# Midterm-2 Project Portion - Instruction

First and last name: Ryan Hilton // Pair's first and last name: Avery Girsky

Submission Date: April 8th, 2021

---

### Midterm-2 Project Instruction

In `Midterm-1 Project`, you have built predictive models using train and test data sets about college students' academic performances and retention status. You fitted four regression models on **Term.GPA** and four classification models on **Persistence.NextYear**. the lowest test score of $MSE_{test}$ achieved on the regression problem was .991 using a simple linear regression, and the highest `accuracy` and `F1` scores obtained were 91.15% and 95.65%, respectively, with the fit of a multiple logistic regression model (equivalently, LDA and QDA give similar performances). Let's call these scores as baseline test scores.

In `Midterm-2 Project`, you will use tree-based methods (trees, random forests, boosting) and artificial neural networks (Modules 5, 6, and 7) to improve the baseline results. There is no any answer key for this midterm: your efforts and justifications will be graded, pick one favorite optimal tree-based method and one optimal ANN architecture for each regression and classification problem (a total of two models for classification and two models for regression), and fit and play with hyperparameters until you get satisfactory improvements in the test data set.

Keep in mind that $Persistence.NextYear$ is not included in as predictor the regression models so use all the predictors except that on the regression. For the classification models, use all the predictors including the term gpa.

First of all, combine the train and test data sets, create dummies for all categorical variables, which include `Entry_Term`, `Gender`, and `Race_Ethc_Visa`, so the data sets are ready to be separated again as train and test. (Expect help on this portion!) You will be then ready to fit models.

---

## A. Improving Regression Models - 15 pts

- Explore tree-based methods, choose the one that is your favorite and yielding optimal results, and then search for one optimal ANN architecture for the regression problem (so two models to report). Fit and make sophisticated decisions by justifying and writing precisely. Report `the test MSE` results in a comparative table along with the methods so the grader can capture all your efforts on building various models in one table.

## B. Improving Classification Models - 20 pts

- Explore tree-based methods, choose the one that is your favorite and yielding optimal results, and then search for one optimal ANN architecture for the classification problem (so two models to report). Fit and make sophisticated decisions by justifying and writing precisely. Report `the test accuracy` and `the test F1` results in a comparative table along with the methods so the grader can capture all your efforts in one table.

## C. Importance Analyses - 15 pts

- Part a. Perform an importance analysis on the best regression model: which three predictors are most important or effective to explain the response variable? Find the relationship and dependence of these predictors with the response variable. Include graphs and comments.
- Part b. Perform an importance analysis on the best classification model: which three predictors are most important or effective to explain the response variable? Find the relationship and dependence of these predictors with the response variable. Include graphs and comments.
- Part c. Write a conclusion paragraph. Evaluate overall what you have achieved. Did the baselines get improved? Why do you think the best model worked well or the models didn't work well? How did you handle issues? What could be done more to get `better` and `interpretable` results? Explain with technical terms.

---

## Project Evaluation

The submitted project report will be evaluated according to the following criteria:

1. All models in the instruction used correctly
2. Completeness and novelty of the model fitting
3. Techniques and theorems of the methods used accurately
4. Reflection of in-class lectures and discussions
5. Achieved reasonable/high performances; insights obtained (patterns of variables)
6. Clear and minimalist write-ups

If the response is not full or not reflecting the correct answer as expected, you may still earn partial points. For each part or model, I formulated this `partial points` as this:

- 20% of pts: little progress with some minor solutions;
- 40% of pts: major calculation mistake(s), but good work, ignored important pieces;
- 60-80% of pts: correct method used, but minor mistake(s).

Additionally, a student who will get the highest performances from both problems in the class (`minimum test MSE` from the regression model and `highest F1` from the classification model) will get a BONUS (up to +2 pts). Just follow up when you think you did good job!

---

## Tips

- `Term.gpa` is an aggregated gpa up until the current semester, however, this does not include this current semester. In the modeling of `gpa`, include all predictors except `persistent`.
- The data shows the `N.Ws`, `N.DFs`, `N.As` as the number of courses withdrawn, D or Fs, A's respectively in the current semester.
- Some rows were made synthetic so may not make sense: in this case, feel free to keep or remove.
- It may be poor to find linear association between gpa and other predictors (don't include `persistent` in `gpa` modeling).
- Scatterplot may mislead since it doesn't show the density.
- You will use the test data set to asses the performance of the fitted models based on the train data set.
- Implementing 5-fold cross validation method while fitting with train data set is strongly suggested.
- You can use any packs (`caret`, `Superml`, `rpart`, `xgboost`, or visit to search more) as long as you are sure what it does and clear to the grader.
- Include helpful and compact plots with titles.
- Keep at most 4 decimals to present numbers and the performance scores.
- When issues come up, try to solve and write up how you solve or can't solve.
- Check this part for updates: the instructor puts here clarifications as asked.

---

## Your Solutions

```r
train <- read.csv("StudentDataTrain.csv")
test <- read.csv("StudentDataTest.csv")

full.data = rbind(train, test)
summary(full.data)
```

```
##  Race_Ethc_Visa        Gender              HSGPA          SAT_Total
##  Length:7435        Length:7435        Min.   : 50.00   Min.   : 900
##  Class :character   Class :character   1st Qu.: 67.00   1st Qu.:1084
##  Mode  :character   Mode  :character   Median : 76.00   Median :1255
##                                        Mean   : 76.51   Mean   :1254
##                                        3rd Qu.: 86.00   3rd Qu.:1425
##                                        Max.   :100.00   Max.   :1600
##                                        NA's   :17       NA's   :12
##    Entry_Term      Term.GPA     Persistence.NextYear N.RegisteredCourse
##  Min.   :2131   Min.   :0.500   Min.   :0.0000       Min.   : 1.00
##  1st Qu.:2131   1st Qu.:1.380   1st Qu.:1.0000       1st Qu.: 2.00
##  Median :2141   Median :2.250   Median :1.0000       Median : 3.00
##  Mean   :2141   Mean   :2.246   Mean   :0.8195       Mean   : 3.58
##  3rd Qu.:2151   3rd Qu.:3.120   3rd Qu.:1.0000       3rd Qu.: 5.00
##  Max.   :2151   Max.   :4.000   Max.   :1.0000       Max.   :11.00
##
##       N.Ws           N.DFs             N.As        N.PassedCourse
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 1.000
##  Median :0.000   Median :0.0000   Median :1.0000   Median : 2.000
##  Mean   :0.545   Mean   :0.6256   Mean   :0.8469   Mean   : 2.409
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 3.000
##  Max.   :6.000   Max.   :7.0000   Max.   :7.0000   Max.   :11.000
##
##  N.CourseTaken    Perc.PassedEnrolledCourse   Perc.Pass        Perc.Withd
##  Min.   : 0.000   Min.   :0.0000            Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 2.000   1st Qu.:0.5000            1st Qu.:0.6000   1st Qu.:0.0000
##  Median : 3.000   Median :0.7143            Median :1.0000   Median :0.0000
##  Mean   : 3.035   Mean   :0.6700            Mean   :0.7882   Mean   :0.1541
##  3rd Qu.: 4.000   3rd Qu.:1.0000            3rd Qu.:1.0000   3rd Qu.:0.2500
##  Max.   :11.000   Max.   :1.0000            Max.   :1.0000   Max.   :1.0000
##                                             NA's   :214
##  N.GraduateCourse FullTimeStudent
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.6173   Mean   :0.5787
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :5.0000   Max.   :1.0000
##
```

```r
full.data$Gender <- ifelse(full.data$Gender == "Male",
    1, 0)  #male = 1, female = 0
full.data$Race_Ethc_Visa <- ifelse(full.data$Race_Ethc_Visa ==
```

```r
    "Afram", 1, ifelse(full.data$Race_Ethc_Visa ==
    "Asian", 2, ifelse(full.data$Race_Ethc_Visa ==
    "Hispanic", 3, ifelse(full.data$Race_Ethc_Visa ==
    "Multi", 4, 0))))

full.data$Entry_Term <- ifelse(full.data$Entry_Term ==
    2131, 0, 1)  #2131 = 0, 2141 = 1
```

```r
summary(full.data)
```

```
##  Race_Ethc_Visa     Gender            HSGPA          SAT_Total   
##  Min.   :0.000   Min.   :0.0000   Min.   : 50.00   Min.   : 900  
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 67.00   1st Qu.:1084  
##  Median :2.000   Median :0.0000   Median : 76.00   Median :1255  
##  Mean   :1.966   Mean   :0.4995   Mean   : 76.51   Mean   :1254  
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.: 86.00   3rd Qu.:1425  
##  Max.   :4.000   Max.   :1.0000   Max.   :100.00   Max.   :1600  
##                  NA's   :3        NA's   :17       NA's   :12    
##    Entry_Term         Term.GPA     Persistence.NextYear N.RegisteredCourse
##  Min.   :0.0000   Min.   :0.500   Min.   :0.0000       Min.   : 1.00     
##  1st Qu.:0.0000   1st Qu.:1.380   1st Qu.:1.0000       1st Qu.: 2.00     
##  Median :1.0000   Median :2.250   Median :1.0000       Median : 3.00     
##  Mean   :0.7197   Mean   :2.246   Mean   :0.8195       Mean   : 3.58     
##  3rd Qu.:1.0000   3rd Qu.:3.120   3rd Qu.:1.0000       3rd Qu.: 5.00     
##  Max.   :1.0000   Max.   :4.000   Max.   :1.0000       Max.   :11.00     
##                                                                          
##       N.Ws           N.DFs             N.As         N.PassedCourse  
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000  
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 1.000  
##  Median :0.000   Median :0.0000   Median :1.0000   Median : 2.000  
##  Mean   :0.545   Mean   :0.6256   Mean   :0.8469   Mean   : 2.409  
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 3.000  
##  Max.   :6.000   Max.   :7.0000   Max.   :7.0000   Max.   :11.000  
##                                                                     
##  N.CourseTaken    Perc.PassedEnrolledCourse   Perc.Pass        Perc.Withd    
##  Min.   : 0.000   Min.   :0.0000            Min.   :0.0000   Min.   :0.0000  
##  1st Qu.: 2.000   1st Qu.:0.5000            1st Qu.:0.6000   1st Qu.:0.0000  
##  Median : 3.000   Median :0.7143            Median :1.0000   Median :0.0000  
##  Mean   : 3.035   Mean   :0.6700            Mean   :0.7882   Mean   :0.1541  
##  3rd Qu.: 4.000   3rd Qu.:1.0000            3rd Qu.:1.0000   3rd Qu.:0.2500  
##  Max.   :11.000   Max.   :1.0000            Max.   :1.0000   Max.   :1.0000  
##                                             NA's   :214                      
##  N.GraduateCourse FullTimeStudent 
##  Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :1.0000  
##  Mean   :0.6173   Mean   :0.5787  
##  3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :5.0000   Max.   :1.0000  
## 
```

```r
# when perc.withd is 1, perc.pass is always NA
# turning perc.pass na values into -1
# full.data$Perc.Pass =
# ifelse(is.na(full.data$Perc.Pass), 0,
# full.data$Perc.Pass) #might try 0
full.data$Perc.Pass = ifelse(is.na(full.data$Perc.Pass),
    -1, full.data$Perc.Pass)
cat("% Complete Cases Before NA Omition: ", sum(complete.cases(full.data)/nrow(full.data)))
```

```
## % Complete Cases Before NA Omition:  0.99731
```

```r
# full.data <- na.omit(full.data)
```

```r
# , include = FALSE, eval = FALSE} if we wanted to
# median imputate the rest of the values instead of
# omiting them gender.na <-
# full.data[is.na(full.data$Gender),] gender.na
# gender.na$Gender <- median(full.data$Gender,
# na.rm=TRUE) gender.na

# HSGPA.na <- full.data[is.na(full.data$HSGPA),]
# HSGPA.na HSGPA.na$HSGPA <-
# median(full.data$HSGPA, na.rm=TRUE) HSGPA.na

# SAT_Total.na <-
# full.data[is.na(full.data$SAT_Total),]
# SAT_Total.na SAT_Total.na$SAT_Total <-
# median(full.data$SAT_Total, na.rm=TRUE)
# SAT_Total.na

full.data[is.na(full.data$Gender), ]$Gender <- median(full.data$Gender,
    na.rm = TRUE)
full.data[is.na(full.data$HSGPA), ]$HSGPA <- median(full.data$HSGPA,
    na.rm = TRUE)
full.data[is.na(full.data$SAT_Total), ]$SAT_Total <- median(full.data$SAT_Total,
    na.rm = TRUE)
```

```r
summary(full.data)
```

```
##  Race_Ethc_Visa      Gender            HSGPA          SAT_Total
##  Min.   :0.000   Min.   :0.0000   Min.   : 50.00   Min.   : 900
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 67.00   1st Qu.:1085
##  Median :2.000   Median :0.0000   Median : 76.00   Median :1255
##  Mean   :1.966   Mean   :0.4993   Mean   : 76.51   Mean   :1254
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.: 86.00   3rd Qu.:1424
##  Max.   :4.000   Max.   :1.0000   Max.   :100.00   Max.   :1600
##    Entry_Term        Term.GPA     Persistence.NextYear N.RegisteredCourse
##  Min.   :0.0000   Min.   :0.500   Min.   :0.0000       Min.   : 1.00
##  1st Qu.:0.0000   1st Qu.:1.380   1st Qu.:1.0000       1st Qu.: 2.00
##  Median :1.0000   Median :2.250   Median :1.0000       Median : 3.00
##  Mean   :0.7197   Mean   :2.246   Mean   :0.8195       Mean   : 3.58
##  3rd Qu.:1.0000   3rd Qu.:3.120   3rd Qu.:1.0000       3rd Qu.: 5.00
```

```
##  Max.   :1.0000   Max.   :4.000   Max.   :1.0000       Max.   :11.00
##       N.Ws           N.DFs             N.As         N.PassedCourse
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 1.000
##  Median :0.000   Median :0.0000   Median :1.0000   Median : 2.000
##  Mean   :0.545   Mean   :0.6256   Mean   :0.8469   Mean   : 2.409
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 3.000
##  Max.   :6.000   Max.   :7.0000   Max.   :7.0000   Max.   :11.000
##  N.CourseTaken    Perc.PassedEnrolledCourse   Perc.Pass          Perc.Withd
##  Min.   : 0.000   Min.   :0.0000            Min.   :-1.0000   Min.   :0.0000
##  1st Qu.: 2.000   1st Qu.:0.5000            1st Qu.: 0.5000   1st Qu.:0.0000
##  Median : 3.000   Median :0.7143            Median : 1.0000   Median :0.0000
##  Mean   : 3.035   Mean   :0.6700            Mean   : 0.7367   Mean   :0.1541
##  3rd Qu.: 4.000   3rd Qu.:1.0000            3rd Qu.: 1.0000   3rd Qu.:0.2500
##  Max.   :11.000   Max.   :1.0000            Max.   : 1.0000   Max.   :1.0000
##  N.GraduateCourse FullTimeStudent
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.6173   Mean   :0.5787
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :5.0000   Max.   :1.0000
```

```r
slr <- lm(Term.GPA ~ . - Persistence.NextYear - N.CourseTaken -
    N.RegisteredCourse - Perc.PassedEnrolledCourse,
    data = full.data)
vif(slr)
```

```
##   Race_Ethc_Visa           Gender            HSGPA        SAT_Total
##         1.001463         1.002181         1.008348         1.003159
##       Entry_Term             N.Ws            N.DFs             N.As
##         1.172681         4.731312         2.822531         1.422215
##   N.PassedCourse        Perc.Pass       Perc.Withd N.GraduateCourse
##         3.461884         3.415285         7.145205         1.178915
##  FullTimeStudent
##         2.555112
```

```r
# Race_Ethc_Visa, Gender, HSGPA, SAT_Total,
# Entry_Term, N.Ws, N.DFs, N.As, N.PassedCourse,
# Perc.Pass, Perc.Withd, N.GraduateCourse,
# FullTimeStudent
```

```r
full.data <- full.data[, c(1, 2, 3, 4, 5, 9, 10, 11,
    12, 15, 16, 17, 18, 8, 13, 14, 6, 7)]  #last rows now response vars
```

```r
names(full.data)
```

```
##  [1] "Race_Ethc_Visa"            "Gender"
##  [3] "HSGPA"                     "SAT_Total"
##  [5] "Entry_Term"                "N.Ws"
##  [7] "N.DFs"                     "N.As"
##  [9] "N.PassedCourse"            "Perc.Pass"
```

```
## [11] "Perc.Withd"              "N.GraduateCourse"
## [13] "FullTimeStudent"         "N.RegisteredCourse"
## [15] "N.CourseTaken"           "Perc.PassedEnrolledCourse"
## [17] "Term.GPA"                "Persistence.NextYear"
```

```
data <- full.data[, c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12, 13, 17, 18)]  #getting rid of problem vars
names(data)
```

```
##  [1] "Race_Ethc_Visa"      "Gender"              "HSGPA"
##  [4] "SAT_Total"           "Entry_Term"          "N.Ws"
##  [7] "N.DFs"               "N.As"                "N.PassedCourse"
## [10] "Perc.Pass"           "Perc.Withd"          "N.GraduateCourse"
## [13] "FullTimeStudent"     "Term.GPA"            "Persistence.NextYear"
```

```
dim(data)
```

```
## [1] 7435   15
```

```
5900/7415
```

```
## [1] 0.7956844
```

```
6000/7415
```

```
## [1] 0.8091706
```

```
normalize <- function(x) {
    return((x - min(x))/(max(x) - min(x)))
}
```

```
scaled.data <- as.data.frame(lapply(data, normalize))
summary(scaled.data)
```

```
##  Race_Ethc_Visa      Gender           HSGPA           SAT_Total
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2500   1st Qu.:0.0000   1st Qu.:0.3400   1st Qu.:0.2643
##  Median :0.5000   Median :0.0000   Median :0.5200   Median :0.5071
##  Mean   :0.4916   Mean   :0.4993   Mean   :0.5302   Mean   :0.5064
##  3rd Qu.:0.7500   3rd Qu.:1.0000   3rd Qu.:0.7200   3rd Qu.:0.7486
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    Entry_Term          N.Ws              N.DFs               N.As
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :1.0000   Median :0.00000   Median :0.00000   Median :0.1429
##  Mean   :0.7197   Mean   :0.09083   Mean   :0.08936   Mean   :0.1210
##  3rd Qu.:1.0000   3rd Qu.:0.16667   3rd Qu.:0.14286   3rd Qu.:0.1429
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  N.PassedCourse      Perc.Pass         Perc.Withd      N.GraduateCourse
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
```

```
##  1st Qu.:0.09091   1st Qu.:0.7500   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.18182   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.21903   Mean   :0.8684   Mean   :0.1541   Mean   :0.1235
##  3rd Qu.:0.27273   3rd Qu.:1.0000   3rd Qu.:0.2500   3rd Qu.:0.2000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  FullTimeStudent     Term.GPA      Persistence.NextYear
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.2514   1st Qu.:1.0000
##  Median :1.0000   Median :0.5000   Median :1.0000
##  Mean   :0.5787   Mean   :0.4989   Mean   :0.8195
##  3rd Qu.:1.0000   3rd Qu.:0.7486   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```r
set.seed(99)
rows <- sample(nrow(data))
train = rows[1:5900]
train.data = data[train, ]
test.data = data[-train, ]
train.data.sc <- scaled.data[train, ]
test.data.sc <- scaled.data[-train, ]
f.train.data = full.data[train, ]
f.test.data = full.data[-train, ]
```

```r
# Summarize univariately
summary(train.data)
```

```
##  Race_Ethc_Visa      Gender           HSGPA          SAT_Total
##  Min.   :0.000   Min.   :0.0000   Min.   : 50.0   Min.   : 900
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 67.0   1st Qu.:1085
##  Median :2.000   Median :0.0000   Median : 75.0   Median :1256
##  Mean   :1.944   Mean   :0.4953   Mean   : 76.4   Mean   :1255
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.: 86.0   3rd Qu.:1425
##  Max.   :4.000   Max.   :1.0000   Max.   :100.0   Max.   :1600
##    Entry_Term          N.Ws              N.DFs            N.As
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.000   Median :1.0000
##  Mean   :0.7192   Mean   :0.5364   Mean   :0.621   Mean   :0.8439
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.0000   Max.   :7.000   Max.   :7.0000
##  N.PassedCourse     Perc.Pass          Perc.Withd     N.GraduateCourse
##  Min.   : 0.000   Min.   :-1.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 1.000   1st Qu.: 0.5000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 2.000   Median : 1.0000   Median :0.0000   Median :0.0000
##  Mean   : 2.419   Mean   : 0.7404   Mean   :0.1516   Mean   :0.6114
##  3rd Qu.: 3.000   3rd Qu.: 1.0000   3rd Qu.:0.2500   3rd Qu.:1.0000
##  Max.   :11.000   Max.   : 1.0000   Max.   :1.0000   Max.   :5.0000
##  FullTimeStudent     Term.GPA     Persistence.NextYear
##  Min.   :0.0000   Min.   :0.500   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.390   1st Qu.:1.0000
##  Median :1.0000   Median :2.260   Median :1.0000
##  Mean   :0.5768   Mean   :2.254   Mean   :0.8212
##  3rd Qu.:1.0000   3rd Qu.:3.130   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :4.000   Max.   :1.0000
```

```r
summary(test.data)
```

```
##  Race_Ethc_Visa      Gender           HSGPA          SAT_Total   
##  Min.   :0.000   Min.   :0.0000   Min.   : 50.00   Min.   : 900  
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 68.00   1st Qu.:1080  
##  Median :2.000   Median :1.0000   Median : 76.00   Median :1246  
##  Mean   :2.051   Mean   :0.5147   Mean   : 76.93   Mean   :1251  
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.: 87.00   3rd Qu.:1422  
##  Max.   :4.000   Max.   :1.0000   Max.   :100.00   Max.   :1600  
##    Entry_Term          N.Ws             N.DFs             N.As       
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000  
##  Median :1.0000   Median :0.0000   Median :0.000   Median :1.0000  
##  Mean   :0.7218   Mean   :0.5778   Mean   :0.643   Mean   :0.8586  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :6.0000   Max.   :5.000   Max.   :6.0000  
##  N.PassedCourse     Perc.Pass          Perc.Withd     N.GraduateCourse
##  Min.   : 0.000   Min.   :-1.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.: 1.000   1st Qu.: 0.5000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median : 2.000   Median : 1.0000   Median :0.0000   Median :0.0000  
##  Mean   : 2.371   Mean   : 0.7227   Mean   :0.1639   Mean   :0.6404  
##  3rd Qu.: 3.000   3rd Qu.: 1.0000   3rd Qu.:0.2500   3rd Qu.:1.0000  
##  Max.   :10.000   Max.   : 1.0000   Max.   :1.0000   Max.   :5.0000  
##  FullTimeStudent     Term.GPA     Persistence.NextYear
##  Min.   :0.0000   Min.   :0.500   Min.   :0.000       
##  1st Qu.:0.0000   1st Qu.:1.340   1st Qu.:1.000       
##  Median :1.0000   Median :2.190   Median :1.000       
##  Mean   :0.5863   Mean   :2.214   Mean   :0.813       
##  3rd Qu.:1.0000   3rd Qu.:3.085   3rd Qu.:1.000       
##  Max.   :1.0000   Max.   :4.000   Max.   :1.000
```

```r
# Dims
dim(train.data)  #5961x18
```

```
## [1] 5900   15
```

```r
dim(test.data)  #1474x18
```

```
## [1] 1535   15
```

```r
# Response variables you may do this
y1 = train.data$Term.GPA  #numerical
y2 = train.data$Persistence.NextYear  #categorical: 0, 1
# you may do this
z1 = test.data$Term.GPA  #numerical
z2 = test.data$Persistence.NextYear  #categorical: 0, 1

y1.sc = train.data.sc$Term.GPA  #numerical
y2.sc = train.data.sc$Persistence.NextYear  #categorical: 0, 1
# you may do this
z1.sc = test.data.sc$Term.GPA  #numerical
z2.sc = test.data.sc$Persistence.NextYear  #categorical: 0, 1
```

## Section A.

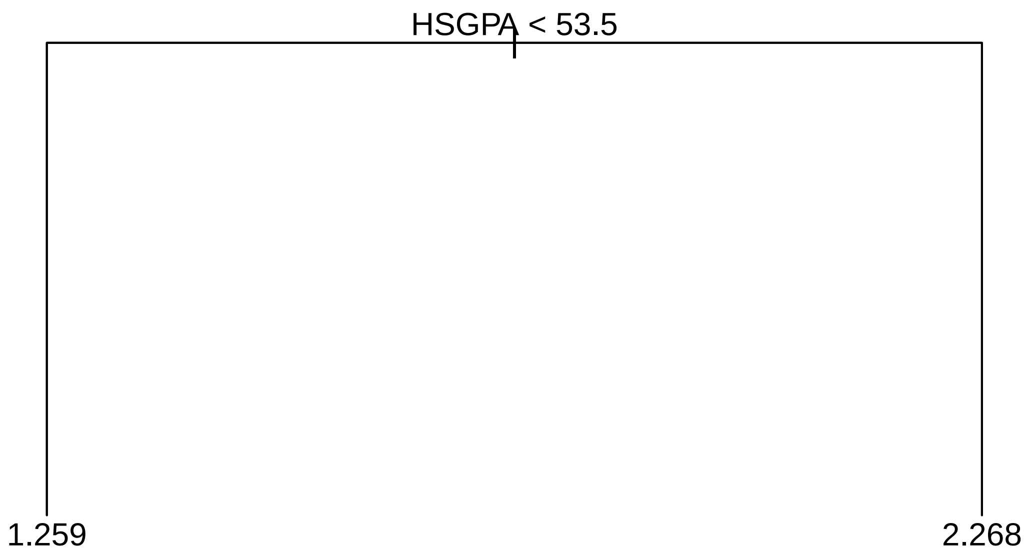
```r
tree.gpa = tree(Term.GPA ~ . - Persistence.NextYear,
    data = train.data)
summary(tree.gpa)
```

```
##
## Regression tree:
## tree(formula = Term.GPA ~ . - Persistence.NextYear, data = train.data)
## Variables actually used in tree construction:
## [1] "HSGPA"
## Number of terminal nodes:  2
## Residual mean deviance:  1.016 = 5990 / 5898
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.768000 -0.848100  0.001905  0.000000  0.861900  2.071000
```

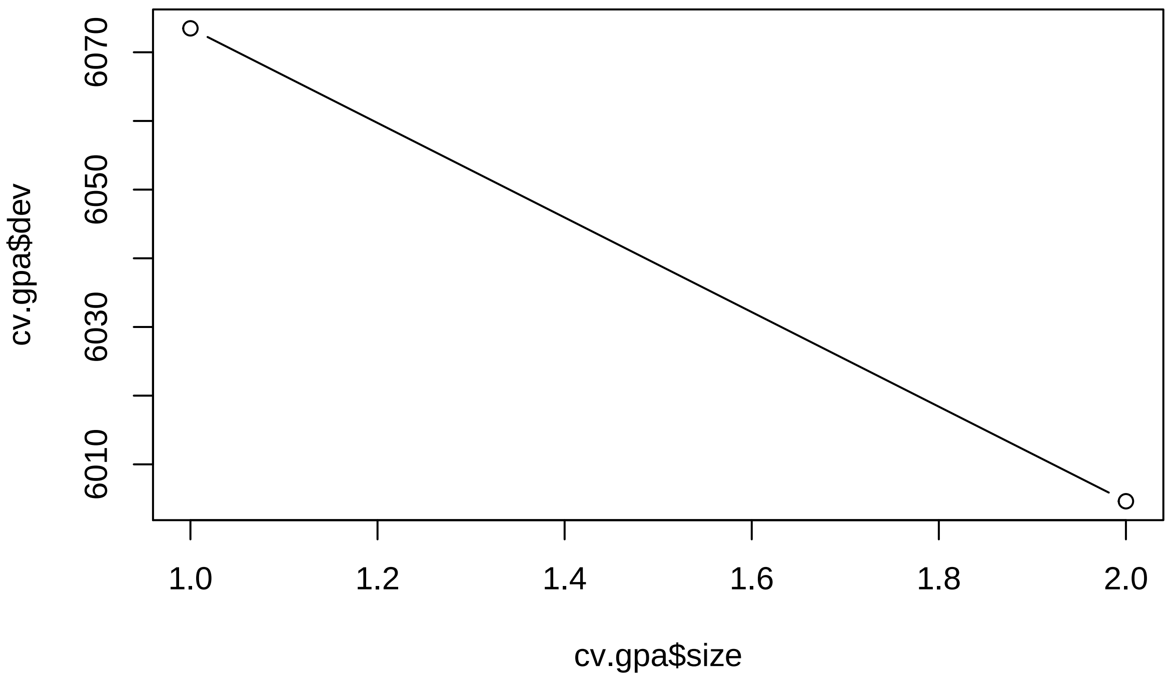
```r
tree.gpa2 = tree(Term.GPA ~ . - Persistence.NextYear,
    data = f.train.data)
summary(tree.gpa2)
```

```
##
## Regression tree:
## tree(formula = Term.GPA ~ . - Persistence.NextYear, data = f.train.data)
## Variables actually used in tree construction:
## [1] "HSGPA"
## Number of terminal nodes:  2
## Residual mean deviance:  1.016 = 5990 / 5898
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.768000 -0.848100  0.001905  0.000000  0.861900  2.071000
```

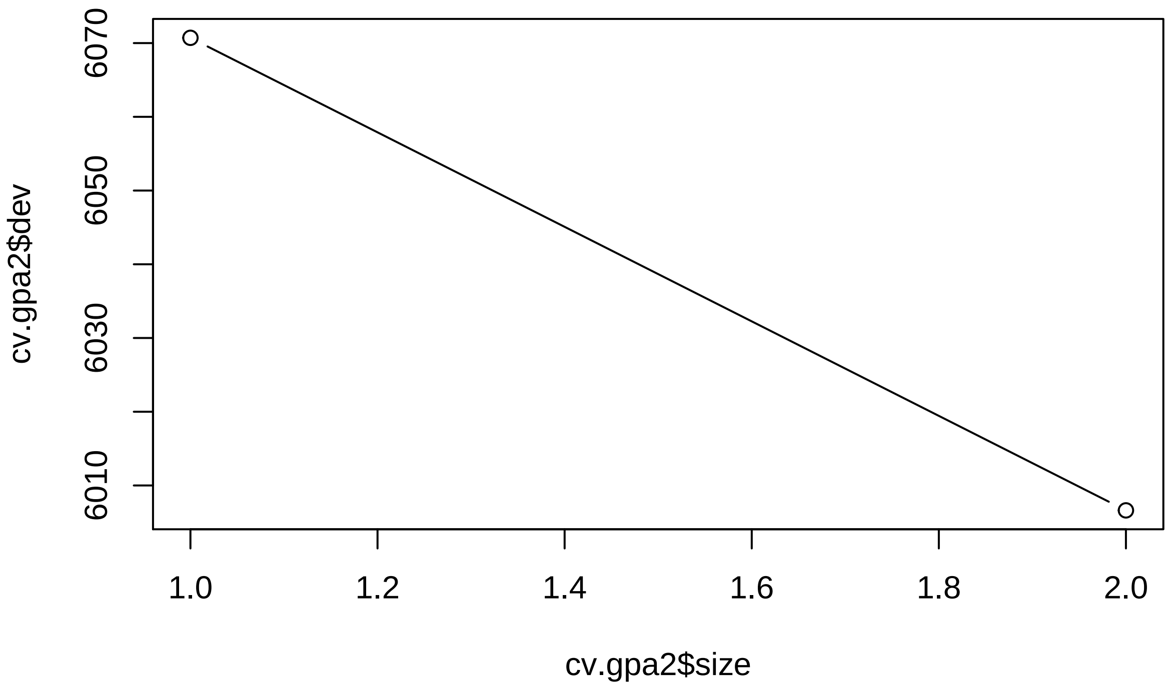
```r
plot(tree.gpa)
text(tree.gpa, pretty = 0)
```




```
cv.gpa = cv.tree(tree.gpa)
plot(cv.gpa$size, cv.gpa$dev, type = "b")
```



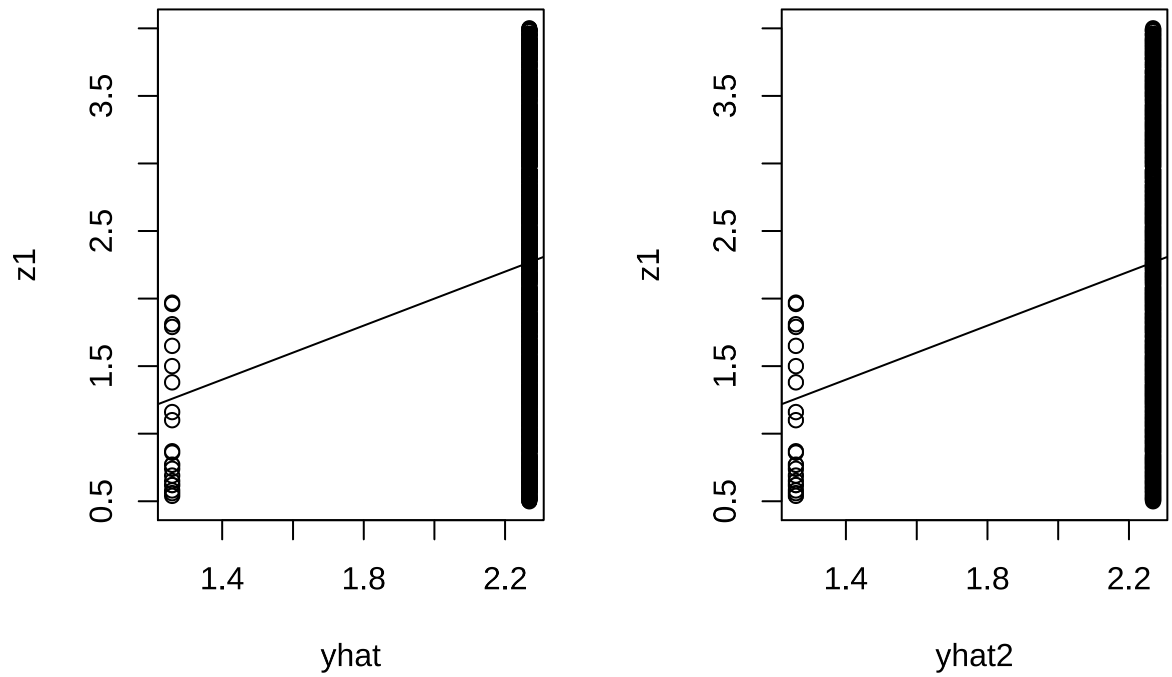

```
cv.gpa2 = cv.tree(tree.gpa2)
plot(cv.gpa2$size, cv.gpa2$dev, type = "b")
```




```r
par(mfrow = c(1, 2))
yhat = predict(tree.gpa, newdata = test.data)
plot(yhat, z1)
abline(0, 1)
cat("Reduced Model Regression Tree Test MSE", mean((yhat -
    z1)^2), "\n")
```

```
## Reduced Model Regression Tree Test MSE 0.9854669
```

```r
yhat2 = predict(tree.gpa2, newdata = f.test.data)
plot(yhat2, z1)
abline(0, 1)
```




```r
cat("Full Model Regression Tree Test MSE", mean((yhat2 -
    z1)^2))
```

```
## Full Model Regression Tree Test MSE 0.9854669
```

```r
set.seed(99)
bag.gpa = randomForest(Term.GPA ~ . - Persistence.NextYear,
    data = train.data, mtry = 13, importance = TRUE)
bag.gpa
```

```
##
## Call:
##  randomForest(formula = Term.GPA ~ . - Persistence.NextYear, data = train.data,      mtry = 13, impo
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 1.055922
##                     % Var explained: -2.63
```

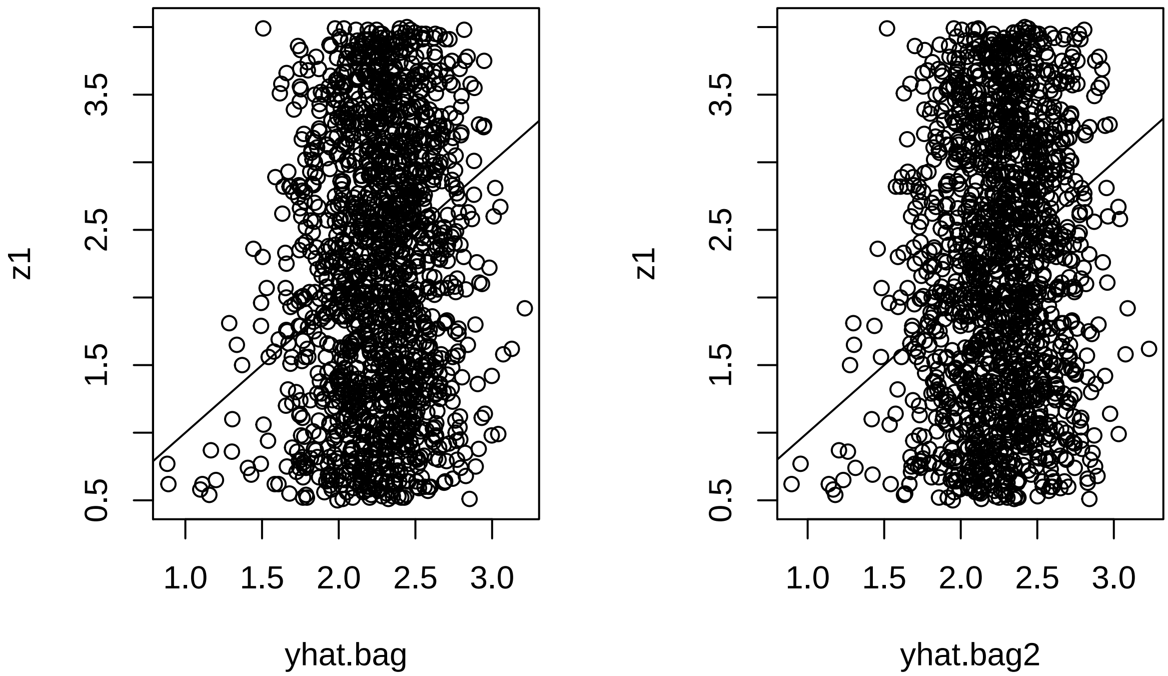
```r
bag.gpa2 = randomForest(Term.GPA ~ . - Persistence.NextYear,
    data = f.train.data, mtry = 16, importance = TRUE)
bag.gpa2
```

```
##
## Call:
##  randomForest(formula = Term.GPA ~ . - Persistence.NextYear, data = f.train.data,      mtry = 16, imp
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 16
##
##           Mean of squared residuals: 1.052768
##                     % Var explained: -2.33
```

```r
par(mfrow = c(1, 2))
yhat.bag = predict(bag.gpa, newdata = test.data)
plot(yhat.bag, z1)
abline(0, 1)
cat("Reduced Model Bagging Test MSE", mean((yhat.bag -
    z1)^2), "\n")
```

```
## Reduced Model Bagging Test MSE 1.03556
```

```r
yhat.bag2 = predict(bag.gpa2, newdata = f.test.data)
plot(yhat.bag2, z1)
abline(0, 1)
```



```r
cat("Full Model Bagging Test MSE", mean((yhat.bag2 -
    z1)^2))
```

```
## Full Model Bagging Test MSE 1.04106
```

```r
set.seed(99)
forest.gpa = randomForest(Term.GPA ~ . - Persistence.NextYear,
    data = train.data, importance = TRUE)
yhat.forest = predict(forest.gpa, newdata = test.data)
cat("Reduced Model Random Forests Test MSE:", mean((yhat.forest -
    z1)^2), "\n")
```

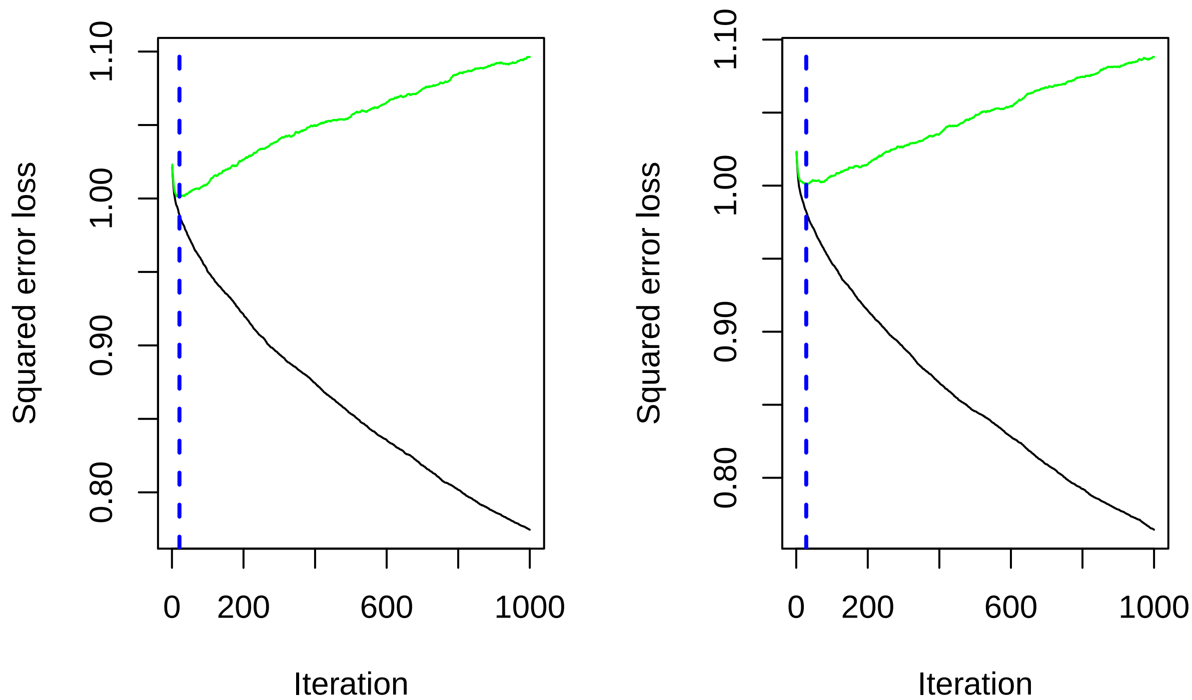
```
## Reduced Model Random Forests Test MSE: 1.022795
```

```r
forest.gpa2 = randomForest(Term.GPA ~ . - Persistence.NextYear,
    data = f.train.data, importance = TRUE)
yhat.forest2 = predict(forest.gpa2, newdata = f.test.data)
cat("Full Model Random Forests Test MSE:", mean((yhat.forest2 -
    z1)^2))
```

```
## Full Model Random Forests Test MSE: 1.028822
```

```r
boost.gpa = gbm(Term.GPA ~ . - Persistence.NextYear,
    data = train.data, distribution = "gaussian", n.trees = 1000,
    interaction.depth = 4, shrinkage = 0.1, cv.folds = 5,
    verbose = F)


boost.gpa2 = gbm(Term.GPA ~ . - Persistence.NextYear,
    data = f.train.data, distribution = "gaussian",
    n.trees = 1000, interaction.depth = 4, shrinkage = 0.1,
    cv.folds = 5, verbose = F)
```

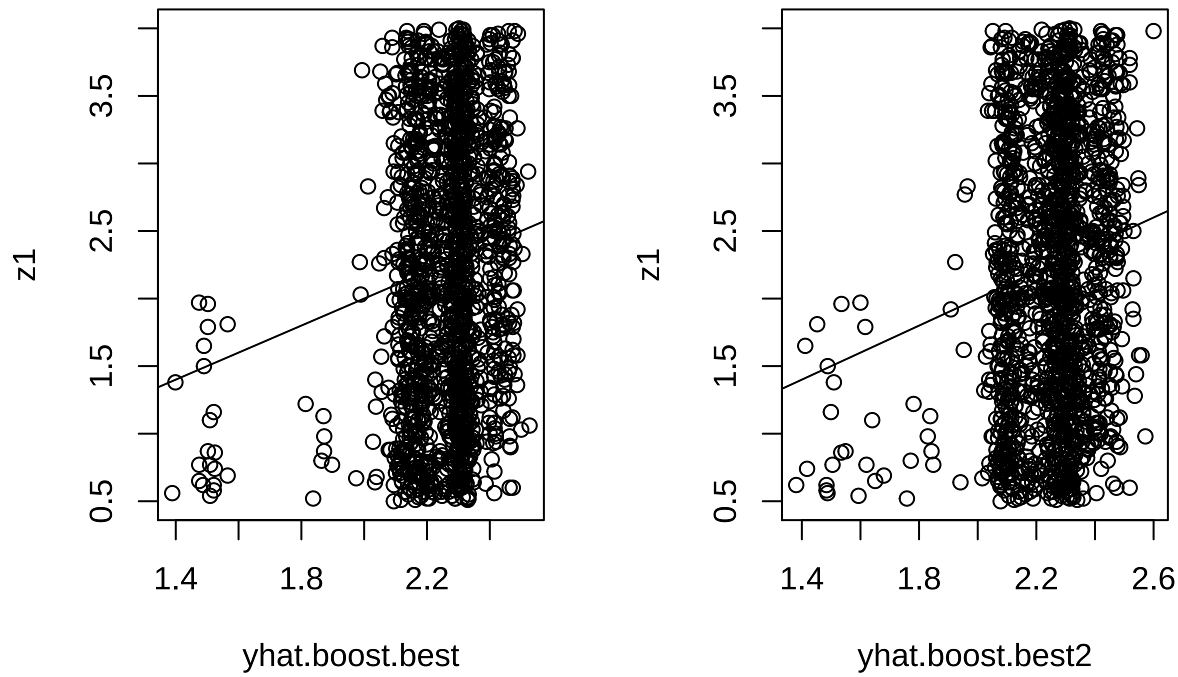
```r
par(mfrow = c(1, 2))
best.iter.gpa = gbm.perf(boost.gpa, method = "cv")
best.iter.gpa2 = gbm.perf(boost.gpa2, method = "cv")
```




```r
par(mfrow = c(1, 2))
yhat.boost.best = predict(boost.gpa, newdata = test.data,
    n.trees = best.iter.gpa)
plot(yhat.boost.best, z1)
abline(0, 1)
cat("Reduced Model Boosting Test MSE:", mean((yhat.boost.best -
    z1)^2), "\n")
```

```
## Reduced Model Boosting Test MSE: 0.9707814
```

```r
yhat.boost.best2 = predict(boost.gpa2, newdata = f.test.data,
    n.trees = best.iter.gpa2)
plot(yhat.boost.best2, z1)
abline(0, 1)
```




```
cat("Full Model Boosting Test MSE:", mean((yhat.boost.best2 -
    z1)^2))
```

```
## Full Model Boosting Test MSE: 0.9709897
```

```
mean((mean(z1) - z1)^2)  #using mean to predict. This should be absolute worst prediction
```

```
## [1] 1.001568
```

```
k = 13
mse.nns <- c()
error.nns <- c()
for (n in (4:k)) {
    nn.gpa = neuralnet(Term.GPA ~ Race_Ethc_Visa +
        Gender + HSGPA + SAT_Total + Entry_Term + N.Ws +
        N.DFs + N.As + N.PassedCourse + Perc.Pass +
        Perc.Withd + N.GraduateCourse + FullTimeStudent,
        data = train.data.sc, hidden = n, stepmax = 3e+05,
        linear.output = TRUE, err.fct = "sse")

    pr.nn <- compute(nn.gpa, test.data.sc[, 1:13])
    pr.nn <- pr.nn$net.result * (max(data$Term.GPA) -
        min(data$Term.GPA)) + min(data$Term.GPA)
    test.cv.r <- (test.data.sc$Term.GPA) * (max(data$Term.GPA) -
```

```
        min(data$Term.GPA)) + min(data$Term.GPA)
    mse.nns <- c(mse.nns, (sum((test.cv.r - pr.nn)^2)/(nrow(test.data.sc) -
        2)))

    print(n)
    print((sum((test.cv.r - pr.nn)^2)/(nrow(test.data.sc) -
        2)))

}
```

```
mse.nns
```

```
# yhat.nn <- predict(nn.gpa,newdata=test.data.sc)
# mse.nns <- c(mse.nns,(sum((yhat.nn -
# z1.sc)^2)/2)/(length(test.data.sc)-2))
yhat.nn = predict(nn.gpa, newdata = test.data.sc[,
    1:13])
sse.nn <- sum((yhat.nn - z1.sc)^2)/2
sse.nn
mse.nn <- sse.nn/(length(test.data.sc) - 2)
mse.nn
```

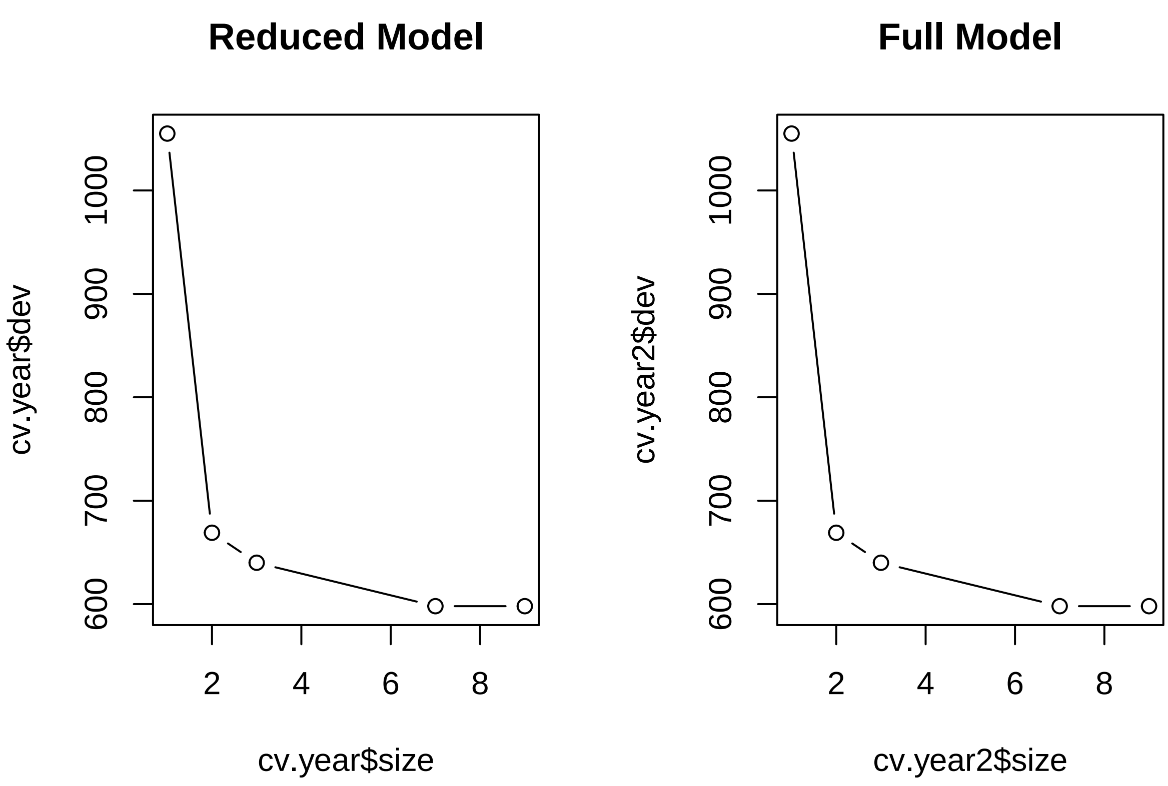
---

**Section B.**

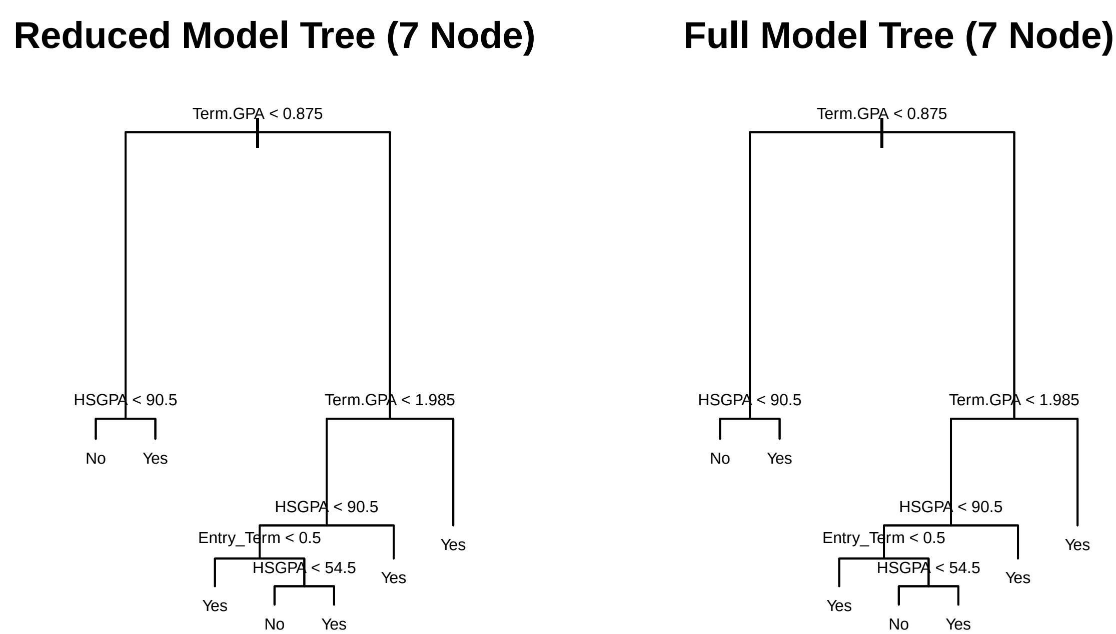
```
train.data$Persistence.NextYear = factor(ifelse(train.data$Persistence.NextYear ==
    1, "Yes", "No"))
test.data$Persistence.NextYear = factor(ifelse(test.data$Persistence.NextYear ==
    1, "Yes", "No"))

f.train.data$Persistence.NextYear = factor(ifelse(f.train.data$Persistence.NextYear ==
    1, "Yes", "No"))
f.test.data$Persistence.NextYear = factor(ifelse(f.test.data$Persistence.NextYear ==
    1, "Yes", "No"))
```

```
set.seed(99)
tree.year = tree(Persistence.NextYear ~ ., data = train.data)
tree.year2 = tree(Persistence.NextYear ~ ., data = f.train.data)
```

```
par(mfrow = c(1, 2))
cv.year = cv.tree(tree.year, FUN = prune.misclass)
plot(cv.year$size, cv.year$dev, type = "b", main = "Reduced Model")

cv.year2 = cv.tree(tree.year2, FUN = prune.misclass)
plot(cv.year2$size, cv.year2$dev, type = "b", main = "Full Model")
```



```r
par(mfrow = c(1, 2))

prune.year = prune.misclass(tree.year, best = 7)
plot(prune.year)
text(prune.year, pretty = 0, cex = 0.5)
title("Reduced Model Tree (7 Node)")

prune.year2 = prune.misclass(tree.year2, best = 7)
plot(prune.year2)
text(prune.year2, pretty = 0, cex = 0.5)
title("Full Model Tree (7 Node)")
```




```r
tree.pred = predict(prune.year, test.data, type = "class")  #should be pruned tree from cv
tree.cm = table(test.data$Persistence.NextYear, tree.pred)
tree.cm
```

```
##      tree.pred
##         No  Yes
##   No   149  138
##   Yes   26 1222
```

```r
recall = tree.cm[2, 2]/(tree.cm[2, 2] + tree.cm[2,
    1])
precision = tree.cm[2, 2]/(tree.cm[2, 2] + tree.cm[1,
```

```r
    2])

tree.f1 = 2 * (recall * precision)/(recall + precision)
cat("Reduced Model Classification Tree Test F1: ",
    tree.f1, "\n")
```

```
## Reduced Model Classification Tree Test F1:  0.9371166
```

```r
tree.accuracy = (tree.cm[1, 1] + tree.cm[2, 2])/(length(test.data$Persistence.NextYear))
cat("Reduced Model Classification Tree Test Accuracy: ",
    tree.accuracy, "\n")
```

```
## Reduced Model Classification Tree Test Accuracy:  0.8931596
```

```r
tree.pred2 = predict(prune.year2, f.test.data, type = "class")  #should be pruned tree from cv
tree.cm2 = table(f.test.data$Persistence.NextYear,
    tree.pred2)
tree.cm2
```

```
##      tree.pred2
##         No  Yes
##   No   149  138
##   Yes   26 1222
```

```r
recall = tree.cm2[2, 2]/(tree.cm2[2, 2] + tree.cm2[2,
    1])
precision = tree.cm2[2, 2]/(tree.cm2[2, 2] + tree.cm2[1,
    2])

tree.f1.2 = 2 * (recall * precision)/(recall + precision)
cat("Full Model Classification Tree Test f1: ", tree.f1.2,
    "\n")
```

```
## Full Model Classification Tree Test f1:  0.9371166
```

```r
tree.accuracy2 = (tree.cm2[1, 1] + tree.cm2[2, 2])/(length(f.test.data$Persistence.NextYear))
cat("Full Model Classification Tree Test Accuracy: ",
    tree.accuracy2, "\n")
```

```
## Full Model Classification Tree Test Accuracy:  0.8931596
```

```r
set.seed(99)
bag.year = randomForest(Persistence.NextYear ~ ., data = train.data,
    mtry = 13, importance = TRUE)
bag.year
```

```
##
## Call:
##  randomForest(formula = Persistence.NextYear ~ ., data = train.data,      mtry = 13, importance = TRU
```

```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 11.08%
## Confusion matrix:
##      No  Yes class.error
## No  651  404  0.38293839
## Yes 250 4595  0.05159959
```

```
bag.year2 = randomForest(Persistence.NextYear ~ .,
    data = f.train.data, mtry = 16, importance = TRUE)
bag.year2
```

```
##
## Call:
##  randomForest(formula = Persistence.NextYear ~ ., data = f.train.data,      mtry = 16, importance = 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 16
##
##         OOB estimate of  error rate: 10.93%
## Confusion matrix:
##      No  Yes class.error
## No  652  403   0.3819905
## Yes 242 4603   0.0499484
```

```
bag.pred = predict(bag.year, test.data, type = "class")
bag.cm = table(test.data$Persistence.NextYear, bag.pred)
bag.cm
```

```
##      bag.pred
##         No  Yes
##   No   174  113
##   Yes   60 1188
```

```
recall = bag.cm[2, 2]/(bag.cm[2, 2] + bag.cm[2, 1])
precision = bag.cm[2, 2]/(bag.cm[2, 2] + bag.cm[1,
    2])

bag.f1 = 2 * (recall * precision)/(recall + precision)
cat("Reduced Model Bagging Test f1: ", bag.f1, "\n")
```

```
## Reduced Model Bagging Test f1:  0.9321302
```

```
bag.accuracy = (bag.cm[1, 1] + bag.cm[2, 2])/(length(test.data$Persistence.NextYear))
cat("Reduced Model Bagging Test Accuracy: ", bag.accuracy,
    "\n")
```

```
## Reduced Model Bagging Test Accuracy:  0.8872964
```

```r
bag.pred2 = predict(bag.year2, f.test.data, type = "class")
bag.cm2 = table(f.test.data$Persistence.NextYear, bag.pred2)
bag.cm2
```

```
##      bag.pred2
##         No  Yes
##   No   172  115
##   Yes   58 1190
```

```r
recall = bag.cm2[2, 2]/(bag.cm2[2, 2] + bag.cm2[2,
    1])
precision = bag.cm2[2, 2]/(bag.cm2[2, 2] + bag.cm2[1,
    2])

bag.f1.2 = 2 * (recall * precision)/(recall + precision)
cat("Full Model Bagging Test f1: ", bag.f1.2, "\n")
```

```
## Full Model Bagging Test f1:  0.9322366
```

```r
bag.accuracy2 = (bag.cm2[1, 1] + bag.cm2[2, 2])/(length(f.test.data$Persistence.NextYear))
cat("Full Model Bagging Test Accuracy: ", bag.accuracy2,
    "\n")
```

```
## Full Model Bagging Test Accuracy:  0.8872964
```

```r
set.seed(99)
forest.year = randomForest(Persistence.NextYear ~ .,
    data = train.data, importance = TRUE)

forest.year2 = randomForest(Persistence.NextYear ~
    ., data = f.train.data, importance = TRUE)
```

```r
forest.pred = predict(forest.year, test.data, type = "class")
forest.cm = table(test.data$Persistence.NextYear, forest.pred)
forest.cm
```

```
##      forest.pred
##         No  Yes
##   No   160  127
##   Yes   42 1206
```

```r
recall = forest.cm[2, 2]/(forest.cm[2, 2] + forest.cm[2,
    1])
precision = forest.cm[2, 2]/(forest.cm[2, 2] + forest.cm[1,
    2])

forest.f1 = 2 * (recall * precision)/(recall + precision)
cat("Reduced Model Random Forests Test f1: ", forest.f1,
    "\n")
```

```
## Reduced Model Random Forests Test f1:  0.9345215
```

```r
forest.accuracy = (forest.cm[1, 1] + forest.cm[2, 2])/(length(test.data$Persistence.NextYear))
cat("Reduced Model Random Forests Test Accuracy: ",
    forest.accuracy, "\n")
```

```
## Reduced Model Random Forests Test Accuracy:  0.8899023
```

```r
forest.pred2 = predict(forest.year2, f.test.data, type = "class")
forest.cm2 = table(f.test.data$Persistence.NextYear,
    forest.pred2)
forest.cm2
```

```
##      forest.pred2
##         No  Yes
##   No   165  122
##   Yes   43 1205
```

```r
recall = forest.cm2[2, 2]/(forest.cm2[2, 2] + forest.cm2[2,
    1])
precision = forest.cm2[2, 2]/(forest.cm2[2, 2] + forest.cm2[1,
    2])

forest.f1.2 = 2 * (recall * precision)/(recall + precision)
cat("Full Model Random Forests Test f1: ", forest.f1.2,
    "\n")
```

```
## Full Model Random Forests Test f1:  0.9359223
```

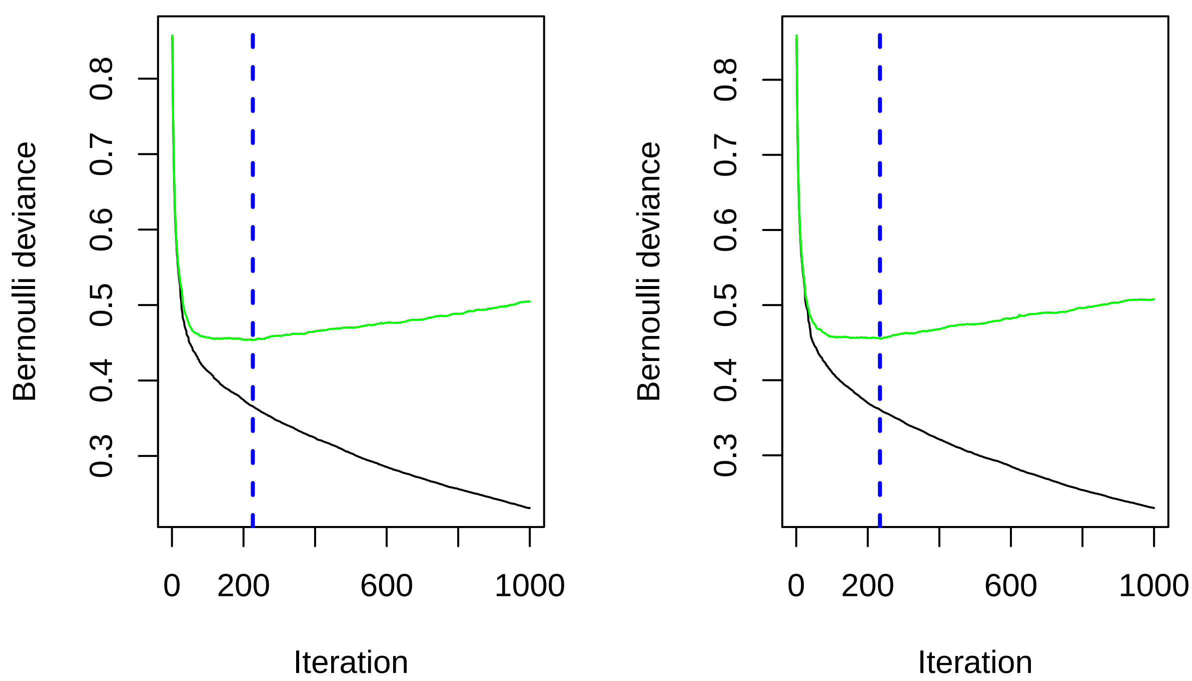
```r
forest.accuracy2 = (forest.cm2[1, 1] + forest.cm2[2,
    2])/(length(f.test.data$Persistence.NextYear))
cat("Full Model Random Forests Test Accuracy: ", forest.accuracy2,
    "\n")
```

```
## Full Model Random Forests Test Accuracy:  0.8925081
```

```r
train.data$Persistence.NextYear = ifelse(train.data$Persistence.NextYear ==
    "Yes", 1, 0)
test.data$Persistence.NextYear = ifelse(test.data$Persistence.NextYear ==
    "Yes", 1, 0)

f.train.data$Persistence.NextYear = ifelse(f.train.data$Persistence.NextYear ==
    "Yes", 1, 0)
f.test.data$Persistence.NextYear = ifelse(f.test.data$Persistence.NextYear ==
    "Yes", 1, 0)
```

```r
par(mfrow = c(1, 2))
boost.year = gbm(Persistence.NextYear ~ ., data = train.data,
    distribution = "bernoulli", n.trees = 1000, interaction.depth = 4,
    shrinkage = 0.1, cv.folds = 5, verbose = F)

best.iter.year = gbm.perf(boost.year, method = "cv")
```

```r
boost.year2 = gbm(Persistence.NextYear ~ ., data = f.train.data,
    distribution = "bernoulli", n.trees = 1000, interaction.depth = 4,
    shrinkage = 0.1, cv.folds = 5, verbose = F)

best.iter.year2 = gbm.perf(boost.year2, method = "cv")
```




```r
boost.probs.i = predict.gbm(boost.year, newdata = test.data,
    n.trees = best.iter.year, type = "response")
boost.probs.i = as.matrix(boost.probs.i)
boost.pred.i <- ifelse(boost.probs.i > 0.5, 1, 0)

boost.cm.i = table(test.data$Persistence.NextYear,
    boost.pred.i)
boost.cm.i
```

```
##    boost.pred.i
##        0    1
##   0  169  118
##   1   42 1206
```

```r
recall = boost.cm.i[2, 2]/(boost.cm.i[2, 2] + boost.cm.i[2,
    1])
precision = boost.cm.i[2, 2]/(boost.cm.i[2, 2] + boost.cm.i[1,
```

```r
    2])

boost.f1.i = 2 * (recall * precision)/(recall + precision)
cat("Reduced Model Best Boosting Test f1: ", boost.f1.i,
    "\n")
```

```
## Reduced Model Best Boosting Test f1:  0.9377916
```

```r
boost.accuracy.i = (boost.cm.i[1, 1] + boost.cm.i[2,
    2])/(length(test.data$Persistence.NextYear))
cat("Reduced Model Best Boosting Test Accuracy: ",
    boost.accuracy.i, "\n")
```

```
## Reduced Model Best Boosting Test Accuracy:  0.8957655
```

```r
boost.probs.i2 = predict.gbm(boost.year2, newdata = f.test.data,
    n.trees = best.iter.year2, type = "response")
boost.probs.i2 = as.matrix(boost.probs.i2)
boost.pred.i2 <- ifelse(boost.probs.i2 > 0.5, 1, 0)

boost.cm.i2 = table(f.test.data$Persistence.NextYear,
    boost.pred.i2)
boost.cm.i2
```

```
##    boost.pred.i2
##        0    1
##   0  166  121
##   1   41 1207
```

```r
recall = boost.cm.i2[2, 2]/(boost.cm.i2[2, 2] + boost.cm.i2[2,
    1])
precision = boost.cm.i2[2, 2]/(boost.cm.i2[2, 2] +
    boost.cm.i2[1, 2])

boost.f1.i2 = 2 * (recall * precision)/(recall + precision)
cat("Full Model Best Boosting Test f1: ", boost.f1.i2,
    "\n")
```

```
## Full Model Best Boosting Test f1:  0.9371118
```

```r
boost.accuracy.i2 = (boost.cm.i2[1, 1] + boost.cm.i2[2,
    2])/(length(f.test.data$Persistence.NextYear))
cat("Full Model Best Boosting Test Accuracy: ", boost.accuracy.i2,
    "\n")
```

```
## Full Model Best Boosting Test Accuracy:  0.8944625
```

**Section C.**

- Part a.

- Part b.

- Part c.

- BONUS.

I hereby write and submit my solutions without violating the academic honesty and integrity. If not, I accept the consequences.

**Write your pair you worked at the top of the page. If no pair, it is ok. List other fiends you worked with (name, last name): . . . Avery Girsky**

**Disclose the resources or persons if you get any help: . . . Lab Codes, Previous Assignments**

**How long did the assignment solutions take?: . . .**

## References

. . .