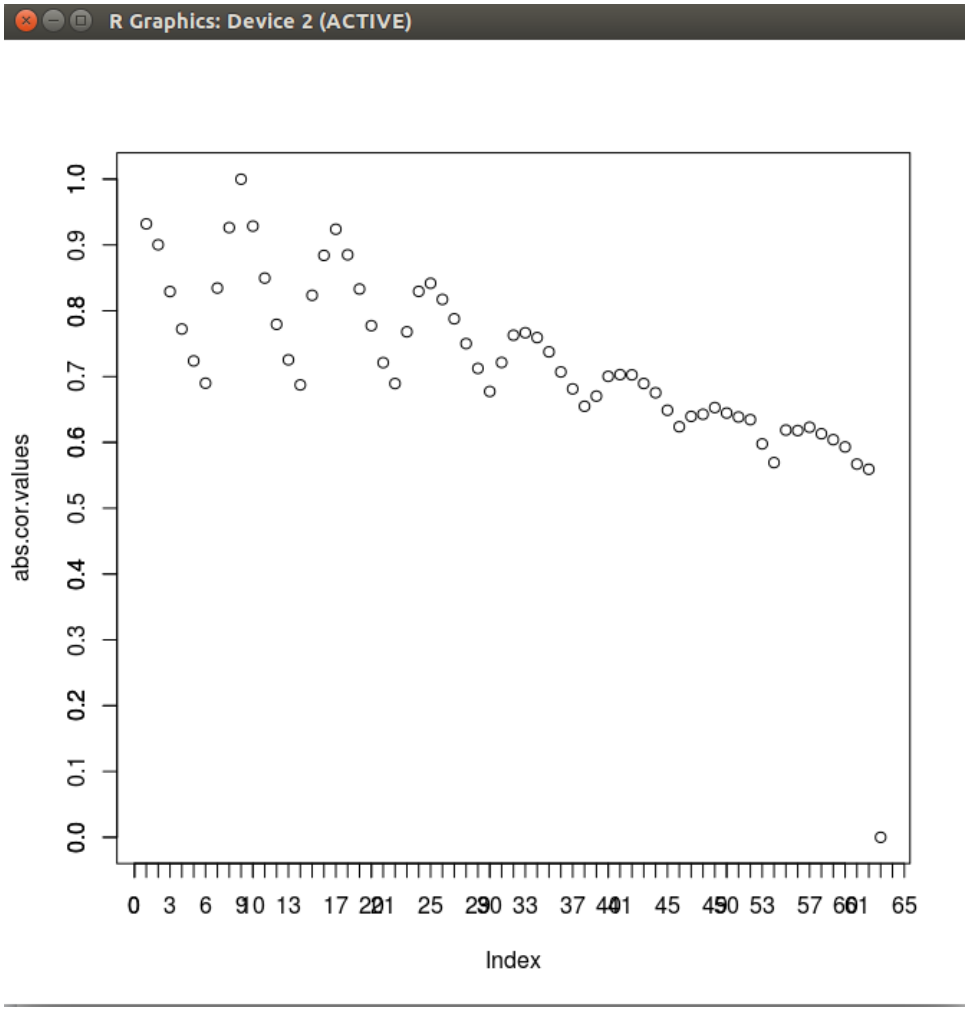

Figure 40: Decision tree for the yogurt dataset Script.

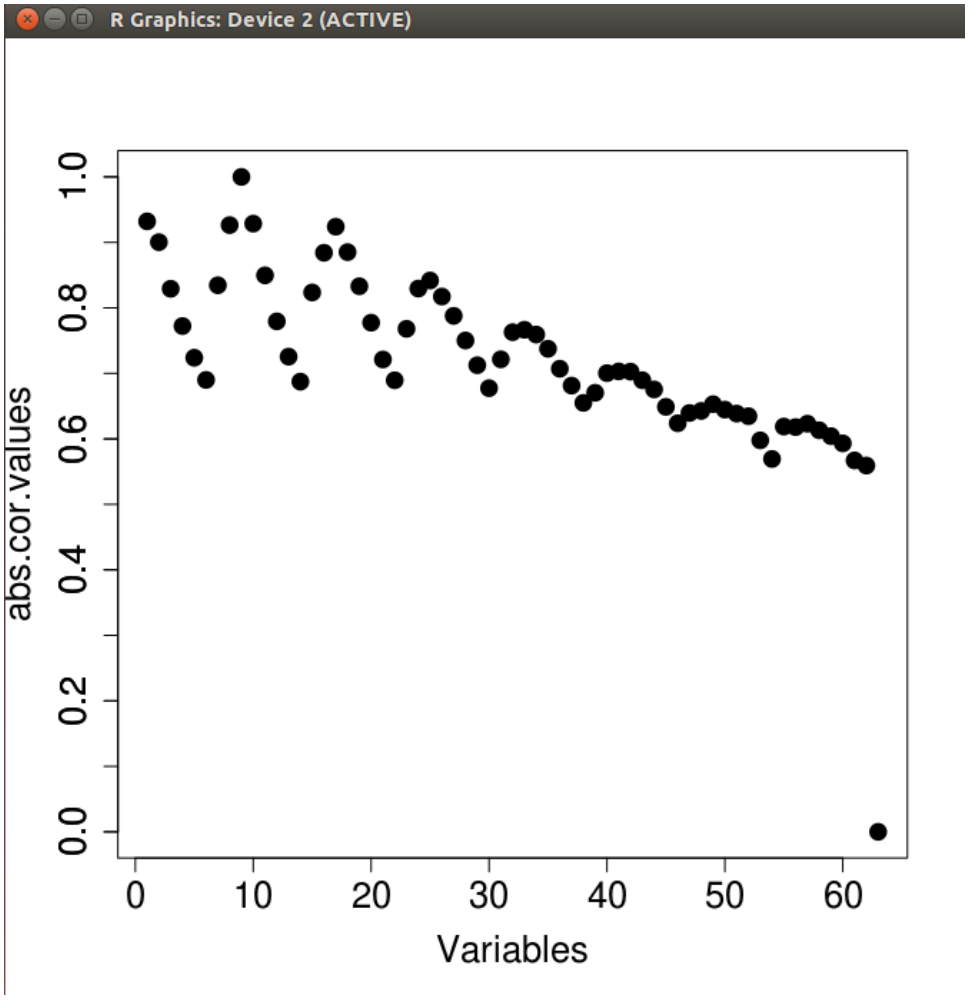

Figure 41: Decision tree for the yogurt dataset Script.

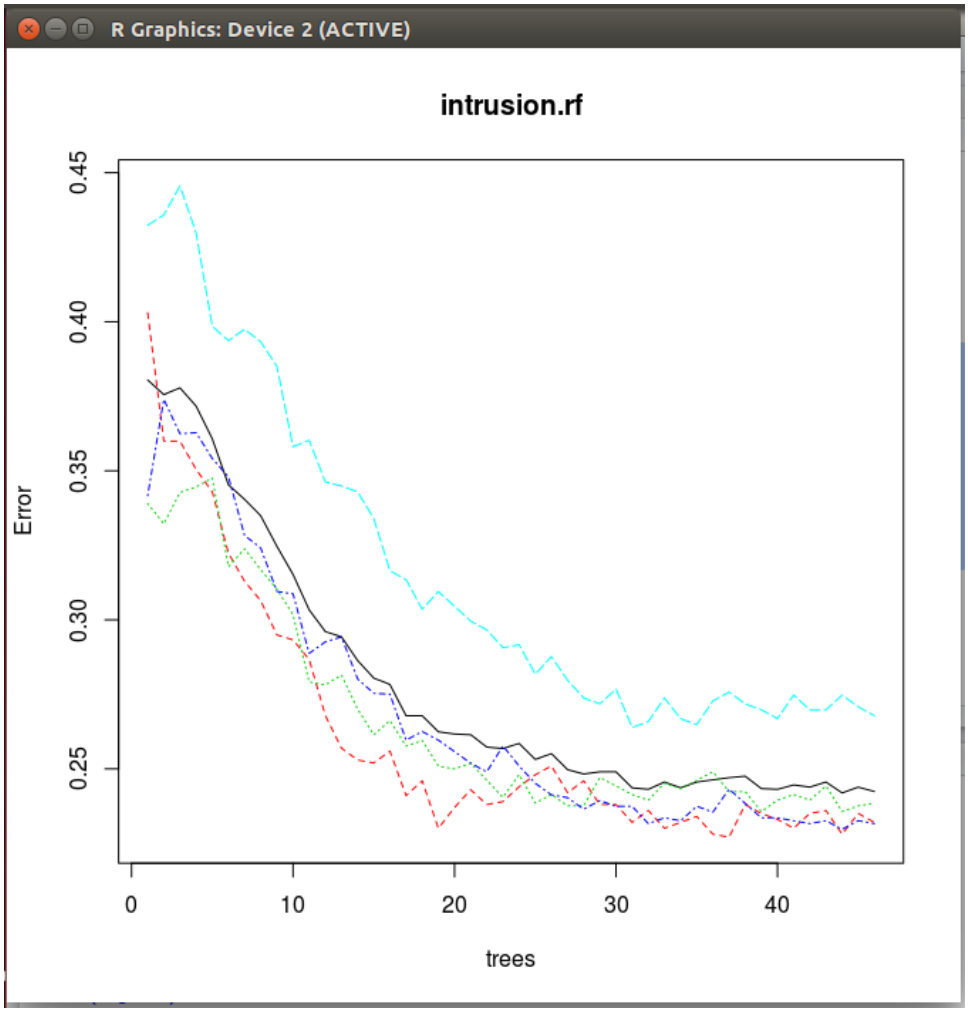

Figure 42: Decision tree for the yogurt dataset Script.

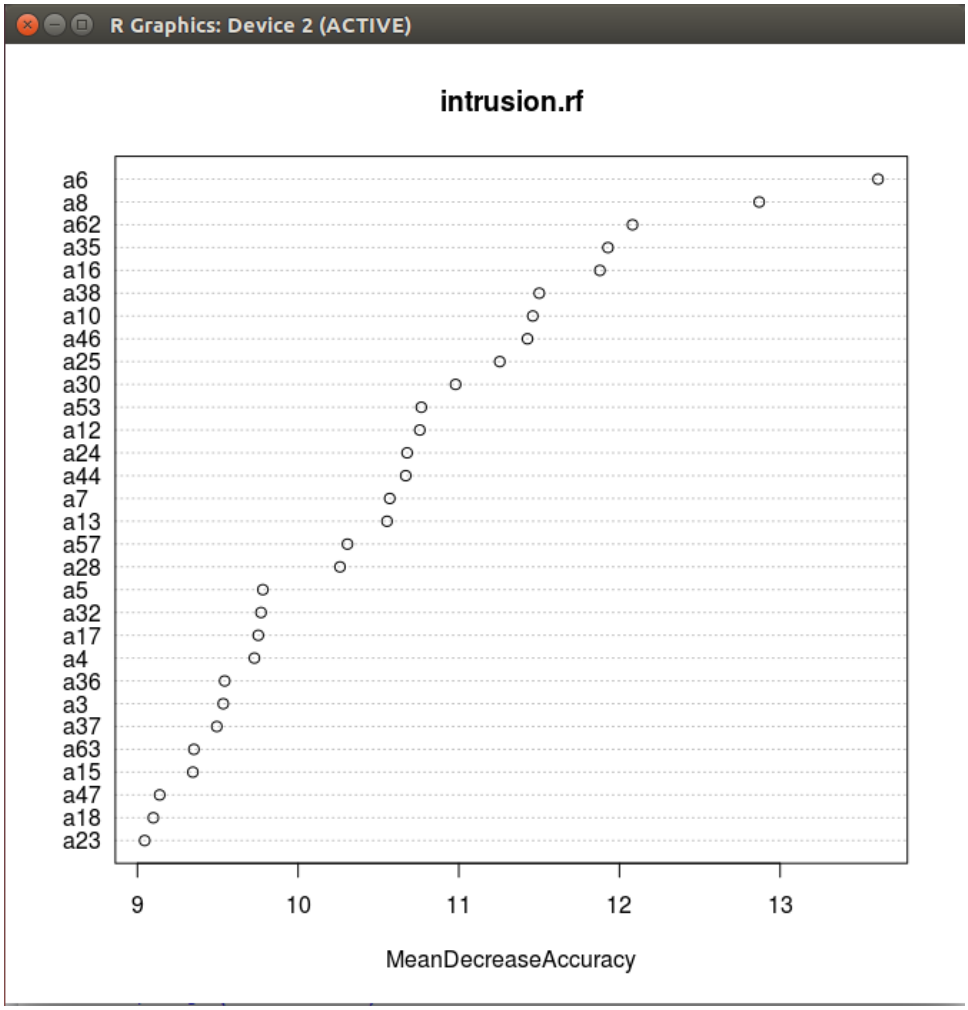

Figure 43: Decision tree for the yogurt dataset Script.

In the next three figure we will take a look at the random forest model's results of the yogurt dataset. Each of these figures have a different amount of trees, one is 250, one is 500, and the last is 1000. The differences between these trees should be substantial, however they're all within 1% OOB estimate range of eachother.

```
Call:
 randomForest(x = v[, 1:64], y = as.factor(v[, 65]), ntree = 250,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 250
No. of variables tried at each split: 8

        OOB estimate of  error rate: 61.14%
Confusion matrix:
    1   2   3   4 class.error
1 356  96  67  87   0.4125413
2 212 170  76  80   0.6840149
3 150  98 133 115   0.7318548
4 156  84  91 175   0.6541502
```

Figure 44: OOB where ntree = 250.

```
        OOB estimate of  error rate: 60.31%
Confusion matrix:
    1   2   3   4 class.error
1 106  91 118 103   0.7464115
2  35 191 110 150   0.6069959
3  41  84 273 130   0.4829545
4  24 141 149 204   0.6061776

$forest

Call:
 randomForest(x = v[, 1:64], y = as.factor(v[, 65]), ntree = 500,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 8

        OOB estimate of  error rate: 60.76%
Confusion matrix:
    1   2   3   4 class.error
1 368  89  60  89   0.3927393
2 228 162  72  76   0.6988848
3 160  87 143 106   0.7116935
4 161  77  99 169   0.6660079

> |
```

Figure 45: OOB where ntree = 500.

```
        OOB estimate of  error rate: 61.85%
Confusion matrix:
    1   2   3   4 class.error
1 100  89 119 110   0.7607656
2  36 189 114 147   0.6111111
3  45  78 262 143   0.5037879
4  24 162 139 193   0.6274131

$forest

Call:
 randomForest(x = v[, 1:64], y = as.factor(v[, 65]), ntree = 1000,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 8

        OOB estimate of  error rate: 60.53%
Confusion matrix:
    1   2   3   4 class.error
1 373  99  51  83   0.3844884
2 220 166  67  85   0.6914498
3 167  90 136 103   0.7258065
4 173  78  83 172   0.6600791

> |
```

Figure 46: OOB where ntree = 1000.

In the cat dataset the number of trees that produces the highest accuracy result was when ntree = 250. Here we can see that the highest accuracy given is when ntree = 1000. Our best accuracy is a whomping 60.53%, a very undesirable score for this random forest algorithm. However, I can point out a few reasons why this machine is having trouble. The images were first reduced from full color images to grayscale images, then that image was stored as data. When these images are all grayscale, it's very hard for even a human to distinguish them. The gray colors look very similar, and even the shapes of the fruits at the bottom of the yogurt lid are nearly identical to each other. A higher number of trees does us more benefit so the machine can scrutinize every feature further.

7.6 Conclusion

Given a great number of levels of decisions, a random forest algorithm can have more success in a lower number of trees. If there are less levels of decision, a higher number of trees can help examine the feature more closely. However, when the observations are nearly identical there will still be a large margin of error. We found that with distinguishable enough data it is possible

for this random forest based machine to differentiate between my cats.

References

[1] K. Hong, *BigData Single Node Cluster Hadoop Installation on Ubuntu*, http:www.bogotobogo.comHadoopBigData_hadoop_Install_on_ubuntu_single_node_cluster.php