Assignment 7 - Response

Timothy Lee

11/21/2019

In the following assignment you will be looking at data from one level of an online geography tutoring system used by 5th grade students. The game involves a pre-test of geography knowledge (pre.test), a series of assignments for which you have the average score (av.assignment.score), the number of messages sent by each student to other students about the assignments (messages), the number of forum posts students posted asking questions about the assignment (forum.posts), a post test at the end of the level (post.test) and whether or not the system allowed the students to go on to the next level (level.up).

Part I

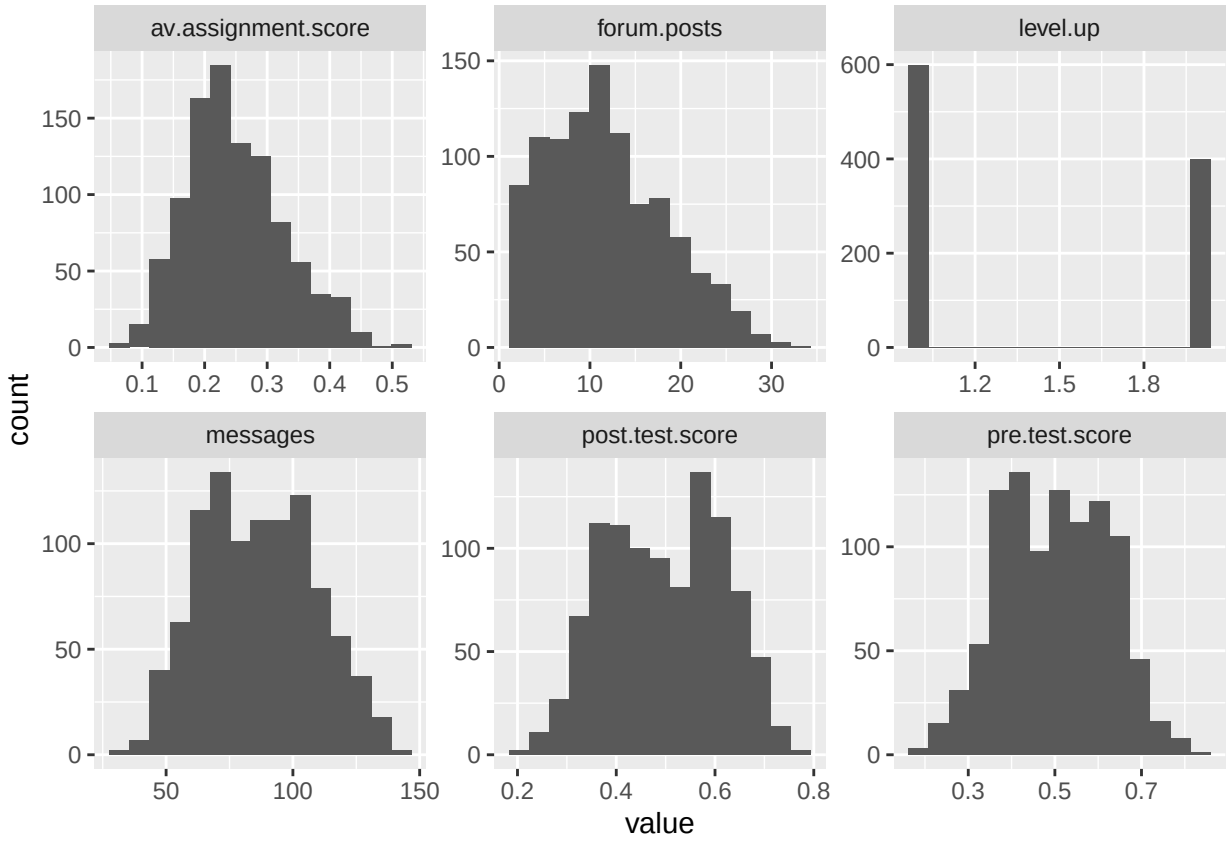
```
#It looks like I'll need a few of these guys
library(tidyverse)

#For Correlation matrix
library(GGally)

#Classification tree
library(rpart)
```

Upload data

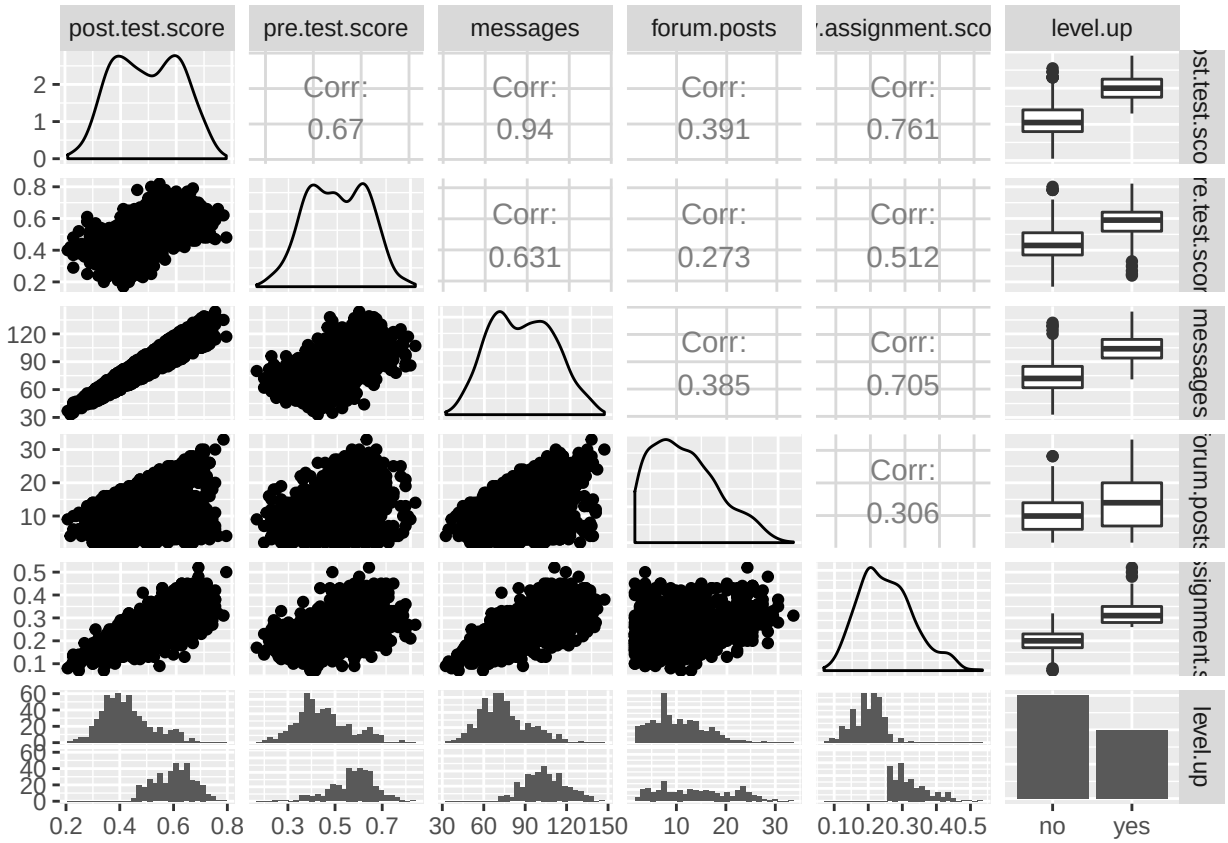
```
geog_its_raw <- read.csv("online.data.csv")
```

Visualization

```
#Start by creating histograms of the distributions for all variables (#HINT: look up "facet" in the ggp
distribution_df <- geog_its_raw %>%
                      select(-id) %>% mutate("messages" = as.numeric(messages),
                                             "forum.posts" = as.numeric(forum.posts),
                                             "level.up" = as.numeric(level.up)
                                             ) %>%
                      gather(key = "key",
                             value = "value")

ggplot(distribution_df) + aes(x = value) + geom_histogram(bins = 15) + facet_wrap(~key, scales = "free"
```




```
#Then visualize the relationships between variables
ggpairs(select(geog_its_raw, -id))
```




```
#Try to capture an intution about the data and the relationships
```

Aside from forum posts and average assignment score, all the continuous variables appear roughly normally distributed. Average assigment score has a slight positive skew, while forum posts has a pronounced positive skew.

All the variables are positively correlated with each other. THe lowest correlation is at .273 (forum posts and pre-test) and the strongest correlation is at .94 (messages and post-test). Most of the correlations between the variables are between 0.5 and 0.8, except the two listed above and the correlation between forum posts and average assignment score (0.306).

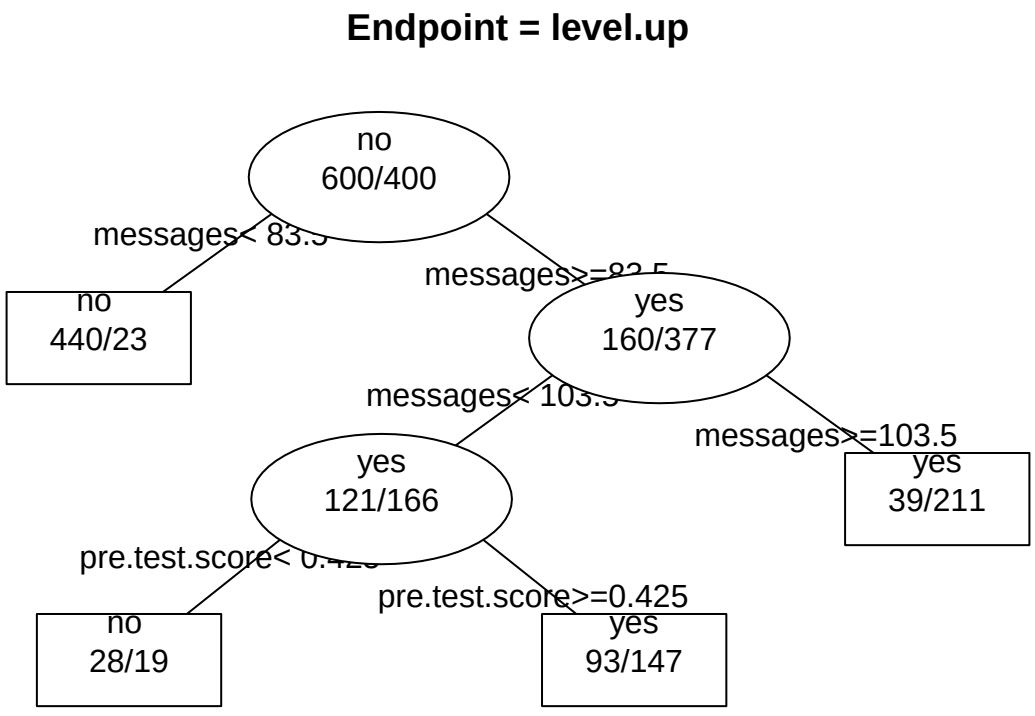
Level up appears to be related to all the variables except forum posts in the same way. Those who leveled up tended to be higher on all the other variables, especially average assignment score, where almost all those who did not level up had scores below 0.25, while almost all who did level up had scores above 0.25. For forum posts, those who did not level up appeared to have made less forum posts, while those who leveled up displayed a more even distribution in forum posts.

Classification tree

```
#Create a classification tree that predicts whether a student "levels up" in the online course using th
ctree_geog_its <- rpart(level.up ~ pre.test.score + messages + forum.posts,
                        data = geog_its_raw)

#Plot and generate a CP table for your tree
printcp(ctree_geog_its)
```

```
##
## Classification tree:
## rpart(formula = level.up ~ pre.test.score + messages + forum.posts,
##     data = geog_its_raw)
##
## Variables actually used in tree construction:
## [1] messages       pre.test.score
##
## Root node error: 400/1000 = 0.4
##
## n= 1000
##
##         CP nsplit rel error xerror     xstd
## 1 0.54250      0    1.0000 1.0000 0.038730
## 2 0.01125      1    0.4575 0.4650 0.030762
## 3 0.01000      3    0.4350 0.4875 0.031322
```

```r
#Pruning since gain to CP at 3rd split is not that much more, with no decrease in xerror
#ctree_geog_its <- prune.rpart(ctree_geog_its, cp = 0.01125)
#Not doing this after all since it kind of ruins the ROC exercise. With few splits there are few unique
#to choose from

#Plot
post(ctree_geog_its, filename = "")
```



```
#Generate a probability value that represents the probability that a student levels up based your class
geog_its_raw$pred <- predict(ctree_geog_its, type = "prob")[,2]

#Last class we used type = "class" which predicted the classification for us, this time we are using ty
#The predict() function with type = "prob" returns a named matrix - each column is the probablility of
```

Part II

Now you can generate the ROC curve for your model. You will need to install the package ROCR to do this.

```
#install.packages("ROCR")
library(ROCR)
```

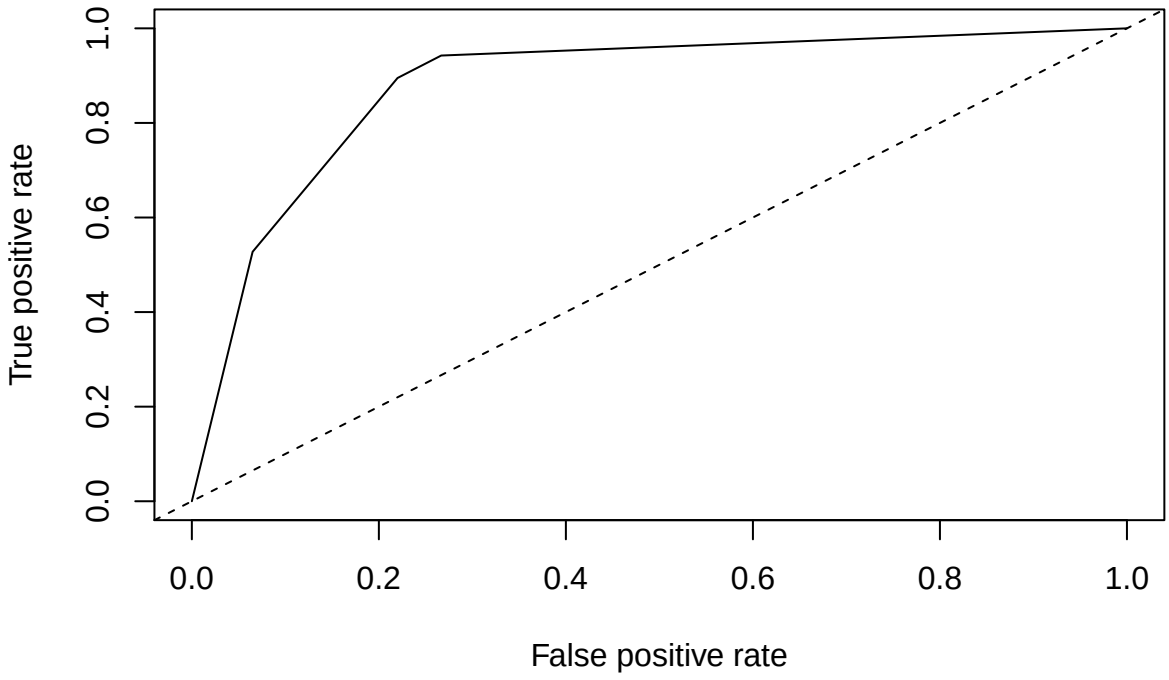
```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

Generating ROC curve

```
#Create prediction object. prediction() transforms the input data of predictions and actual labels into
pred.detail <- prediction(geog_its_raw$pred, geog_its_raw$level.up)

#performance() calculates performance metrics from prediction objects.
#"tpr" - true positive rate
#"fpr" - false positive rate
#These are the axes for the ROC curve
perf.detail <- (performance(pred.detail, "tpr", "fpr"))
plot(perf.detail)
abline(0, 1, lty = 2)
```




```
#Calculate the Area Under the Curve, which can be done using the performance() function
#performance() returns a class, which has slots. slot() extracts the item in a slot, here a list from t
#Classes and slots (referred to with @) work like lists, but have no indexes. See ?Classes for more inf
#unlist() is used to get the elements in the list that was in the slot object - here the AUC value
unlist(slot(performance(pred.detail, "auc"), "y.values"))
```

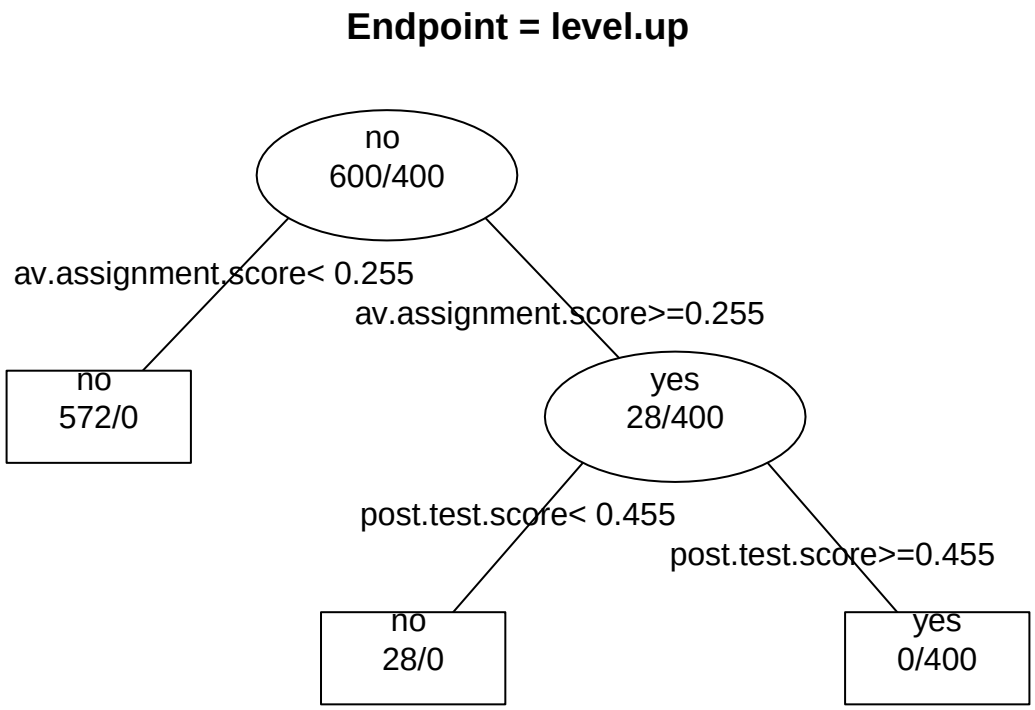
```
## [1] 0.8825125
```

```
#Now repeat this process, but using the variables you did not use for the previous model and compare th

#Create a classification tree that predicts whether a student "levels up" using different variables
ctree_geog_its2 <- rpart(level.up ~ post.test.score + av.assignment.score,
                         data = geog_its_raw)

#Plot and generate a CP table for your tree
printcp(ctree_geog_its2)
```

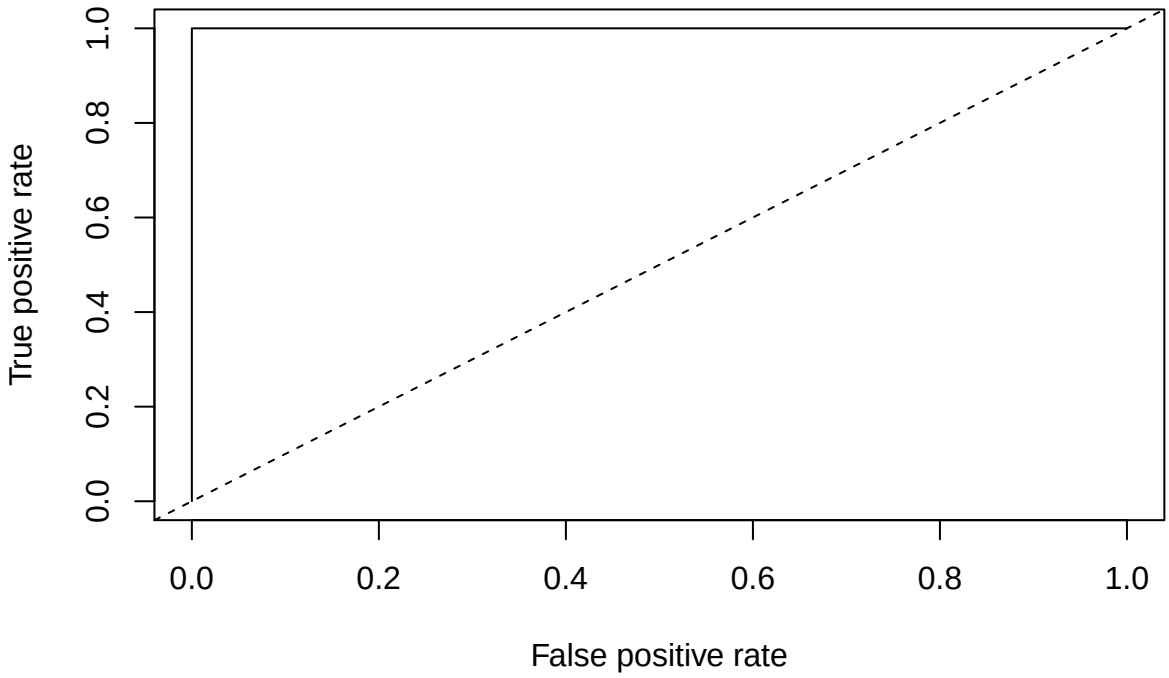
```
##
## Classification tree:
## rpart(formula = level.up ~ post.test.score + av.assignment.score,
##     data = geog_its_raw)
##
## Variables actually used in tree construction:
## [1] av.assignment.score post.test.score
##
```

```
## Root node error: 400/1000 = 0.4
##
## n= 1000
##
##     CP nsplit rel error xerror     xstd
## 1 0.93      0      1.00   1.00 0.038730
## 2 0.07      1      0.07   0.07 0.013042
## 3 0.01      2      0.00   0.00 0.000000
```

```
post(ctree_geog_its2, filename = "")
```




```
#Generate a probability value that represents the probability that a student levels up based your class
geog_its_raw$pred2 <- predict(ctree_geog_its2, type = "prob")[,2]

#Create prediction object. prediction() transforms the input data of predictions and actual labels into
pred.detail2 <- prediction(geog_its_raw$pred2, geog_its_raw$level.up)

#performance() calculates performance metrics from prediction objects.
plot(performance(pred.detail2, "tpr", "fpr"))
abline(0, 1, lty = 2)
```




```
#Calculate the Area Under the Curve
unlist(slot(performance(pred.detail2, "auc"), "y.values"))
```

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

The second model is superior. It has a perfect ROC curve, classification accuracy, and 10-fold CV error. This seems unrealistic for real life data, but is believable in this context. The ITS probably uses these same metrics to assess student mastery of content, and decide on whether the student should level up. Hence, any model including these two important determinants of levelling up should be highly accurate in predicting whether the ITS actually allowed the students to level up.

Part III

Thresholds

```
#Look at the ROC plot for your first model. Based on this plot choose a probability threshold that bala

#Create a data frame - I think a table would be useful to make the choice
pred_detail_df <- data.frame(pred.detail@cutoffs,
                             pred.detail@tp,
                             pred.detail@fp,
                             pred.detail@tn,
                             pred.detail@fn,
                             perf.detail@y.values,
```

```
                            perf.detail@x.values
                            )
colnames(pred_detail_df) <- c("Cutoffs", "TP", "FP", "TN", "FN", "TPR", "FPR")

#Print the DF - can compare with plot above
pred_detail_df
```

```
##      Cutoffs  TP  FP  TN  FN    TPR       FPR
## 1        Inf   0   0 600 400 0.0000 0.0000000
## 2 0.84400000 211  39 561 189 0.5275 0.0650000
## 3 0.61250000 358 132 468  42 0.8950 0.2200000
## 4 0.40425532 377 160 440  23 0.9425 0.2666667
## 5 0.04967603 400 600   0   0 1.0000 1.0000000
```

```
#Set treshold at 0.6125 - this gives a TPR of 0.895 and a FPR of 0.22
#On the graph this is right where the second bend is, which I think is a maximising point.
#This would be particularly true if we assume that detecting true positives (sucessfully predicting tha
#false positives (predicting that a student will level up when they didn't).
#This is the case here - a false positive is more punishing than a false negative (student leveled up,
#A false positive might deprive a student of support to level up, whereas a false negative is a happy s
geog_its_raw$threshold.pred1 <- geog_its_raw$pred > 0.6124

#Had to drop threshold because it seems $prob calculated earlier exceeded the float. See test below
# > 0.6125 >= 0.6125
# [1] TRUE
# > geog_its_raw$pred[100]
# [1] 0.6125
#> geog_its_raw$pred[100] >= 0.6125
# [1] FALSE

#Now generate three diagnostics
#Accuracy = (TP + TN)/Total
accuracy.model1 <- (pred_detail_df$TP[[3]] + pred_detail_df$TN[[3]])/1000
writeLines(c("Accuracy", accuracy.model1, ""))
```

```
## Accuracy
## 0.826
```

```
#Precision = TP/(TP + FP)
precision.model1 <- pred_detail_df$TP[[3]]/(pred_detail_df$TP[[3]] + pred_detail_df$FP[[3]])
writeLines(c("Precision", precision.model1, ""))
```

```
## Precision
## 0.730612244897959
```

```
#Recall = TP/(TP + FN)
recall.model1 <- pred_detail_df$TP[[3]]/(pred_detail_df$TP[[3]] + pred_detail_df$FN[[3]])
writeLines(c("Recall", recall.model1, ""))
```

```
## Recall
## 0.895
```

```r
#Finally, calculate Kappa for your model according to:

#First generate the table of comparisons
table1 <- table(geog_its_raw$level.up, geog_its_raw$threshold.pred1)
table1
```

```
##
##       FALSE TRUE
##   no    468  132
##   yes    42  358
```

```r
#Convert to matrix
matrix1 <- as.matrix(table1)

#Calculate kappa
kappa(matrix1, exact = TRUE)/kappa(matrix1)
```

```
## [1] 0.9944954
```

```r
#Now choose a different threshold value and repeat these diagnostics. What conclusions can you draw abo
geog_its_raw$threshold.pred2 <- geog_its_raw$pred > 0.8439

#Now generate three diagnostics
#Accuracy = (TP + TN)/Total
accuracy.model2 <- (pred_detail_df$TP[[2]] + pred_detail_df$TN[[2]])/1000
writeLines(c("Accuracy", accuracy.model2, ""))
```

```
## Accuracy
## 0.772
```

```r
#Precision = TP/(TP + FP)
precision.model2 <- pred_detail_df$TP[[2]]/(pred_detail_df$TP[[2]] + pred_detail_df$FP[[2]])
writeLines(c("Precision", precision.model2, ""))
```

```
## Precision
## 0.844
```

```r
#Recall = TP/(TP + FN)
recall.model2 <- pred_detail_df$TP[[2]]/(pred_detail_df$TP[[2]] + pred_detail_df$FN[[2]])
writeLines(c("Recall", recall.model2, ""))
```

```
## Recall
## 0.5275
```

```r
#Finally, calculate Kappa for your model according to:

#First generate the table of comparisons
table2 <- table(geog_its_raw$level.up, geog_its_raw$threshold.pred2)
table2
```

```
##
##       FALSE TRUE
##   no    561   39
##   yes   189  211
```

```
#Convert to matrix
matrix2 <- as.matrix(table2)

#Calculate kappa
kappa(matrix2, exact = TRUE)/kappa(matrix2)
```

```
## [1] 1.040758
```

```
#How did I arrive at a kappa of more than 1?
#Time to do manual calculations
#First model
p01 <- (468 + 358)/1000
pe1 <- (((468 + 132)/1000) * ((468 + 42)/1000)) + (((42 + 358)/1000) * ((132 + 358)/1000))
manual_kappa1 <- (p01 - pe1)/(1 - pe1)
writeLines(c("Cohen's Kappa for the first model = ", manual_kappa1))
```

```
## Cohen's Kappa for the first model =
## 0.650602409638554
```

```
#2nd model
p02 <- (561 + 211)/1000
pe2 <- (((561 + 39)/1000) * ((561 + 189)/1000)) + (((189 + 211)/1000) * ((39 + 211)/1000))
manual_kappa2 <- (p02 - pe2)/(1 - pe2)
writeLines(c("Cohen's Kappa for the second model = ", manual_kappa2))
```

```
## Cohen's Kappa for the second model =
## 0.493333333333333
```

```
#Testing with psych package
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
cohen.kappa(matrix1)
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa   0.6     0.65   0.7
## weighted kappa     0.6     0.65   0.7
##
##  Number of subjects = 1000
```

```r
cohen.kappa(matrix2)
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa  0.44     0.49  0.55
## weighted kappa    0.44     0.49  0.55
##
##  Number of subjects = 1000
```

Conclusions that can be drawn:

1. The first threshold has a higher kappa, accuracy, and recall.
2. The second threshold has a higher precision.
3. The first threshold is likely the better one to use. It shows more agreement between model and data (kappa), has a higher accuracy and recall. Although it has a lower precision, the reduction in precision (driven mostly by a lower false positive rate) is not sufficient to justify the tradeoff to accuracy and recall (driven mostly by the decreased true positive rate/increased false negative rate, which is larger than the decrease in FPR). The overestimation of students who might not level up would be a strain on resources.

To Submit Your Assignment

Please submit your assignment by first "knitting" your RMarkdown document into an html file and then commit, push and pull request both the RMarkdown file and the html file.