PSTAT 131 Project Report

Nathan Fritter

12/3/2016

Abstract/Executive Summary

For my project, I took a dataset containing attributes about students in secondary school and attempted to predict their final grades using the rest of the attributes. I also attempted to find any and all interesting relationships between sets of variables, and transform certain variables as necessary. For this project I utilized a decision tree for classification with a random forest added on as well as a Support Vector Machine and cross validation to find the best model parameters. The biggest key result I learned from this project was that how data sets are formatted, as well as the type of data collected can completely alter the direction of its analysis. I had to reformat a significant amount of the data manually; from this I also learned that numeric data features can provide much more insight than categorical features. Methods like Principal Component Analysis (PCA) and checking correlation could not be used; I could only test for significance in predicting the final grade variable.

While I could have done other things with this dataset, the type of data collected really limited the type of analysis I could do. Perhaps if the surveys the students took asked for more specific numbers for the features I would have been able to perform a more meaningful analysis. However, understanding that some of that information is sensitive for a student to put on a survey (even anonymously), this was a pretty cool collection of data points to look at. If there ever is a point where that sort of data could be recorded, I believe that more interesting insights could be discovered; and perhaps a more accurate model that isn't so dependent on previous grades for accuracy. I believe I got the best possible model with the dataset given to me.

Introduction

Using this dataset I am attempting to predict the student's final grades (response feature) from the various explanatory features included in the data. The explanatory features in the data set include school, gender, family size, mothers/fathers education and profession, travel time to school, daily and weekly alcohol consumption, number of class failures, freetime, and more. When looking at the data, I asked myself questions such as:

- Can I predict student alcohol consumption based on the other features in the dataset (which is what the authors of the paper have encouraged to do)?
- Should I try and predict final grades instead (which they also added as another output target)?
- If so, does there need to be any transformations for any of the variables (even final grades)?
- What relationships (if any) can I find between certain variables?

I found this data set not only interesting because I am still in school and it made me think about what attributes I would have if I was in one of those two classes, but also because the authors of the paper wanted to see if student alcohol consumption had any significance with grades (or other features). The intention of the project is sound and something I would get behind, but the writers unfortunately did not do a great job at achieving this goal. In fact, the paper that came with this had never been published, and attempted to merge two different datasets because some of the same students took both classes and took the survey twice. The merge led to confusing results, so I decided to just analyze one of the classes (Portugese).

In analyzing this dataset, I am planning to address student education and how it might be improved by looking at factors that could impact student grades. For example, if I were to find that travel time to school had a big effect on final grades (most likely a negative relationship), this could suggest that investing in services that improve the lives of students that live far away could be to the school's benefit. Of course, figuring out whether or not alcohol consumption has an effect on student's final grades would be a huge finding and would further give instituions reason to crack down on underage drinking and discourage it using any hypothetical findings from this report.

I decided to go with classification for this project; because of this I used a decision tree plus random forest as my first model, and a Support Vector Machine for the second model. I also ran the models through 21 class classification (original scale of 0 - 20), 5 class classification (letter scale of A, B, C, D, and F), and binary classification (Pass/No Pass). The last of the three methods yielded the best results, and the Random Forest model had the lower test error rate of the two.

The positives to this was that I was able to get a solid model with a low test error rate. The correct models were also cross validated, so I know that the best possible parameters were chosen. However, the two explanatory grade variables were highly correlated and contributed to almost all of the influence in predicting the final grade; after removing both of them to see what would happen, none of the other variables could prove significant in predicting the final grade variable. This also means that there are really no interesting or surprising insights from the non-grade variables. It did not help that a lot of the data was in categories; this eliminated a good amount of further analysis that could have been done.

For my project, I took the "Student Alcohol Consumption" dataset from the UCI Machine Learning archive (https://archive.ics.uci.edu/ml/datasets/STUDENT+ALCOHOL+CONSUMPTION), and this project was done using Rstudio. The rest of this project will contain more information about my dataset, processing steps and analysis, plots and data mining techniques comparing two classification algorithms, and the results of my analysis with their subsequent conclusions and all resources used for the project.

Main Body

Description/Preprocessing Steps

As said before, this dataset contains various features detailing student's lives (age, gender, travel time to school, free time, mother/father education, daily/weekly alcohol consumption, grades, etc.) in one of two classes: math and portugese. The two classes came separated into two data frames, and came with a mini R script that was to be used to merge the two data sets based on students (as some students were in both classes and took the survey both times); however, merging the datasets and using the merged one led to many issues and thus I decided to simply use the portugese class data because there was more observations than the math class data set.

After choosing to go with the Portugese class data set, I noticed that a lot of the variables (espcially the interesting ones like travel time and alcohol consumption) were discrete numeric variables (on a scale; usually 1-5 but sometimes other numbers) and R was reading them as continuous numeric variables. This was affecting my analysis, so I changed the variables into ordinal factors. The reason for choosing ordinal factors (factors that have relative value to each other) rather than regular factors is because if we had used regular variables, the relationship between the values of the variable is lost. The data was read in and modified as followed:

```r
d1=read.table("~/Documents/student-mat.csv",sep=";",header=TRUE)
d2=read.table("~/Documents/student-por.csv",sep=";",header=TRUE)

# For this project I will only be analyzing d2, which is the portugese class grades
# Most of these variables need to be converted
# They are numbers on a scale (ordinal)
# R studio is reading them in as integers and making them continuous
# This will be the case for both regression and classification
# First check structure pre-transformation
str(d2)
```

```
## 'data.frame':    649 obs. of  33 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ Pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ Fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  4 2 6 0 0 6 0 2 0 0 ...
##  $ G1        : int  0 9 12 14 11 12 13 10 15 12 ...
##  $ G2        : int  11 11 13 14 13 12 12 13 16 12 ...
##  $ G3        : int  11 11 12 14 13 13 13 13 17 13 ...
```

```r
# Now reformat everything
d2$age <- ordered(d2$age,
                  levels = c(15:22),
                  labels = c("15", "16", "17", "18", "19", "20", "21", "22"))

d2$famsize <- ordered(d2$famsize,
                      levels = c("LE3", "GT3"))

d2$Medu <- ordered(d2$Medu,
                   levels = c(0:4),
                   labels = c("0", "1", "2", "3", "4"))

d2$Fedu <- ordered(d2$Fedu,
                   levels = c(0:4),
                   labels = c("0", "1", "2", "3", "4"))
```

```r
d2$traveltime <- ordered(d2$traveltime,
                   levels = c(1:4),
                   labels = c("1", "2", "3", "4"))

d2$studytime <- ordered(d2$studytime,
                         levels = c(1:4),
                         labels = c("1", "2", "3", "4"))

d2$failures <- ordered(d2$failures,
                         levels = c(0:3),
                         labels = c("0", "1", "2", "3"))

d2$famrel <- ordered(d2$famrel,
                        levels = c(1:5),
                        labels = c("1", "2", "3", "4", "5"))

d2$freetime <- ordered(d2$freetime,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

d2$goout <- ordered(d2$goout,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

d2$famrel <- ordered(d2$famrel,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

d2$Dalc <- ordered(d2$Dalc,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

d2$Walc <- ordered(d2$Walc,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

d2$health <- ordered(d2$health,
                     levels = c(1:5),
                     labels = c("1", "2", "3", "4", "5"))

# Check structure of dataset now
str(d2)
```

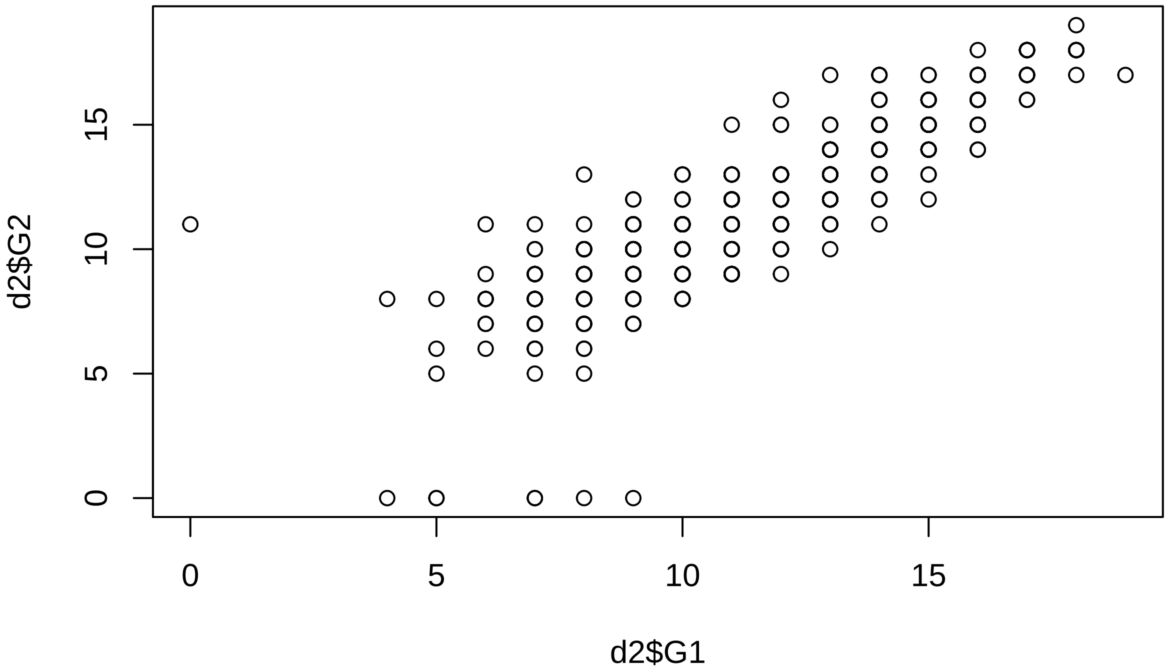
```
## 'data.frame':    649 obs. of  33 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : Ord.factor w/ 8 levels "15"<"16"<"17"<..: 4 3 1 1 2 2 2 3 1 1 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Ord.factor w/ 2 levels "LE3"<"GT3": 2 2 1 2 2 1 1 2 1 2 ...
##  $ Pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ Medu      : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 5 2 2 5 4 5 3 5 4 4 ...
##  $ Fedu      : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 5 2 2 3 4 4 3 5 3 5 ...
##  $ Mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ Fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<..: 3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  4 2 6 0 0 6 0 2 0 0 ...
##  $ G1        : int  0 9 12 14 11 12 13 10 15 12 ...
##  $ G2        : int  11 11 13 14 13 12 12 13 16 12 ...
##  $ G3        : int  11 11 12 14 13 13 13 13 17 13 ...
```

After running the random forest and support vector machine models with all of the variables included, I found that the first and second grades explained almost all of the variance. I inferred that these variables would be highly correlated, and decided to check the correlation between the first grade and the second grade as numeric variables via the cor() function. I graphed the variables against each other as well; here are the results:

```
cor(d2$G1, d2$G2)
```

```
## [1] 0.8649816
```

```
plot(d2$G1, d2$G2)
```



Looking at the above output, it is clear the variables are highly correlated. Thus, this prompted me to remove the first grade variable, as it contributed to less of the variance in the final grade than the second grade. I will do this below:

```
d2$G1 <- NULL
```

Now the variable has been removed, and we can continue with the analysis.

Visualization Techniques

Since most of the data was categorical (only maybe three or four variables were numeric continuous), nice visualizations like Principal Component Analysis (PCA) and correlation plots (fitted vs residuals) could not be done. I was able to acquire the distributions of the various categorical variables, and here are some of them below:

```
table(d2$Medu)
```

```
##
##   0   1   2   3   4
##   6 143 186 139 175
```

```
table(d2$Fedu)
```

```
##
##   0   1   2   3   4
##   7 174 209 131 128
```

```
table(d2$traveltime)
```

```
##
##   1   2   3   4
## 366 213  54  16
```

```
table(d2$studytime)
```

```
##
##   1   2   3   4
## 212 305  97  35
```

```
table(d2$famrel)
```

```
##
##   1   2   3   4   5
##  22  29 101 317 180
```

```
table(d2$freetime)
```

```
##
##   1   2   3   4   5
##  45 107 251 178  68
```

```
table(d2$goout)
```

```
##
##   1   2   3   4   5
##  48 145 205 141 110
```

```
table(d2$Dalc)
```

```
##
##   1   2   3   4   5
## 451 121  43  17  17
```

```
table(d2$Walc)
```

```
##
##   1   2   3   4   5
## 247 150 120  87  45
```

```
table(d2$health)
```

```
##
##   1   2   3   4   5
##  90  78 124 108 249
```

From these tables we can see that factors such as Mother/Father education hover in the mid to high range categories, travel time and study time hover in the mid lower categories, and both alcohol consumption variables stay in the lower percentages (although it would be nice to see the numbers go all the way down to zero in the high percentages).

While I could have turned these numbers into bar graphs, the time constraints of this project coupled with the fact that these numbers don't really add much significance to the project led me to hold back on doing so. This is something I intend on doing in the future as a next set of steps for the project.

Data Mining Techniques

For this project I utilized a decision tree for classification with a random forest added on to improve results. Due to the fact that most of the variables were either binary or ordinal factors, a decision tree made a lot of sense (after transforming the grade variables into ordinal categories and then binary for Pass/No Pass). I pruned the tree before using a forest, found that it did not change the tree very much, and went ahead with the random forest model on the whole data set.

I also utlized a Support Vector Machine to handle possible non-linear decision boundaries between the different class labels. A Support Vector Machine is an excellent choice for classification due to its ability to map certain non-linear variables into a higher dimension space, making the decision boundaries more linear in nature. I used cross validation to find the best parameters for the model, then using the actual SVM() method with the parameters to get the best fine-tuned model.

But before I implement my models, I need to transform the grade variable into a binary variable and split the data into train and test sets.

Binary Transformation

While I did try three separate methods for classification (21 classes for the original scale, 5 classes for letter grades, and binary for Pass/No Pass), the binary method ended up being the most accurate. I will demonstrate this in the next code chunk.

First, I created a new variable holding the portugese class data frame. Then I divided the numbers by 20 to get percentages, and set all percentages 70% or greater as "Pass" and the rest as "No Pass".

```r
# Let's try if we were to turn the grades into pass/no pass
portug.grades.pass <- d2

# Divide grades by 20 to get percentages
portug.grades.pass$G2 <- portug.grades.pass$G2 / 20
portug.grades.pass$G3 <- portug.grades.pass$G3 / 20

# And turn into Pass/No Pass
portug.grades.pass$G2 <- cut(portug.grades.pass$G2,
                             c(0, 0.7, Inf),
                             right=FALSE,
                             labels=c("No Pass", "Pass"))
portug.grades.pass$G3 <- cut(portug.grades.pass$G3,
                             c(0, 0.7, Inf),
                             right=FALSE,
                             labels=c("No Pass", "Pass"))
```

Train/Test Split

Next, I created a random sample of indices that made up 80% of the dataset, and split up the data into training and testing (for the decision tree and for the Support Vector Machine after cross validation).

```r
# Split data into train and test set
ratio = 0.80
train.num <- ratio * nrow(portug.grades.pass)
train.ind <- sample.int(nrow(portug.grades.pass), train.num)

# Create X and Y
```

```
portug.grades.pass.Y <- portug.grades.pass$G3
portug.grades.pass.X <- subset(portug.grades.pass, select = -c(G3))


# Extract the training set using our train indices
portug.grades.pass.X.train <- portug.grades.pass.X[train.ind,]
portug.grades.pass.Y.train <- portug.grades.pass.Y[train.ind]
portug.grades.pass.train <- portug.grades.pass[train.ind,]

# Get the test set from the rest
portug.grades.pass.X.test <- portug.grades.pass.X[-c(train.ind),]
portug.grades.pass.Y.test <- portug.grades.pass.Y[-c(train.ind)]
portug.grades.pass.test <- portug.grades.pass[-train.ind,]
```

Decision Tree

I created a decision tree with the following commands, and got a tree that was pretty accurate and with second grade variable (G2) as the main predictor of the final grade. You can see the initial tree below:

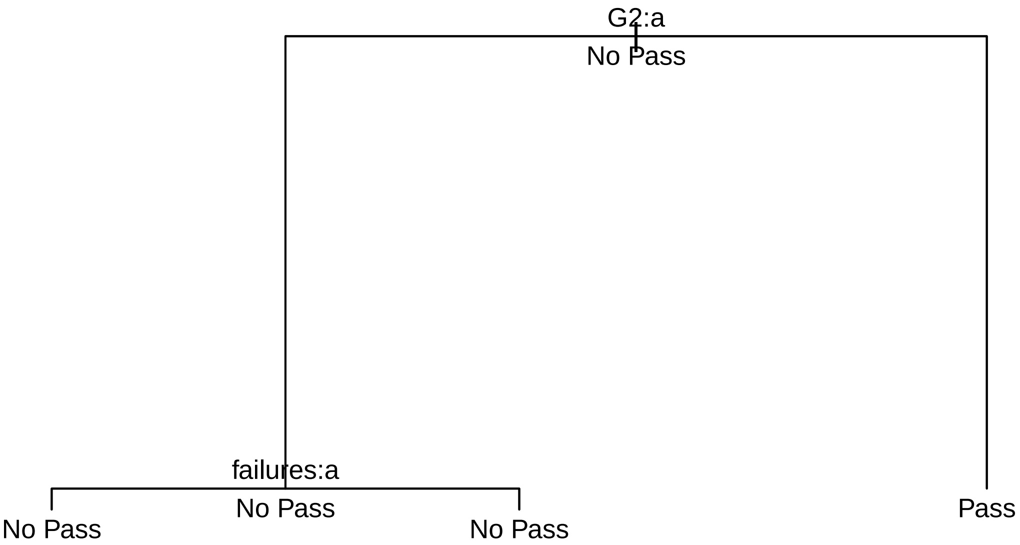
```
# Decision tree
portug.dec.tree.pass <- tree(G3 ~ ., data = portug.grades.pass, subset = train.ind)
portug.dec.tree.pass
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 519 632.90 No Pass ( 0.70135 0.29865 )
##   2) G2: No Pass 397 241.50 No Pass ( 0.90932 0.09068 )
##     4) failures: 0 318 224.60 No Pass ( 0.88679 0.11321 ) *
##     5) failures: 1,2,3 79   0.00 No Pass ( 1.00000 0.00000 ) *
##   3) G2: Pass 122  28.16 Pass ( 0.02459 0.97541 ) *
```

```
plot(portug.dec.tree.pass)
text(portug.dec.tree.pass, use.n=TRUE, all=TRUE, cex=.8)
```

```
## Warning in text.default(xy$x[ind], xy$y[ind] + 0.5 * charht, rows[ind], :
## "use.n" is not a graphical parameter
```

```
## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 * charht, labels =
## stat, : "use.n" is not a graphical parameter
```

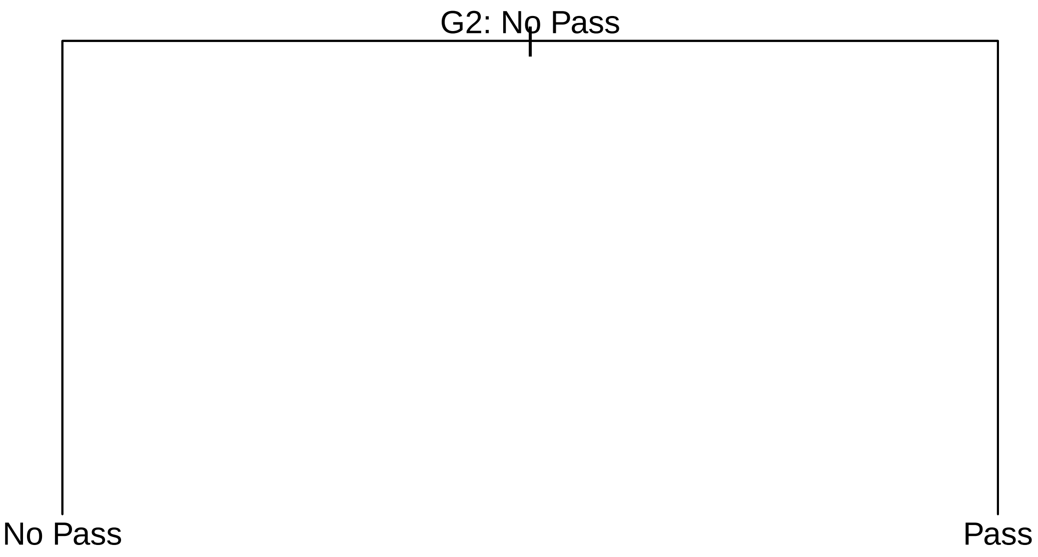


I also pruned the tree (via cross validation) to get the ideal number of branches, and found that it ended up using the one grade variable remaining to predict the other.

```
# Let's try pruning the tree
portug.dec.tree.cv <- cv.tree(portug.dec.tree.pass, FUN = prune.misclass)
portug.dec.tree.cv
```

```
## $size
## [1] 3 2 1
##
## $dev
## [1]  44  44 155
##
## $k
## [1] -Inf    0  116
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
# We wish to match up the size of the trees with the lowest number in the "dev" section
# size = 2 has the lowest dev value, so we will pick this
portug.dec.tree.prune <- prune.misclass(portug.dec.tree.pass, best = 2)
plot(portug.dec.tree.prune)
text(portug.dec.tree.prune, pretty = 0,
     main = "Pruned Classification Tree for 21 Ordinal Class Decision Tree")
```



Since there was such a big change, I stuck with the initial tree and used it to predict the test data set. I printed out the confusion matrix to see how the algorithm in terms of test error rate (the diagonals are correct labels, everything else is incorrect; the left side represents the prediction by the decision tree, and the right side shows the actual value):

```
portug.dec.pred.pass <- predict(portug.dec.tree.pass,
                                portug.grades.pass.X.test,
                                type = "class")
conf.matrix.dec.tree <- table(portug.dec.pred.pass,
                              portug.grades.pass.Y.test)
conf.matrix.dec.tree
```

```
##                       portug.grades.pass.Y.test
## portug.dec.pred.pass No Pass Pass
##              No Pass      91    9
##              Pass          0   30
```

```
diag(conf.matrix.dec.tree)
```

```
## No Pass    Pass
##      91      30
```

Using the confusion matrix I am able to calculate the test error rate by taking the sum of the diagonals (correct predictions), dividing that by the number of observations in the test set, then taking the compliment of this to get the percent of observations incorrectly labeled:

```
# Calculate test error rate; looks good so far
correct.pass <- sum(diag(conf.matrix.dec.tree))
total.pass <- length(portug.grades.pass.Y.test)
total.pass
```

```
## [1] 130
```

```
test.error.rate.pass <- 1 - (correct.pass / total.pass)
test.error.rate.pass
```

```
## [1] 0.06923077
```

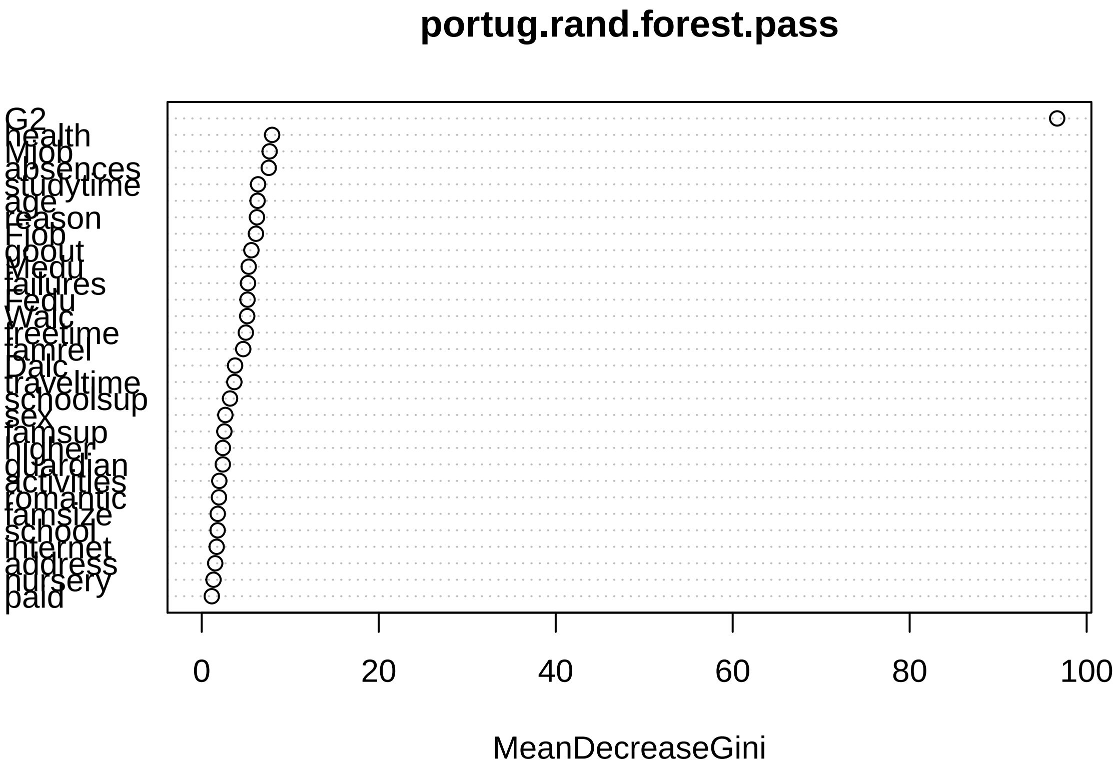
Random Forest

This above model has a good test error rate for predicting the final grades (around 10%); but implementing a random forest would improve the model and lower the test error rate. A Random Forest is decision trees mixed with cross validation; when implementing the random forest, different combinations of training and testing data sets are extracted and models created using these various train/test data sets. The best models will be outputted with the code below, along with the ideal number of branches for the decision tree (should be similar to the decision tree + cross validation before):

```
# Random Forest
portug.rand.forest.pass <- randomForest(G3 ~ .,
                                        data = portug.grades.pass,
                                        subset = train.ind,
                                        norm.votes=FALSE)
portug.rand.forest.pass
```

```
##
## Call:
##  randomForest(formula = G3 ~ ., data = portug.grades.pass, norm.votes = FALSE,      subset = train.i
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 7.51%
## Confusion matrix:
##         No Pass Pass class.error
## No Pass     361    3 0.008241758
## Pass         36  119 0.232258065
```

```
varImpPlot(portug.rand.forest.pass)
```



The accuracy here, in terms of test error rate, is the best out of all the decision tree models attempted.

Support Vector Machine

As with decision trees, I tried all three classification methods and the binary Pass/No Pass method ended up achieving the highest accuracy. Here are the steps I performed to create a Support Vector Machine model:

Since this was the second model built, I simply used the same dataset and train/test split as before to fit the model. But before that, I ran repeated 10-fold cross validation using the "caret" package and the "svmRadial" method (which means Support Vector Machines for classification):

```
# Now let's compare this to another classification algorithm
# A Support Vector Machine can utilize non linear decision boundaries to make a decsion
# So let's use a Support Vector Machine
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3)
portug.grades.svm.cv <- train(G3 ~ ., data = portug.grades.pass,
                 method = "svmRadial",
                 preProcess = c("center", "scale"),
                 tuneLength = 10,
                 trControl=control)
```

```
## Loading required package: kernlab
```

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

```
portug.grades.svm.cv
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 649 samples
##  31 predictor
##   2 classes: 'No Pass', 'Pass'
##
## Pre-processing: centered (76), scaled (76)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 585, 584, 585, 585, 584, 583, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.8875270  0.6963892
##     0.50  0.9260137  0.8100671
##     1.00  0.9260137  0.8100671
##     2.00  0.9260137  0.8100671
##     4.00  0.9219109  0.8005960
##     8.00  0.9136813  0.7813583
##    16.00  0.9054521  0.7635051
##    32.00  0.9054521  0.7635520
##    64.00  0.9044107  0.7612672
##   128.00  0.9033851  0.7589985
##
## Tuning parameter 'sigma' was held constant at a value of 0.006911013
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.006911013 and C = 0.5.
```

Above, the ideal cost and sigma values were outputted; I took those and fit a SVM model with those parameters below:

```
portug.grades.svm <- svm(G3 ~ .,
                         data = portug.grades.pass.train,
                         kernel = "radial",
                         cost = 0.5,
                         sigma = 0.007116499)
summary(portug.grades.svm)
```

```
##
## Call:
## svm(formula = G3 ~ ., data = portug.grades.pass.train, kernel = "radial",
##     cost = 0.5, sigma = 0.007116499)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.5
##       gamma:  0.01298701 
## 
## Number of Support Vectors:  344
## 
##  ( 189 155 )
## 
## 
## Number of Classes:  2 
## 
## Levels: 
##  No Pass Pass
```

The model looks good. I then took the model and predicted the test set responses, and created a confusion matrix to show the prediction versus the actual values:

```
portug.grades.svm.pred <- predict(portug.grades.svm,
                                  portug.grades.pass.X.test)
conf.matrix.svm <- table(portug.grades.svm.pred,
                         portug.grades.pass.Y.test)
conf.matrix.svm
```

```
##                         portug.grades.pass.Y.test
## portug.grades.svm.pred No Pass Pass
##                  No Pass      91   10
##                  Pass          0   29
```

And finally, I calculated the test error rate using the ideal parameters below:

```
# Get test error rate
correct.svm <- sum(diag(conf.matrix.svm))
total.svm <- length(portug.grades.pass.Y.test)
test.error.svm <- 1 - (correct.svm / total.svm)
test.error.svm
```

```
## [1] 0.07692308
```

Conclusion

In conclusion I was able to make a pretty good model with the data given. However, since the model mainly used the other grade variable, and most of the other variables were factors, this greatly inhibited the analysis that I could have done with this dataset. I do understand that asking questions with scales (i.e. 1-5) made collecting the data much easier, as well as allowing students to not have to write down information they might not be comfortable sharing. If there the variables were instead numeric that would lead to a much more meaningful analysis. But with the data I was given, I feel like I did what I could.

The classification methods each yielded different results. The 21 class classification (which I tried at the beginning) yielded a high variance and test error rate, so I decided not to stick with this type of model. Turning the grades into letter grades (A, B, C, D, F) that were ordinal factors helped with the test error rate but still was not the best model to move ahead with. Finally, turning the grades into a binary variable (Pass/No Pass) yielded the best test error rate (both models getting around 7 - 11% accuracy, with the random forest getting the lower test error rate). Since a goal of classification is to try and have the lowest number of classes to try and predict from, the transformation to a binary variable did the trick.

My friend, Raul Eulogio, helped me a lot for this project. He graduated last year with an Applied Statistics degree, and also took this class in the spring. He had a solid knowledge of the data mining process, and gave the suggestion for the decision tree + random forest as well.

As for next steps, me and Raul are planning to analyze this dataset more, as well as try a neural network model to see if anything changes occur. I might also look into turning the distributions of the categorical variables into actual bar graphs and other visual objects.

References

This was the citation required for using this dataset:

"Using Data Mining To Predict Secondary School Student Alcohol Consumption. Fabio Pagnotta, Hossain Mohammad Amran Department of Computer Science, University of Camerino"

I also used these sites as resources for this project:

Kabacoff, R. I., Ph.D. (n.d.). Quick-R. Retrieved December 10, 2016, from http://www.statmethods.net/advstats/cart.html

Decision tree model evaluation for "training set" vs "testing set" in R. (n.d.). Retrieved December 10, 2016, from http://stats.stackexchange.com/questions/49416/decision-tree-model-evaluation-for-training-set-vs-testing-set-in-r

Convert percentage to letter grade in R. (n.d.). Retrieved December 10, 2016, from http://stackoverflow.com/questions/27415071/convert-percentage-to-letter-grade-in-r