Final Project (CIS-544)

Javier Rojas. jrojas@stu.edu

Due 12/12/16 by 6 PM EST

Introduction and Motivation

- As with any school or higher education institution, one of their key goals is to continously find various alternatives that may increase their student retention and graduation rate.
- With the current advances in information technology, research fields incompassing Business Intelligence (BI) and Data Mining (DM) have opened windows of opportunities to effectively model student performance, which can lead to early detection of students who may be prone to falter in their academic discipline.
- The tools associated with these fields can consequently direct us to the discovery of key features or characteristics that may hinder students from achieving optimal academic performance.
- As such, all the valuable information acquired from utilizing such tools can assist educators in determining appropriate methods which can aid falty students in surpassing their difficulties and continue smoothly toward the completion of their intended academic path.

Data Description

- In keeping with the same mindset explained above, for this project, I focused on conducting a similar investigative approach exceuted by Cortez and Silva in their work titled "Using Data Mining to Predict Secondary School Student Performance".
- In such work, they studied student data collected during the 2005-2006 school year from two public Portuguese secondary schools with regards to the students' performances in two distinct courses, Mathematics (MAT) and Portuguese (POR).
- The information contained within this dataset provided insight into each student's corresponding grades, demographic, social and school related features. Specifically, below are the attributes contained within both the `student-mat.csv` (Math course) and `student-por.csv` (Portuguese language course) datasets:

1) school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2) sex - student's sex (binary: 'F' - female or 'M' - male)
3) age - student's age (numeric: from 15 to 22)
4) address - student's home address type (binary: 'U' - urban or 'R' - rural)
5) famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6) Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7) Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education.)
8) Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education.)
9) Mjob - mother's job (nominal: teacher, health care related, civil services (e.g. administrative or police), at home or other.)
10) Fjob - father's job (nominal: teacher, health care related, civil services (e.g. administrative or police), at home or other.)
11) reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')

12) guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13) traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14) studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15) failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16) schoolsup - extra educational support (binary: yes or no)
17) famsup - family educational support (binary: yes or no)
18) paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19) activities - extra-curricular activities (binary: yes or no)
20) nursery - attended nursery school (binary: yes or no)
21) higher - wants to take higher education (binary: yes or no)
22) internet - Internet access at home (binary: yes or no)
23) romantic - with a romantic relationship (binary: yes or no)
24) famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25) freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26) goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27) Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28) Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29) health - current health status (numeric: from 1 - very bad to 5 - very good)
30) absences - number of school absences (numeric: from 0 to 93)
31) G1 - first period grade (numeric: from 0 to 20)
32) G2 - second period grade (numeric: from 0 to 20)
33) G3 - final grade (numeric: from 0 to 20, output target)

- For the purposes of this project, I focused solely on the Mathematics (MAT) dataset. In regards to this dataset, I modeled the Math course under three DM goals:

i) binary classification (pass/fail)
ii) classification with five levels (from I very good or excellent to V - insufficient); and
iii) regression, with a numeric output that ranges between zero (0%) and twenty (100%).

- For each of these approaches, three input setups (e.g. with and without the school period grades) and four DM algorithms (e.g. Decision Trees, Random Forest, Support Vector Machine and 10-Fold Cross Validation) were tested. In addition, an explanatory analysis was performed over the best models in order to identify the most relevant features.

- Specifically, the input configurations consisted of the following three scenarios:

1) A - with all variables from dataset except G3 (the output);
2) B - similar to A but without G2 (the second period grade); and
3) C - similar to B but without G1 (the first period grade).

Methods

- As can be concluded from the above mentioned procedures, this project was divided into two general forms of analysis, which are classification and regression analysis. In both cases, these analyses are implemented when dealing with supervised learning scenarios, which consist of exposing your established model to a set of labeled training examples from your dataset in order for it to ultimately correctly determine the class labels for unseen instances (testing set).

- With regards to classification models, I examined their performance based on their associated Percentage of Correct Classifications (PCC), while in regression the Root Mean Squared Error (RMSE - variance of the residuals (errors) - indicator of how close the observed data points are to the model's predicted

values) is a popular metric. A high PCC (i.e. near 100%) suggests a good classifier, while a regressor should present a low global error (i.e. RMSE close to zero).

10-Fold Cross Validation

- In the case of the 10-fold cross validation, we are randomly partitioning the data into k parts or "folds", set one fold aside for testing, train a model on the remaining *k-1* folds and evaluate it on the test fold. This process is repreated k times until each fold has been used for testing once.
- For this work, 20 runs of a 10-fold cross-validation were applied to each configuration. Under such scheme, for a given run the data is randomly divided in 10 subsets of equal size. Sequentially, one different subset is tested (with 10% of the data) and the remaining data used to fit the DM technique. At the end of this process, the evaluated test set contains the whole dataset, although 10 variations of the same DM model are used to create the predictions.
- In comparison to the other DM algorithms with regards to classification, this algorithm's performance in general was better if pruning is not executed with the Decision Tree and Random Forest models. Below are the results of the 10-fold Cross Validation for all possible classification cases.

A-Setup

- 10-fold Cross Validation (Binary)

```
# read the caret library.
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
# load the data:
MAT <- read.table("student-mat.csv",sep=";",header=TRUE)
attach(MAT)
# Binary response
Score <- factor(ifelse(G3 >= 10, 1, 0),
                labels = c("fail", "pass"))
MAT <- data.frame(MAT, Score)
```

```
# define training control
set.seed(1337)
train_control<- trainControl(method="cv", number=10, repeats=20, savePredictions = TRUE)
# train the model
model<- train(Score ~. - G3, data=MAT, trControl=train_control, method="rpart")
```

```
## Loading required package: rpart
```

```
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
##  33 predictor
##   2 classes: 'fail', 'pass'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 356, 355, 356, 356, 355, 355, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.8937821  0.7518868
##   0.01282051  0.9111538  0.8009587
##   0.75384615  0.7349359  0.2222032
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.01282051.
```

```
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 41)
##
##               Overall
## G2            100.000
## G1             64.455
## failures       15.641
## goout           5.561
## age             3.825
## health          0.000
## famrel          0.000
## freetime        0.000
## Fjobother       0.000
## paidyes         0.000
## romanticyes     0.000
## nurseryyes      0.000
## addressU        0.000
## sexM            0.000
## Fjobservices    0.000
## higheryes       0.000
## Fedu            0.000
## schoolsupyes    0.000
## guardianother   0.000
## internetyes     0.000
```

- 10-fold Cross Validation (5-level classification)

```
rm(list = ls())

# load the data:
MAT <- read.table("student-mat.csv",sep=";",header=TRUE)
# 5-level response
Score <- ifelse(G3 >= 0 & G3 <= 9, "fail", ifelse(G3 >= 10 & G3 <= 11, "sufficient", ifelse(G3 >= 12 & G
MAT <- data.frame(MAT, Score)
```

```
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
```

```
##  33 predictor
##   5 classes: 'excellent/very good', 'fail', 'good', 'satisfactory', 'sufficient'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 355, 356, 356, 355, 356, 356, ...
## Resampling results across tuning parameters:
##
##   cp         Accuracy   Kappa
##   0.1018868  0.6834615  0.58004043
##   0.2150943  0.5341667  0.35180059
##   0.2679245  0.3818590  0.09725188
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.1018868.
```

```
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 41)
##
##               Overall
## G2            100.000
## G1             69.608
## failures       10.419
## Medu            6.195
## goout           3.543
## Dalc            2.641
## schoolsupyes    2.401
## freetime        0.000
## Walc            0.000
## Fjobother       0.000
## activitiesyes   0.000
## famrel          0.000
## higheryes       0.000
## age             0.000
## sexM            0.000
## Fjobservices    0.000
## internetyes     0.000
## Fedu            0.000
## famsupyes       0.000
## guardianother   0.000
```

B-Setup

- 10-fold Cross Validation (Binary)

```
# define training control
set.seed(1337)
train_control<- trainControl(method="cv", number=10, repeats=20, savePredictions = TRUE)
# train the model
model<- train(Score ~. - G2 - G3, data=MAT, trControl=train_control, method="rpart")
```

```r
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
##  33 predictor
##   2 classes: 'fail', 'pass'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 356, 355, 356, 356, 355, 355, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03846154  0.8304487  0.6104561
##   0.06153846  0.8152564  0.6228311
##   0.43846154  0.7187821  0.2741535
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03846154.
```

```r
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 40)
##
##               Overall
## G1            100.000
## failures       31.491
## age            17.604
## absences       10.049
## guardianother   7.549
## goout           7.296
## higheryes       4.789
## romanticyes     2.713
## Walc            0.000
## health          0.000
## Fjobother       0.000
## activitiesyes   0.000
## Dalc            0.000
## internetyes     0.000
## addressU        0.000
## sexM            0.000
## Fjobservices    0.000
## famrel          0.000
## Fedu            0.000
## famsupyes       0.000
```

- 10-fold Cross Validation (5-level classification)

```r
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
##  33 predictor
##   5 classes: 'excellent/very good', 'fail', 'good', 'satisfactory', 'sufficient'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 355, 356, 356, 355, 356, 356, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.07924528  0.5064103  0.3423619
##   0.08301887  0.4835897  0.3159242
##   0.20377358  0.4028846  0.1684877
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.07924528.
```

```
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 40)
##
##               Overall
## G1            100.000
## failures       20.564
## absences       10.110
## Medu            9.933
## goout           5.620
## Dalc            3.312
## famrel          0.000
## freetime        0.000
## Fjobother       0.000
## paidyes         0.000
## romanticyes     0.000
## nurseryyes      0.000
## age             0.000
## sexM            0.000
## Fjobservices    0.000
## higheryes       0.000
## Fedu            0.000
## schoolsupyes    0.000
## guardianother   0.000
## internetyes     0.000
```

C-Setup

- 10-fold Cross Validation (Binary)

```
# define training control
set.seed(1337)
train_control<- trainControl(method="cv", number=10, repeats=20, savePredictions = TRUE)
```

```r
# train the model
model<- train(Score ~. - G1 - G2 - G3, data=MAT, trControl=train_control, method="rpart")
```

```r
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
##  33 predictor
##   2 classes: 'fail', 'pass'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 356, 355, 356, 356, 355, 355, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02115385  0.6857692  0.1928890
##   0.03076923  0.7085897  0.2469571
##   0.16153846  0.6809615  0.1107344
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03076923.
```

```r
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 39)
##
##               Overall
## failures       100.00
## goout           35.55
## age             24.46
## higheryes       23.34
## absences        21.61
## freetime         0.00
## Dalc             0.00
## Fjobother        0.00
## paidyes          0.00
## famrel           0.00
## nurseryyes       0.00
## addressU         0.00
## sexM             0.00
## Fjobservices     0.00
## internetyes      0.00
## Fedu             0.00
## schoolsupyes     0.00
## guardianother    0.00
## romanticyes      0.00
## Mjobhealth       0.00
```

- 10-fold Cross Validation (5-level classification)

```r
# Find PCC (Percent Correct Classification)
print(model)
```

```
## CART
##
## 395 samples
##  33 predictor
##   5 classes: 'excellent/very good', 'fail', 'good', 'satisfactory', 'sufficient'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 355, 356, 356, 355, 356, 356, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01886792  0.3617949  0.11912768
##   0.03144654  0.3441026  0.08354359
##   0.03396226  0.3188462  0.02255030
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.01886792.
```

```r
# Most important variables
varImp(model)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 39)
##
##               Overall
## absences       100.00
## failures        86.49
## Medu            80.24
## Walc            59.69
## Dalc            51.11
## guardianother   41.86
## goout           29.41
## schoolsupyes    24.03
## higheryes       22.19
## Fjobother        0.00
## nurseryyes       0.00
## health           0.00
## romanticyes      0.00
## age              0.00
## sexM             0.00
## Fjobservices     0.00
## famrel           0.00
## Fedu             0.00
## paidyes          0.00
## traveltime       0.00
```

Random Forest

- Now, in terms of regression analysis, the best scenario would be to implement a random forest algorithm based on its overall performance with respect said analysis and compared to the results of the other algorithms.
- With random forest, we are essentially building various decision trees, but when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. With these bushy trees, the random forest algorithm later takes their average in order to reduce the variance.

A-Setup

- Regression (Random Forest)

```
rm(list = ls())
MAT <- read.table("student-mat.csv",sep=";",header=TRUE)

# Train and test sets
set.seed(1337)
ind <- sample(2, nrow(MAT), replace = TRUE, prob = c(0.8, 0.2))
train <- MAT[ind==1,]
test <- MAT[ind==2,]
```

```
# Create random forest: G3 as a function of all variables
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
rf.MAT = randomForest(G3 ~ ., data = train)
rf.MAT
```

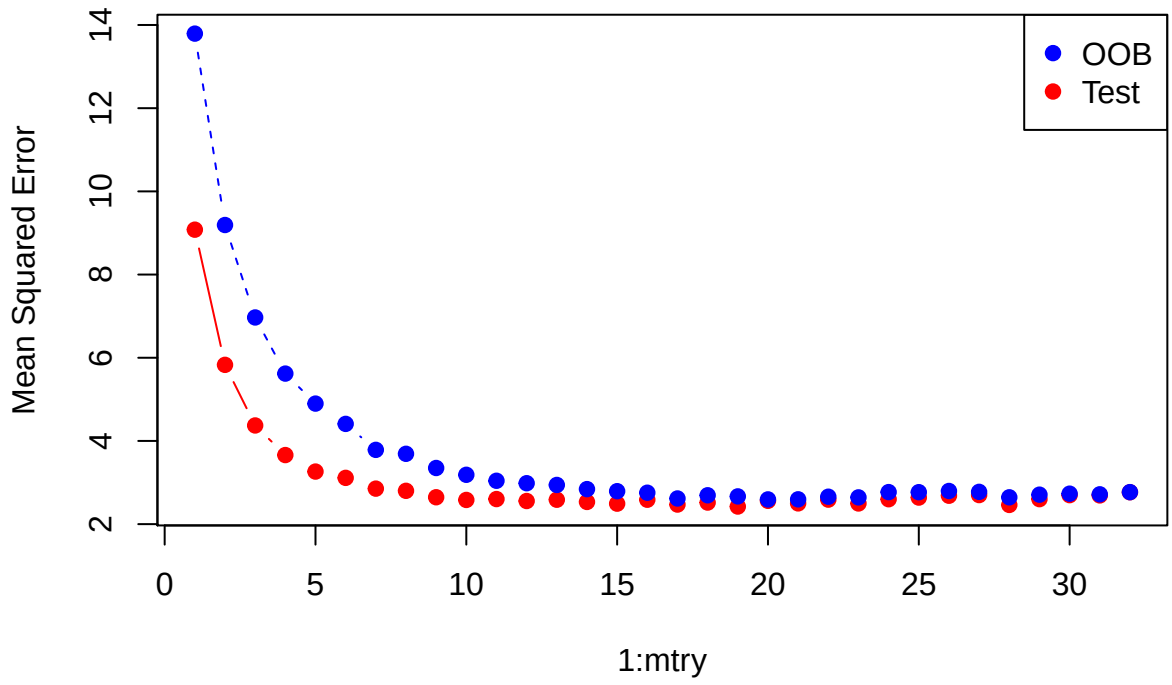
```
##
## Call:
##  randomForest(formula = G3 ~ ., data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 3.283751
##                     % Var explained: 85.17
```

```
# Out-of-bag error vectors
oob.err = double(32)
# Test error vector
test.err = double(32)
# Run every value of mtry
for (mtry in 1:32) {
# fit a random forest in the training data with 400 trees
```

```
fit = randomForest(G3 ~ ., data = train, mtry = mtry, ntree = 400)
# Record OOB error
oob.err[mtry] = fit$mse[400]
# Calculate predictions
pred = predict(fit, test)
# Record test error
test.err[mtry] = with(test, mean((G3 - pred)^2))
# Display the value of mtry
cat(mtry, " ")
}
```

```
## 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
```

```
# Create plot using function matplot (since we have two
# vectors: test.error and oob.error) type = 'b' plots both
# points
matplot(1:mtry, cbind(test.err, oob.err), pch = 19, col = c("red",
"blue"), type = "b", ylab = "Mean Squared Error")
# Legend in the top-right corner
legend("topright", legend = c("OOB", "Test"), pch = 19, col = c("blue",
"red"))
```




```
# Find MSE
which.min(oob.err)
```

```
## [1] 20
```

```
sqrt(oob.err[20])
```

```
## [1] 1.610602
```

```
which.min(test.err)
```

```
## [1] 19
```

```
sqrt(test.err[19])
```

```
## [1] 1.556641
```

```
# Extract tree structure of randomForest object with least oob.err
tree <- getTree(fit, k=20, labelVar = TRUE)
# Output the variables used in such object
unique(na.omit(tree$`split var`))
```

```
##  [1] G2         absences   reason     Mjob       age        failures
##  [7] schoolsup  freetime   goout      Fedu       activities famrel
## [13] famsize    paid       Walc       studytime  G1         guardian
## [19] Fjob       traveltime
## 20 Levels: absences activities age failures famrel famsize Fedu ... Walc
```

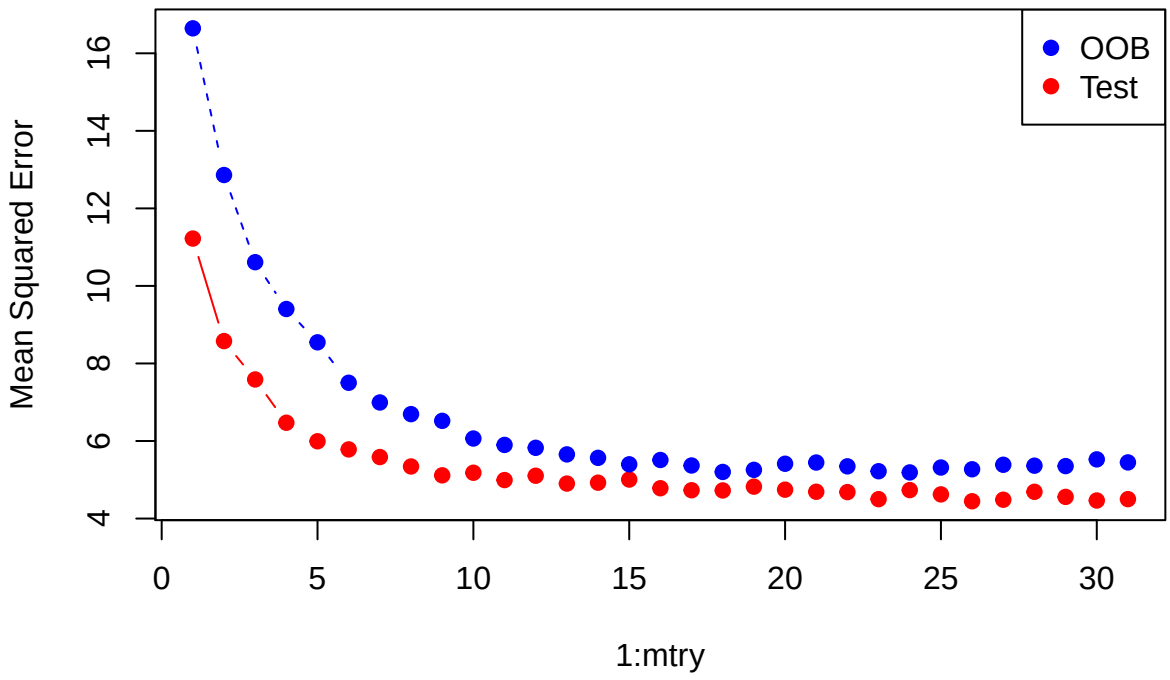
B-Setup

- Regression (Random Forest)

```
# Create random forest: G3 as a function of all variables except G2
rf.MAT = randomForest(G3 ~ . - G2, data = train)
rf.MAT
```

```
##
## Call:
##  randomForest(formula = G3 ~ . - G2, data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 6.101527
##                     % Var explained: 72.44
```

```
## 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
```




```
# Find MSE
which.min(oob.err)
```

```
## [1] 24
```

```
sqrt(oob.err[20])
```

```
## [1] 2.326636
```

```
which.min(test.err)
```

```
## [1] 26
```

```
sqrt(test.err[31])
```

```
## [1] 2.121357
```

```
# Extract tree structure of randomForest object with least oob.err
tree <- getTree(fit, k=20, labelVar = TRUE)
# Output the variables used in such object
unique(na.omit(tree$`split var`))
```

```
##  [1] G1         absences   failures   schoolsup  activities reason
##  [7] famrel     Mjob       Medu       age        goout      Fedu
## [13] guardian   famsize    sex        freetime   paid       famsup
## [19] Fjob       address    health     traveltime Walc       Dalc
## [25] nursery    studytime  school
## 27 Levels: absences activities address age Dalc failures ... Walc
```

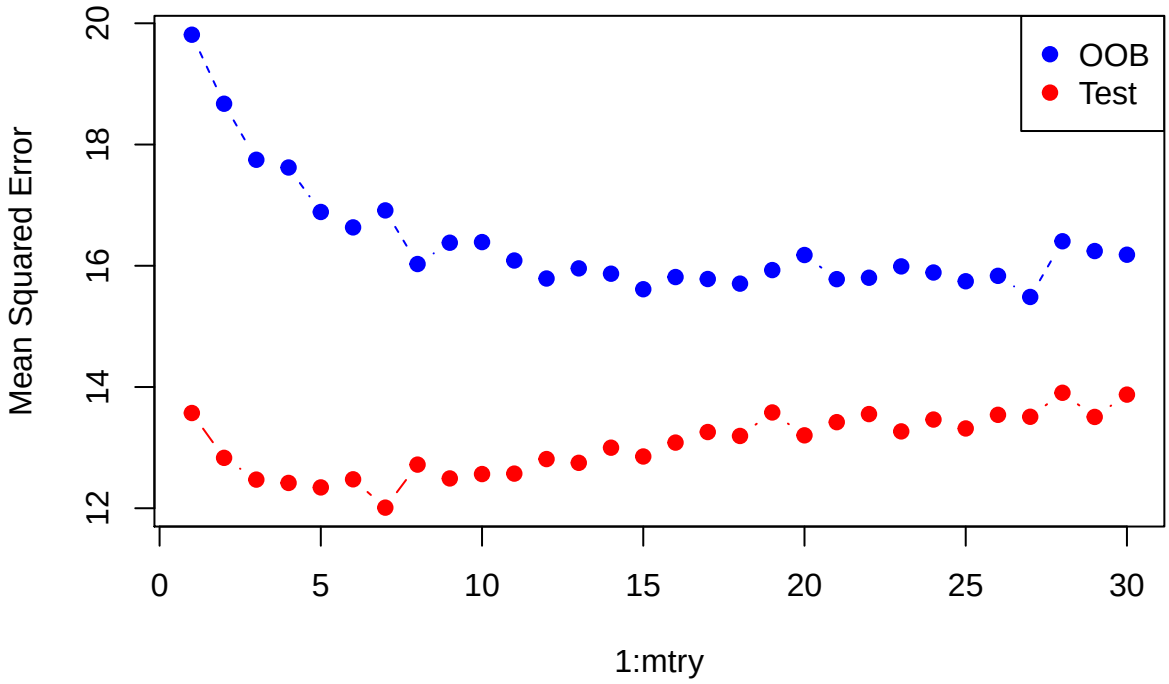
C-Setup

- Regression (Random Forest)

```
# Create random forest: G3 as a function of all variables except G1 and G2
rf.MAT = randomForest(G3 ~ . - G1 - G2, data = train)
rf.MAT
```

```
##
## Call:
##  randomForest(formula = G3 ~ . - G1 - G2, data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 16.34387
##                     % Var explained: 26.18
```

```
## 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
```




```
# Find MSE
which.min(oob.err)
```

```
## [1] 27
```

```
sqrt(oob.err[20])
```

```
## [1] 4.022133
```

```r
which.min(test.err)
```

```
## [1] 7
```

```r
sqrt(test.err[5])
```

```
## [1] 3.513371
```

```r
# Extract tree structure of randomForest object with least oob.err
tree <- getTree(fit, k=20, labelVar = TRUE)
# Output the variables used in such object
unique(na.omit(tree$`split var`))
```

```
##  [1] failures   higher     absences   Medu       Dalc       Mjob
##  [7] schoolsup  guardian   Fjob       goout      studytime  famrel
## [13] freetime   romantic   reason     age        famsup     Fedu
## [19] address    school     famsize    health     Walc       paid
## [25] traveltime Pstatus
## 26 Levels: absences address age Dalc failures famrel famsize ... Walc
```

Conclusion and Future Work

- As can be witnessed and confirmed from the results, we can be more reliant on achieving overall best results from utilizing 20 runs of a 10-fold cross validation model for classification and randomForest for the case of regression.
- For reference, I will include an "Appendix" section where the viewer may find the implementation of the previously mentioned DM algorithms so as to compare the results of those algorithms with the ones described above.
- As an extra note, for future work, I may execute this same investigation actually for the upcoming Spring semester with regards to STU student data and see the results of these algorithms with this dataset so as to maybe pinpoint areas where STU's approach towards its students may improve.

References

Cortez, P., & Silva, A. (n.d.). Using Data Mining To Predict Secondary School Student Performance. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.540.8151&rep=rep1&type=pdf

James, G. (2013). An introduction to statistical learning: With applications in R.