Midterm 2

Soumya Goswami // Graduate Student Done alone

04/09/2021

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 and four classification models on . 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.

- 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.
- 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.
- 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.

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:

- 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!

- `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

Section A:

```
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 Introduct
ion to statistics/Project1_midterm"
```

```
#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
```

```
library(fastDummies)
```

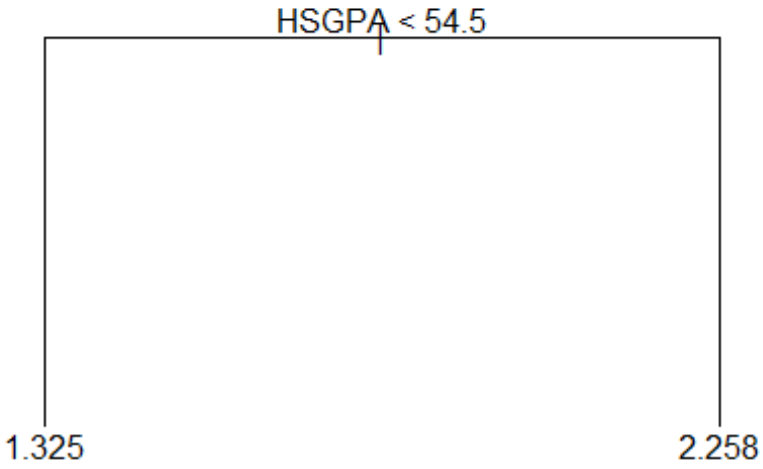
```
## Warning: package 'fastDummies' was built under R version 4.0.5
```

```
View(train<-fastDummies::dummy_cols(train,
select_columns=c('Entry_Term', 'Gender', 'Race_Ethc_Visa'),
remove_first_dummy = TRUE, remove_selected_columns=TRUE)
)
train=subset(train,select=-Entry_Term_2151)
```

```
#test data set with dummies, original cols removed,
View(test<-fastDummies::dummy_cols(test,
select_columns=c('Entry_Term', 'Gender', 'Race_Ethc_Visa'),
remove_first_dummy = TRUE, remove_selected_columns=TRUE)
)

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

library(tree) #class and reg trees

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



#fit the model on train
tree.GPA=tree(Term.GPA~.-Persistence.NextYear,train)

#predict the test
tree.pred=predict(tree.GPA,test)
MSE_tree=mean((tree.pred-z1)^2)
MSE_tree

## [1] 0.9835952
```

So MSE on test set is very low which is good . We will try to reduce it further. We will apply tree with pruning.

```
#plot
plot(tree.GPA)
text(tree.GPA,pretty=0)
```




```
prune.GPA=prune.tree(tree.GPA,best=6)

## Warning in prune.tree(tree.GPA, best = 6): best is bigger than tree size

tree.pred=predict(prune.GPA,test)
MSE_tree_prun=mean((tree.pred-z1)^2)
MSE_tree_prun

## [1] 0.9835952
```

So pruning does not really improve our model.

We will try Random fores here

```
## Bagging and Random Forests
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

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

#when m=p, this is just a random forest
set.seed(99)
#shrn mtry=p, it is bagging
bag.GPA=randomForest(Term.GPA~.-Persistence.NextYear,data=train,
                         mtry=10,
                         importance=TRUE)
```

```
bag.GPA

#discuss the input, arguments and output

#predict
yhat.bag = predict(bag.GPA,newdata=test)

##
## Call:
##  randomForest(formula = Term.GPA ~ . - Persistence.NextYear, data = train,
mtry = 10, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 1.044794
##                     % Var explained: -1.56

MSE_bag=mean((yhat.bag-z1)^2)
MSE_bag

## [1] 1.044909
```

So MSE on test set is 0.88186 which is lower than the decision tree case.

Now we implement ANN with two hidden layers

```
require(neuralnet)

## Loading required package: neuralnet

## Warning: package 'neuralnet' was built under R version 4.0.5

set.seed(99)
nn_sc2=neuralnet(Term.GPA~.-Persistence.NextYear,
  data=train,
  hidden=10,
  threshold = 0.01,
  stepmax = 1e+05,
  act.fct = "logistic",
  linear.output = TRUE,
  learningrate=NULL,
  startweights = NULL, #NULL for random initialization
  algorithm = "rprop+",
  err.fct = "sse",
  likelihood = FALSE,
)
x0hat=predict(nn_sc2,test)
MSE_ANN=mean((x0hat-z1)^2)
MSE_ANN

## [1] 0.9946358
```

So we chose the optimal ANN with a single hidden layer as we obtained more testing MSE with multiple hidden layers. Hence we fixed a single hidden layer

```
a=rbind(MSE_tree,MSE_bag,MSE_ANN)
rownames(a)=c("Decision tree","random forest","ANN")
colnames(a)=c("MSE on test set")
knitr::kable(a, caption = "Model Metrics")
```

Model Metrics

	MSE on test set
Decision tree	0.9835952
random forest	1.0449092
ANN	0.9946358

So comparing all the models, we see that decision tree and ANN does the best job in this case compared to RANDOM FOREST. Since we have 21 parameters,ANN would give the optimum results. Random forest did not perform well because the number of paarmeters in this case is low.

Section B:

```
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))
}

library(tree) #class and reg trees
#fit the model on train
tree.persistence=tree(Persistence.NextYear~.,train)
#predict the test
tree.pred=predict(tree.persistence,test)
tree.pred <- ifelse(tree.pred>0.5, 1, 0)
tree_table=table(tree.pred, z2)
me_log_tree=perfcheck(tree_table)
me_log_tree

##      Metrics     Values
## [1,] "Accuracy"  "95.64"
## [2,] "Recall"    "97.54"
## [3,] "Type1"     "21.68"
## [4,] "Precision" "97.62"
## [5,] "Type2"     "2.46"
## [6,] "F1"        "97.58"

prune.persistence=prune.tree(tree.persistence,best=8)

tree.pred=predict(prune.persistence,test)
tree.pred <- ifelse(tree.pred>0.5, 1, 0)
tree_prune_table=table(tree.pred, z2)
me_log_tree_prune=perfcheck(tree_prune_table)
me_log_tree_prune

##      Metrics     Values
## [1,] "Accuracy"  "95.64"
## [2,] "Recall"    "97.54"
## [3,] "Type1"     "21.68"
## [4,] "Precision" "97.62"
## [5,] "Type2"     "2.46"
## [6,] "F1"        "97.58"
```

Now we implement random forest

```
## Bagging and Random Forests
library(randomForest)

#when m=p, this is just a random forest
set.seed(99)
#shrn mtry=p, it is bagging
bag.persistence=randomForest(Persistence.NextYear~.,data=train,mtry=60,
                         ntree=10,
                         importance=TRUE,
                         proximity=TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
valid
## range

bag.persistence

#discuss the input, arguments and output

#predict
yhat.bag = predict(bag.persistence,newdata=test)

##
## Call:
##  randomForest(formula = Persistence.NextYear ~ ., data = train,      mtry
= 60, ntree = 10, importance = TRUE, proximity = TRUE)
##                Type of random forest: regression
##                      Number of trees: 10
## No. of variables tried at each split: 20
##
##           Mean of squared residuals: 0.1084083
##                     % Var explained: 32.92

yhat.bag <- ifelse(yhat.bag>0.5, 1, 0)
bag_table=table(yhat.bag, z2)
me_log_bag=perfcheck(bag_table)
me_log_bag

##      Metrics     Values
## [1,] "Accuracy"  "92.46"
## [2,] "Recall"    "97.68"
## [3,] "Type1"     "41.03"
## [4,] "Precision" "93.85"
## [5,] "Type2"     "2.32"
## [6,] "F1"        "95.73"
```

implementing ANN

```
require(neuralnet)
set.seed(99)
nn_sc2=neuralnet(Persistence.NextYear~.,
  data=train,
  hidden=3,
  threshold = 0.01,
  stepmax = 1e+05,
  act.fct = "logistic",
  linear.output = FALSE,
  learningrate=NULL,
  startweights = NULL, #NULL for random initialization
  algorithm = "rprop+",
  err.fct = "ce",
  likelihood = FALSE,
)
#x0hat=predict(nn_sc2,subset(test,select=-c(Term.GPA,Persistence.NextYear)))
#MSE_ANN=mean((x0hat-z1)^2)
#MSE_ANN
x0hat <- compute(nn_sc2,test)$net.result
x0hat <- ifelse(x0hat>0.5, 1, 0)
ANN_table=table(x0hat, z2)
me_log_ANN=perfcheck(ANN_table)
me_log_ANN
```

```
##      Metrics     Values
## [1,] "Accuracy"  "89.83"
## [2,] "Recall"    "96.99"
## [3,] "Type1"     "50.69"
## [4,] "Precision" "91.54"
## [5,] "Type2"     "3.01"
## [6,] "F1"        "94.19"
```

```
a=rbind(c(me_log_tree[1,2],me_log_tree[6,2]),c(me_log_bag[1,2],me_log_bag[6,2
]),c(me_log_ANN[1,2],me_log_ANN[6,2]))
rownames(a)=c("Decision tree","random forest","ANN")
colnames(a)=c("Accuracy","F1-score")
knitr::kable(a, caption = "Model Metrics")
```

Model Metrics

	Accuracy	F1-score
Decision tree	95.64	97.58
random forest	92.46	95.73
ANN	89.83	94.19

So we see that decision tree gives us the best accuracy of 95.64% and F1-score of 97.58% which is pretty good.

Section C:

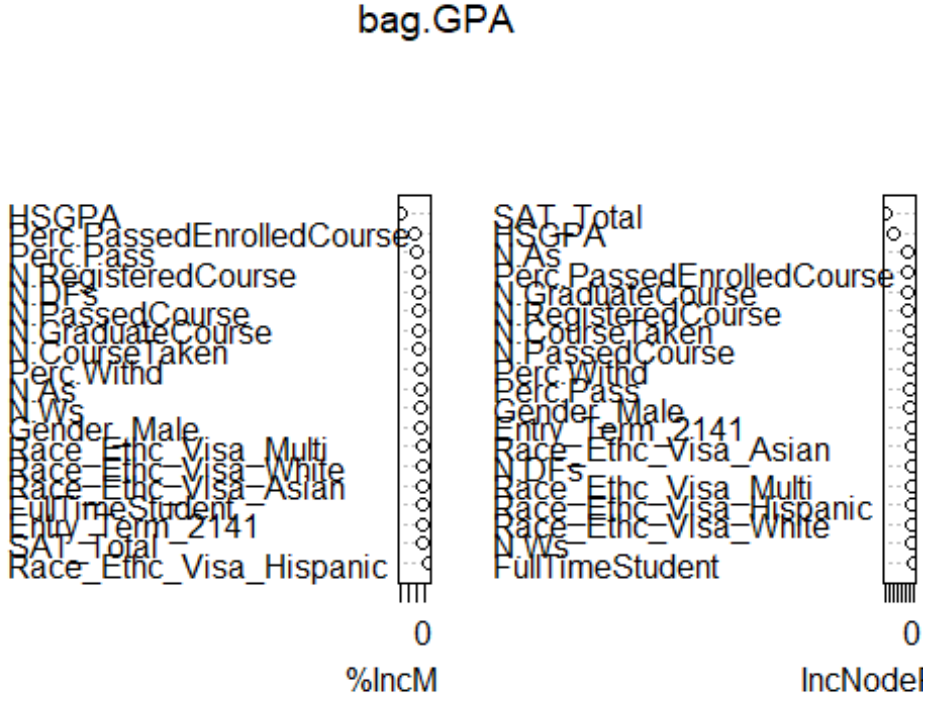
- Part a.

Here we will evaluate the three important variables for random forest regression model.

```
#get importance analysis and plot
importance(bag.GPA)
```

```
##                               %IncMSE IncNodePurity
## HSGPA                       30.318759     997.35922
## SAT_Total                    1.063390    1383.76321
## N.RegisteredCourse           6.058315     256.30733
## N.Ws                         5.018899     112.44018
## N.DFs                        5.727123     124.96462
## N.As                         5.243408     276.52972
## N.PassedCourse               5.641953     195.55576
## N.CourseTaken                5.412312     199.61287
## Perc.PassedEnrolledCourse 11.451892     260.17571
## Perc.Pass                    9.935950     171.51881
## Perc.Withd                   5.282103     194.09524
## N.GraduateCourse             5.470327     258.82439
## FullTimeStudent              1.316084      38.29271
## Entry_Term_2141              1.169190     154.69055
## Gender_Male                  2.994553     161.25362
## Race_Ethc_Visa_Asian         1.484453     126.44708
## Race_Ethc_Visa_Hispanic     -6.684568     122.00358
## Race_Ethc_Visa_Multi         2.037014     122.26061
## Race_Ethc_Visa_White         1.616204     121.62644
```

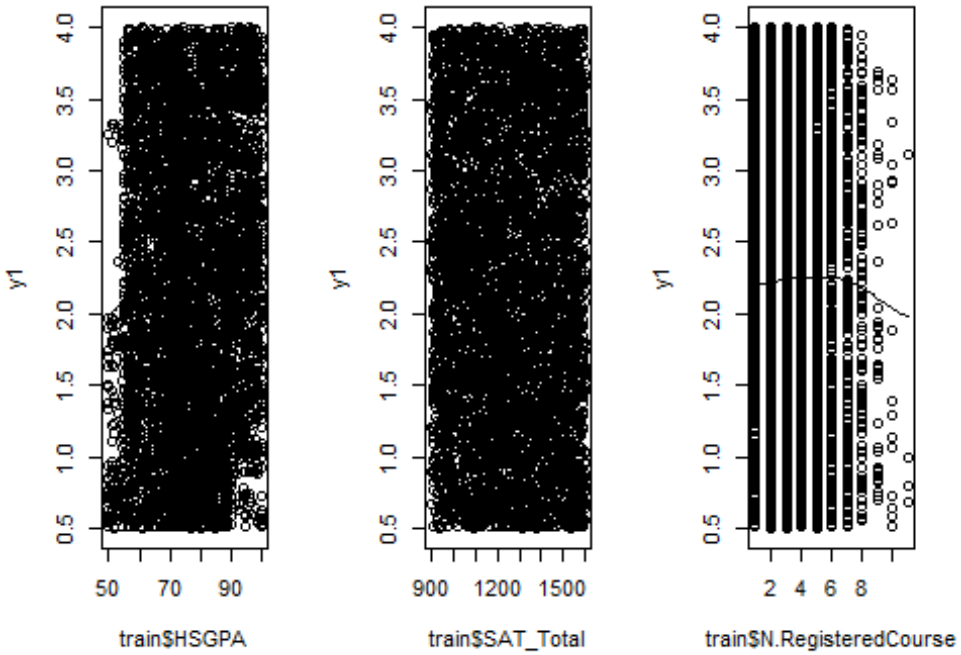
```
varImpPlot(bag.GPA)
```



So we find based on IncMSE and IncNodePurity, that the best three predictors are HSGPA, SAT_Total & N.RegisteredCourse.

We also do a correlation analysis of each predictor an the response variable

```
par(mfrow=c(1,3))
scatter.smooth(train$HSGPA,y1)
scatter.smooth(train$SAT_Total,y1)
scatter.smooth(train$N.RegisteredCourse,y1)
```



so we see from scatterplot there is not much strong correlation between tthe varaibales and the response Term.GPA

Now we calculate the correlation

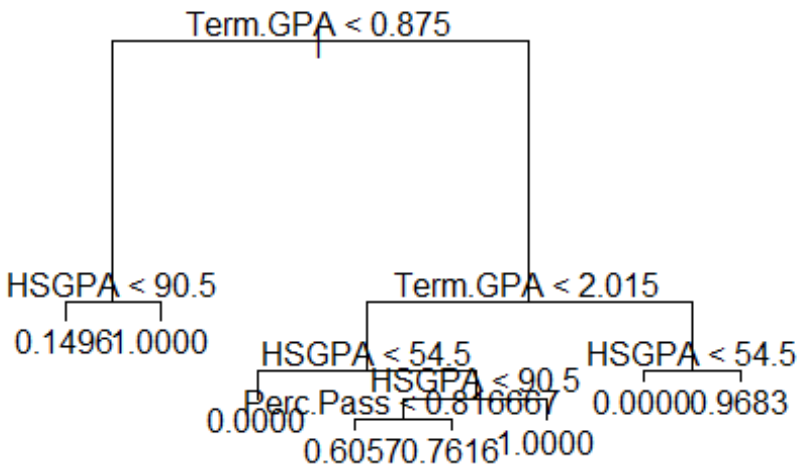
```
cat('the correlation between HSGPA & Term.GPA', cor(train$HSGPA,y1), 'between
SAT_Total & Term.GPA', cor(train$SAT_Total,y1),'between N.RegisteredCourse &
Term.GPA', cor(train$N.RegisteredCourse,y1))
```

```
## the correlation between HSGPA & Term.GPA 0.06028004 between SAT_Total & Te
rm.GPA -0.02262783 between N.RegisteredCourse & Term.GPA -0.001990355
```

- Part b.

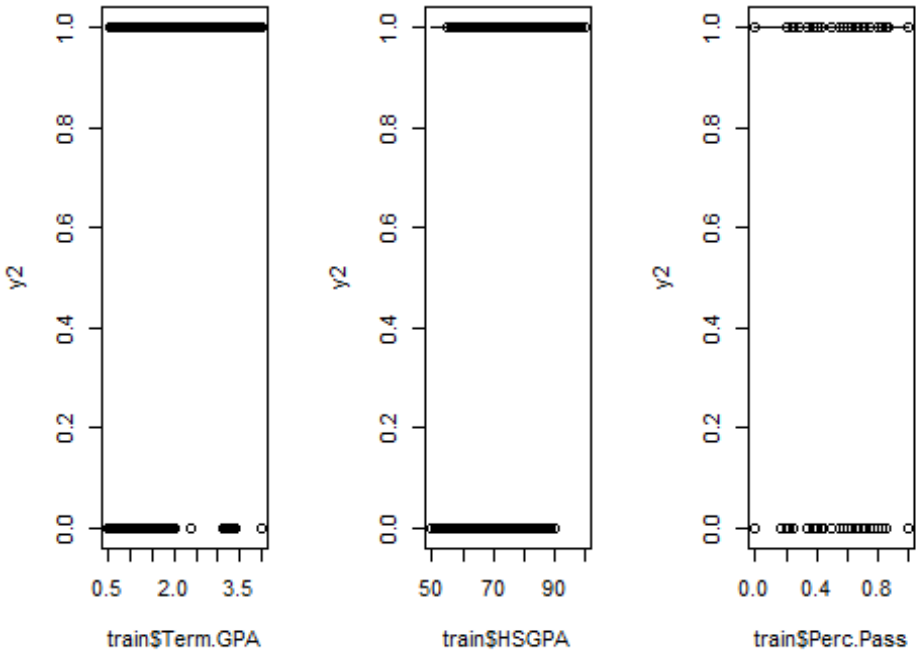
Here we will perform importance analysis on decision tree based classification

```
plot(prune.persistence)
text(prune.persistence,pretty=0)
```



So based on the tree, we see that three best predictors in order are Term.GPA, HSGPA AND Perc.Pass

```
par(mfrow=c(1,3))
scatter.smooth(train$Term.GPA,y2)
scatter.smooth(train$HSGPA,y2)
scatter.smooth(train$Perc.Pass,y2)
```



So we see good dependence between Term.GPA and Persistence.NextYear

```
cat('the correlation between Term.GPA & Persistence.NextYear', cor(train$Term
.GPA,y2), 'between HSGPA & Persistence.NextYear', cor(train$HSGPA,y2),'betwee
n Perc.Pass & Persistence.NextYear', cor(train$Perc.Pass,y2))
```

```
## the correlation between Term.GPA & Persistence.NextYear 0.4775424 between
HSGPA & Persistence.NextYear 0.1745613 between Perc.Pass & Persistence.NextYe
ar 0.06583424
```

- Part c.

In conclusion, we can say that in the regression model, ANN, decision tree gave comparable results. Since we have already seen that none of the predictors have a strong relation with the Term.GPA response variable. So prunning of trees did not help us in this case . In the first tree that we fir, we get the optimum results with HSGPA as the best predictor variable. For classification, we did pretty good job in the tree model, random forest and ANN as well. But in this case decision tree gave us the best result with 97% accuracy and 95% F1-SCORE. This indicates we dont have class imbalance problem here and also train and test performance is very close we observed for all the modles. So that further suggests our data is pretty consistent.

Did the baselines get improved?

Ans: Yes we improved the baselines in both the cases.

In Regression analysis, we observed MSE_Test of 0.983 with decision tree compared to baseline score of 0.991. This suggests Term.GPA as response variable can be well fit using our model.

In classification analysis, we again improved the baseline score of 91.15% accuracy and 95.65% F1-score by decision tree which gave us accuracy 95.64%, F1-SCORE 97.58%, by random forest accuracy 92.46%, F1-SCORE 95.73% and by ANN F1-SCORE 94.19%. So we outperform baseline results.

Why do you think the best model worked well or the models didn't work well?

Ans: In regression decision tree and ANN work well because our train test data is consistent,. Even if its noisy, the distribution is pretty consistent. So thats why we didnot overfit and our test accuracy is pretty high.

In classification ANN accuracy is not high becasue maybe here we have very few parameters to train on. With more parameters the performance would have improved. Whereas in decision tree, the less the parameters the better the generalization of the model.

How did you handle issues?

Ans: I converted the categorical variables into binary (numerical). Also we did a cross fold validation. In the classification problem, we did pruning of trees.

What could be done more to get better and interpretable results?

Ans: Maybe in regression problem we could try to normalize all predictors and see the performance. Because none of the predictors have a strong correlation with the response. Although here we dont have data imbalance, we could have done some sampling(undersampling for majority class and undersampling for minority class)

For the ANN model we could have chosen more hiden layers. Here we have tried with two only. These are possible improvements.

BONUS.

Yes we improved the baselines in both the cases.

In Regression analysis, we observed MSE_Test of 0.983 with decision tree compared to baseline score of 0.991. This suggests Term.GPA as response variable can be well fit using our model.

In classification analysis, we again improved the baseline score of 91.15% accuracy and 95.65% F1-score by decision tree which gave us accuracy 95.64%, F1-SCORE 97.58%, by random forest accuracy 92.46%, F1-SCORE 95.73% and by ANN F1-SCORE 94.19%. So we outperform baseline results.

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

…