Supervised Learning Methods

Srikant Vasudevan

11/12/2019

ABSTRACT

This report will use a dataset from the UCI Machine Learning Repository titled "MAGIC Gamma Telescope Data Set". This dataset contains several attributes for telescope images and a binary class attribute (class) which serves as a distinction between being signal gamma detection or background hadron particle detection. Within this report, several different supervised learning models will be utilized and assessed to their effectiveness. There is little scientific significance in this report, as it is primarily examining supervised learning techniques, rather than analyzing specific scientific data.

INTRODUCTION

Supervised learning analyses consist of many different models. One widely used model that can provide seamless statistical visualization is called a decision tree model. The decision tree is a support tool that uses probabilistic decisions, and their possible consequences to map out chances and outcomes in a tree-like format. These trees are useful in visualizing probabilistic data using statistical significance and p-values (these will be discussed later). These trees are very versatile methods to visualize certain data.

Support Vector Machine (svm or ksvm) is one of many supervised learning models used for classification and regression analysis. The svm method of classification uses a training set to record the assignment of data to either of two categories, this method then builds a classification model to use on a testing set (a binary linear classification method). SVMs can also create non-linear classifications by using the kernel trick, which maps their inputs into large multi-dimensional spaces (models). SVMs (and any supervised learning model for that matter) require labelled data, that is, data that is distinctly identified to be of a certain attribute or variable.

Another widely used supervised learning model is called a neural network. Put simply, a neural network is a set of algorithms, that are resemblant of the human brain, that interpret "sensory" data through a given classification method. Neural networks, aside from being an effective method for classification and clustering, can also maintain features that would otherwise be lost within analysis (when using other supervised learning algorithms). Many consider the neural network machine learning method to be the most effective.

In addition to the aforementioned methods of supervised machine learning, many use the random forest method (random decision forest, consisting of several different decision trees). The key to a random forest model is low correlation between individual tree models, this would produce ensemble predictions that equate to a more accurate prediction than the sum of each individual random tree prediction. Random forest models can serve as rich collections of several smaller tree models that diversify and strengthen classification predictions.

Naive Bayesian Analysis consists of a family of "probabilistic classifiers" that use a theorem referred to as "Bayes' theorem" to create independence assumptions between variables within the data. Naive Bayesian models work in extensive networks but can also be used in a simplified manner.

To calculate statistical significance within the dataset, an algorithm called the Gini index is used. The higher the value of the gini index for an attribute, there is a greater indication of inequality within that variable, therefore making it less likely to contribute to classification with statistical significance.

Tidyverse and GridExtra are both packages that allow for advanced and seamless visualizations of data.Rpart, party, randomForest, nnet, kernlab and e1071 are all packages that contain pre-developed functions for each supervised learning algorithm that will be used in this analysis.

METHODS

1. The data that was used in this analysis was downloaded from the UCI Machine Learning repository in the form of a .data file. The .data file was converted into a .csv file and headers were added through R code
2. To analyze the .csv file, R (an open source statistical programming language) and RStudio (a free IDE for the R language) are used, these tools read the contents of the .csv file and format them into a data frame.

3. Within R, we will be using a process called “data munging”, this process is used to clean and format the data to be free of errors and easily understood. This process includes replacing or eliminating missing variables, renaming variable headers to either be more understandable or more concise, and logically organizing the data so it is in a sensical order, prepped for analysis. The dataset used for this classifier has minimal “dirty data” and is a fairly clean dataset, therefore this step will be less extensive.
4. Additionally, a multitude of R packages will be used in the analysis process.
5. To analyze our data, multitudes of visual and numerical representations are used. The majority of the analysis is through several supervised learning models, which allow us to assess the significance of attributes and the accuracy of said models

RESULTS

To set up our dataset for our supervised learning methods, we need to clear the workspace, turn warnings off (be careful when doing this), set our working directory, source any needed files and libraries and read the csv (turning headers off so we can add them in later).

```
rm(list = ls())
options(warn=-1)

setwd("C:/Users/Srikant/Desktop/Data Science/Week 11")
source("./gini.R")

library(tidyverse)
```

```
## -- Attaching packages ------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(rpart)
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

##
## Attaching package: 'strucchange'

## The following object is masked from 'package:stringr':
##
##     boundary
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(nnet)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:modeltools':
##
##     prior

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(e1071)

data <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/magic/magic04.data"), header=FALS
```

The names of the headers of the data are extrapolated from the documentation of the dataset (UCI Machine Learning Repository). First, we look at the summary and dimensions of the data to make sure that we are entering the right data and the right amount of data. Next, we will enter the names and view the summary to make sure the names correspond with the correct columns in the data table. Finally, for the "class" variable, we need to make sure it is a factor variable type for our supervised learning.

```
summary(data)
```

```
##        V1                V2               V3              V4
##  Min.   :  4.284   Min.   :  0.00   Min.   :1.941   Min.   :0.0131
##  1st Qu.: 24.336   1st Qu.: 11.86   1st Qu.:2.477   1st Qu.:0.2358
##  Median : 37.148   Median : 17.14   Median :2.740   Median :0.3542
##  Mean   : 53.250   Mean   : 22.18   Mean   :2.825   Mean   :0.3803
##  3rd Qu.: 70.122   3rd Qu.: 24.74   3rd Qu.:3.102   3rd Qu.:0.5037
##  Max.   :334.177   Max.   :256.38   Max.   :5.323   Max.   :0.8930
##        V5                V6                V7                V8
##  Min.   :0.0003   Min.   :-457.916   Min.   :-331.78   Min.   :-205.8947
##  1st Qu.:0.1285   1st Qu.: -20.587   1st Qu.: -12.84   1st Qu.: -10.8494
##  Median :0.1965   Median :   4.013   Median :  15.31   Median :   0.6662
##  Mean   :0.2147   Mean   :  -4.332   Mean   :  10.55   Mean   :   0.2497
##  3rd Qu.:0.2852   3rd Qu.:  24.064   3rd Qu.:  35.84   3rd Qu.:  10.9464
##  Max.   :0.6752   Max.   : 575.241   Max.   : 238.32   Max.   : 179.8510
##        V9               V10          V11
##  Min.   : 0.000   Min.   :  1.283   g:12332
##  1st Qu.: 5.548   1st Qu.:142.492   h: 6688
##  Median :17.680   Median :191.851
##  Mean   :27.646   Mean   :193.818
##  3rd Qu.:45.884   3rd Qu.:240.564
##  Max.   :90.000   Max.   :495.561
```

```
dim(data)
```

```
## [1] 19020    11
```

```
names(data) <- c("fLength", "fWidth", "fSize",
                 "fConc", "fConc1", "fAsym", "fM3Long", "fM3Trans",
                 "fAlpha", "fDist", "class")
summary(data)
```

```
##     fLength           fWidth           fSize           fConc
##  Min.   :  4.284   Min.   :  0.00   Min.   :1.941   Min.   :0.0131
##  1st Qu.: 24.336   1st Qu.: 11.86   1st Qu.:2.477   1st Qu.:0.2358
##  Median : 37.148   Median : 17.14   Median :2.740   Median :0.3542
##  Mean   : 53.250   Mean   : 22.18   Mean   :2.825   Mean   :0.3803
##  3rd Qu.: 70.122   3rd Qu.: 24.74   3rd Qu.:3.102   3rd Qu.:0.5037
##  Max.   :334.177   Max.   :256.38   Max.   :5.323   Max.   :0.8930
##      fConc1           fAsym           fM3Long           fM3Trans
##  Min.   :0.0003   Min.   :-457.916   Min.   :-331.78   Min.   :-205.8947
##  1st Qu.:0.1285   1st Qu.: -20.587   1st Qu.: -12.84   1st Qu.: -10.8494
##  Median :0.1965   Median :   4.013   Median :  15.31   Median :   0.6662
##  Mean   :0.2147   Mean   :  -4.332   Mean   :  10.55   Mean   :   0.2497
##  3rd Qu.:0.2852   3rd Qu.:  24.064   3rd Qu.:  35.84   3rd Qu.:  10.9464
##  Max.   :0.6752   Max.   : 575.241   Max.   : 238.32   Max.   : 179.8510
##      fAlpha           fDist          class
##  Min.   : 0.000   Min.   :  1.283   g:12332
```

```
##  1st Qu.: 5.548   1st Qu.:142.492   h: 6688
##  Median :17.680   Median :191.851
##  Mean   :27.646   Mean   :193.818
##  3rd Qu.:45.884   3rd Qu.:240.564
##  Max.   :90.000   Max.   :495.561
```

```
data$class <- as.factor(data$class)
```

Descriptive Statistics

For a quick and better understanding of our dataset, we can look at descriptive statistics of our dataset. These various statistics allow us to see sections of our dataset or specific attributes of it (such as size and distribution).

```
summary(data)
```

```
##     fLength            fWidth           fSize           fConc
##  Min.   :  4.284   Min.   :  0.00   Min.   :1.941   Min.   :0.0131
##  1st Qu.: 24.336   1st Qu.: 11.86   1st Qu.:2.477   1st Qu.:0.2358
##  Median : 37.148   Median : 17.14   Median :2.740   Median :0.3542
##  Mean   : 53.250   Mean   : 22.18   Mean   :2.825   Mean   :0.3803
##  3rd Qu.: 70.122   3rd Qu.: 24.74   3rd Qu.:3.102   3rd Qu.:0.5037
##  Max.   :334.177   Max.   :256.38   Max.   :5.323   Max.   :0.8930
##      fConc1           fAsym            fM3Long           fM3Trans
##  Min.   :0.0003   Min.   :-457.916   Min.   :-331.78   Min.   :-205.8947
##  1st Qu.:0.1285   1st Qu.: -20.587   1st Qu.: -12.84   1st Qu.: -10.8494
##  Median :0.1965   Median :   4.013   Median :  15.31   Median :   0.6662
##  Mean   :0.2147   Mean   :  -4.332   Mean   :  10.55   Mean   :   0.2497
##  3rd Qu.:0.2852   3rd Qu.:  24.064   3rd Qu.:  35.84   3rd Qu.:  10.9464
##  Max.   :0.6752   Max.   : 575.241   Max.   : 238.32   Max.   : 179.8510
##      fAlpha           fDist         class
##  Min.   : 0.000   Min.   :  1.283   g:12332
##  1st Qu.: 5.548   1st Qu.:142.492   h: 6688
##  Median :17.680   Median :191.851
##  Mean   :27.646   Mean   :193.818
##  3rd Qu.:45.884   3rd Qu.:240.564
##  Max.   :90.000   Max.   :495.561
```

```
head(data)
```

```
##    fLength   fWidth  fSize  fConc fConc1    fAsym  fM3Long fM3Trans
## 1  28.7967  16.0021 2.6449 0.3918 0.1982  27.7004  22.0110  -8.2027
## 2  31.6036  11.7235 2.5185 0.5303 0.3773  26.2722  23.8238  -9.9574
## 3 162.0520 136.0310 4.0612 0.0374 0.0187 116.7410 -64.8580 -45.2160
## 4  23.8172   9.5728 2.3385 0.6147 0.3922  27.2107  -6.4633  -7.1513
## 5  75.1362  30.9205 3.1611 0.3168 0.1832  -5.5277  28.5525  21.8393
## 6  51.6240  21.1502 2.9085 0.2420 0.1340  50.8761  43.1887   9.8145
##    fAlpha    fDist class
## 1 40.0920  81.8828     g
## 2  6.3609 205.2610     g
## 3 76.9600 256.7880     g
## 4 10.4490 116.7370     g
## 5  4.6480 356.4620     g
## 6  3.6130 238.0980     g
```

```
dim(data)
```

```
## [1] 19020    11
```

```
glimpse(data)
```

```
## Observations: 19,020
## Variables: 11
## $ fLength  <dbl> 28.7967, 31.6036, 162.0520, 23.8172, 75.1362, 51.6240...
## $ fWidth   <dbl> 16.0021, 11.7235, 136.0310, 9.5728, 30.9205, 21.1502,...
## $ fSize    <dbl> 2.6449, 2.5185, 4.0612, 2.3385, 3.1611, 2.9085, 3.033...
## $ fConc    <dbl> 0.3918, 0.5303, 0.0374, 0.6147, 0.3168, 0.2420, 0.252...
## $ fConc1   <dbl> 0.1982, 0.3773, 0.0187, 0.3922, 0.1832, 0.1340, 0.151...
## $ fAsym    <dbl> 27.7004, 26.2722, 116.7410, 27.2107, -5.5277, 50.8761...
## $ fM3Long  <dbl> 22.0110, 23.8238, -64.8580, -6.4633, 28.5525, 43.1887...
## $ fM3Trans <dbl> -8.2027, -9.9574, -45.2160, -7.1513, 21.8393, 9.8145,...
## $ fAlpha   <dbl> 40.0920, 6.3609, 76.9600, 10.4490, 4.6480, 3.6130, 4....
## $ fDist    <dbl> 81.8828, 205.2610, 256.7880, 116.7370, 356.4620, 238....
## $ class    <fct> g, g, g, g, g, g, g, g, g, g, g, g, g, g, g, g, g, g,...
```

This distribution has 11 variables and 19020 observations. The distribution of "class" is 12332 gamma signal data points and 6688 hadron background data points.

Gini Algorithm

The gini algorithm was an algorithm developed to determine which variables in the dataset are the most significant to the determining of the class (whether it is 'g' or 'h').

We are going to use a for loop, with the "gini" value starting at 1 and being changed whenever a lower value comes along. This loop will also tell us the number of the column of the value that produces the lowest gini value.

```
value <- 1
for(i in 1:10){
  ginii <- gini_process(data[,i])
  if(ginii < value){
    value <- ginii
    number <- i
  }
  else{
    value <- value
  }
}

value
```

```
## [1] 0.9996622
```

```
number
```

```
## [1] 3
```

```
names(data)
```

```
##  [1] "fLength"  "fWidth"   "fSize"    "fConc"    "fConc1"   "fAsym"
##  [7] "fM3Long"  "fM3Trans" "fAlpha"   "fDist"    "class"
```

As a result of this for loop, the lowest gini value present in the dataset is .9996622, which is produced by the 3rd column in the dataset, or "fSize".

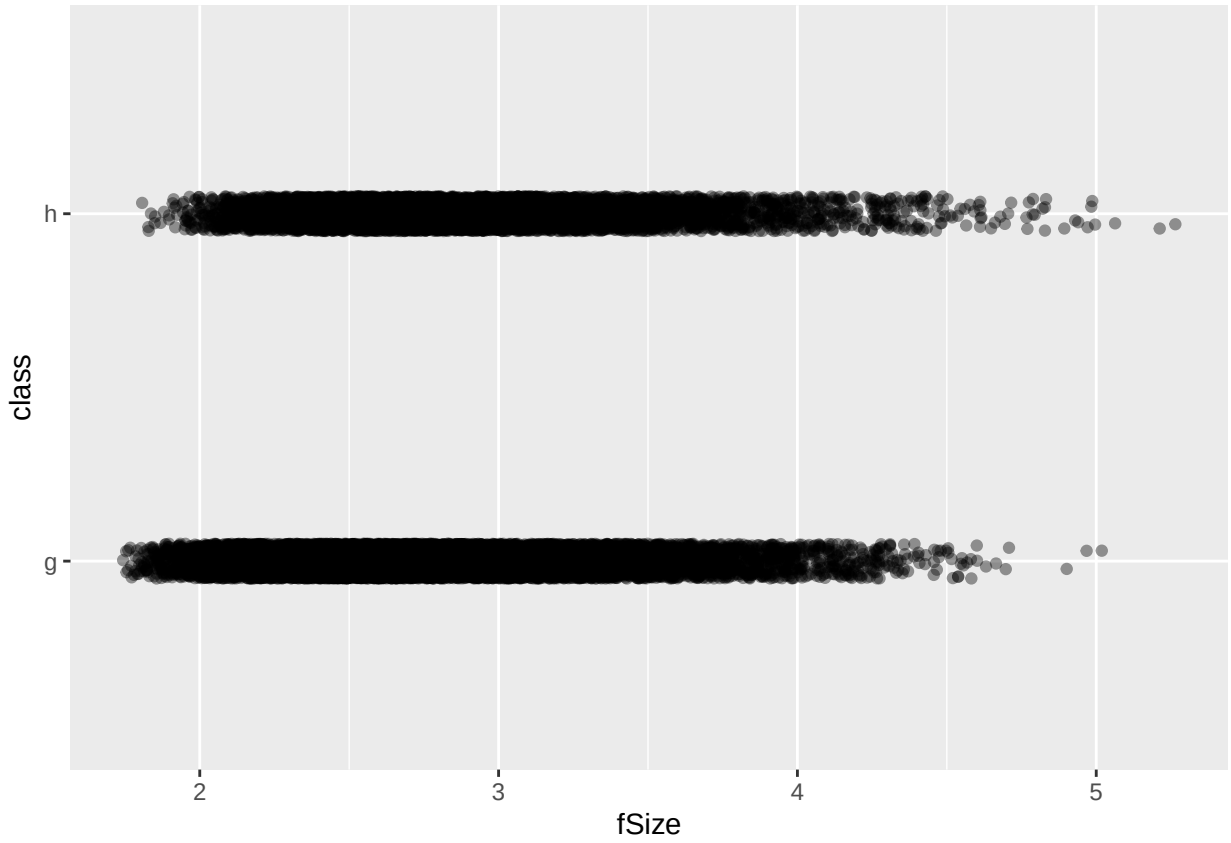
NOTE: This gini value, being the lowest one in the dataset, is very high. This means that classification and supervised learning within the dataset will be largely inaccurate moving forward.

Regression Plots

Below we will produce a regression plot, first without any regression line shown in the plot, and then with both a linear and logistic regression line shown.

The variable being closely examined in these plots will be the the "fSize" because it produced the lowest gini value in our dataset.

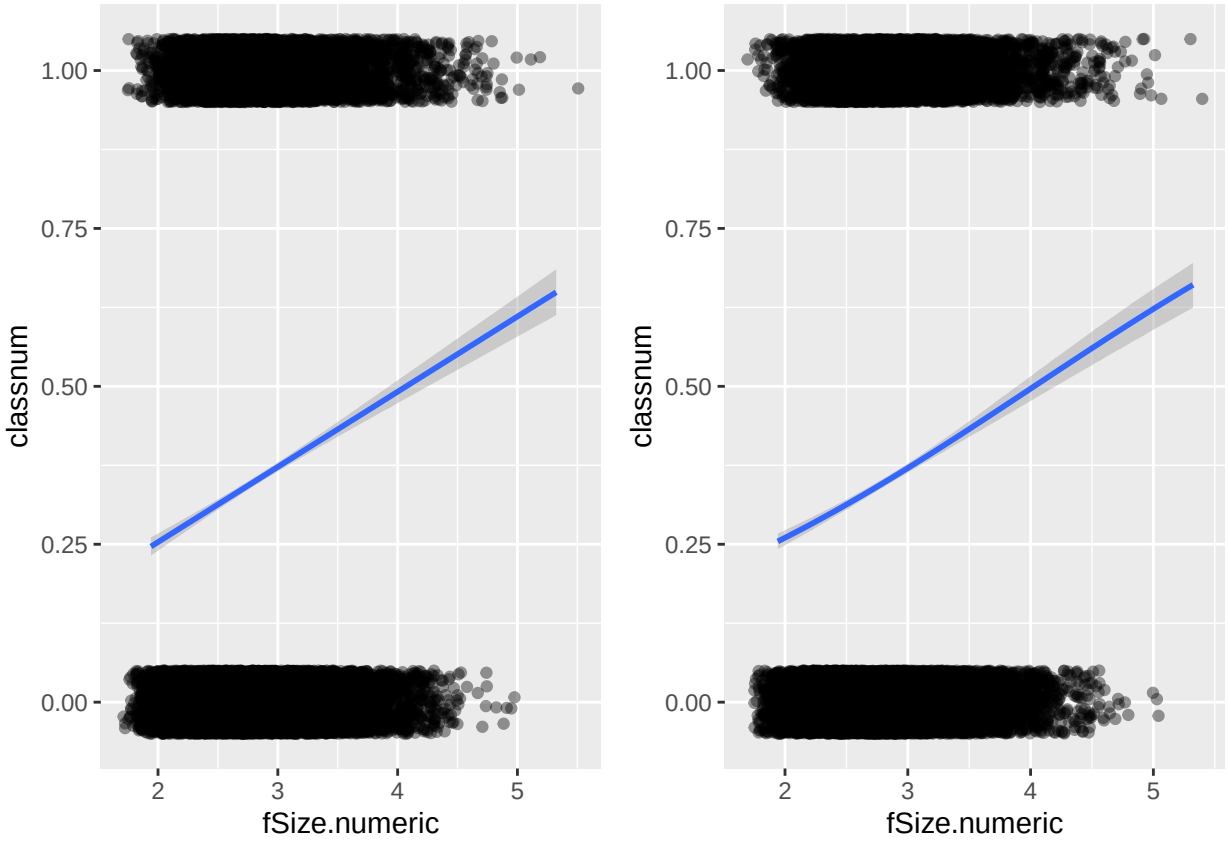
```
data %>%
  ggplot(aes(x=fSize, y=class)) +
  geom_jitter(height=0.05, width=0.3, alpha=0.4)
```




```
linear <- data %>%
  mutate(fSize.numeric = as.numeric(as.character(fSize))) %>%
  mutate(classnum = ifelse(class == "g", 0, 1)) %>%
  ggplot(aes(x= fSize.numeric, y=classnum)) +
  geom_jitter(height=.05, width=.3, alpha=.4) +
  geom_smooth(method = "lm",
              method.args = list(family="binomial"))

logistic <- data %>%
  mutate(fSize.numeric = as.numeric(as.character(fSize))) %>%
  mutate(classnum = ifelse(class == "g", 0, 1)) %>%
  ggplot(aes(x= fSize.numeric, y=classnum)) +
  geom_jitter(height=.05, width=.3, alpha=.4) +
  geom_smooth(method = "glm",
              method.args = list(family="binomial"))


grid.arrange(linear, logistic, nrow=1, ncol=2)
```



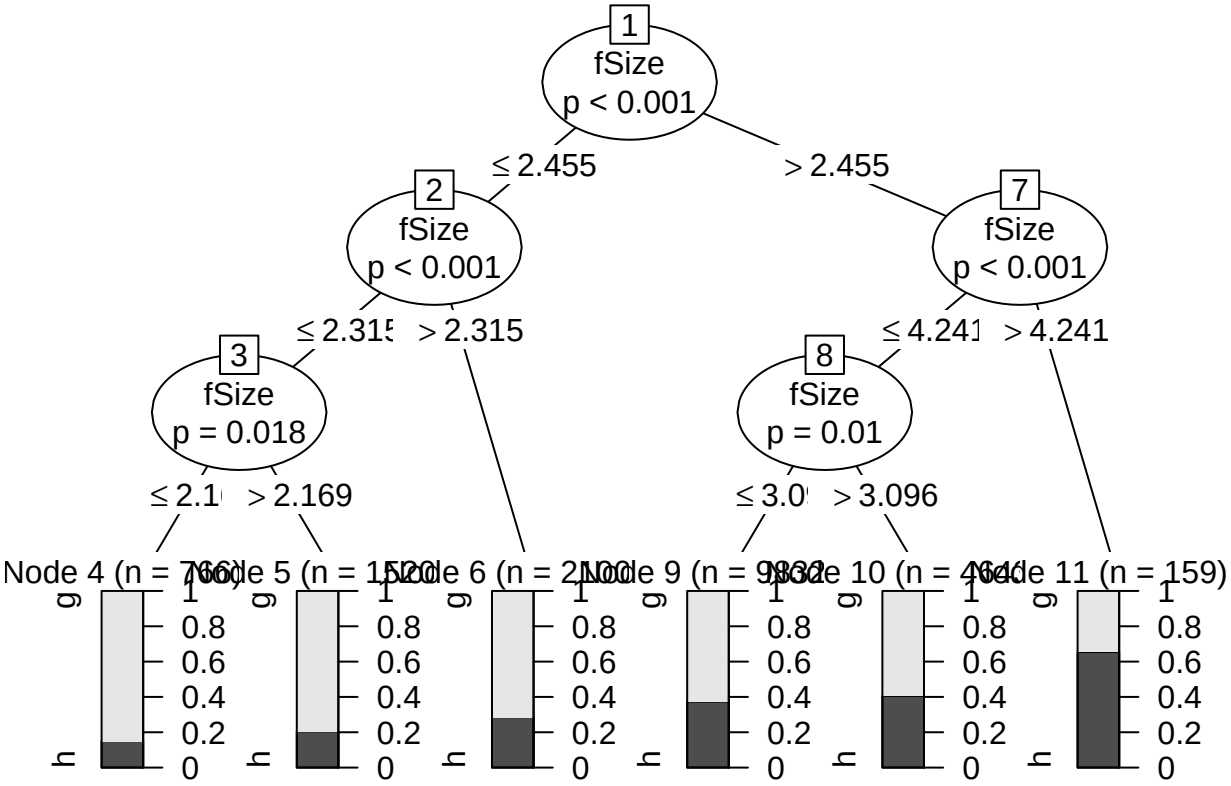
The plot, as expected (due to the large nature of the gini values within our dataset) has no apparent distinction between the "fSize" attribute for the "g" class and the "h" class. This can also be seen in the regression lines, as the linear regression line only varies slightly from the logistic regression, hinting that there is no real correlation between the variable and the classes.

SUPERVISED LEARNING MODELS

Below are 6 various supervised classification methods:

Decision Tree Analysis

```
modelCT <- data %>%
  ctree(class~fSize, data = .)%>% plot
```



In this decision tree, using p-values (probability value representing the probability of a sample statistic occuring (fSize) if the population paramater is false, lower means a more accurate result) of the fSize variable, we can see that 6 bins of probabilities is the optimal value for classification. Since we have determined that classification using the "fSize" variable will reap inaccurate results, the inaccuracy of this tree comes to no surprise. First of all, if majority determines classification, this method only predicts 159 of the data points to be in the "h" class, when there are in fact 6688 data points of the "h" class. This same error can be seen in the "g" class, as the amount of data points is largely over-estimated.

Partition Model Analysis

```
modelPT <- data %>%
  rpart(class~fSize, data = .)
predict(modelPT, data) %>% head
```

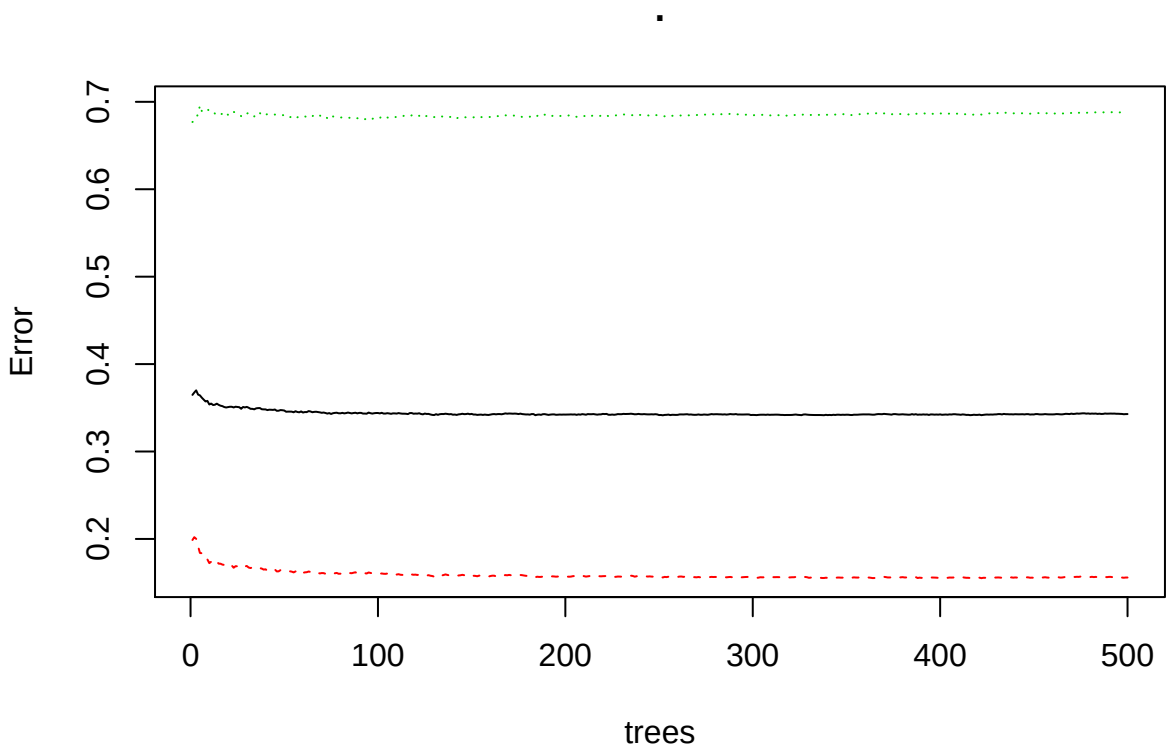
```
##              g         h
## [1,] 0.6483701 0.3516299
## [2,] 0.6483701 0.3516299
## [3,] 0.6483701 0.3516299
## [4,] 0.6483701 0.3516299
## [5,] 0.6483701 0.3516299
## [6,] 0.6483701 0.3516299
```

Viewing the head (first 5 data points) of the partition model, there is an apparent redundancy that diminishes the veritability of this model. For all of the first 5 values in this model, the probability for the "g" class is .6483701 and the probability for the "h" class is .3516299. These probabilities are equivalent to the distribution of the dataset.

Dataset distribution: g; 12332/19020 = .64837013 h: 6688/19020 = .35162986

Random Forest Analysis

```
modelRF <- data %>%
  randomForest(class~fSize, data = .)%>% plot
```



This random forest analysis graph shows the error of the algorithm rather than the output. The error seems to vary greatly for each tree and seems to follow a predictable regression, dismissing the veritability of this model

Neural Network Analysis

```
modelNN <- data %>%
  nnet(class~fSize, data= ., size = 5)
```

```
## # weights:  16
## initial  value 13864.441533
## iter  10 value 12181.858660
## iter  20 value 12129.003341
## iter  30 value 12119.380249
## iter  40 value 12106.970144
## iter  50 value 12101.109793
## iter  60 value 12099.764898
## iter  70 value 12098.570520
## iter  80 value 12097.842145
## iter  90 value 12097.102201
## iter 100 value 12095.769562
## final  value 12095.769562
## stopped after 100 iterations
```

```
predict(modelNN, data) %>% head
```

```
##        [,1]
```

```
## 1 0.3852883
## 2 0.3627799
## 3 0.4715139
## 4 0.2593226
## 5 0.3774733
## 6 0.3810902
```

In the head of this dataset, we can see probability values, all of which are under .5, signifying that the neural network believes that (for at least the first 5 values of the dataset), they would all belong to the same class. This is in fact the case, even though the predictor does vary, this is a decently accurate predictor.

Support Vector Machine Analysis

```
modelSVM <- data %>%
  ksvm(class~fSize, data = .)

table(data$class, predict(modelSVM, data))
```

```
##
##         g     h
##   g 12288    44
##   h  6598    90
```

Naive Bayesian Analysis

```
modelNB <- data %>%
  naiveBayes(class~fSize, data=.)
  predict(modelNB, data) %>% head
```

```
## [1] g g h g g g
## Levels: g h
```

```
  table(data$class, predict(modelNB, data))
```

```
##
##         g     h
##   g 11906   426
##   h  6324   364
```

Both the support vector machine model and the naive bayesian model output a table where predicted values and observed values merge. When the row and column are both "g" or both "h", this indicates that the class has been correctly predicted. Otherwise, the class has not been correctly predicted.

As seen in both these models, more "g" classes were predicted in the SVM model, but more "h" classes were predicted in the Naive Bayes analysis. Both of these models, though, do not serve as explicitly accurate for both classes.

REFERENCES

Magerman (1995) Caruana and Niculescu-Mizil (2006) Sathya and Abraham (2013)

Caruana, Rich, and Alexandru Niculescu-Mizil. 2006. "An Empirical Comparison of Supervised Learning Algorithms." In *Proceedings of the 23rd International Conference on Machine Learning*, 161–68. ACM.

Magerman, David M. 1995. "Statistical Decision-Tree Models for Parsing." In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, 276–83. Association for Computational Linguistics.

Sathya, R, and Annamma Abraham. 2013. "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification." *International Journal of Advanced Research in Artificial Intelligence* 2 (2): 34–38.