Midterm-1 Project Portion - Version 1

First and last name: Soumya Goswami // Pair's first and last name: ____ _____

Submission Date: 03/09/2021

Read and Delete This Part Before Submission

- Find a pair, work together, split parts among each of you, explain your findings to each other, make sure you understand all, combine work, and submit separately. It is fine if your codes and results are the same. I expect comments will be your own. If you don't have pair, it is ok.
- Give a name to this rmd file: `Midterm1_Submission_FirstName_LastName.rmd`.
- You will then submit two files to Blackboard: `.rmd` and the knitted `.pdf` files.
- Grading will be based on the pdf file uploaded. Make easy and readable. Grader or me may take a look at the rmd file.
- Unless otherwise specified, use a 5% level for statistical significance.
- Always include your comments on results: don't just leave the numbers without explanations. Use full sentences, structured paragraphs if needed, correct grammar, and proofreading.
- Show your knowledge with detailed work in consistency with course materials.
- Show code. Don't include irrelevant or uncommented outputs. Compact the code and results.
- TAs will grade your and pair's submission.

Midterm-1 Project Instruction

Midterm-1 has test and project portions. This is the project portion. Based on what we covered on the modules 1, 2 and 3, you will reflect statistical methods by analyzing data and building predictive models using train and test data sets. The data sets are about college students and their academic performances and retention status, which include categorical and numerical variables.

Throughout the data analysis, we will consider only two response variables, 1) current GPA of students, a numerical response variable, call it = and 2) Persistence of student for following year, a binary response variable (0: not persistent on the next term, 1:persistent on the next term), call it =.

Briefly, you will fit regression models on $y1$ and classification models on $y2$ using the subset of predictors in the data set. Don't use all predictors in any model.

- Import Data Set and Set Up:

Open the data set . Be familiar with the data and variables. Start exploring it. Practice the code at the bottom and do the set-up.

- Do Exploratory Data Analysis:

Start with Exploratory Data Analysis (EDA) before running models. Visually or aggregatedly you can include the description and summary of the variables (univariate, and some bivariate analyses). If you keep this part very simple, it is ok.

Build linear regressions as listed below the specific four models to predict $y1$ with a small set of useful predictors. Please fit all these by justifying why you do (I expect grounding justifications and technical terms used), report the performance indicators in a comparative table, MSE_{train}, MSE_{test}, $R^2_{adj,train}$ and $R^2_{adj,test}$ using train and test data sets. The regression models you will fit:

For tuning parameter, justify with statistical methods/computations why you choose.

Build four classification models as below. Please fit all these, include performance indicators for train and test data sets, separately. Include confusion matrix for each. For each `train` and `test` data set, report: `accuracy`, `recall`, `precision`, and `f1` in a cooperative table. For LR or LDA, include ROC curve, area and interpretation. The classification models you will fit:

Justify why you choose specific K in KNN with a grid search or CV methods.

Briefly, make critiques of the models fitted and write the conclusion (one sentence for each model, one sentence for each problem - regression and classificaton problems we have here). Also, just address one of these: diagnostics, violations, assumptions checks, overall quality evaluations of the models, importance analyses (which predictors are most important or effects of them on response), outlier analyses. You don't need to address all issues. Just show the reflection of our course materials.

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

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:

- 25% of pts: little progress with some minor solutions;
- 50% of pts: major calculation mistake(s), but good work;
- 75% 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 precision rate` from the classification model) will get a BONUS.

- You will use the test data set to asses the performance of the fitted models based on train data set.
- Implementing 5-fold cross validation method while fitting with train data set is suggested.
- You can use any packs as long as you are 100% sure what it does and clear to the grader.
- Include compact other useful measurements and plots. Not too many! Report some useful results in a comparative table each.
- Include helpful compact plots with titles.
- Keep at most 4 decimals to present numbers and the performance scores.
- What other models could be used to get better results? This is an extra if you like to discuss.

Your Solutions

```
getwd() #gets what working directory is

# Create a RStudio Project and work under it.

#Download, Import and Assign
train <- read.csv("StudentDataTrain.csv")
test <- read.csv("StudentDataTest.csv")

## [1] "C:/Users/soumy/Documents/rochester/coursework/Computational
Introduction to statistics/Project1_midterm"

#Summarize univariately
summary(train)

##  Race_Ethc_Visa        Gender               HSGPA          SAT_Total
##  Length:5961        Length:5961        Min.   : 50.00   Min.   : 900
##  Class :character   Class :character   1st Qu.: 67.00   1st Qu.:1085
##  Mode  :character   Mode  :character   Median : 76.00   Median :1256
##                                        Mean   : 76.48   Mean   :1255
##                                        3rd Qu.: 86.00   3rd Qu.:1426
##                                        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.   : 1.000
##  1st Qu.:2141   1st Qu.:1.360   1st Qu.:1.0000       1st Qu.: 2.000
##  Median :2141   Median :2.230   Median :1.0000       Median : 3.000
##  Mean   :2143   Mean   :2.241   Mean   :0.7992       Mean   : 3.586
##  3rd Qu.:2151   3rd Qu.:3.130   3rd Qu.:1.0000       3rd Qu.: 5.000
##  Max.   :2151   Max.   :4.000   Max.   :1.0000       Max.   :11.000
##
##       N.Ws             N.DFs             N.As        N.PassedCourse
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 1.000
##  Median :0.0000   Median :0.0000   Median :1.0000   Median : 2.000
##  Mean   :0.5954   Mean   :0.7675   Mean   :0.7792   Mean   : 2.223
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 3.000
##  Max.   :6.0000   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.:0.3333            1st Qu.:0.5000   1st
Qu.:0.0000
##  Median : 3.000   Median :0.6667            Median :1.0000   Median
:0.0000
##  Mean   : 2.991   Mean   :0.6156            Mean   :0.7398   Mean
:0.1687
##  3rd Qu.: 4.000   3rd Qu.:1.0000            3rd Qu.:1.0000   3rd
Qu.:0.3333
```

```
##  Max.   :11.000   Max.   :1.0000            Max.   :1.0000   Max.
:1.0000
##                                             NA's   :186
##  N.GraduateCourse FullTimeStudent
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.6182   Mean   :0.5685
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :5.0000   Max.   :1.0000
##
```

```
summary(test)
```

```
##  Race_Ethc_Visa        Gender              HSGPA          SAT_Total
##  Length:1474        Length:1474        Min.   : 50.00   Min.   : 900
##  Class :character   Class :character   1st Qu.: 67.00   1st Qu.:1081
##  Mode  :character   Mode  :character   Median : 75.00   Median :1252
##                                        Mean   : 76.62   Mean   :1254
##                                        3rd Qu.: 87.00   3rd Qu.:1418
##                                        Max.   :100.00   Max.   :1600
##
##    Entry_Term      Term.GPA     Persistence.NextYear N.RegisteredCourse
##  Min.   :2131   Min.   :0.500   Min.   :0.0000       Min.   : 1.000
##  1st Qu.:2131   1st Qu.:1.442   1st Qu.:1.0000       1st Qu.: 2.000
##  Median :2131   Median :2.270   Median :1.0000       Median : 3.000
##  Mean   :2132   Mean   :2.266   Mean   :0.9016       Mean   : 3.554
##  3rd Qu.:2131   3rd Qu.:3.110   3rd Qu.:1.0000       3rd Qu.: 5.000
##  Max.   :2141   Max.   :4.000   Max.   :1.0000       Max.   :10.000
##
##       N.Ws            N.DFs              N.As       N.PassedCourse
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.: 2.000
##  Median :0.0000   Median :0.00000   Median :1.000   Median : 3.000
##  Mean   :0.3412   Mean   :0.05156   Mean   :1.121   Mean   : 3.161
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:2.000   3rd Qu.: 4.000
##  Max.   :4.0000   Max.   :4.00000   Max.   :7.000   Max.   :10.000
##
##  N.CourseTaken    Perc.PassedEnrolledCourse   Perc.Pass        Perc.Withd
##  Min.   : 0.000   Min.   :0.0000            Min.   :0.0000   Min.
:0.00000
##  1st Qu.: 2.000   1st Qu.:0.8333            1st Qu.:1.0000   1st
Qu.:0.00000
##  Median : 3.000   Median :1.0000            Median :1.0000   Median
:0.00000
##  Mean   : 3.213   Mean   :0.8896            Mean   :0.9815   Mean
:0.09517
##  3rd Qu.: 4.000   3rd Qu.:1.0000            3rd Qu.:1.0000   3rd
Qu.:0.12500
##  Max.   :10.000   Max.   :1.0000            Max.   :1.0000   Max.
```

```
:1.00000
##                                                            NA's   :28
##  N.GraduateCourse FullTimeStudent
##  Min.   :0.000    Min.   :0.0000
##  1st Qu.:0.000    1st Qu.:0.0000
##  Median :0.000    Median :1.0000
##  Mean   :0.614    Mean   :0.6201
##  3rd Qu.:1.000    3rd Qu.:1.0000
##  Max.   :4.000    Max.   :1.0000
##
```

```
#Dims
dim(train) #5961x18
dim(test) #1474x18
```

```
## [1] 5961   18
## [1] 1474   18
```

```
#Without NA's
dim(na.omit(train)) #5757x18
dim(na.omit(test)) #1445x18
```

```
## [1] 5757   18
## [1] 1445   18
```

```
#Perc of complete cases
sum(complete.cases(train))/nrow(train)
sum(complete.cases(test))/nrow(test)
```

```
## [1] 0.9657776
## [1] 0.9803256
```

```
#Delete or not? Don't delete!! Use Imputation method to fill na's
train <- na.omit(train)
test <- na.omit(test)
dim(train)
```

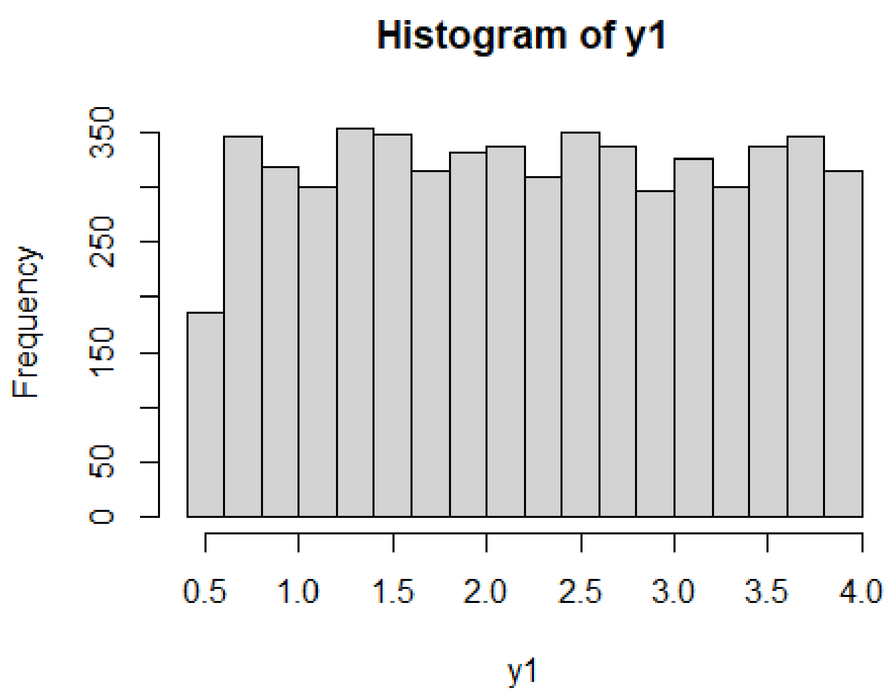
```
## [1] 5757   18
```

#you can create new columns based on features

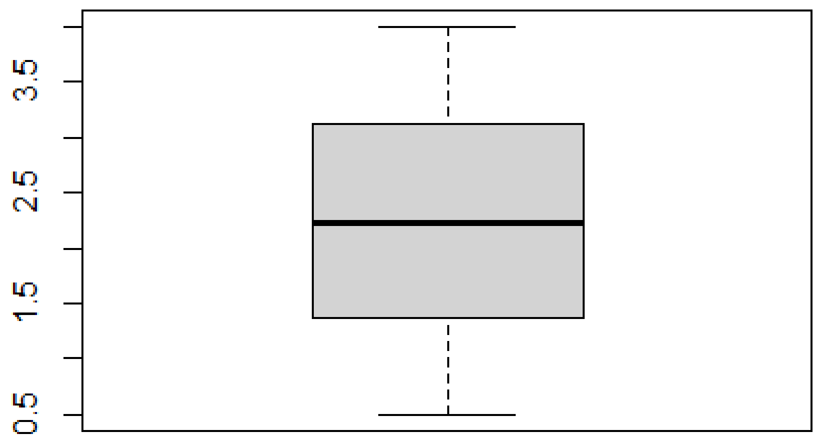
```
#Variable/Column names
colnames(test)
```

```
##  [1] "Race_Ethc_Visa"            "Gender"
##  [3] "HSGPA"                     "SAT_Total"
##  [5] "Entry_Term"                "Term.GPA"
##  [7] "Persistence.NextYear"      "N.RegisteredCourse"
##  [9] "N.Ws"                      "N.DFs"
## [11] "N.As"                      "N.PassedCourse"
## [13] "N.CourseTaken"             "Perc.PassedEnrolledCourse"
```

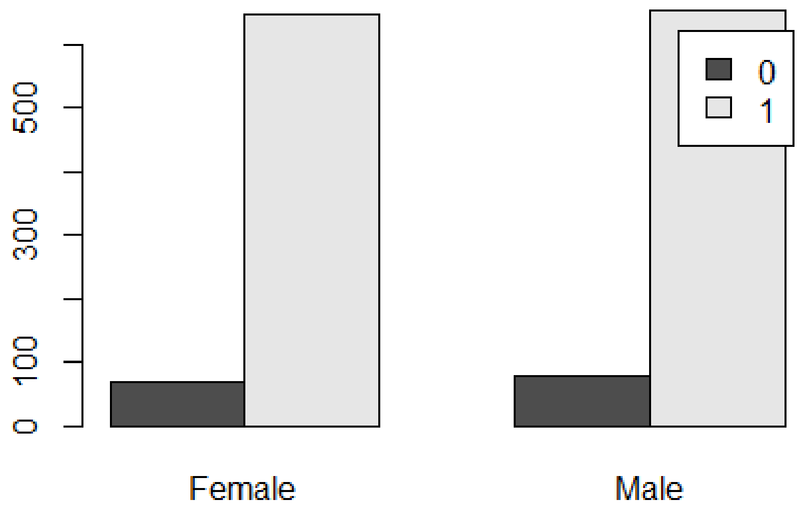
```
## [15] "Perc.Pass"                   "Perc.Withd"
## [17] "N.GraduateCourse"            "FullTimeStudent"

#Response variables
#Do this for train after processing the data AND for test data sets)
y1=train$Term.GPA #numerical
y2=train$Persistence.NextYear #categorical

##Summarize
#y1
hist(y1)
```



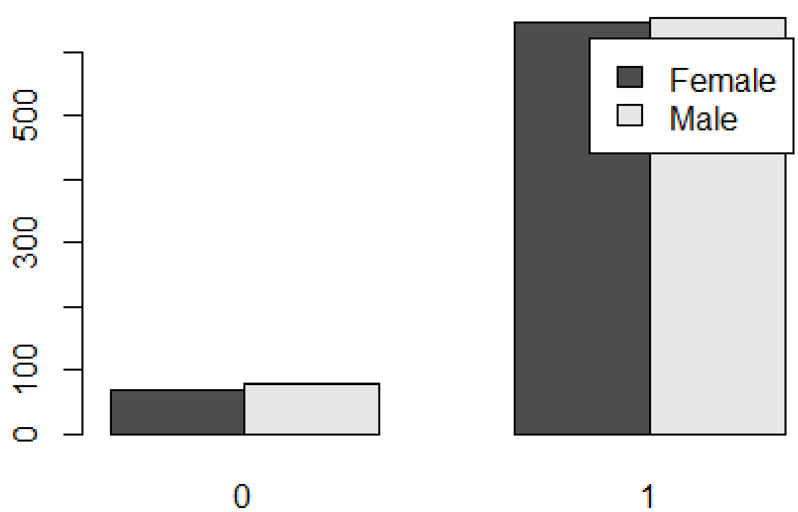

```
boxplot(y1)
```



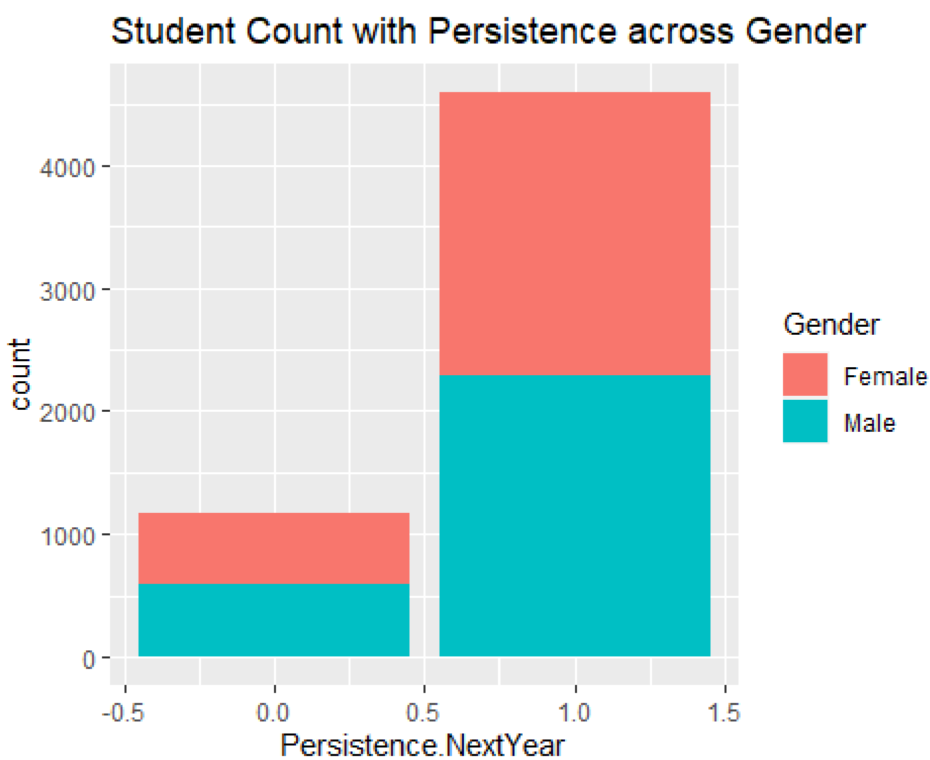

```
#y2: 0 - not persistent (drop), 1 - persistent (stay)
table(y2)

## y2
##    0    1
## 1167 4590

#Persistence
aa=table(test$Persistence.NextYear, test$Gender)
addmargins(aa)
prop.table(aa,2)
barplot(aa,beside=TRUE,legend=TRUE) #counts
```



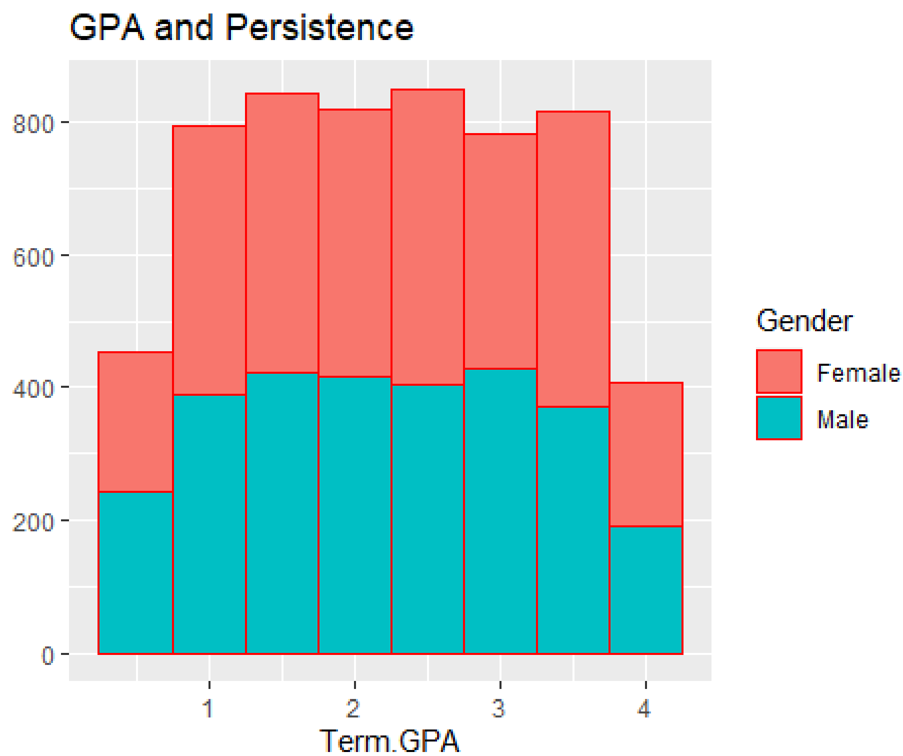

```
barplot(t(aa),beside=TRUE,legend=TRUE)
```



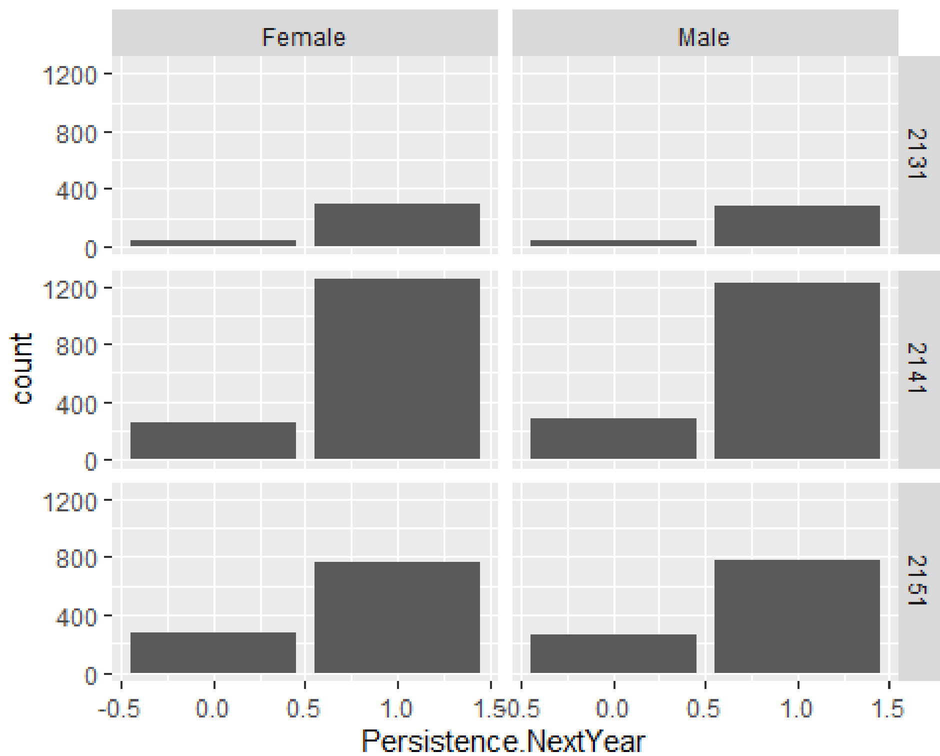
```
##
##       Female Male  Sum
##   0       67   77  144
##   1      648  653 1301
##   Sum    715  730 1445
##
##          Female       Male
##   0 0.09370629 0.10547945
##   1 0.90629371 0.89452055

#ggplots: just read more with help(ggplot2) and play
## Persistence percent by Year
library(ggplot2)
ggplot(data=train, aes(x=Persistence.NextYear, fill=Gender))+          #,
fill=gender
  geom_bar(stat="count")+ ggtitle("Student Count with Persistence across
Gender") #bar chart
```



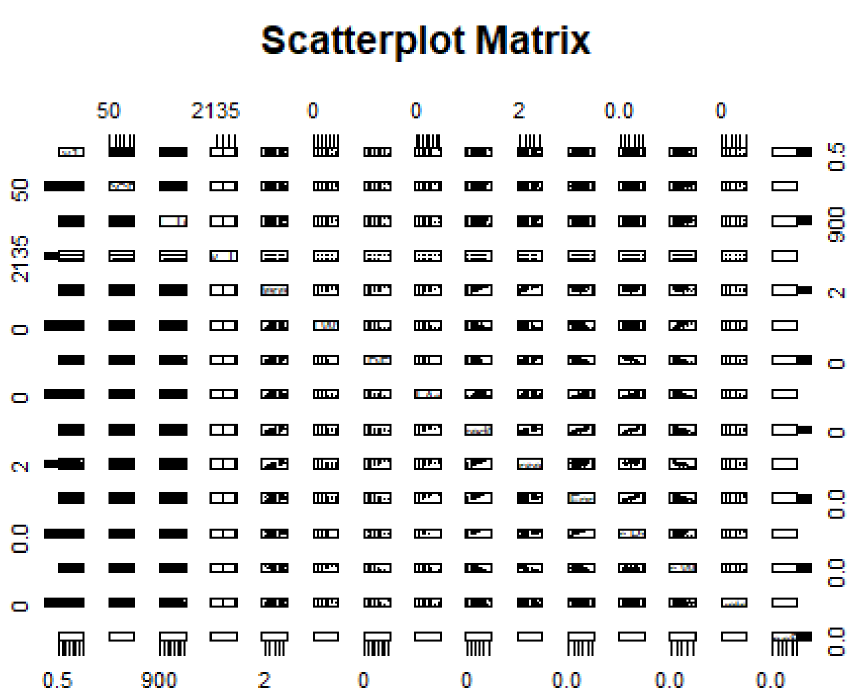

```
##GPA and Persistence by Gender and (Entry) Years
qplot(Term.GPA, data=train[which(!is.na(train$Gender)),], fill=Gender, colour
= I("red"),
      na.rm = TRUE, main="GPA and Persistence", binwidth=.5)
```




```
##just template
ggplot(data=train, aes(x=Persistence.NextYear))+        #, fill=gender
  geom_bar(stat="count", position = position_dodge())+  #stat="bin"
  facet_grid(Entry_Term ~ Gender)
```




```
pairs(y1~HSGPA+SAT_Total+Entry_Term+N.RegisteredCourse+N.Ws+N.DFs+N.As+N.PassedCourse+N.CourseTaken+Perc.PassedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCourse+FullTimeStudent, data= train, main="Scatterplot Matrix", pch='.')
```



```
cor_mat=cor(train[sapply(train, is.numeric)])
round(cor_mat, 2)
```

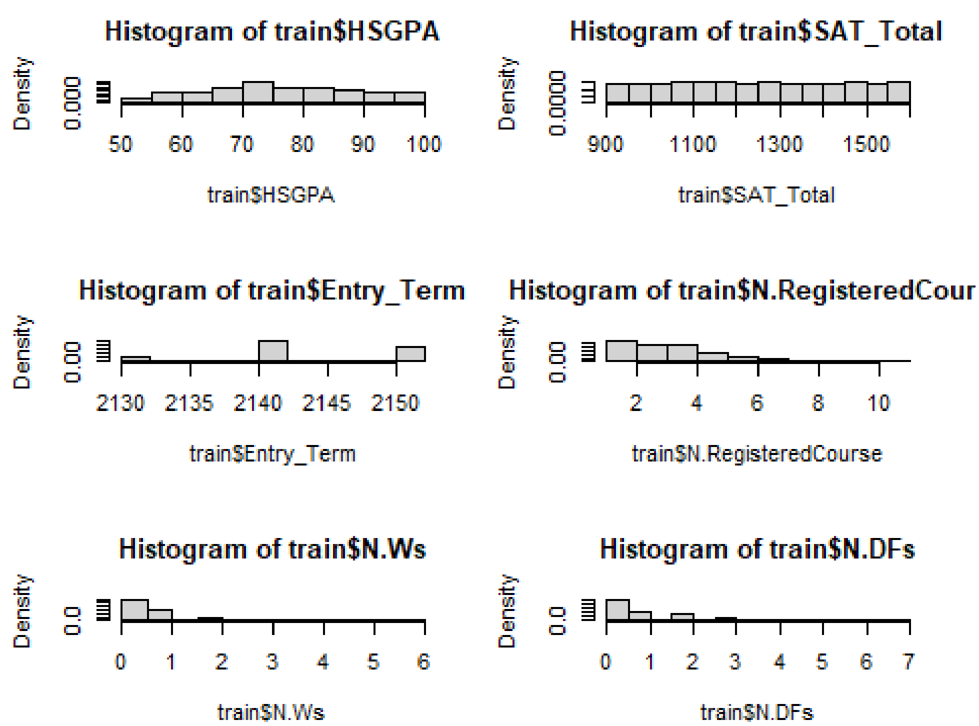
```
##                            HSGPA SAT_Total Entry_Term Term.GPA
## HSGPA                       1.00      0.00      -0.10     0.06
## SAT_Total                   0.00      1.00      -0.04    -0.02
## Entry_Term                 -0.10     -0.04       1.00     0.00
## Term.GPA                    0.06     -0.02       0.00     1.00
## Persistence.NextYear        0.17     -0.01      -0.11     0.48
## N.RegisteredCourse          0.02      0.00       0.00     0.00
## N.Ws                        0.02      0.02      -0.18    -0.01
## N.DFs                      -0.03     -0.02       0.23     0.01
## N.As                        0.01     -0.01      -0.12     0.02
## N.PassedCourse              0.03      0.00      -0.06     0.00
## N.CourseTaken               0.01     -0.01       0.09     0.00
## Perc.PassedEnrolledCourse  0.02      0.00      -0.08     0.00
## Perc.Pass                   0.03      0.02      -0.23     0.00
## Perc.Withd                  0.01      0.03      -0.19    -0.01
## N.GraduateCourse            0.02     -0.02       0.00     0.01
## FullTimeStudent             0.00      0.00       0.08     0.01
##                            Persistence.NextYear N.RegisteredCourse  N.Ws
N.DFs
## HSGPA                                      0.17               0.02  0.02 -
0.03
## SAT_Total                                 -0.01               0.00  0.02 -
0.02
## Entry_Term                                -0.11               0.00 -0.18
0.23
## Term.GPA                                   0.48               0.00 -0.01
0.01
## Persistence.NextYear                       1.00               0.00  0.02 -
0.07
## N.RegisteredCourse                         0.00               1.00  0.39
0.34
## N.Ws                                       0.02               0.39  1.00
0.00
## N.DFs                                     -0.07               0.34  0.00
1.00
## N.As                                       0.08               0.39  0.01 -
0.18
## N.PassedCourse                             0.04               0.69 -0.08 -
0.28
## N.CourseTaken                             -0.01               0.89 -0.08
0.37
## Perc.PassedEnrolledCourse                  0.05              -0.05 -0.42 -
0.66
## Perc.Pass                                  0.07               0.02  0.00 -
0.79
## Perc.Withd                                 0.01               0.14  0.89 -
0.09
```

```
## N.GraduateCourse                            0.00               0.39  0.15
0.12
## FullTimeStudent                             0.00               0.68 -0.07
0.30
##                           N.As N.PassedCourse N.CourseTaken
## HSGPA                     0.01           0.03          0.01
## SAT_Total                -0.01           0.00         -0.01
## Entry_Term               -0.12          -0.06          0.09
## Term.GPA                  0.02           0.00          0.00
## Persistence.NextYear      0.08           0.04         -0.01
## N.RegisteredCourse        0.39           0.69          0.89
## N.Ws                      0.01          -0.08         -0.08
## N.DFs                    -0.18          -0.28          0.37
## N.As                      1.00           0.55          0.42
## N.PassedCourse            0.55           1.00          0.79
## N.CourseTaken             0.42           0.79          1.00
## Perc.PassedEnrolledCourse  0.33           0.60          0.16
## Perc.Pass                 0.33           0.54          0.02
## Perc.Withd               -0.10          -0.25         -0.29
## N.GraduateCourse          0.15           0.28          0.34
## FullTimeStudent           0.32           0.60          0.77
##                          Perc.PassedEnrolledCourse Perc.Pass Perc.Withd
## HSGPA                                         0.02      0.03       0.01
## SAT_Total                                     0.00      0.02       0.03
## Entry_Term                                   -0.08     -0.23      -0.19
## Term.GPA                                      0.00      0.00      -0.01
## Persistence.NextYear                          0.05      0.07       0.01
## N.RegisteredCourse                           -0.05      0.02       0.14
## N.Ws                                         -0.42      0.00       0.89
## N.DFs                                        -0.66     -0.79      -0.09
## N.As                                          0.33      0.33      -0.10
## N.PassedCourse                                0.60      0.54      -0.25
## N.CourseTaken                                 0.16      0.02      -0.29
## Perc.PassedEnrolledCourse                     1.00      0.86      -0.48
## Perc.Pass                                     0.86      1.00      -0.01
## Perc.Withd                                   -0.48     -0.01       1.00
## N.GraduateCourse                             -0.01      0.01       0.06
## FullTimeStudent                               0.14      0.01      -0.27
##                          N.GraduateCourse FullTimeStudent
## HSGPA                                0.02            0.00
## SAT_Total                           -0.02            0.00
## Entry_Term                           0.00            0.08
## Term.GPA                             0.01            0.01
## Persistence.NextYear                 0.00            0.00
## N.RegisteredCourse                   0.39            0.68
## N.Ws                                 0.15           -0.07
## N.DFs                                0.12            0.30
## N.As                                 0.15            0.32
## N.PassedCourse                       0.28            0.60
## N.CourseTaken                        0.34            0.77
```

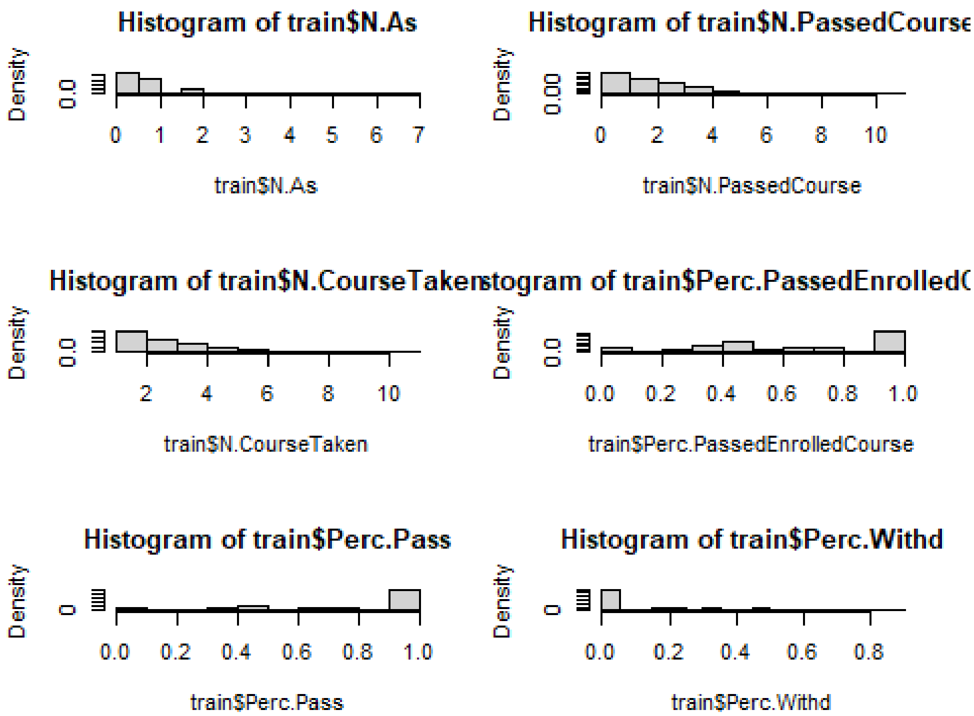
```
## Perc.PassedEnrolledCourse            -0.01            0.14
## Perc.Pass                             0.01            0.01
## Perc.Withd                            0.06           -0.27
## N.GraduateCourse                      1.00            0.27
## FullTimeStudent                       0.27            1.00
```

So correlation matrix shows that Term.GPA and Persistence.NextYear have good correlation.

```r
# graph analysis (histogram plot)
par(mfrow=c(3,2))
hist(train$HSGPA, prob=T)
hist(train$SAT_Total, prob=T)
hist(train$Entry_Term, prob=T)
hist(train$N.RegisteredCourse, prob=T)
hist(train$N.Ws, prob=T)
hist(train$N.DFs, prob=T)
```




```r
par(mfrow=c(3,2))
hist(train$N.As, prob=T)
hist(train$N.PassedCourse, prob=T)
hist(train$N.CourseTaken, prob=T)
hist(train$Perc.PassedEnrolledCourse, prob=T)
hist(train$Perc.Pass, prob=T)
hist(train$Perc.Withd, prob=T)
```


Histogram of train$N.As
Density
train$N.As
Histogram of train$N.PassedCourse
train$N.PassedCourse
Histogram of train$N.CourseTaken
train$N.CourseTaken
train$Perc.PassedEnrolledCourse
Histogram of train$Perc.Pass
train$Perc.Pass
Histogram of train$Perc.Withd
train$Perc.Withd

Section B

- Model 1.

Here, we see that y1 (Term.GPA) has no good correlation with any of the predictors from the correlation matrix plot in Section A. So we will check a few variables one ata a time based on correlation with y1,i.e., HSGPA, Sat_Total, N.As

For each case we do a Cross validation test

```
## k-Fold CV
k=5
set.seed(99)
folds=sample(1:k,nrow(train),replace=TRUE)

#install.packages('modelr')
library(modelr)

## Warning: package 'modelr' was built under R version 4.0.4
```

First we find if any variables are collinear

```
train$genderD <- ifelse(train$Gender=="Male", 1, 0)
test$genderD <- ifelse(test$Gender=="Male", 1, 0)
# VIF
library(car)

## Loading required package: carData

lm.fit=lm(Term.GPA~HSGPA+SAT_Total+N.RegisteredCourse+N.Ws+N.DFs+N.As+Perc.Pa
ssedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCourse+FullTimeStudent+gend
erD,data=train)
vif(lm.fit)

##                      HSGPA                  SAT_Total
N.RegisteredCourse
##                   1.003047                   1.003179
4.519071
##                       N.Ws                      N.DFs
N.As
##                   7.493807                   5.067874
1.448187
## Perc.PassedEnrolledCourse                  Perc.Pass
Perc.Withd
##                  23.288664                  17.263960
10.736864
##           N.GraduateCourse            FullTimeStudent
genderD
##                   1.178131                   2.514490
1.002481
```

So from the multicolinearity test by VIF, we observe that some predictors are collinear whose VIF > 5 .

In doing so we reduce our variable set to HSGPA, SAT_Total, N.As, N.GraduateCourse, FullTimeStuden, genderD, N.RegisteredCourse

```
# first we will predict y1 based on HSGPA
#For OLS method find train_MSE and test_MSE

cv.errors_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
cv.errors_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
# now, looping/k-fold procedure
for(j in 1:k){
  best.fit=lm(Term.GPA~HSGPA,data=train[folds!=j,])
    # predict the held data for validation MSE
  pred=predict(best.fit,train[folds==j,],interval="confidence")[,1]
  cv.errors_valid_OLS1[j,]=mean( (train$Term.GPA[folds==j]-pred)^2)
  Rsquare_valid_OLS1[j,]=rsquare(best.fit, train[folds==j,])
  # predict the train MSE for k-1 dataset
  pred2=predict(best.fit,train[folds!=j,],interval="confidence")[,1]
  cv.errors_train_OLS1[j,]=mean( (train$Term.GPA[folds!=j]-pred2)^2)
  Rsquare_train_OLS1[j,]=rsquare(best.fit, train[folds!=j,])
}
mean.cv.errors_train_OLS1=apply(cv.errors_train_OLS1,2,mean)
mean.cv.errors_valid_OLS1=apply(cv.errors_valid_OLS1,2,mean)
mean.Rsquare_train_OLS1=apply(Rsquare_train_OLS1,2,mean)
mean.Rsquare_valid_OLS1=apply(Rsquare_valid_OLS1,2,mean)

pred_test=predict(best.fit,test,interval="confidence")[,1]
mean.cv.errors_test_OLS1=mean( (test$Term.GPA-pred_test)^2)
mean.Rsquare_test_OLS1=rsquare(best.fit, test)



MSE_OLS=c(mean.cv.errors_train_OLS1,mean.cv.errors_valid_OLS1,mean.cv.errors_
test_OLS1)
Rsq_OLS=c(mean.Rsquare_train_OLS1,mean.Rsquare_valid_OLS1,mean.Rsquare_test_O
LS1)
SS_OLS <- cbind(MSE_OLS,Rsq_OLS)
SS_OLS=round(SS_OLS,3)
colnames(SS_OLS) <- c("MSE","Rsq_adj")
rownames(SS_OLS) <- c("Train set", "Valid set", "test set")
addmargins(SS_OLS)
knitr::kable(SS_OLS, caption = "SLR Model quality check HSGPA predictor")

##             MSE Rsq_adj   Sum
## Train set 1.025   0.004 1.029
## Valid set 1.026   0.003 1.029
```

```
## test set  0.991   0.003 0.994
## Sum       3.042   0.010 3.052
```

SLR Model quality check HSGPA predictor

	MSE	Rsq_adj
Train set	1.025	0.004
Valid set	1.026	0.003
test set	0.991	0.003

```
# we will predict y1 based on SAT_Total
#For OLS method find train_MSE and test_MSE

cv.errors_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
cv.errors_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
# now, looping/k-fold procedure
for(j in 1:k){
  best.fit=lm(Term.GPA~SAT_Total,data=train[folds!=j,])
    # predict the held data for validation MSE
  pred=predict(best.fit,train[folds==j,],interval="confidence")[,1]
  cv.errors_valid_OLS1[j,]=mean( (train$Term.GPA[folds==j]-pred)^2)
  Rsquare_valid_OLS1[j,]=rsquare(best.fit, train[folds==j,])
  # predict the train MSE for k-1 dataset
  pred2=predict(best.fit,train[folds!=j,],interval="confidence")[,1]
  cv.errors_train_OLS1[j,]=mean( (train$Term.GPA[folds!=j]-pred2)^2)
  Rsquare_train_OLS1[j,]=rsquare(best.fit, train[folds!=j,])
}
mean.cv.errors_train_OLS1=apply(cv.errors_train_OLS1,2,mean)
mean.cv.errors_valid_OLS1=apply(cv.errors_valid_OLS1,2,mean)
mean.Rsquare_train_OLS1=apply(Rsquare_train_OLS1,2,mean)
mean.Rsquare_valid_OLS1=apply(Rsquare_valid_OLS1,2,mean)

pred_test=predict(best.fit,test,interval="confidence")[,1]
mean.cv.errors_test_OLS1=mean( (test$Term.GPA-pred_test)^2)
mean.Rsquare_test_OLS1=rsquare(best.fit, test)



MSE_OLS=c(mean.cv.errors_train_OLS1,mean.cv.errors_valid_OLS1,mean.cv.errors_
test_OLS1)
Rsq_OLS=c(mean.Rsquare_train_OLS1,mean.Rsquare_valid_OLS1,mean.Rsquare_test_O
LS1)
SS_OLS <- cbind(MSE_OLS,Rsq_OLS)
SS_OLS=round(SS_OLS,3)
colnames(SS_OLS) <- c("MSE","Rsq_adj")
rownames(SS_OLS) <- c("Train set", "Valid set", "test set")
addmargins(SS_OLS)
knitr::kable(SS_OLS, caption = "SLR Model quality check SAT_Total")
```

```
##              MSE Rsq_adj   Sum
## Train set 1.028   0.001 1.029
## Valid set 1.029   0.000 1.029
## test set  0.997  -0.002 0.995
## Sum       3.054  -0.001 3.053
```

SLR Model quality check SAT_Total

	MSE	Rsq_adj
Train set	1.028	0.001
Valid set	1.029	0.000
test set	0.997	-0.002

```
# we will predict y1 based on N.As
#For OLS method find train_MSE and test_MSE

cv.errors_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
cv.errors_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_train_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
Rsquare_valid_OLS1=matrix(NA,k,1, dimnames=list(NULL, paste("OLS")))
# now, looping/k-fold procedure
for(j in 1:k){
  best.fit=lm(Term.GPA~N.As,data=train[folds!=j,])
    # predict the held data for validation MSE
  pred=predict(best.fit,train[folds==j,],interval="confidence")[,1]
  cv.errors_valid_OLS1[j,]=mean( (train$Term.GPA[folds==j]-pred)^2)
  Rsquare_valid_OLS1[j,]=rsquare(best.fit, train[folds==j,])
  # predict the train MSE for k-1 dataset
  pred2=predict(best.fit,train[folds!=j,],interval="confidence")[,1]
  cv.errors_train_OLS1[j,]=mean( (train$Term.GPA[folds!=j]-pred2)^2)
  Rsquare_train_OLS1[j,]=rsquare(best.fit, train[folds!=j,])
}
mean.cv.errors_train_OLS1=apply(cv.errors_train_OLS1,2,mean)
mean.cv.errors_valid_OLS1=apply(cv.errors_valid_OLS1,2,mean)
mean.Rsquare_train_OLS1=apply(Rsquare_train_OLS1,2,mean)
mean.Rsquare_valid_OLS1=apply(Rsquare_valid_OLS1,2,mean)

pred_test=predict(best.fit,test,interval="confidence")[,1]
mean.cv.errors_test_OLS1=mean( (test$Term.GPA-pred_test)^2)
mean.Rsquare_test_OLS1=rsquare(best.fit, test)


MSE_OLS=c(mean.cv.errors_train_OLS1,mean.cv.errors_valid_OLS1,mean.cv.errors_
test_OLS1)
Rsq_OLS=c(mean.Rsquare_train_OLS1,mean.Rsquare_valid_OLS1,mean.Rsquare_test_O
LS1)
SS_OLS <- cbind(MSE_OLS,Rsq_OLS)
SS_OLS=round(SS_OLS,3)
colnames(SS_OLS) <- c("MSE","Rsq_adj")
```

```r
rownames(SS_OLS) <- c("Train set", "Valid set", "test set")
addmargins(SS_OLS)
knitr::kable(SS_OLS, caption = "SLR Model quality check N.As")

##              MSE Rsq_adj   Sum
## Train set 1.028       0 1.028
## Valid set 1.030       0 1.030
## test set  0.994       0 0.994
## Sum       3.052       0 3.052
```

SLR Model quality check N.As

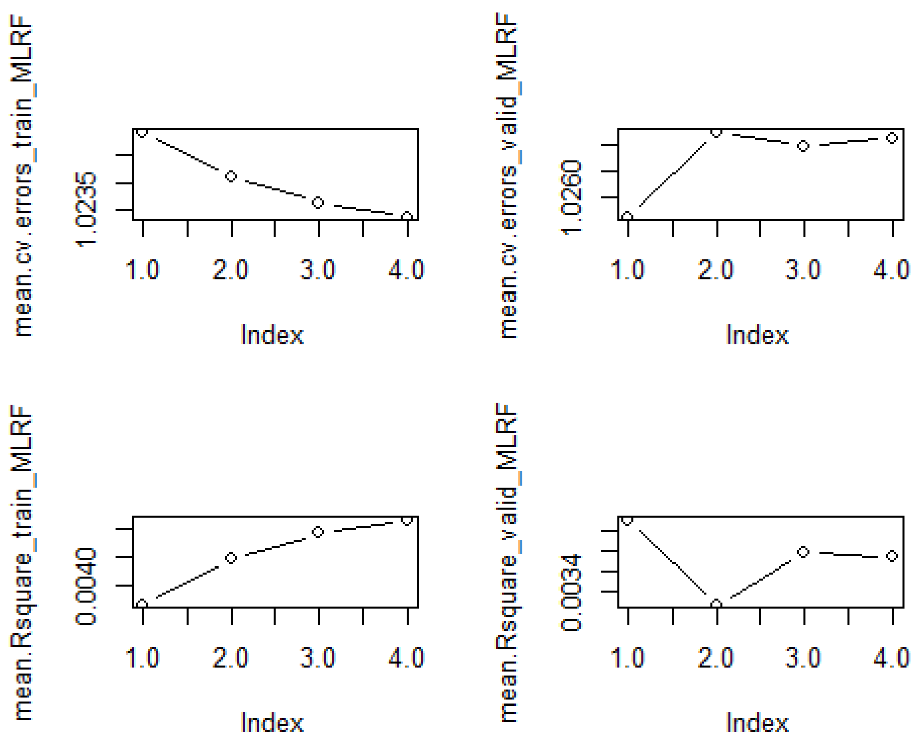
	MSE	Rsq_adj
Train set	1.028	0
Valid set	1.030	0
test set	0.994	0

So from The Rsq-adjusted value we see that for predictors SAT_Total and N.As are negative meaning very poor. Only for HSGPA we get positive Rsq-adjusted although the value is very less. So the best model of OLS SLR is with predictor HSGPA. ***

- Model 2.

```r
#For Forward selection method
library(leaps)

library(caret)

## Loading required package: lattice

predict_regsubsets=function(object, newdata, id, ...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi #prediction or fitted results
}
predictor_no=4
cv.errors_valid_MLRF=matrix(NA,k,predictor_no, dimnames=list(NULL,
paste(1:predictor_no)))
cv.errors_train_MLRF=matrix(NA,k,predictor_no, dimnames=list(NULL,
paste(1:predictor_no)))
Rsquare_train_MLRF=matrix(NA,k,predictor_no, dimnames=list(NULL,
paste(1:predictor_no)))
Rsquare_valid_MLRF=matrix(NA,k,predictor_no, dimnames=list(NULL,
paste(1:predictor_no)))
# now, looping/k-fold procedure
for(j in 1:k){

best.fit=regsubsets(Term.GPA~HSGPA+SAT_Total+N.As+N.GraduateCourse+FullTimeSt
```

```
udent+N.RegisteredCourse+genderD,data=train[folds!=j,],nvmax=6, method = 
"forward")
  for(i in 1:predictor_no){
    # predict the held data for test MSE
    pred=predict_regsubsets(best.fit,train[folds==j,],id=i)
    cv.errors_valid_MLRF[j,i]=mean( (train$Term.GPA[folds==j]-pred)^2)
    Rsquare_valid_MLRF[j,i]=R2(pred, train$Term.GPA[folds==j])
  # predict the train MSE for k-1 dataset
    pred2=predict_regsubsets(best.fit,train[folds!=j,],id=i)
    cv.errors_train_MLRF[j,i]=mean( (train$Term.GPA[folds!=j]-pred2)^2)
    Rsquare_train_MLRF[j,i]=R2(pred2, train$Term.GPA[folds!=j])
  }
}
mean.cv.errors_train_MLRF=apply(cv.errors_train_MLRF,2,mean)
mean.cv.errors_valid_MLRF=apply(cv.errors_valid_MLRF,2,mean)
mean.Rsquare_train_MLRF=apply(Rsquare_train_MLRF,2,mean)
mean.Rsquare_valid_MLRF=apply(Rsquare_valid_MLRF,2,mean)
par(mfrow=c(2,2))
plot(mean.cv.errors_train_MLRF,type='b')
plot(mean.cv.errors_valid_MLRF,type='b')
plot(mean.Rsquare_train_MLRF,type='b')
plot(mean.Rsquare_valid_MLRF,type='b')
```



So the best OLS MLR model is with 3 predictors as evident from MSE and Rsquared graph of validation set.

```
coef(best.fit,3)
```

```
##   (Intercept)         HSGPA     SAT_Total       genderD 
##  2.1012637119  0.0049074880 -0.0001635333 -0.0534801439
```

So the best 3 combinations of predictors are HSGPA , SAT_Total and genderD

```
pred_test=predict_regsubsets(best.fit,test,id=3)
mean.cv.errors_test_MLRF=mean( (test$Term.GPA-pred_test)^2)
mean.Rsquare_test_MLRF=R2(pred_test, test$Term.GPA)


MSE_MLRF=c(mean.cv.errors_train_MLRF[3],mean.cv.errors_valid_MLRF[3],mean.cv.
errors_test_MLRF)
Rsq_MLRF=c(mean.Rsquare_train_MLRF[3],mean.Rsquare_valid_MLRF[3],mean.Rsquare
_test_MLRF)
SS_MLRF <- cbind(MSE_MLRF,Rsq_MLRF)
SS_MLRF=round(SS_MLRF,3)
colnames(SS_MLRF) <- c("MSE","Rsq_adj")
rownames(SS_MLRF) <- c("Train set", "Valid set", "test set")
addmargins(SS_MLRF)
knitr::kable(SS_MLRF, caption = "MLR Model quality")
```

```
##             MSE Rsq_adj   Sum
## Train set 1.024   0.005 1.029
## Valid set 1.027   0.004 1.031
## test set  0.997   0.000 0.997
## Sum       3.048   0.009 3.057
```

MLR Model quality

	MSE	Rsq_adj
Train set	1.024	0.005
Valid set	1.027	0.004
test set	0.997	0.000

- Model 3.

Ridge regression

```
# Ridge Regression
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1
```

```
# grid for lambda
grid=10^seq(10,-2,length=100)
x=model.matrix(Term.GPA~HSGPA+SAT_Total+N.As+N.GraduateCourse+FullTimeStudent
+N.RegisteredCourse+genderD,train)[,-1]
```

```r
y=na.omit(train$Term.GPA)

x_test=model.matrix(Term.GPA~HSGPA+SAT_Total+N.As+N.GraduateCourse+FullTimeSt
udent+N.RegisteredCourse+genderD,test)[,-1]
y_test=na.omit(test$Term.GPA)

library(glmnet)
#For Ridge regression method find train_MSE and test_MSE
cv.errors_train_Ridge=matrix(NA,k,1, dimnames=list(NULL, paste("ridge")))
cv.errors_valid_Ridge=matrix(NA,k,1, dimnames=list(NULL, paste("ridge")))
Rsquare_train_Ridge=matrix(NA,k,1, dimnames=list(NULL, paste("ridge")))
Rsquare_valid_Ridge=matrix(NA,k,1, dimnames=list(NULL, paste("ridge")))
# now, looping/k-fold procedure
for(j in 1:k){
  #fit on train data
  ridge.mod=glmnet(x[folds==j,],y[folds==j],alpha=0,lambda=grid, thresh=1e-
12)
  cv.out=cv.glmnet(x[folds==j,],y[folds==j],alpha=0)
  bestlam=cv.out$lambda.min
  # predict the held data for test MSE
  ridge.pred=predict(ridge.mod,s=bestlam,newx=x[folds!=j,])
  cv.errors_valid_Ridge[j,]=mean((ridge.pred-y[folds!=j])^2) #test MSE
associated with best lambda
  Rsquare_valid_Ridge[j,]=R2(ridge.pred, y[folds!=j])
  # predict the train MSE for k-1 dataset
  ridge.pred=predict(ridge.mod,s=bestlam,newx=x[folds==j,])
  cv.errors_train_Ridge[j,]=mean((ridge.pred-y[folds==j])^2)
  Rsquare_train_Ridge[j,]=R2(ridge.pred, y[folds==j])
}
mean.cv.errors_train_ridge=apply(cv.errors_train_Ridge,2,mean)
mean.cv.errors_valid_ridge=apply(cv.errors_valid_Ridge,2,mean)
mean.Rsquare_train_ridge=apply(Rsquare_train_Ridge,2,mean)
mean.Rsquare_valid_ridge=apply(Rsquare_valid_Ridge,2,mean)

ridge.pred2=predict(ridge.mod,s=bestlam,newx=x_test)

mean.cv.errors_test_ridge=mean( (y_test-ridge.pred2)^2)
mean.Rsquare_test_ridge=R2(ridge.pred2, y_test)



MSE_ridge=c(mean.cv.errors_train_ridge,mean.cv.errors_valid_ridge,mean.cv.err
ors_test_ridge)
Rsq_ridge=c(mean.Rsquare_train_ridge,mean.Rsquare_valid_ridge,mean.Rsquare_te
st_ridge)
SS_ridge <- cbind(MSE_ridge,Rsq_ridge)
SS_ridge=round(SS_ridge,3)
colnames(SS_ridge) <- c("MSE","Rsq_adj")
rownames(SS_ridge) <- c("Train set", "Valid set", "test set")
addmargins(SS_ridge)
knitr::kable(SS_ridge, caption = "MLR Ridge Model quality")
```

```
##              MSE Rsq_adj   Sum
## Train set 1.023   0.010 1.033
## Valid set 1.028   0.002 1.030
## test set  0.996   0.004 1.000
## Sum       3.047   0.016 3.063
```

MLR Ridge Model quality

	MSE	Rsq_adj
Train set	1.023	0.010
Valid set	1.028	0.002
test set	0.996	0.004

```
#refit model with best lambda and get the coefficients
out1=glmnet(x,y,alpha=0)
predict(out1,type="coefficients",s=bestlam)[1:4,]
```

```
##   (Intercept)         HSGPA     SAT_Total          N.As
##  2.239958e+00  4.946112e-39 -1.156448e-40  2.068790e-38
```

Here the best subset of coefficints are HSGPA, SAT_Total and N.As ***

- Model 4.

Lasso expression

```
library(glmnet)
#For Lasso regression method find train_MSE and test_MSE
cv.errors_train_Lasso=matrix(NA,k,1, dimnames=list(NULL, paste("lasso")))
cv.errors_valid_Lasso=matrix(NA,k,1, dimnames=list(NULL, paste("lasso")))
Rsquare_train_Lasso=matrix(NA,k,1, dimnames=list(NULL, paste("lasso")))
Rsquare_valid_Lasso=matrix(NA,k,1, dimnames=list(NULL, paste("lasso")))
# now, looping/k-fold procedure
for(j in 1:k){
  #fit on train data
  lasso.mod=glmnet(x[folds==j,],y[folds==j],alpha=1,lambda=grid, thresh=1e-
12) # alpha = 1 for lasso
  cv.out=cv.glmnet(x[folds==j,],y[folds==j],alpha=1)
  bestlam=cv.out$lambda.min
  # predict the held data for test MSE
  lasso.pred=predict(lasso.mod,s=bestlam,newx=x[folds!=j,])
  cv.errors_valid_Lasso[j,]=mean((lasso.pred-y[folds!=j])^2) #test MSE
associated with best lambda
  Rsquare_valid_Lasso[j,]=R2(lasso.pred, y[folds!=j])
  # predict the train MSE for k-1 dataset
  lasso.pred=predict(lasso.mod,s=bestlam,newx=x[folds==j,])
  cv.errors_train_Lasso[j,]=mean((lasso.pred-y[folds==j])^2)
  Rsquare_train_Lasso[j,]=R2(lasso.pred, y[folds==j])
}
mean.cv.errors_train_lasso=apply(cv.errors_train_Lasso,2,mean)
mean.cv.errors_valid_lasso=apply(cv.errors_valid_Lasso,2,mean)
```

```
mean.Rsquare_train_lasso=apply(Rsquare_train_Lasso,2,mean)
mean.Rsquare_valid_lasso=apply(Rsquare_valid_Lasso,2,mean)

lasso.pred2=predict(lasso.mod,s=bestlam,newx=x_test)

mean.cv.errors_test_lasso=mean( (y_test-lasso.pred2)^2)
mean.Rsquare_test_lasso=R2(lasso.pred2, y_test)



MSE_lasso=c(mean.cv.errors_train_lasso,mean.cv.errors_valid_lasso,mean.cv.err
ors_test_lasso)
Rsq_lasso=c(mean.Rsquare_train_lasso,mean.Rsquare_valid_lasso,mean.Rsquare_te
st_lasso)
SS_lasso <- cbind(MSE_lasso,Rsq_lasso)
SS_lasso=round(SS_lasso,3)
colnames(SS_lasso) <- c("MSE","Rsq_adj")
rownames(SS_lasso) <- c("Train set", "Valid set", "test set")
addmargins(SS_lasso)
knitr::kable(SS_lasso, caption = "MLR Lasso Model quality")

##             MSE Rsq_adj   Sum
## Train set 1.023   0.008 1.031
## Valid set 1.029   0.002 1.031
## test set  0.995   0.003 0.998
## Sum       3.047   0.013 3.060
```

MLR Lasso Model quality

	MSE	Rsq_adj
Train set	1.023	0.008
Valid set	1.029	0.002
test set	0.995	0.003

```
#refit model with best lambda and get the coefficients
out1=glmnet(x,y,alpha=1)
predict(out1,type="coefficients",s=bestlam)[1:4,]

## (Intercept)       HSGPA   SAT_Total        N.As
##    2.239958    0.000000    0.000000    0.000000
```

From the lasso model we see that none of the predictors are doing good in predicting Term.GPA. Term.GPA is expressed by intercept which actually is not true. SO lasso expression is not doing good in that respect.

Section C

- Model 1.

```
perfcheck <- function(ct) {
  Accuracy <- (ct[1]+ct[4])/sum(ct)
  Recall <- ct[4]/sum((ct[2]+ct[4]))      #TP/P   or Power, Sensitivity, TPR
  Type1 <- ct[3]/sum((ct[1]+ct[3]))       #FP/N   or 1 - Specificity , FPR
  Precision <- ct[4]/sum((ct[3]+ct[4]))   #TP/P*
  Type2 <- ct[2]/sum((ct[2]+ct[4]))       #FN/P
  F1 <- 2/(1/Recall+1/Precision)
  Values <- as.vector(round(c(Accuracy, Recall, Type1, Precision, Type2,
F1),4)) *100
  Metrics = c("Accuracy", "Recall", "Type1", "Precision", "Type2", "F1")
  cbind(Metrics, Values)
  #list(Performance=round(Performance, 4))
}

# we will use cross validation first.
## k-Fold CV

k=5
set.seed(99)
folds=sample(1:k,nrow(train),replace=TRUE)
#logistic regression
error.log.valid=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
error.log.train=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
# now, looping/k-fold procedure
for(j in 1:k){

glm.fits=glm(Persistence.NextYear~Term.GPA+HSGPA+SAT_Total+N.RegisteredCourse
+N.Ws+N.DFs+N.As+Perc.PassedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCou
rse+FullTimeStudent,data=train[folds!=j,],family=binomial)
  glm.probs.train=predict(glm.fits,train[folds!=j,],type="response")

  glm.probs.valid=predict(glm.fits,train[folds==j,],type="response")

  # predict the held data for train MSE
  glm.pred.train=rep(0,length(glm.probs.train))
  glm.pred.train[glm.probs.train>.5]=1

error.log.train[j,]=mean(glm.pred.train!=train$Persistence.NextYear[folds!=j]
) #error

  # predict the held data for test MSE
  glm.pred.valid=rep(0,length(glm.probs.valid))
  glm.pred.valid[glm.probs.valid>.5]=1

error.log.valid[j,]=mean(glm.pred.valid!=train$Persistence.NextYear[folds==j]
```

```
) #error
}
mean.errors.log.train=apply(error.log.train,2,mean)
mean.errors.log.valid=apply(error.log.valid,2,mean)

cat('The MSE for train and valid set for logistic regression are',
mean.errors.log.train,'&', mean.errors.log.valid)

## The MSE for train and valid set for logistic regression are 0.1486868 &
0.1495244

#confusion matrix for logistic regression on train set
ct1_train=table(train$Persistence.NextYear[folds!=k], glm.pred.train)
cat('the confusion matrix is',sep="\n\n")

ct1_train
me_log_train=perfcheck(ct1_train)

## the confusion matrix is
##    glm.pred.train
##        0    1
##   0  465  474
##   1  199 3475

#confusion matrix for logistic regression on valid set
ct1_valid=table(train$Persistence.NextYear[folds==k], glm.pred.valid)
cat('the confusion matrix is',sep="\n\n")

ct1_valid
me_log_valid=perfcheck(ct1_train)

## the confusion matrix is
##    glm.pred.valid
##       0   1
##   0 113 115
##   1  61 855

#confusion matrix for logistic regression on test set
glm.probs.test=predict(glm.fits,test,type="response")
glm.pred.test=rep(0,length(glm.probs.test))
glm.pred.test[glm.probs.test>.5]=1
ct1_test=table(test$Persistence.NextYear, glm.pred.test)
cat('the confusion matrix is',sep="\n\n")

ct1_test
me_log_test=perfcheck(ct1_test)

## the confusion matrix is
##    glm.pred.test
##        0    1
```

```
## 0 75 69
## 1 69 1232
```

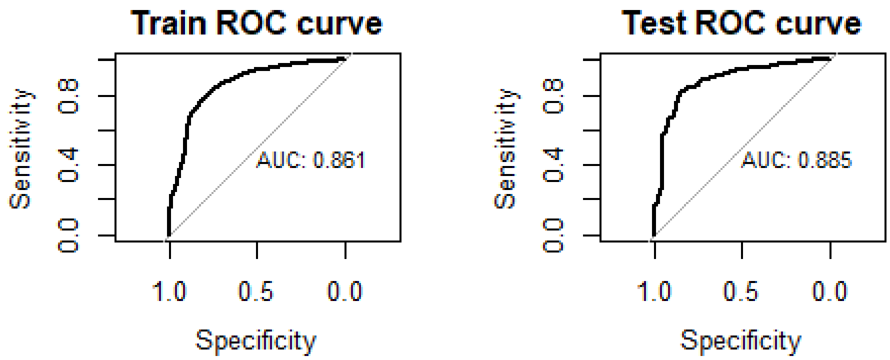
```
#performance metric for logistic regression
a=cbind(me_log_train[,1],me_log_train[,2],me_log_valid[,2],me_log_test[,2])
colnames(a)=c("Metrics","Train","valid","test")
knitr::kable(a, caption = "Model Metrics Logistic Regression")
```

Model Metrics Logistic Regression

Metrics	Train	valid	test
Accuracy	85.41	85.41	90.45
Recall	94.58	94.58	94.7
Type1	50.48	50.48	47.92
Precision	88	88	94.7
Type2	5.42	5.42	5.3
F1	91.17	91.17	94.7

So what we see is Logistic regression performs very good in terms of accuracy, precision, recall and F1-score for both train, valid and test set.There is not much gap between train and test set. So it suggests that data is well distributed.

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

par(mfrow=c(2,2))
train_prob = predict(glm.fits, newdata = train, type = "response")
train_roc = roc(train$Persistence.NextYear ~ train_prob, plot = TRUE,
print.auc = TRUE, main="Train ROC curve")

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

test_prob = predict(glm.fits, newdata = test, type = "response")
test_roc = roc(test$Persistence.NextYear ~ test_prob, plot = TRUE, print.auc
= TRUE, main="Test ROC curve")

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



So the ROC curve for train and test set shows AUC of 0.861 and 0.885. So we are doing a very decent job.

- Model 2.

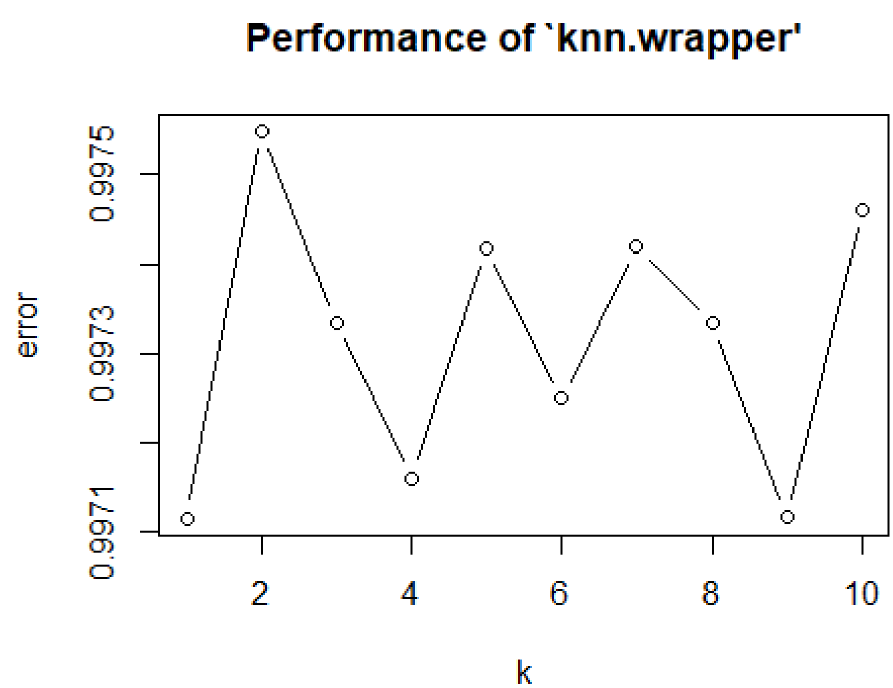
#knn with grid search

```
#install.packages('e1071')
library('e1071')

## Warning: package 'e1071' was built under R version 4.0.4

x1 <- train[,-1]
x =x1[,-1]
x=x[,-3]
x=x[,-4]
x=x[,-4]
y <- factor(train[,6])# getting the response variable
obj2 <- tune.knn(x, y, k = 1:10, tunecontrol = tune.control(sampling =
"boot"))
summary(obj2)

##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: bootstrapping
##
```

```
## - best parameters:
##  k
##  1
##
## - best performance: 0.9971148
##
## - Detailed performance results:
##     k     error   dispersion
## 1   1 0.9971148 0.0010678415
## 2   2 0.9975486 0.0005800924
## 3   3 0.9973343 0.0007251480
## 4   4 0.9971607 0.0012252235
## 5   5 0.9974179 0.0010074710
## 6   6 0.9972497 0.0006762468
## 7   7 0.9974196 0.0009779034
## 8   8 0.9973343 0.0008303711
## 9   9 0.9971163 0.0007729647
## 10 10 0.9974611 0.0009439150
```

So the dispersion is lowest with k of 9. So we need 9 nearest neighbor points for best knn performance.

```
plot(obj2)
```



So the best K is 9

```
attach(train)
#confusion matrix for knn for valid set
train.X=cbind(Term.GPA, HSGPA, SAT_Total, N.RegisteredCourse, N.Ws, N.DFs,
```

```
N.As, N.PassedCourse, N.CourseTaken, Perc.PassedEnrolledCourse, Perc.Pass,
N.GraduateCourse)

test.X=cbind(Term.GPA, HSGPA, SAT_Total, N.RegisteredCourse, N.Ws, N.DFs,
N.As, N.PassedCourse, N.CourseTaken, Perc.PassedEnrolledCourse, Perc.Pass,
N.GraduateCourse)

library(class)
knn.pred=knn(train.X,test.X,train$Persistence.NextYear,k=obj2$best.parameters
[1])
ct5_knn=table(train$Persistence.NextYear, knn.pred)
cat('the confusion matrix is',sep="\n\n")
ct5_knn
me_knn_valid=perfcheck(ct5_knn)

## the confusion matrix is
##    knn.pred
##        0    1
##   0 1167    0
##   1    0 4590

#confusion matrix for knn for valid set
detach(train)
attach(test)
train.X=cbind(Term.GPA, HSGPA, SAT_Total, N.RegisteredCourse, N.Ws, N.DFs,
N.As, N.PassedCourse, N.CourseTaken, Perc.PassedEnrolledCourse, Perc.Pass,
N.GraduateCourse)

test.X=cbind(Term.GPA, HSGPA, SAT_Total, N.RegisteredCourse, N.Ws, N.DFs,
N.As, N.PassedCourse, N.CourseTaken, Perc.PassedEnrolledCourse, Perc.Pass,
N.GraduateCourse)

knn.pred=knn(train.X,test.X,test$Persistence.NextYear,k=obj2$best.parameters[
1])
ct5_knn=table(test$Persistence.NextYear, knn.pred)
cat('the confusion matrix is',sep="\n\n")

ct5_knn
me_knn_test=perfcheck(ct5_knn)

## the confusion matrix is
##    knn.pred
##        0    1
##   0  144    0
##   1    0 1301

#performance metric for logistic regression
a=cbind(me_knn_valid[,1],me_knn_valid[,2],me_knn_test[,2])
colnames(a)=c("Metrics","valid","test")
knitr::kable(a, caption = "Model Metrics KNN")
```

Model Metrics KNN

Metrics	valid	test
Accuracy	100	100
Recall	100	100
Type1	0	0
Precision	100	100
Type2	0	0
F1	100	100

So for KNN the metrics are 100% which is kind of justifies here, because there is no seprate train, test, we are separating the class based on nearest neighbours.

- Model 3.

LDA model

```
# we will use cross validation first. on LDA
## k-Fold CV
library(MASS)
#LDA regression
error.lda.valid=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
error.lda.train=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
# now, looping/k-fold procedure
for(j in 1:k){

lda.fit=lda(Persistence.NextYear~Term.GPA+HSGPA+SAT_Total+N.RegisteredCourse+
N.Ws+N.DFs+N.As+Perc.PassedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCour
se+FullTimeStudent,data=train[folds!=j,])
  #PREDICT on train and test data
  lda.pred.train=predict(lda.fit, train[folds!=j,])
  lda.pred.valid=predict(lda.fit, train[folds==j,])
  lda.class.train=lda.pred.train$class
  lda.class.valid=lda.pred.valid$class


error.lda.train[j,]=mean(lda.class.train!=train$Persistence.NextYear[folds!=j
])


error.lda.valid[j,]=mean(lda.class.valid!=train$Persistence.NextYear[folds==j
])
}
mean.errors.lda.train=apply(error.lda.train,2,mean)
mean.errors.lda.valid=apply(error.lda.valid,2,mean)
```

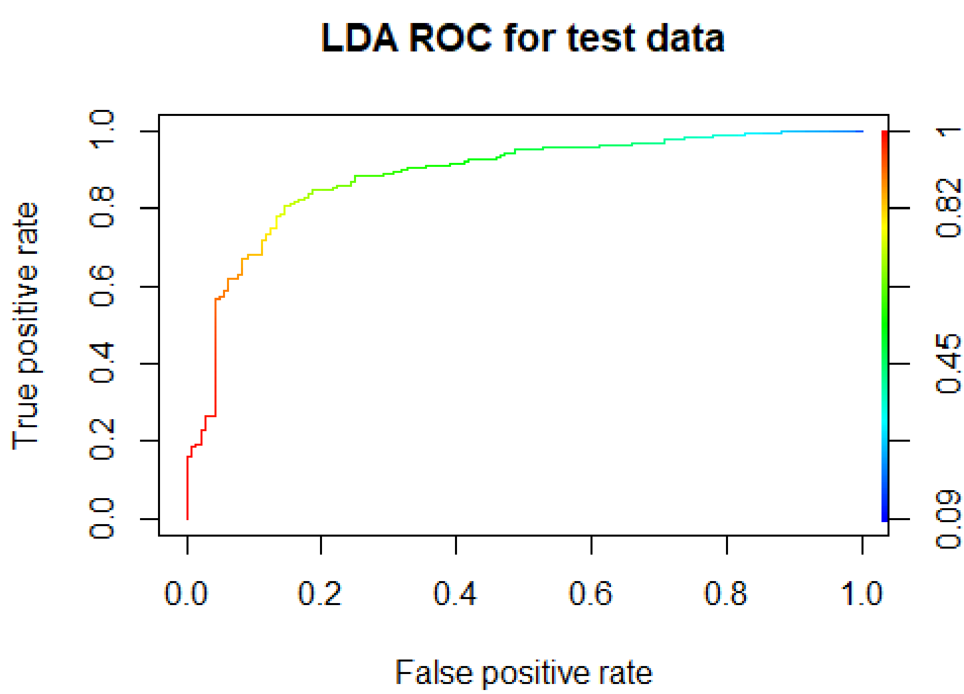
```
cat('The train and test error rate for LDA are', mean.errors.lda.train, '&',
mean.errors.lda.valid)

## The train and test error rate for LDA are 0.1494714 & 0.1498732

#confusion matrix for LDA on train set
ct1_train=table(train$Persistence.NextYear[folds!=k], lda.class.train)
cat('the confusion matrix is',sep="\n\n")

ct1_train
me_lda_train=perfcheck(ct1_train)

## the confusion matrix is
##    lda.class.train
##        0    1
##   0  471  468
##   1  200 3474

#confusion matrix for LDA on valid set
ct1_valid=table(train$Persistence.NextYear[folds==k], lda.class.valid)
cat('the confusion matrix is',sep="\n\n")

ct1_valid
me_lda_valid=perfcheck(ct1_valid)

## the confusion matrix is
##    lda.class.valid
##       0   1
##   0 114 114
##   1  62 854

#confusion matrix for LDA on test set
lda.pred.test=predict(lda.fit, test)
lda.class.test=lda.pred.test$class
ct1_test=table(test$Persistence.NextYear, lda.class.test)
cat('the confusion matrix is',sep="\n\n")

ct1_test
me_lda_test=perfcheck(ct1_test)

## the confusion matrix is
##    lda.class.test
##        0    1
##   0   74   70
##   1   70 1231

#performance metric for LDA
a=cbind(me_lda_train[,1],me_lda_train[,2],me_lda_valid[,2],me_lda_test[,2])
colnames(a)=c("Metrics","Train","valid","test")
knitr::kable(a, caption = "Model Metrics LDA")
```

Model Metrics LDA

Metrics	Train	valid	test
Accuracy	85.52	84.62	90.31
Recall	94.56	93.23	94.62
Type1	49.84	50	48.61
Precision	88.13	88.22	94.62
Type2	5.44	6.77	5.38
F1	91.23	90.66	94.62

So agin, for LDA, we are doing good with both train and test set.

```
library(ROCR)

## Warning: package 'ROCR' was built under R version 4.0.4

# choose the posterior probability column carefully, it may be
# lda.pred$posterior[,1] or lda.pred$posterior[,2], depending on your factor
levels
pred <- prediction(lda.pred.test$posterior[,2], test$Persistence.NextYear)
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE, main="LDA ROC for test data")
```



So again the ROC curve is very good with AUC of 0.88 for LDA

- Model 4.

QDA model

```
# we will use cross validation first. on QDA
## k-Fold CV

#LDA regression
error.qda.valid=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
error.qda.train=matrix(NA,k,1, dimnames=list(NULL, paste("Log")))
# now, looping/k-fold procedure
for(j in 1:k){

qda.fit=qda(Persistence.NextYear~Term.GPA+HSGPA+SAT_Total+N.RegisteredCourse+
N.Ws+N.DFs+N.As+Perc.PassedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCour
se+FullTimeStudent,data=train[folds!=j,])
  #PREDICT on train and test data
  qda.pred.train=predict(qda.fit, train[folds!=j,])
  qda.pred.valid=predict(qda.fit, train[folds==j,])
  qda.class.train=qda.pred.train$class
  qda.class.valid=qda.pred.valid$class


error.qda.train[j,]=mean(qda.class.train!=train$Persistence.NextYear[folds!=j
])


error.qda.valid[j,]=mean(qda.class.valid!=train$Persistence.NextYear[folds==j
])
}
mean.errors.qda.train=apply(error.qda.train,2,mean)
mean.errors.qda.valid=apply(error.qda.valid,2,mean)

cat('The train and test error rate for QDA are', mean.errors.qda.train, '&',
mean.errors.qda.valid)

## The train and test error rate for QDA are 0.1741443 & 0.1822701

#confusion matrix for QDA on train set
ct1_train=table(train$Persistence.NextYear[folds!=k], qda.class.train)
ct1_train
me_qda_train=perfcheck(ct1_train)

##    qda.class.train
##        0    1
##   0  457  482
##   1  304 3370

#confusion matrix for QDA on valid set
ct1_valid=table(train$Persistence.NextYear[folds==k], qda.class.valid)
```

```
ct1_valid
me_qda_valid=perfcheck(ct1_valid)
```

```
##    qda.class.valid
##       0   1
##   0 117 111
##   1  95 821
```

```
#confusion matrix for QDA on test set
qda.pred.test=predict(qda.fit, test)
qda.class.test=qda.pred.test$class
ct1_test=table(test$Persistence.NextYear, qda.class.test)
ct1_test
me_qda_test=perfcheck(ct1_test)
```

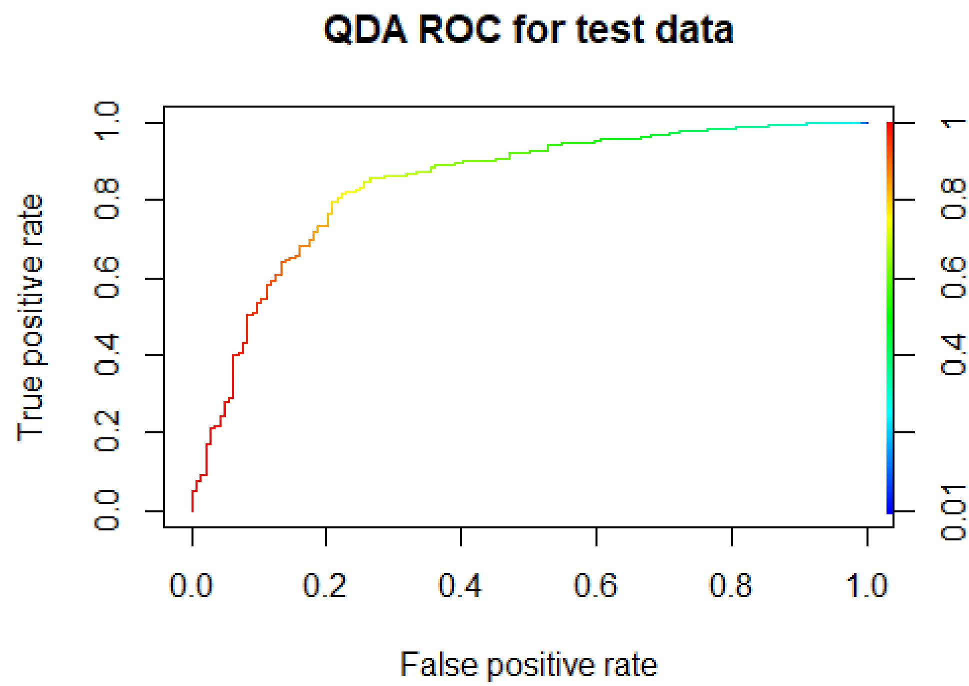
```
##    qda.class.test
##        0    1
##   0   52   92
##   1   56 1245
```

```
#performance metric for LDA
a=cbind(me_qda_train[,1],me_qda_train[,2],me_qda_valid[,2],me_qda_test[,2])
colnames(a)=c("Metrics","Train","valid","test")
knitr::kable(a, caption = "Model Metrics QDA")
```

Model Metrics QDA

Metrics	Train	valid	test
Accuracy	82.96	81.99	89.76
Recall	91.73	89.63	95.7
Type1	51.33	48.68	63.89
Precision	87.49	88.09	93.12
Type2	8.27	10.37	4.3
F1	89.56	88.85	94.39

```
library(ROCR)
# choose the posterior probability column carefully, it may be
# lda.pred$posterior[,1] or lda.pred$posterior[,2], depending on your factor
levels
pred <- prediction(qda.pred.test$posterior[,2], test$Persistence.NextYear)
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE, main="QDA ROC for test data")
```



So with QDA the recall score is higher than LDA. That means LDA has more false negatives (FN) compared to QDA. Whereas LDA has higher precision than QDA. That means, QDA has more false negatives compared to LDA.

So depending on situation we can say that we will choose LDA when we want precision in our result and QDA when we want high recall in our result.

Section 4.

Conclusion on each model.

1. OLS SLR- Here we are doing not a good job best on single predictor HSGPA is the best of the lot with a positive Rsq-adjusted value on tet set.
2. OLS MLR- WE coose forward selection model and found out that 3 predictors is best for the model, HSGPA, SAT_Total and gender. Although again the Rsq value is pretty low. however we are doing good compared to SLR model.
3. MLR Ridge- ridge regression gives better result compared to MLR Forward selection with optimal Lambda. Although the predictors in this Case are diffrenet. the best three predictors we got as HSGPA, SAT_Tota and N.As.
4. MLR Lasso- This model is not doing good and actually worst of all the regression models. since Lasso regression reduces the coefficients to zero, so we obseved only an intercept with non-zero concept, which actually is the mean value of all classes. So its is not good at all.

Assumption checks: 1. For SLR model we checked the assumptions, all the predictors did not have a linear relationship with Term.GPA thats why we didnot predict good except for HSGPA.

2. For MLR model, we checked if any of the predictors are collinear and we found many of the them except for 5 as mentioned in MLR model.
3. For Ridge model, there exists conditionally independent Gaussian residuals with zero mean and constant variance across the range of the explanatory variable(s), HSGPA, SAT_total and N.As.
4. Lasso model is very poorly fit. We only get intercept coefficient as non-zero.
5. For Logistic regression, we observed independence of errors, linearity in the logit for continuous variables, absence of multicollinearity, and lack of strongly influential outliers.
6. For KNN, similar things exist in close proximity which in our case is absolutely valid. We did a great in separating the clasess with KNN.
7. For LDA model, our data is Gaussian, that each variable is is shaped like a bell curve when plotted. In our case Term.GPA is like that. So we can say that To an extent our assumption is valid if we consider the varaiable Term.GPA.
8. QDA assumes that each class has its own covariance matrix, Here class 1 has a different covariance than class 0.

- BONUS.

```
library(e1071)
svmfit =
svm(Persistence.NextYear~Term.GPA+HSGPA+SAT_Total+N.RegisteredCourse+N.Ws+N.D
Fs+N.As+Perc.PassedEnrolledCourse+Perc.Pass+Perc.Withd+N.GraduateCourse+FullT
imeStudent, data = train, kernel = "radial", cost = 10, scale = TRUE)
print(svmfit)

##
## Call:
## svm(formula = Persistence.NextYear ~ Term.GPA + HSGPA + SAT_Total +
##     N.RegisteredCourse + N.Ws + N.DFs + N.As + Perc.PassedEnrolledCourse +
##     Perc.Pass + Perc.Withd + N.GraduateCourse + FullTimeStudent,
##     data = train, kernel = "radial", cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.08333333
##     epsilon:  0.1
##
##
## Number of Support Vectors:  3293

pred <- predict(svmfit, test)
pred.test=rep(0,length(pred))
pred.test[pred>.5]=1
ct1_test=table(test$Persistence.NextYear, pred.test)
cat('the confusion matrix is',sep="\n\n")

ct1_test
me_svm_test=perfcheck(ct1_test)

## the confusion matrix is
##    pred.test
##        0    1
##   0   82   62
##   1   55 1246

me_svm_test

##      Metrics     Values
## [1,] "Accuracy"  "91.9"
## [2,] "Recall"    "95.77"
## [3,] "Type1"     "43.06"
```

```
## [4,] "Precision" "95.26"
## [5,] "Type2"     "4.23"
## [6,] "F1"        "95.52"
```

So we see that a nonlinear SVM gives much better result compared to LDA, QDA, AND logistic regression. So the varaiables in our case are better fitted with an nonlinear SVM kernel.

Another thing to note here have majority class as 1 and minority class as 0, so our data is imbalanced. It would be good to do a balanced weighting for SVM to solve class imbalance.

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): …

Disclose the resources or persons if you get any help: …

How long did the assignment solutions take?: …

References

…