SDS 293 - Results Check-In

Team Avocado Toast

11/21/2018

During our preliminary methods test, we determined that regression trees gave the lowest MSE values of all of the regression methods we tried (other methods were subset selection and lasso). For this check-in, we will complete tree analysis from start to finish.

We will use our `combined_final` (no dummy variables) dataset for this method because this method can easily use categorical predictors.

```r
# Read in the math student dataset and the portuguese student dataset and combine
math <- read.csv(file = "student-mat.csv", header = TRUE, sep = ",")

portuguese <- read.csv(file = "student-por.csv", header = TRUE, sep = ",")

combined <- rbind(math, portuguese)

# Remove G1 and G2 (alternate response variables that we're not
# interested in using as response or predictors)
combined_final <- combined %>%
  dplyr::select(-G1, -G2)
```

First, we will look at a basic regression tree from the `tree` package.

```r
# Get a training and test set
# Used to evaluate the success of the method on test data
set.seed(1)
train <- combined_final %>%
  sample_frac(0.75)

test = combined_final %>%
  setdiff(train)
```

```r
tree_combined = tree(G3~., train)
summary(tree_combined)
```

We can look at the unpruned tree we get from the model:

```r
plot(tree_combined)
text(tree_combined, pretty= 0)
```

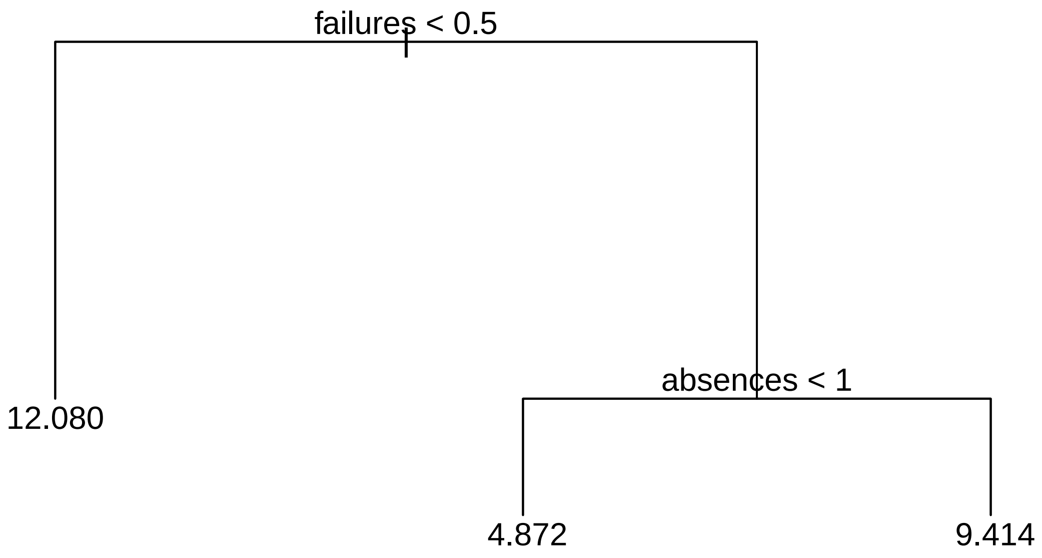
Since this tree is not pruned, it is likely that it includes some predictors that are not particularly useful. We can use cross-validation to prune our tree by determining the optimum number of terminal nodes:

```r
set.seed(1)
# Plot number of nodes vs error rate
cv_combined <- cv.tree(tree_combined)
plot(cv_combined$size, cv_combined$dev, type = "b", xlab = "# of Terminal Nodes", ylab = "Cross-Val Err
points(x=3,y=min(cv_combined$dev), col = "red", cex = 2, pch = 20)
```

A 3 node tree appears to be best.

```r
# Plot the pruned tree
pruned_tree <- prune.tree(tree_combined, best = 3)
```

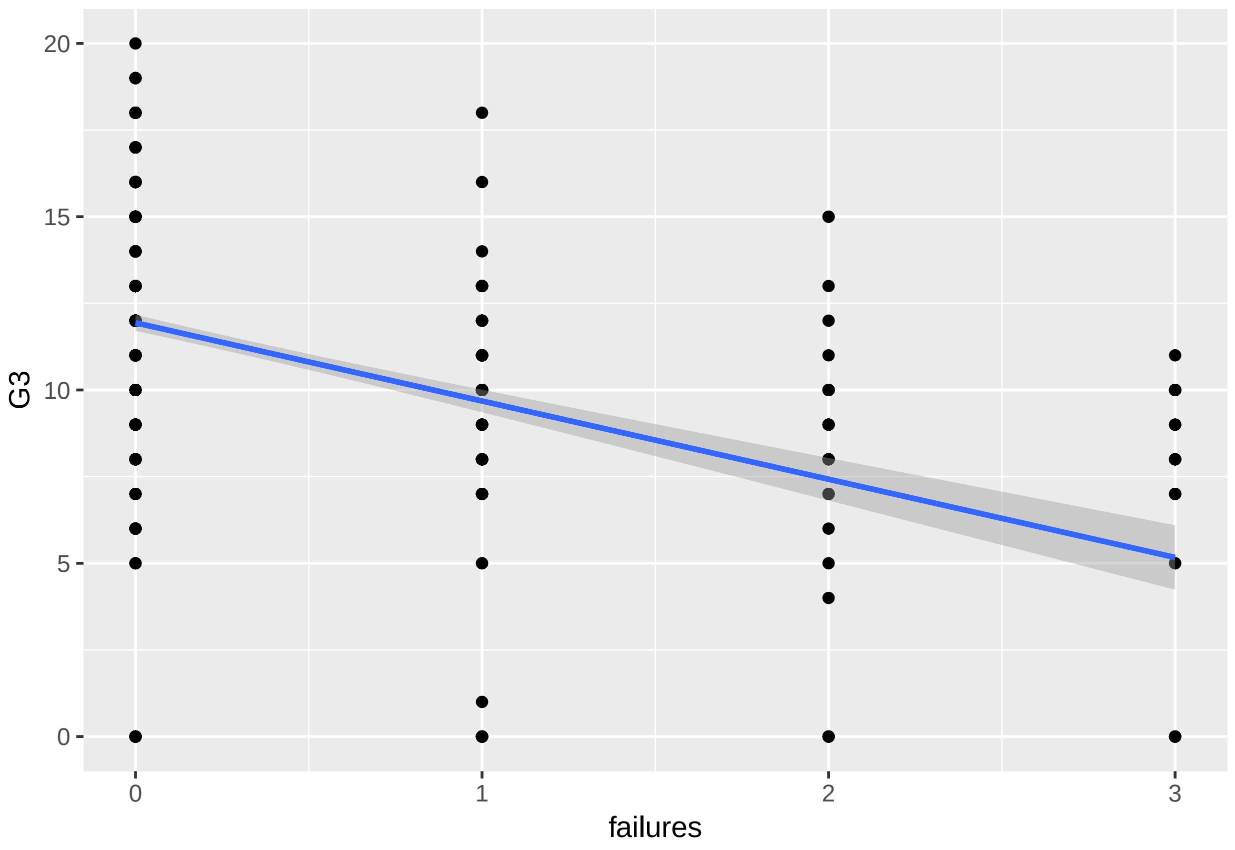
```r
plot(pruned_tree)
text(pruned_tree, pretty = 0)
```



The pruned tree uses only two predictors: failures and absences. The split on `failures` here makes sense, but the `absences` distinction is quite strange. Perhaps students with `absences` < 1 are outliers for some reason. We would typically expect students with more absences to have worse grades, using intuition, but this model does not show that to be the case. If we examine some scatterplots more closely, we might see why absences < 1 has such a low grade.

Let's look at the scatterplot of failures vs final grade:

```r
ggplot(aes(x = failures, y = G3), data = combined_final) +
  geom_point() + geom_smooth(method = "lm", formula = y~x)
```

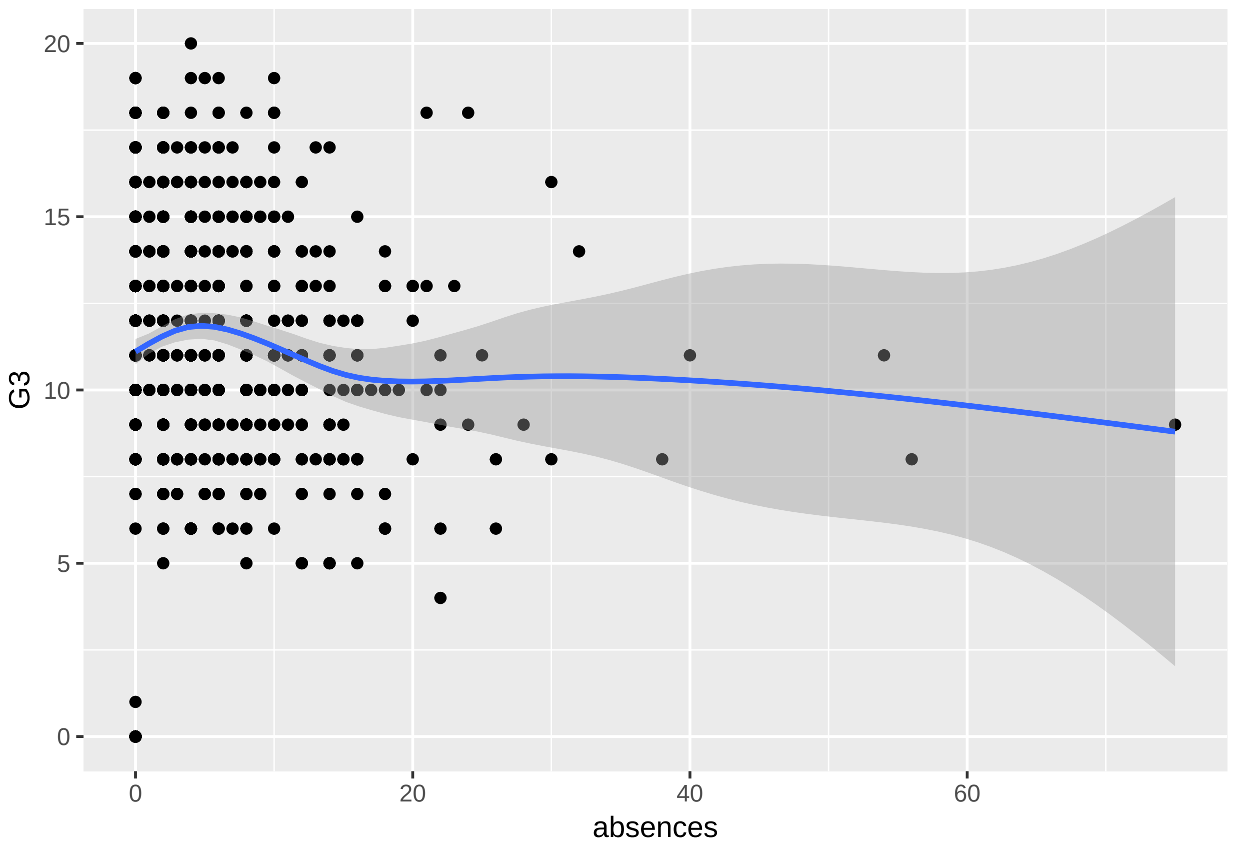


Failures appear to follow our intuition. More failures is associated with a lower final grade.

We can also look at absences vs. final grade:

```r
ggplot(aes(x = absences, y = G3), data = combined_final) +
  geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



There does not appear to be a clear upward or downward trend in this data. In fact, the area between around 0 and 5 absences runs counter to our intuition. We would expect more absences to be correlated with a lower grade, because missing school would mean that you're missing out on learning, but that is not true for the first few absences. However, this is likely a result of the fact that there are several outliers with very low grades and zero absences that appear to be bringing down the grade average for low-absence students.

In addition to looking at these scatterplots, we can use this pruned tree to make predictions on the test dataset.

```
single_tree_estimate <- predict(pruned_tree, newdata = test)

mean((single_tree_estimate - test$G3)^2)
```

The MSE for this pruned tree is 12.88.

Next, we can try bagging.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

```
##
##     margin
```

```r
set.seed(1)

bag_combined <- randomForest(G3~.,
                             data = train,
                             mtry = 30,
                             importance = TRUE)
bag_combined
```

Let's look at variable importance:

```r
importance(bag_combined)
varImpPlot(bag_combined)
```

Failures and absences are most important in this model.

We can compare the test error rate of this model to the pruned tree from earlier:

```r
bagged_estimate <- predict(bag_combined,
                           newdata = test)
mean((bagged_estimate - test$G3)^2)
```

The MSE for this bagged tree is better than the first tree attempt at 11.7. This signifies that we probably have several very strong predictors in our data, since bagging often pushes these to the forefront.

Next, we can try random forest:

```r
library(randomForest)

set.seed(1)

# Restricting to 1/3 of predictors at each split
rf_combined <- randomForest(G3~.,
                            data = train,
                            mtry = 10,
                            importance = TRUE)

rf_estimate <- predict(rf_combined,
                       newdata = test)

mean((rf_estimate - test$G3)^2)
```

The MSE for the random forest tree is 11.69 (the lowest so far), which is slightly better than the result we got from bagging.

We can also look at the importance of each predictor in the random forest:

```r
importance(rf_combined)
```

```r
varImpPlot(rf_combined)
```

`failures` and `absences` are by far the two most important predictors, as with all the other models tested earlier, followed by `Mjob`, `goout`, and `reason`.

Now, we can try a boosted tree:

```r
set.seed(1)
```

```r
boost_combined <- gbm(G3~., data = train,
                      distribution = "gaussian",
                      n.trees = 5000,
                      interaction.depth = 4)
```

We can use this model to predict final grades:

```r
boost_estimate <- predict(boost_combined,
                          newdata = test,
                          n.trees = 5000)

mean((boost_estimate - test$G3)^2)
```

The MSE for this boosted model is the worst of all of the regression tree attempts (15.9825),

However, we know that there are several parameters we can tune with boosting to achieve better results. Let's try running the same model but with a different shrinkage parameter (slower learning):

```r
set.seed(1)
boost_combined2 = gbm(G3~., data = train, distribution = "gaussian",
                    n.trees = 5000,
                    interaction.depth = 4,
                    shrinkage = 0.01,
                    verbose = F)
boost_estimate2 = predict(boost_combined2, newdata = test, n.trees = 5000)
mean((boost_estimate2-test$G3)^2)
```

Slowing the learning rate significantly reduces the MSE to 11.78, which is similarly good as the other regression tree models.

Since the boosted tree performed really well, we wanted to see if we could tune the learning rate to lower the MSE even more. We tested several different learning rates:

```r
rates = c(0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1)
```

```r
set.seed(1)
for (rate in rates){

boost = gbm(G3~., data = train, distribution = "gaussian",
                    n.trees = 5000,
                    interaction.depth = 4,
                    shrinkage = rate,
                    verbose = F)
estimate = predict(boost, newdata = test, n.trees = 5000)
mse =  mean((estimate-test$G3)^2)
print(mse)

}
```

0.001 was the best learning rate tested with an MSE of 11.35, so we'll use that rate in our final model.

We can also try different values for `interaction.depth` (number of splits to perform on a tree):

```r
# Try 1 through 8 for interaction.depth
# and report the MSE values
set.seed(1)
for (i in 1:8){
```

```r
boost = gbm(G3~., data = train, distribution = "gaussian",
                    n.trees = 5000,
                    interaction.depth = i,
                    shrinkage = 0.001,
                    verbose = F)
estimate = predict(boost, newdata = test, n.trees = 5000)
mse =  mean((estimate-test$G3)^2)
print(mse)
}
```

An interaction depth of 8 appears best. We can now examine the model with our best interaction depth and best learning speed:

```r
set.seed(1)

best_boost = gbm(G3~., data = train, distribution = "gaussian",
                    n.trees = 5000,
                    interaction.depth = 8,
                    shrinkage = 0.001,
                    verbose = F)
estimate = predict(best_boost, newdata = test, n.trees = 5000)
mean((estimate-test$G3)^2)
```

It has an MSE of 11.13576, which is very good.

```r
summary(best_boost)
```

The top 5 variables in terms of importance appear to be: 1) failures 2) absences 3) Mjob 4) reason 5) goout

We can look at scatterplots of each of these vs final grade to get an idea of how they may be affecting predictions:

failures

```r
ggplot(aes(x = failures, y = G3), data = combined_final) +
  geom_point() + geom_smooth(method = "lm", formula = y~x) +
  ggtitle("#1) Failures")
```

Failures have a clear negative relationship to G3.

absences (natural fit)

```r
ggplot(aes(x = absences, y = G3), data = combined_final) +
  geom_point() +
  geom_smooth() +
  ggtitle("#2) Absences (smoothed)")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

absences (linear)

```r
ggplot(aes(x = absences, y = G3), data = combined_final) +
  geom_point() +
  geom_smooth(method = "lm", formula = y~x) +
  ggtitle("#2) Absences (linear)")
```

Absences do not have a clear relationship to G3 overall. It is possible that they were picked by the tree model due to the seemingly strong positive correlation with final grade at very low values for absences.

Mjob

```r
ggplot(aes(x = Mjob, y = G3, group = Mjob), data = combined_final) +
  geom_boxplot() +
  ggtitle("#3) Mjob")
```

Students appear to do better when their mother's job is in health, services, or teaching than if she is a stay at home job or works in another industry.

reason

```r
ggplot(aes(x = reason, y = G3, group = reason), data = combined_final) +
  geom_boxplot() +
  ggtitle("#4) reason")
```

The reason to choose the school also had an effect on final grade. Students who chose schools because they were close to home or had a good reputation performed slightly better than students who chose schools because of a specific course or due to another reason.

goout

```r
ggplot(aes(x = goout, y = G3, group = goout), data = combined_final) +
  geom_boxplot() +
  ggtitle("#5) goout")
```

The amount that students went out also appeared related to final grade. There was a positive increase in grades as students went from going out very little to going out a little bit more, but any additional increase in going out led to, on average, a decrease in student grades.

Summary

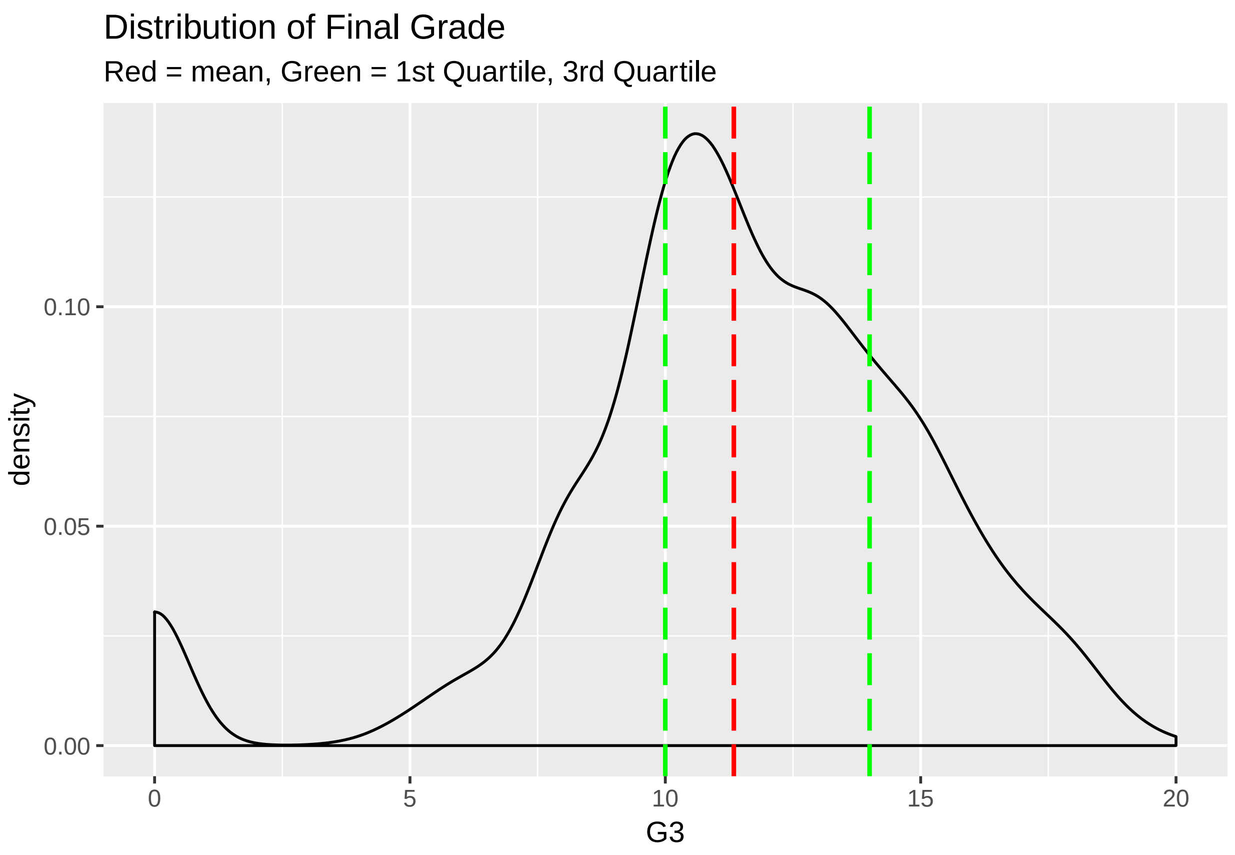
For all of the tree methods we tested, the best MSE value we found was 11.13576. The root mean squared error (RMSE) is 3.337. RMSE is a good measure of how far our predictions strayed from the true values

```r
sqrt(11.13576)
```

We can compare this RMSE to the distribution of our response variable, final grade:

```r
summary(combined_final$G3)
```

```r
ggplot(aes(x=G3), data = combined_final) +
  geom_density() +
  ggtitle("Distribution of Final Grade", subtitle = "Red = mean, Green = 1st Quartile, 3rd Quartile") +
  geom_vline(aes(xintercept = mean(G3, na.rm = T)),
             colour = "red", linetype ="longdash", size = 0.8) +
  geom_vline(aes(xintercept = 10), colour = "green", linetype = "longdash", size = 0.8) +
  geom_vline(aes(xintercept = 14), colour = "green", linetype = "longdash", size = 0.8)
```



The majority of grades lie between the two green lines (a range of only 4). Considering the small range in which the majority of grades fall, a root mean squared error of 3.34 is rather poor. This is almost as wide as the width that the majority of our data falls between. This error means that even with our best model, there does not appear to be a clear trend in the data that we have. We can not make accurate predictions of final grade given the predictors/data points given to us, no matter which method we try.

References:

https://www.kaggle.com/uciml/student-alcohol-consumption