ENSEMBLE METHODS and SUPER LEARNERS to predict the GRADE

Bogdan Tanasa

1. INTRODUCTION

2. DATA EXPLORATION

3. DATA SELECTION

4. DATA FILTERING

5. TRAINING AND TEST SETS

6. TRAINING AND PREDICTIONS WITH ANN (CARET)

7. TRAINING AND PREDICTIONS WITH ANN

8. TRAINING AND PREDICTIONS WITH SVM (CARET)

9. TRAINING AND PREDICTIONS WITH SVM

10. CONCLUSIONS

1. INTRODUCTION

We are using the data from **UCI** : !(https://archive.ics.uci.edu/ml/datasets/Student+Performance)

We are reading a file about **STUDENTS**, and we aim to predict whether they have passed or not the exams **(PASS/no_PASS)**;

The attributes in the **INPUT FILE** are the following :

- 1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
- 2 sex - student's sex (binary: "F" - female or "M" - male)
- 3 age - student's age (numeric: from 15 to 22)
- 4 address - student's home address type (binary: "U" - urban or "R" - rural)
- 5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
- 6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
- 7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
- 8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
- 9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
- 10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
- 11 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
- 12 guardian - student's guardian (nominal: "mother", "father" or "other")
- 13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- 14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- 15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
- 16 schoolsup - extra educational support (binary: yes or no)
- 17 famsup - family educational support (binary: yes or no)
- 18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- 19 activities - extra-curricular activities (binary: yes or no)
- 20 nursery - attended nursery school (binary: yes or no)
- 21 higher - wants to take higher education (binary: yes or no)
- 22 internet - Internet access at home (binary: yes or no)
- 23 romantic - with a romantic relationship (binary: yes or no)
- 24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)

- 25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
- 26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
- 27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- 28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- 29 health - current health status (numeric: from 1 - very bad to 5 - very good)
- 30 absences - number of school absences (numeric: from 0 to 93)

NOTES

DATA EXPLORATION and **DATA SELECTION** and **DATA FILTERING** have been presented also in the previous documents, and here, we have not fully included all the figures in those sections.

2. DATA EXPLORATION

```r
options(warn=-1)
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(reshape2))
suppressPackageStartupMessages(library(readxl))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(tidyr))
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(ggpubr))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(tibble))
suppressPackageStartupMessages(library(class))
suppressPackageStartupMessages(library(gmodels))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(ISLR))
suppressPackageStartupMessages(library(pROC))
suppressPackageStartupMessages(library(lattice))
suppressPackageStartupMessages(library(kknn))
suppressPackageStartupMessages(library(multiROC))
suppressPackageStartupMessages(library(MLeval))
suppressPackageStartupMessages(library(AppliedPredictiveModeling))
suppressPackageStartupMessages(library(corrplot))
suppressPackageStartupMessages(library(Hmisc))
suppressPackageStartupMessages(library(rattle))
suppressPackageStartupMessages(library(Hmisc))
suppressPackageStartupMessages(library(broom)) # to add : AUGMENT
suppressPackageStartupMessages(library(rattle))
suppressPackageStartupMessages(library(quantmod))
suppressPackageStartupMessages(library(nnet))
suppressPackageStartupMessages(library(NeuralNetTools))
suppressPackageStartupMessages(library(neuralnet))
suppressPackageStartupMessages(library(klaR))
suppressPackageStartupMessages(library(kernlab))

#########################################################
#########################################################

FILE1="student.mat.txt"

#########################################################
#########################################################
# FILE2="student.por.txt"
# FILE3="student.mat.and.por.txt"
#########################################################
#########################################################

#########################################################
#########################################################

student <- read.delim(FILE1, sep="\t", header=T, stringsAsFactors=F)

#########################################################
#########################################################
```

```r
summary(student)
```

```
##     school              sex                 age          address
##  Length:395         Length:395         Min.   :15.0   Length:395
##  Class :character   Class :character   1st Qu.:16.0   Class :character
##  Mode  :character   Mode  :character   Median :17.0   Mode  :character
##                                        Mean   :16.7
##                                        3rd Qu.:18.0
##                                        Max.   :22.0
##    famsize            Pstatus               Medu            Fedu
##  Length:395         Length:395         Min.   :0.000   Min.   :0.000
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :3.000   Median :2.000
##                                        Mean   :2.749   Mean   :2.522
##                                        3rd Qu.:4.000   3rd Qu.:3.000
##                                        Max.   :4.000   Max.   :4.000
##      Mjob               Fjob              reason            guardian
##  Length:395         Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    traveltime      studytime       failures       schoolsup
##  Min.   :1.000   Min.   :1.000   Min.   :0.0000   Length:395
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   Class :character
##  Median :1.000   Median :2.000   Median :0.0000   Mode  :character
##  Mean   :1.448   Mean   :2.035   Mean   :0.3342
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.0000
##  Max.   :4.000   Max.   :4.000   Max.   :3.0000
##     famsup              paid            activities          nursery
##  Length:395         Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     higher            internet           romantic             famrel
##  Length:395         Length:395         Length:395         Min.   :1.000
##  Class :character   Class :character   Class :character   1st Qu.:4.000
##  Mode  :character   Mode  :character   Mode  :character   Median :4.000
##                                                           Mean   :3.944
##                                                           3rd Qu.:5.000
##                                                           Max.   :5.000
##     freetime         goout            Dalc            Walc
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :3.000   Median :3.000   Median :1.000   Median :2.000
##  Mean   :3.235   Mean   :3.109   Mean   :1.481   Mean   :2.291
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:3.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##      health         absences            G1              G2
##  Min.   :1.000   Min.   : 0.000   Min.   : 3.00   Min.   : 0.00
```

```
##  1st Qu.:3.000   1st Qu.: 0.000   1st Qu.: 8.00   1st Qu.: 9.00
##  Median :4.000   Median : 4.000   Median :11.00   Median :11.00
##  Mean   :3.554   Mean   : 5.709   Mean   :10.91   Mean   :10.71
##  3rd Qu.:5.000   3rd Qu.: 8.000   3rd Qu.:13.00   3rd Qu.:13.00
##  Max.   :5.000   Max.   :75.000   Max.   :19.00   Max.   :19.00
##        G3
##  Min.   : 0.00
##  1st Qu.: 8.00
##  Median :11.00
##  Mean   :10.42
##  3rd Qu.:14.00
##  Max.   :20.00
```

```r
str(student)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ school    : chr  "GP" "GP" "GP" "GP" ...
##  $ sex       : chr  "F" "F" "F" "F" ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : chr  "U" "U" "U" "U" ...
##  $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
##  $ Pstatus   : chr  "A" "T" "T" "T" ...
##  $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob      : chr  "at_home" "at_home" "at_home" "health" ...
##  $ Fjob      : chr  "teacher" "other" "other" "services" ...
##  $ reason    : chr  "course" "course" "other" "home" ...
##  $ guardian  : chr  "mother" "father" "mother" "mother" ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : chr  "yes" "no" "yes" "no" ...
##  $ famsup    : chr  "no" "yes" "no" "yes" ...
##  $ paid      : chr  "no" "no" "yes" "yes" ...
##  $ activities: chr  "no" "no" "no" "yes" ...
##  $ nursery   : chr  "yes" "no" "yes" "yes" ...
##  $ higher    : chr  "yes" "yes" "yes" "yes" ...
##  $ internet  : chr  "no" "yes" "yes" "yes" ...
##  $ romantic  : chr  "no" "no" "no" "yes" ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ G1        : int  5 5 7 15 6 15 12 6 16 14 ...
##  $ G2        : int  6 5 8 14 10 15 12 5 18 15 ...
##  $ G3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```r
class(student)
```

```
## [1] "data.frame"
```

Here we are starting to display the data for visual exploration.

```r
################################################################################################
################################################################################################
# 1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)

# unique(student$school)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=school, fill=school))

# ggsave("display.1.school.png")
# student$school = as.factor(student$school)
student$school = as.character(student$school)

################################################################################################
################################################################################################
# 2 sex - student's sex (binary: "F" - female or "M" - male)

# unique(student$sex)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=sex , fill=sex))

# ggsave("display.2.sex.png")
student$sex = as.factor(student$sex)

################################################################################################
################################################################################################
# 3 age - student's age (numeric: from 15 to 22)

# unique(student$age)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=age , fill=age))

# ggplot(data=student, aes(x=age)) +
#       geom_histogram(aes(y=..density..), colour="black", fill="white")+
#       geom_density(alpha=.2, fill="#FF6666")

# ggsave("display.3.age.png")
# AGE is already on the numerical scale !!
student$age = as.integer(student$age)

################################################################################################
################################################################################################
# 4 address - student's home address type (binary: "U" - urban or "R" - rural)

# unique(student$address) ## [1] "U" "R"

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=address, fill=address))

# ggsave("display.4.address.png")
student$address = as.factor(student$address)
```

```r
##############################################################################################
##############################################################################################
# 5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)

# unique(student$famsize)

# ggplot(data = student) +
#         geom_bar(mapping = aes(x=famsize, fill=famsize))

# ggsave("display.5.famsize.png")
student$famsize = as.factor(student$famsize)

##############################################################################################
##############################################################################################
# 6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)

# unique(student$Pstatus)

# ggplot(data = student) +
#         geom_bar(mapping = aes(x=Pstatus, fill=Pstatus))

# ggsave("display.6.Pstatus.png")
student$Pstatus = as.factor(student$Pstatus)

##############################################################################################
##############################################################################################
# 7 Medu - mother's education (numeric: 0 - none,  1 - primary education (4th grade),
# 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

# unique(student$Medu)

# ggplot(data = student) +
#         geom_bar(mapping = aes(x=Medu, fill=Medu))

# ggsave("display.7.Medu.png")
# we may wanna use the numerical values in various regression models
student$Medu = as.integer(student$Medu)

##############################################################################################
##############################################################################################
# 8 Fedu - father's education (numeric: 0 - none,  1 - primary education (4th grade),
# 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

# unique(student$Fedu)

# ggplot(data = student) +
#         geom_bar(mapping = aes(x=Fedu, fill=Fedu))

# ggsave("display.8.Fedu.png")
# we may wanna use the numerical values in various regression models
student$Fedu = as.integer(student$Fedu)

##############################################################################################
```

```
##########################################################################################
# 9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services"
# (e.g. administrative or police), "at_home" or "other")

# unique(student$Mjob)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=Mjob, fill=Mjob))

# ggsave("display.9.Mjob.png")
student$Mjob = as.factor(student$Mjob)

##########################################################################################
##########################################################################################
# 10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services"
# (e.g. administrative or police), "at_home" or "other")

# unique(student$Fjob)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=Fjob, fill=Fjob))

# ggsave("display.10.Fjob.png")
student$Fjob = as.factor(student$Fjob)

##########################################################################################
##########################################################################################
# 11 reason - reason to choose this school
# (nominal: close to "home", school "reputation", "course" preference or "other")

# unique(student$reason)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=reason, fill=reason))

# ggsave("display.11.reason.png")
student$reason = as.factor(student$reason)

##########################################################################################
##########################################################################################
# 12 guardian - student's guardian (nominal: "mother", "father" or "other")

# unique(student$guardian)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=guardian, fill=guardian))

# ggsave("display.12.guardian.png")
student$guardian = as.factor(student$guardian)

##########################################################################################
##########################################################################################
# 13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min.,
```

```
# 3 - 30 min. to 1 hour, or 4 - >1 hour)

# unique(student$traveltime)

# ggplot(data = student) +
#       geom_bar(mapping = aes(x=traveltime, fill=traveltime))

# ggsave("display.13.traveltime.png")
# we may wanna use the NUMERICAL VALUES :
student$traveltime = as.integer(student$traveltime)

################################################################################################
################################################################################################
# 14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours,
# 3 - 5 to 10 hours, or 4 - >10 hours)

# unique(student$studytime)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=studytime, fill=studytime))

# ggsave("display.14.studytime.png")
# we may wanna use the NUMERICAL VALUES :
student$studytime = as.integer(student$studytime)

################################################################################################
################################################################################################
# 15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)

# unique(student$failures)

# ggplot(data = student) +
#       geom_bar(mapping = aes(x=failures, fill=failures))

# ggsave("display.15.failures.png")
# we may wanna use the NUMERICAL VALUES :
student$failures = as.integer(student$failures)

################################################################################################
################################################################################################
# 16 schoolsup - extra educational support (binary: yes or no)

# unique(student$schoolsup)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=schoolsup, fill=schoolsup))

# ggsave("display.16.schoolsup.png")
student$schoolsup = as.factor(student$schoolsup)

################################################################################################
################################################################################################
# 17 famsup - family educational support (binary: yes or no)
```

```r
# unique(student$famsup)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=famsup, fill=famsup))

# ggsave("display.17.famsup.png")
student$famsup = as.factor(student$famsup)

################################################################################################
################################################################################################
# 18 paid - extra paid classes within the course subject (Math or Portuguese)
# (binary: yes or no)

# unique(student$paid)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=paid, fill=paid))

# ggsave("display.18.paid.png")
student$paid = as.factor(student$paid)

################################################################################################
################################################################################################
# 19 activities - extra-curricular activities (binary: yes or no)

# unique(student$activities)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=activities, fill=activities))

# ggsave("display.19.activities.png")
student$activities = as.factor(student$activities)

################################################################################################
################################################################################################
# 20 nursery - attended nursery school (binary: yes or no)

# unique(student$nursery)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=nursery, fill=nursery))

# ggsave("display.20.nursery.png")
student$nursery = as.factor(student$nursery)

################################################################################################
################################################################################################
# 21 higher - wants to take higher education (binary: yes or no)

# unique(student$higher)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=higher, fill=higher))
```

```r
# ggsave("display.21.higher.png")
student$higher = as.factor(student$higher)

###############################################################################################
###############################################################################################
# 22 internet - Internet access at home (binary: yes or no)

# unique(student$internet)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=internet, fill=internet))

# ggsave("display.22.internet.png")
student$internet = as.factor(student$internet)

###############################################################################################
###############################################################################################
# 23 romantic - with a romantic relationship (binary: yes or no)

# unique(student$romantic)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=romantic, fill=romantic))

# ggsave("display.23.romantic.png")
student$romantic = as.factor(student$romantic)

###############################################################################################
###############################################################################################
# 24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)

# unique(student$famrel)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=famrel, fill=famrel))

# ggsave("display.24.famrel.png")
# i believe that we can keep these as numerical :
student$famrel = as.integer(student$famrel)

###############################################################################################
###############################################################################################
# 25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)

# unique(student$freetime)

# ggplot(data = student) +
#       geom_bar(mapping = aes(x=freetime, fill=freetime))

# ggsave("display.25.freetime.png")
# i believe that we can keep these as numerical :
student$freetime = as.integer(student$freetime)
```

```r
################################################################################################
################################################################################################
# 26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)

# unique(student$goout)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=goout, fill=goout))

# ggsave("display.26.goout.png")
# i believe that we can keep these as numerical :
student$goout = as.integer(student$goout)

################################################################################################
################################################################################################
# 27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)

# unique(student$Dalc)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=Dalc, fill=Dalc))

# ggsave("display.27.Dalc.png")
# i believe that we can keep these as numerical :
student$Dalc = as.integer(student$Dalc)

################################################################################################
################################################################################################
# 28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)

# unique(student$Walc)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=Walc, fill=Walc))

# ggsave("display.28.Walc.png")
# i believe that we can keep these as numerical :
student$Walc = as.integer(student$Walc)

################################################################################################
################################################################################################
# 29 health - current health status (numeric: from 1 - very bad to 5 - very good)

# unique(student$health)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=health, fill=health))

# ggsave("display.29.health.png")
# i believe that we can keep these as numerical :
student$health = as.integer(student$health)

################################################################################################
```

```r
#################################################################################################
# 30 absences - number of school absences (numeric: from 0 to 93)

# unique(student$absences)

# ggplot(data = student) +
#       geom_bar(mapping = aes(x=absences, fill=absences))

# ggplot(data=student, aes(x=absences)) +
#      geom_histogram(aes(y=..density..), colour="black", fill="white")+
#      geom_density(alpha=.2, fill="#FF6666")

# ggsave("display.30.absences.png")
# i believe that we can keep these as numerical :
student$absences = as.integer(student$absences)

#################################################################################################
#################################################################################################
# $ G1        : int  5 5 7 15 6 15 12 6 16 14 ...

# unique(student$G1)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=G1, fill=G1))

# ggplot(data=student, aes(x=G1)) +
#       geom_histogram(aes(y=..density..), colour="black", fill="white")+
#       geom_density(alpha=.2, fill="#FF6666")

# ggsave("display.0.G1.png")
# i believe that we can keep these as numerical, although we may not need it :
student$G1 = as.factor(student$G1)

#################################################################################################
#################################################################################################
# $ G2        : int  6 5 8 14 10 15 12 5 18 15 ...

# unique(student$G2)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=G2, fill=G2))

# ggplot(data=student, aes(x=G2)) +
#       geom_histogram(aes(y=..density..), colour="black", fill="white")+
#       geom_density(alpha=.2, fill="#FF6666")

# ggsave("display.0.G2.png")
# i believe that we can keep these as numerical, although we may not need it :
student$G2 = as.factor(student$G2)

#################################################################################################
#################################################################################################
# $ G3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```r
# unique(student$G3)

# ggplot(data = student) +
#        geom_bar(mapping = aes(x=G3, fill=G3))

# ggplot(data=student, aes(x=G3)) +
#       geom_histogram(aes(y=..density..), colour="black", fill="white")+
#       geom_density(alpha=.2, fill="#FF6666")

# ggsave("display.0.G3.png")
# i believe that we can covert it into RANGES of VALUES :
student$G3 = as.factor(student$G3)

################################################################################################
summary(student)
```

```
##     school              sex          age       address famsize   Pstatus
##  Length:395         F:208   Min.   :15.0   R: 88   GT3:281   A: 41
##  Class :character   M:187   1st Qu.:16.0   U:307   LE3:114   T:354
##  Mode  :character           Median :17.0
##                             Mean   :16.7
##                             3rd Qu.:18.0
##                             Max.   :22.0
##
##       Medu            Fedu                Mjob           Fjob              reason
##  Min.   :0.000   Min.   :0.000   at_home : 59   at_home : 20   course    :145
##  1st Qu.:2.000   1st Qu.:2.000   health  : 34   health  : 18   home      :109
##  Median :3.000   Median :2.000   other   :141   other   :217   other     : 36
##  Mean   :2.749   Mean   :2.522   services:103   services:111   reputation:105
##  3rd Qu.:4.000   3rd Qu.:3.000   teacher : 58   teacher : 29
##  Max.   :4.000   Max.   :4.000
##
##    guardian     traveltime      studytime        failures       schoolsup
##  father: 90   Min.   :1.000   Min.   :1.000   Min.   :0.0000   no :344
##  mother:273   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   yes: 51
##  other : 32   Median :1.000   Median :2.000   Median :0.0000
##               Mean   :1.448   Mean   :2.035   Mean   :0.3342
##               3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.0000
##               Max.   :4.000   Max.   :4.000   Max.   :3.0000
##
##  famsup     paid     activities nursery   higher    internet   romantic
##  no :153   no :214   no :194    no : 81   no : 20   no : 66   no :263
##  yes:242   yes:181   yes:201    yes:314   yes:375   yes:329   yes:132
##
##
##
##
##
##      famrel         freetime          goout            Dalc
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:4.000   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1.000
##  Median :4.000   Median :3.000   Median :3.000   Median :1.000
##  Mean   :3.944   Mean   :3.235   Mean   :3.109   Mean   :1.481
```

```
##   3rd Qu.:5.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:2.000
##   Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##
##        Walc           health        absences            G1              G2
##   Min.   :1.000   Min.   :1.000   Min.   : 0.000   10      : 51   9       : 50
##   1st Qu.:1.000   1st Qu.:3.000   1st Qu.: 0.000   8       : 41   10      : 46
##   Median :2.000   Median :4.000   Median : 4.000   11      : 39   12      : 41
##   Mean   :2.291   Mean   :3.554   Mean   : 5.709   7       : 37   13      : 37
##   3rd Qu.:3.000   3rd Qu.:5.000   3rd Qu.: 8.000   12      : 35   11      : 35
##   Max.   :5.000   Max.   :5.000   Max.   :75.000   13      : 33   15      : 34
##                                                    (Other):159   (Other):152
##         G3
##   10      : 56
##   11      : 47
##   0       : 38
##   15      : 33
##   8       : 32
##   12      : 31
##   (Other):158
```

```r
str(student)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ school    : chr  "GP" "GP" "GP" "GP" ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ Pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ Fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ G1        : Factor w/ 17 levels "3","4","5","6",..: 3 3 5 13 4 13 10 4 14 12 ...
##  $ G2        : Factor w/ 17 levels "0","4","5","6",..: 4 3 6 12 8 13 10 3 16 13 ...
```

```
##  $ G3        : Factor w/ 18 levels "0","4","5","6",..: 4 4 8 13 8 13 9 4 17 13 ...
```

```r
class(student)
```

```
## [1] "data.frame"
```

```r
###############################################################################################
# knitr::kable(summary(student, format = "html"))
###############################################################################################
```

3. DATA SELECTION

```r
## the OUTPUT VARIABLES is G3
## we may remove G1 and G2
## and some other features

student1 <- subset(student, select = -c(G1, G2))

student2 <- subset(student1,
                   select = -c(school, sex, address, famsize, Pstatus,
                   Mjob, Fjob, reason, guardian, schoolsup, famsup,
                   paid, activities, nursery,
                   higher, internet, romantic))

### shall we decide to keep ALL the FEATURES (ATTRIBUTES)
student2 = student1

str(student2)
```

```
## 'data.frame':    395 obs. of  31 variables:
##  $ school    : chr  "GP" "GP" "GP" "GP" ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ Pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ Fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ G3        : Factor w/ 18 levels "0","4","5","6",..: 4 4 8 13 8 13 9 4 17 13 ...
```

```r
student2$G3 = as.factor(student2$G3)

table(student2$G3)
```

```
##
```

```
##  0  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## 38  1  7 15  9 32 28 56 47 31 31 27 33 16  6 12  5  1
```

```
### for simplicity, to work with a copy of STUDENT2, let's call it STUDENT3

student3 = subset(student2,
                  select= c(age, traveltime, studytime, failures, absences, G3))

### shall we decide to keep ALL the FEATURES (ATTRIBUTES)
### student3 = student2

table(student3$G3)
```

```
##
##  0  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## 38  1  7 15  9 32 28 56 47 31 31 27 33 16  6 12  5  1
```

4. DATA FILTERING

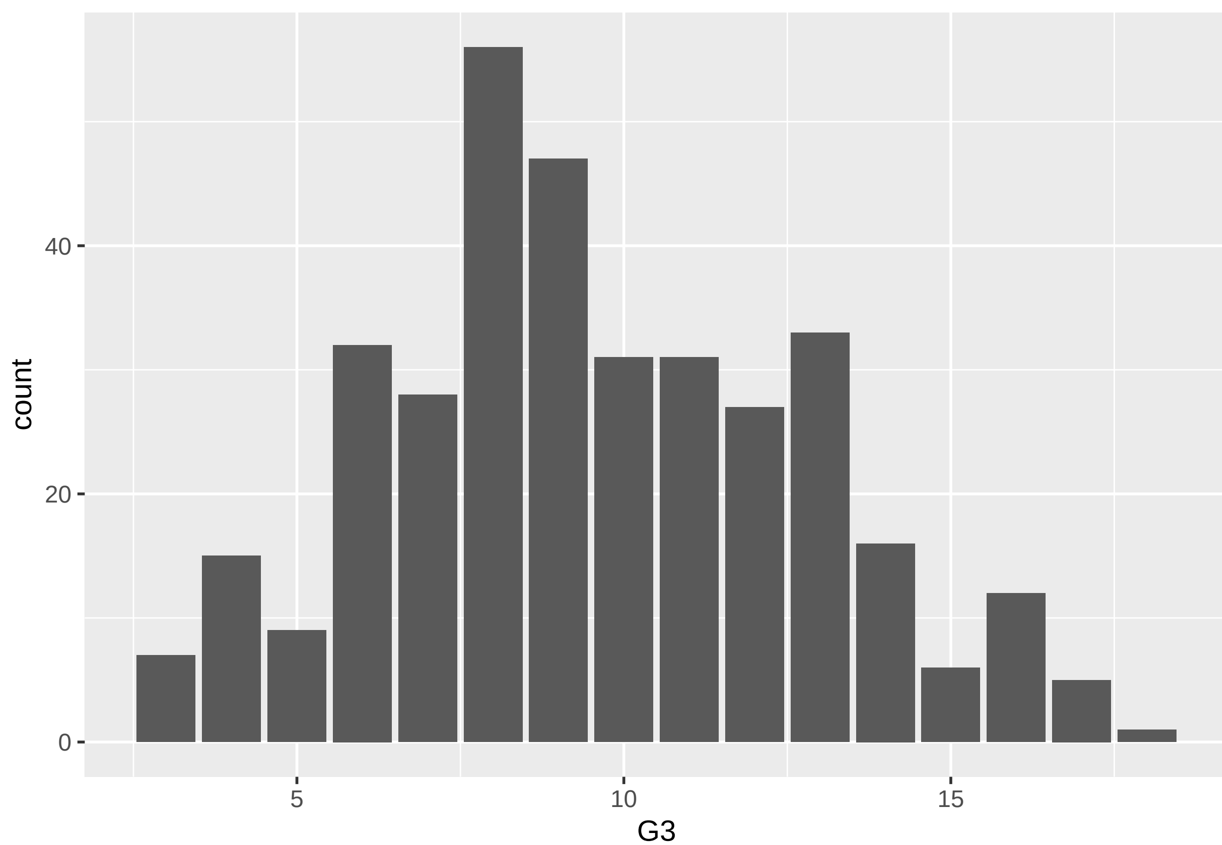
```
## in order to KEEP the RECORDS where the GRADE 3 is > 2 :

dim(student3)
```

```
## [1] 395   6
```

```
student3$G3 = as.integer(student3$G3)

student4 = student3[student3$G3 > 2, ]

dim(student4)
```

```
## [1] 356   6
```

```
ggplot(data = student4) +
       geom_bar(mapping = aes(x=G3, fill=G3))
```




```
ggsave("display.0.G3.after.filtering.grade3.frequency.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
student3 = student4

## TRANSFORMING G3 into RANGES of PASS and NO-PASS :

student3$G3 = as.integer(student3$G3)

student3$RESULT[student3$G3 <= 10] = "NO_PASS"
```

```r
student3$RESULT[student3$G3 >=10 ] = "PASS"

student3 <- subset(student3, select = -c(G3))

student3$RESULT = as.factor(student3$RESULT)

## DISPLAYING THE FEATURES (ATTRIBUTES) in THE CURRENT DATASET :

colnames(student3)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
```

5. TRAINING AND TEST SETS

```r
### CHOOSING the **TRAINING** and the **TESTING** SETS

set.seed(123)

indxTrain <- createDataPartition(student3$RESULT,
                                 p = 0.70,
                                 list = FALSE)

training <- student3[indxTrain,]
# training

testing <- student3[-indxTrain,]
# testing

dim(student3)
```

```
## [1] 356   6
```

```r
dim(training)
```

```
## [1] 250   6
```

```r
dim(testing)
```

```
## [1] 106   6
```

6. TRAINING AND PREDICTIONS WITH ANN (CARET)

6.1. TRAINING

```r
set.seed(123)

TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeats=10)

# nnnet package by defualt uses the Logistic Activation function
fit.nn <- train( RESULT~ .,
                    data = training,
                    method = "nnet",
                    trControl = TrainingParameters,
                    preProcess = c("center","scale"),
                    trace=FALSE,
                    verbose=FALSE,
                    # tuneLength = 20,
                    na.action = na.omit)

## The OUTPUT of nnet

# Size: Number of Hidden Layers.
# Decay: Is the regularization factor that offsets overfitting.
# Kappa: Evaluates the match is significant or by chance.

head(fit.nn$results)
```

```
##   size decay  Accuracy      Kappa AccuracySD   KappaSD
## 1    1 0e+00 0.5379718 0.08125237 0.09268560 0.1802034
## 2    1 1e-04 0.5337782 0.07162751 0.09435217 0.1851266
## 3    1 1e-01 0.5659128 0.12777881 0.09111360 0.1849630
## 4    3 0e+00 0.5533064 0.09840619 0.10007823 0.2013225
## 5    3 1e-04 0.5638295 0.12010537 0.09913340 0.1995938
## 6    3 1e-01 0.5878128 0.16401772 0.09417460 0.1927093
```

```r
tail(fit.nn$results)
```

```
##   size decay  Accuracy      Kappa AccuracySD   KappaSD
## 4    3 0e+00 0.5533064 0.09840619 0.10007823 0.2013225
## 5    3 1e-04 0.5638295 0.12010537 0.09913340 0.1995938
## 6    3 1e-01 0.5878128 0.16401772 0.09417460 0.1927093
## 7    5 0e+00 0.5281000 0.05295349 0.09658857 0.1941188
## 8    5 1e-04 0.5495308 0.09158380 0.08345964 0.1695250
## 9    5 1e-01 0.5712346 0.12995194 0.09160381 0.1868849
```

```r
print(fit.nn)
```

```
## Neural Network
##
## 250 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
```

```
## Summary of sample sizes: 225, 225, 225, 226, 225, 225, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##   1     0e+00  0.5379718  0.08125237
##   1     1e-04  0.5337782  0.07162751
##   1     1e-01  0.5659128  0.12777881
##   3     0e+00  0.5533064  0.09840619
##   3     1e-04  0.5638295  0.12010537
##   3     1e-01  0.5878128  0.16401772
##   5     0e+00  0.5281000  0.05295349
##   5     1e-04  0.5495308  0.09158380
##   5     1e-01  0.5712346  0.12995194
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
```

```r
# plot(fit.nn)
```

6.2. PREDICTIONS

```r
## colnames(testing)
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
## nn_predict <- predict(nn_model, testing[-6])

fit.nn.predict <- predict(fit.nn, newdata = testing)
```

We would aim to optimize the model by FEATURE SELECTION or by including NEW FEATURES from the data that is available (we have excluded at the beginning many features).

6.3. THE CONFUSION MATRIX

```r
confusionMatrix(fit.nn.predict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      41   23
##    PASS         17   25
##
##                Accuracy : 0.6226
##                  95% CI : (0.5233, 0.715)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : 0.0710
##
##                   Kappa : 0.2302
##
```

```
##  Mcnemar's Test P-Value : 0.4292
##
##             Sensitivity : 0.7069
##             Specificity : 0.5208
##          Pos Pred Value : 0.6406
##          Neg Pred Value : 0.5952
##              Prevalence : 0.5472
##          Detection Rate : 0.3868
##    Detection Prevalence : 0.6038
##       Balanced Accuracy : 0.6139
##
##        'Positive' Class : NO_PASS
##
```

The ACCURACY of the MODEL is :

```r
mean(fit.nn.predict == testing$RESULT)
```

```
## [1] 0.6226415
```

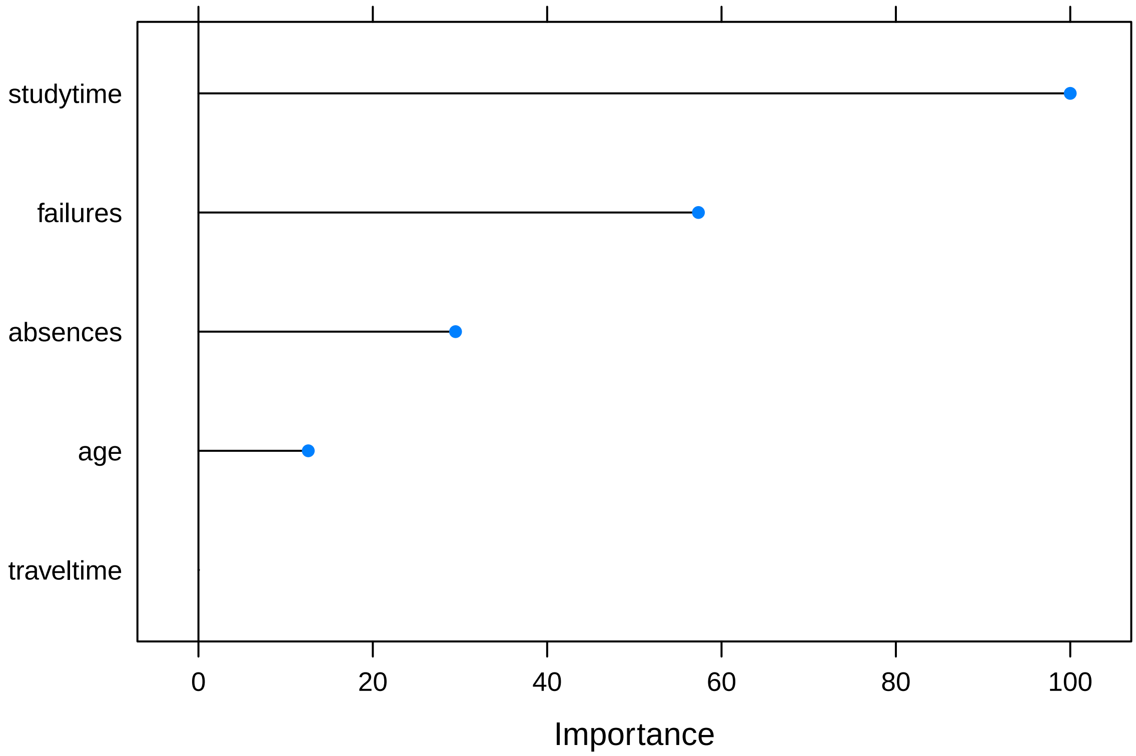
```r
# dim(student3)
# accuracy <- sum(nn_predict == (testing$RESULT))/length(testing$RESULT)
# print(accuracy)
```

6.4. THE VARIABLE IMPORTANCE

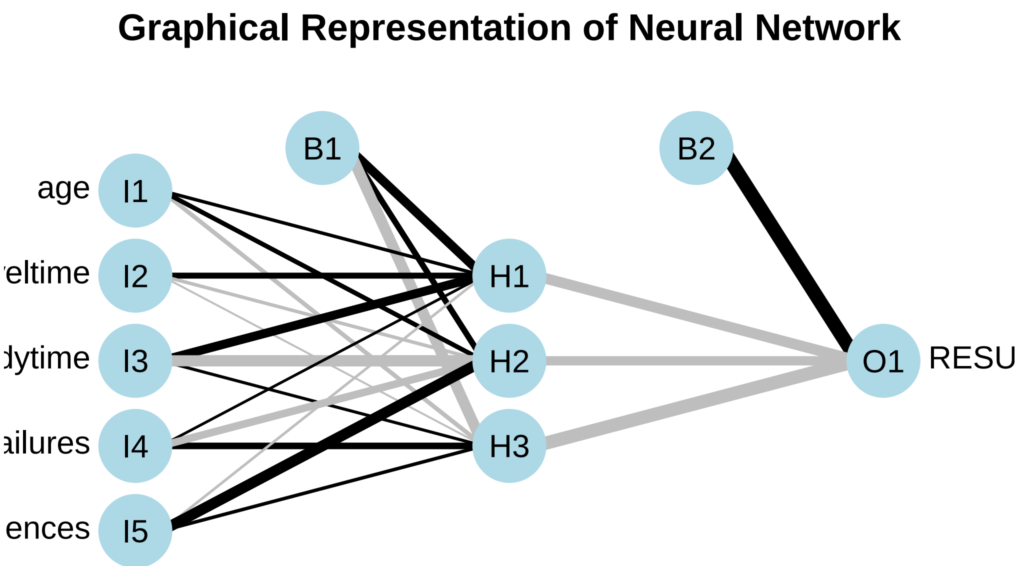
```r
X <- varImp(fit.nn)
print(X)
```

```
## nnet variable importance
##
##            Overall
## studytime   100.00
## failures     57.35
## absences     29.49
## age          12.60
## traveltime    0.00
```

```r
plot(X)
```




```
plotnet(fit.nn)
title("Graphical Representation of Neural Network")
```



Graphical Representation of Neural Network
B1
B2
age
I1
eltime
I2
dytime
I3
ailures
I4
ences
I5
H1
H2
H3
O1
RESU

7. TRAINING AND PREDICTIONS WITH ANN

We are using the package "neuralnet" available on CRAN :

https://cran.r-project.org/web/packages/neuralnet/index.html

and according to the description in the book :

"Machine Learning with R"

7.1. TRAINING

```
set.seed(123)

model.nn1 <- neuralnet(RESULT ~ age + traveltime + studytime + failures + absences,
                       data = training,
                       hidden=2,
                       act.fct = "logistic",
                       linear.output = FALSE)

plot(model.nn1)

model.nn2 <- neuralnet(RESULT ~ age + traveltime + studytime + failures + absences,
                       data = training,
                       hidden=2,
                       act.fct = "tanh",
                       linear.output = FALSE)

plot(model.nn2)
```

7.2. PREDICTIONS

```
set.seed(123)

model.nn1.results <- neuralnet::compute(model.nn1, testing)
head(model.nn1.results$net.result)
```

```
##         [,1]       [,2]
## 1  0.4448504 0.55515359
## 2  0.4448504 0.55515359
## 3  0.9593022 0.04078078
## 6  0.6280588 0.37191597
## 9  0.4448504 0.55515359
## 19 0.9593022 0.04078078
```

```
model.nn2.results <- neuralnet::compute(model.nn2, testing)
head(model.nn2.results$net.result)
```

```
##         [,1]       [,2]
## 1  0.6139445  0.4146450
## 2  0.4823239  0.5195626
## 3  0.9887364 -0.1373829
## 6  0.4680685  0.5290555
```

```
## 9  0.3650816  0.5899601
## 19 0.9915722 -0.2372633
```

We have followed also the materials from online resources :

https://datascienceplus.com/neuralnet-train-and-test-neural-networks-using-r/

although using CARET package is simpler and easier, and permits also an easy calculation of the CONFUSION MATRIX and the VARIABLE IMPORTANCE.

8. TRAINING AND PREDICTIONS WITH SVM (CARET)

8.A. using SVM_LINEAR

8.1. TRAINING

```
set.seed(123)

TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeats=10)
# grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))

svm_Linear <- train( RESULT~ .,
                     data = training,
                     method = "svmLinear",
                     trControl = TrainingParameters,
                     preProcess = c("center","scale"),
                     trace=FALSE,
                     verbose=FALSE,
                     # tuneGrid = grid,
                     # tuneLength = 20,
                     na.action = na.omit)

## The OUTPUT of svm_Linear

head(svm_Linear$results)
```

```
##   C  Accuracy     Kappa AccuracySD  KappaSD
## 1 1 0.5599474 0.1263012 0.09445751 0.188907
```

```
tail(svm_Linear$results)
```

```
##   C  Accuracy     Kappa AccuracySD  KappaSD
## 1 1 0.5599474 0.1263012 0.09445751 0.188907
```

```
print(svm_Linear)
```

```
## Support Vector Machines with Linear Kernel
##
## 250 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 225, 225, 225, 226, 225, 225, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5599474  0.1263012
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# plot(svm_Linear)
```

8.2. PREDICTIONS

```r
set.seed(123)
svm_Linear_predict <- predict(svm_Linear, newdata = testing)
```

8.3. THE CONFUSION MATRIX

```r
set.seed(123)
confusionMatrix(svm_Linear_predict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      34   17
##    PASS         24   31
##
##                Accuracy : 0.6132
##                  95% CI : (0.5137, 0.7062)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : 0.1019
##
##                   Kappa : 0.2292
##
##  Mcnemar's Test P-Value : 0.3487
##
##             Sensitivity : 0.5862
##             Specificity : 0.6458
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5636
##              Prevalence : 0.5472
##          Detection Rate : 0.3208
##    Detection Prevalence : 0.4811
##       Balanced Accuracy : 0.6160
##
##        'Positive' Class : NO_PASS
##
```

The ACCURACY of the MODEL is :

```r
mean(svm_Linear_predict == testing$RESULT)
```

```
## [1] 0.6132075
```

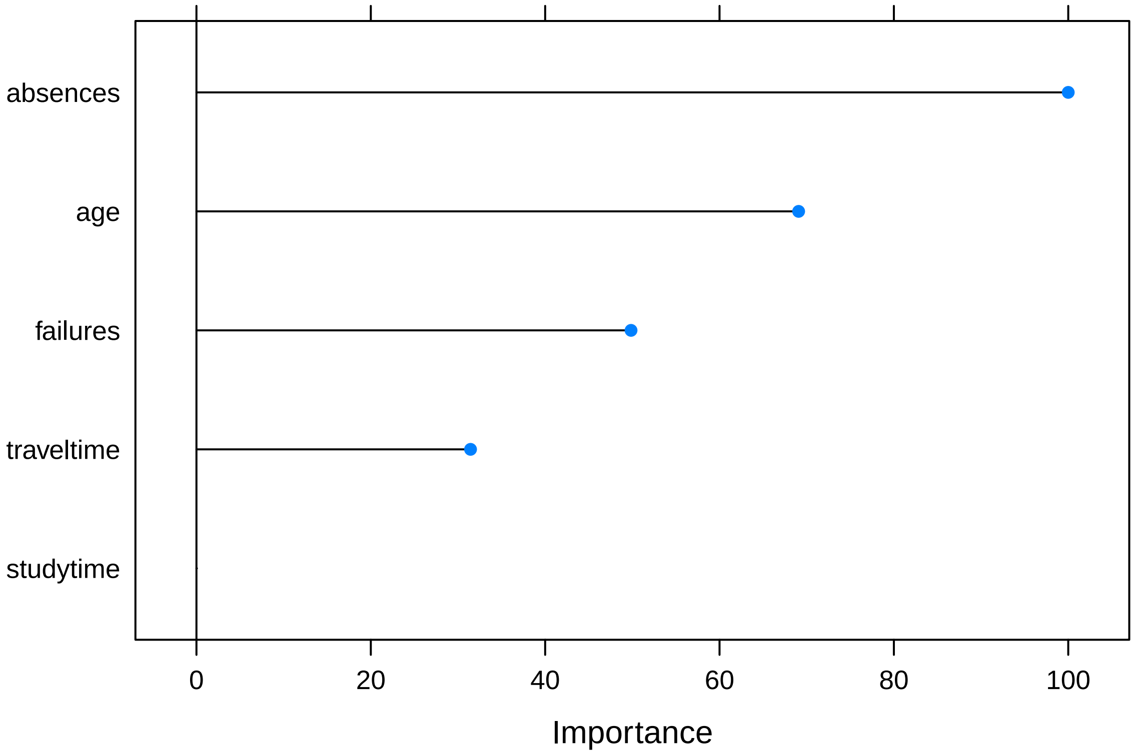
8.4. THE VARIABLE IMPORTANCE

```r
X <- varImp(svm_Linear)
print(X)
```

```
## ROC curve variable importance
##
```

```
##              Importance
## absences       100.00
## age             69.07
## failures        49.85
## traveltime      31.44
## studytime        0.00
```

```r
plot(X)
```



8.B. using SVM_RADIAL

```
set.seed(123)

TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeats=10)
# grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5))

svm_Radial <- train( RESULT~ .,
                   data = training,
                   method = "svmRadial",
                   trControl = TrainingParameters,
                   preProcess = c("center","scale"),
                   trace=FALSE,
                   verbose=FALSE,
                   # tuneGrid = grid,
                   # tuneLength = 20,
                   na.action = na.omit)

## The OUTPUT of svm_Radial

head(svm_Radial$results)
```

```
##       sigma    C  Accuracy      Kappa AccuracySD   KappaSD
## 1 0.2569873 0.25 0.5617000 0.08695771 0.07965907 0.1615056
## 2 0.2569873 0.50 0.5966115 0.17562602 0.09097046 0.1853376
## 3 0.2569873 1.00 0.6003103 0.18501000 0.09549276 0.1935835
```

```
tail(svm_Radial$results)
```

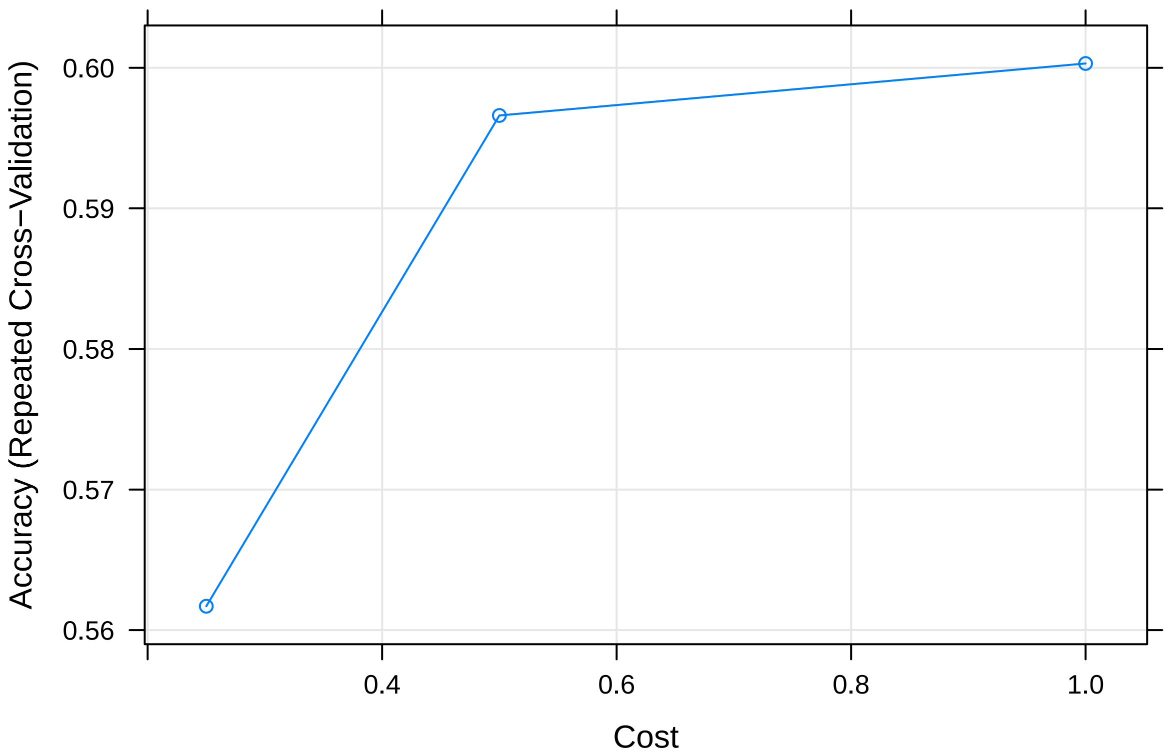
```
##       sigma    C  Accuracy      Kappa AccuracySD   KappaSD
## 1 0.2569873 0.25 0.5617000 0.08695771 0.07965907 0.1615056
## 2 0.2569873 0.50 0.5966115 0.17562602 0.09097046 0.1853376
## 3 0.2569873 1.00 0.6003103 0.18501000 0.09549276 0.1935835
```

```
print(svm_Radial)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 250 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 225, 225, 225, 226, 225, 225, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.5617000  0.08695771
##   0.50  0.5966115  0.17562602
##   1.00  0.6003103  0.18501000
##
## Tuning parameter 'sigma' was held constant at a value of 0.2569873
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.2569873 and C = 1.
```

```r
plot(svm_Radial)
```



8.6. PREDICTIONS

```r
set.seed(123)
svm_Radial_predict <- predict(svm_Radial, newdata = testing)
```

8.7. THE CONFUSION MATRIX

```r
set.seed(123)
confusionMatrix(svm_Radial_predict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      39   25
##    PASS         19   23
```

```
##
##                Accuracy : 0.5849
##                  95% CI : (0.4851, 0.6798)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : 0.2479
##
##                   Kappa : 0.1532
##
##  Mcnemar's Test P-Value : 0.4510
##
##             Sensitivity : 0.6724
##             Specificity : 0.4792
##          Pos Pred Value : 0.6094
##          Neg Pred Value : 0.5476
##              Prevalence : 0.5472
##          Detection Rate : 0.3679
##    Detection Prevalence : 0.6038
##       Balanced Accuracy : 0.5758
##
##        'Positive' Class : NO_PASS
##
```

The ACCURACY of the MODEL is :

```
mean(svm_Radial_predict == testing$RESULT)
```

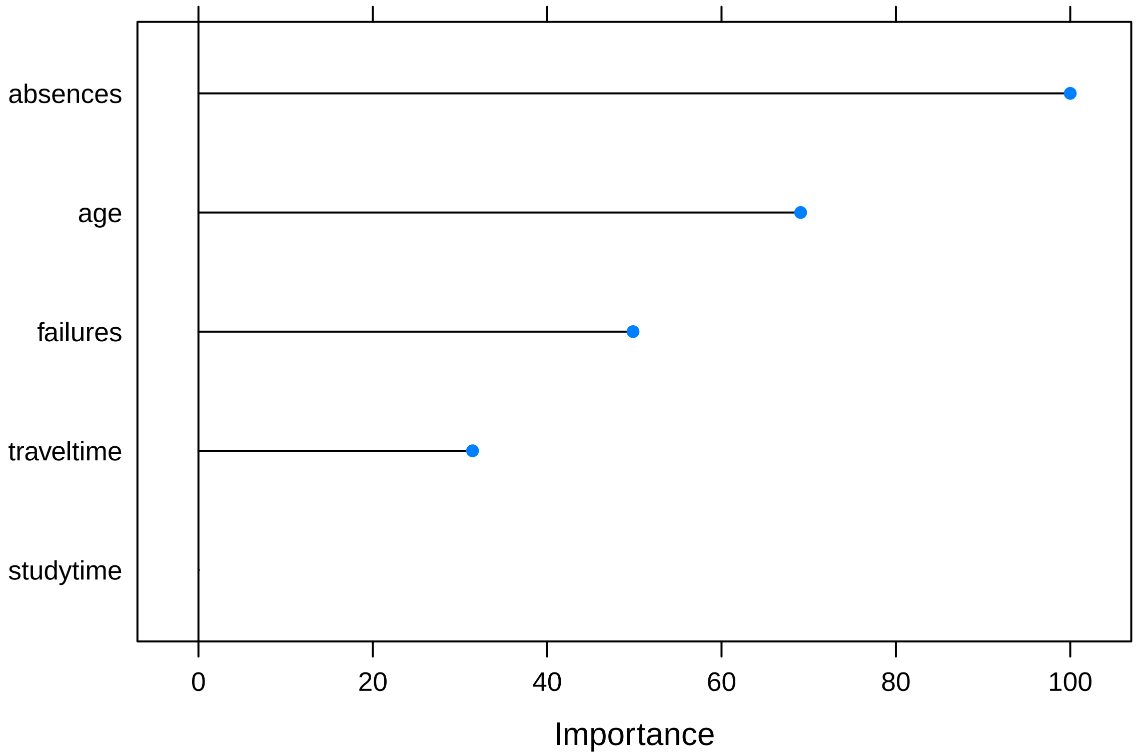
```
## [1] 0.5849057
```

8.8. THE VARIABLE IMPORTANCE

```
X <- varImp(svm_Radial)
print(X)
```

```
## ROC curve variable importance
##
##            Importance
## absences       100.00
## age             69.07
## failures        49.85
## traveltime      31.44
## studytime        0.00
```

```
plot(X)
```


absences
age
failures
traveltime
studytime
0
20
40
60
80
100
Importance

9. TRAINING AND PREDICTIONS WITH SVM

As it is described in the book of the class "Machine Learning with R", we may also use the package "kernlab" where ksvm() function uses the Gaussian RBF kernel (by default),

or it may use the following other kernels:

- 'rbfdot' Radial Basis kernel "Gaussian"
- 'polydot' Polynomial kernel
- 'vanilladot' Linear kernel
- 'tanhdot' Hyperbolic tangent kernel
- 'laplacedot' Laplacian kernel
- 'besseldot' Bessel kernel
- 'anovadot' ANOVA RBF kernel
- 'splinedot' Spline kernel
- 'stringdot' String kernel

We have chosen to work below with **rbfdot** and **tanhdot**.

9.1. TRAINING

```
suppressPackageStartupMessages(library(klaR))
suppressPackageStartupMessages(library(kernlab))

set.seed(123)

model.ksvm1 <- ksvm(RESULT ~ age + traveltime + studytime + failures + absences,
                    data = training,
                    kernel="rbfdot")

model.ksvm1
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.256987320110779
##
## Number of Support Vectors : 226
##
## Objective Function Value : -186.0894
## Training error : 0.316
```

```
model.ksvm2 <- ksvm(RESULT ~ age + traveltime + studytime + failures + absences,
                    data = training,
                    kernel="tanhdot")
```

```
##  Setting default kernel parameters
```

```r
model.ksvm2
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Hyperbolic Tangent kernel function.
##  Hyperparameters : scale =  1  offset =  1
##
## Number of Support Vectors : 140
##
## Objective Function Value : -874.9083
## Training error : 0.536
```

9.2. PREDICTIONS

```r
set.seed(123)

model.ksvm1.results <- predict(model.ksvm1, testing, type="response")
head(model.ksvm1.results)
```

```
## [1] NO_PASS NO_PASS NO_PASS NO_PASS PASS    NO_PASS
## Levels: NO_PASS PASS
```

```r
table(model.ksvm1.results, testing$RESULT)
```

```
##
## model.ksvm1.results NO_PASS PASS
##             NO_PASS      39   25
##             PASS         19   23
```

```r
agreement1 <-  model.ksvm1.results == testing$RESULT
table(agreement1)
```

```
## agreement1
## FALSE  TRUE
##    44    62
```

```r
prop.table(table(agreement1))
```

```
## agreement1
##     FALSE      TRUE
## 0.4150943 0.5849057
```

```r
model.ksvm2.results <- predict(model.ksvm2, testing, type="response")
head(model.ksvm2.results)
```

```
## [1] PASS    PASS    NO_PASS PASS    PASS    PASS
## Levels: NO_PASS PASS
```

```r
table(model.ksvm2.results, testing$RESULT)
```

```
##
## model.ksvm2.results NO_PASS PASS
```

```
##             NO_PASS      35   32
##             PASS         23   16
```

```
agreement2 <-  model.ksvm2.results == testing$RESULT
table(agreement2)
```

```
## agreement2
## FALSE  TRUE
##    55    51
```

```
prop.table(table(agreement2))
```

```
## agreement2
##     FALSE      TRUE
## 0.5188679 0.4811321
```

We have followed above the materials from the book recommended in the class :

https://www.amazon.com/Machine-Learning-R-Brett-Lantz-ebook/dp/B00G9581JM

although using CARET package is simpler and easier, and permits also a direct calculation of the CONFUSION MATRIX and VARIABLE IMPORTANCE.

10. CONCLUSIONS

Here below we are comparing the algorithms that we have used above,

particularly ANN,

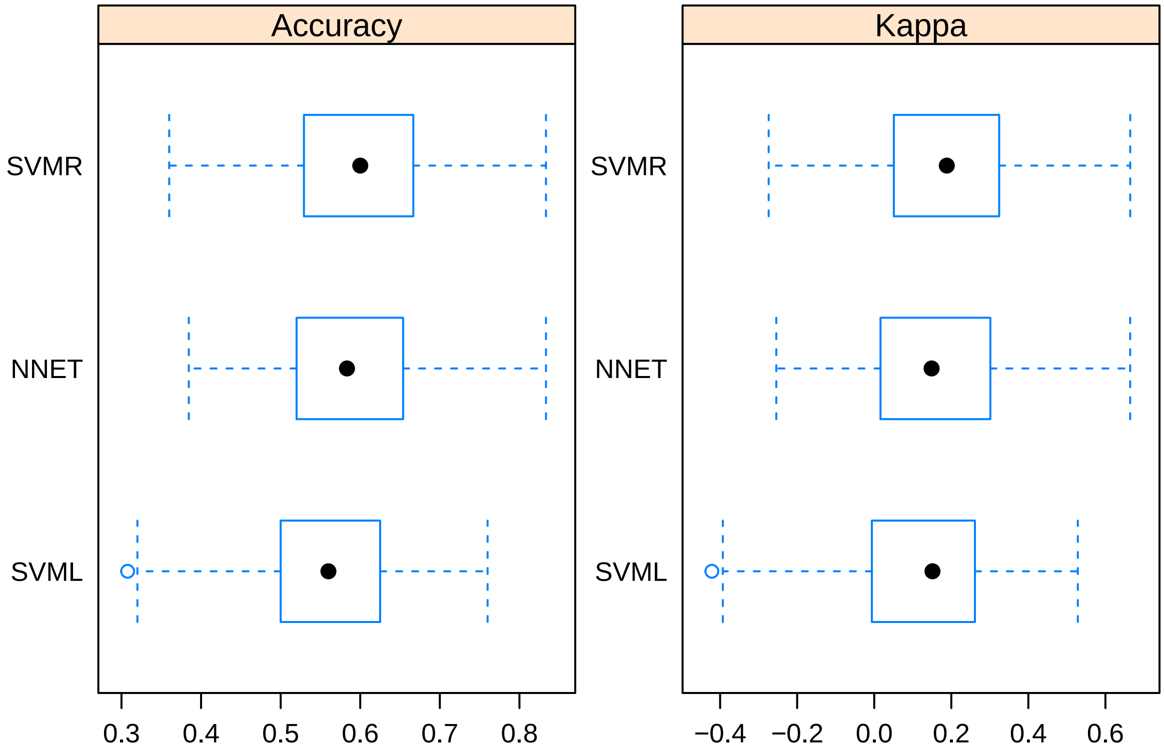
and SVM with a Linear Kernel,

and SVM with a Radial Basis Function (RBF) Kernel.

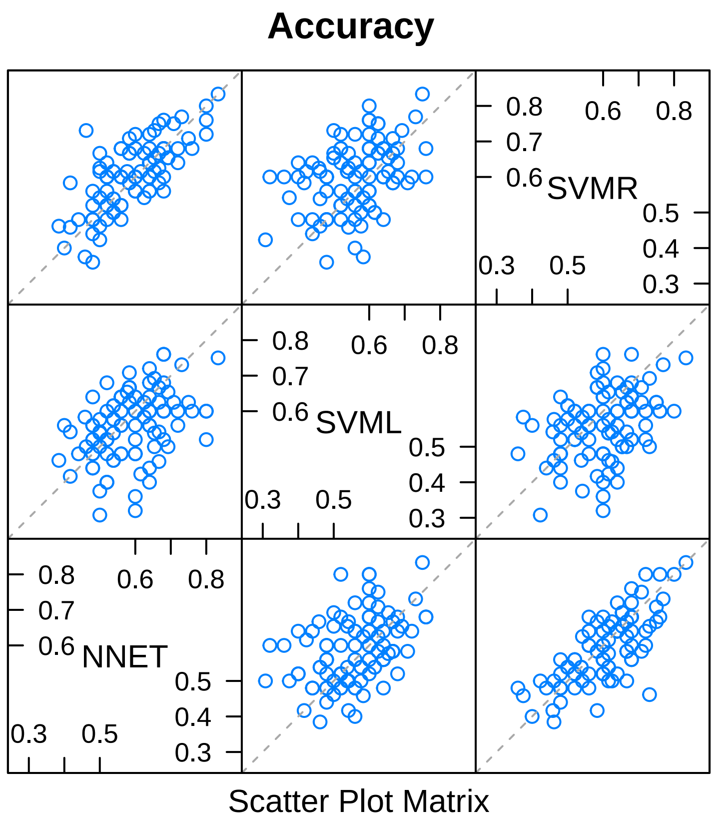
```
set.seed(123)

algo_results <- resamples(list(NNET=fit.nn,
                               SVML=svm_Linear,
                               SVMR=svm_Radial))

summary(algo_results)
```

```
##
## Call:
## summary.resamples(object = algo_results)
##
## Models: NNET, SVML, SVMR
## Number of resamples: 100
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NNET 0.3846154 0.5200000 0.5833333 0.5878128 0.6538462 0.8333333    0
## SVML 0.3076923 0.5000000 0.5600000 0.5599474 0.6250000 0.7600000    0
## SVMR 0.3600000 0.5338462 0.6000000 0.6003103 0.6666667 0.8333333    0
##
## Kappa
##            Min.     1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NNET -0.2541806  0.02113615 0.1489362 0.1640177 0.3004023 0.6643357    0
## SVML -0.4214047 -0.00619195 0.1512265 0.1263012 0.2607792 0.5283019    0
## SVMR -0.2738854  0.05400364 0.1883117 0.1850100 0.3243243 0.6643357    0
```

```
scales <- list(x=list(relation="free"), y=list(relation="free"))

bwplot(algo_results, scales=scales)
```




```
splom(algo_results)
```




```
diffs <- diff(algo_results)

# summarize p-values for pair-wise comparisons

summary(diffs)
```

```
##
## Call:
## summary.diff.resamples(object = diffs)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##      NNET      SVML      SVMR
## NNET            0.02787  -0.01250
## SVML 0.0237832           -0.04036
## SVMR 0.2034813 0.0006784
##
## Kappa
##      NNET    SVML     SVMR
## NNET         0.03772 -0.02099
## SVML 0.20771         -0.05871
## SVMR 0.37175 0.01808
```

As we can see, by comparing the **ACCURACY** on Box-and-Whisker plots, the ML model that is based on

SVM-RBF performs better than the ML models that are based on ANN and SVM-LK, although the precise **ACCURACY** that we have obtained with SVM-RBF is only 0.58.

In fact, considering the precise values of the **ACCURACY**, we could rank **ANN** (0.62), followed by **SVM-LK** (0.61), and **SVM-RBF** (0.58).

Also the **FEATURES** that are considered as important differ between these three models :

in **ANN** model, the order of feature importance is :

“studytime”, failures“,”absences“,”age" ;

in sharp contrast with the model based on **SVM (LK or RBF)**, that place more emphasis on :

“absences”, “age”, “failures”, and “traveltime” (SVM-LK and SVM-RBF).

10. adding another models

4. DATA SUMMARY and VISUALIZATION

In the section 4, we aim to address the following Q1 from the course.

STEP 1 Data Descriptive Statistics

Q1. Amongst the variables of interest identify one that is categorical and one that is quantitative and then provide the following descriptive deliverables:

Summaries (Do this for at least one categorical and one quantitative variable).

a) For the categorical variable create a frequency distribution.

b) For the categorical variable create a bar diagram.

c) For the quantitative variable create numerical summaries grouped by a categorical variable.

d) For the quantitative variable create a histogram and a boxplot grouped by categorical variable.

4. DATA SUMMARY and VISUALIZATION

We display the GRADE G3 (NO PASS/PASS), function of AGE

```r
## after we REMOVE the RECORDS where the GRADE G3 is > 2 ;
## we add a new piece of R code where we display the GRADE G3, function of AGE

student3 %>%
  group_by(RESULT, age) %>%
  summarise (n = n()) %>%
  mutate(freq = n / sum(n))
```

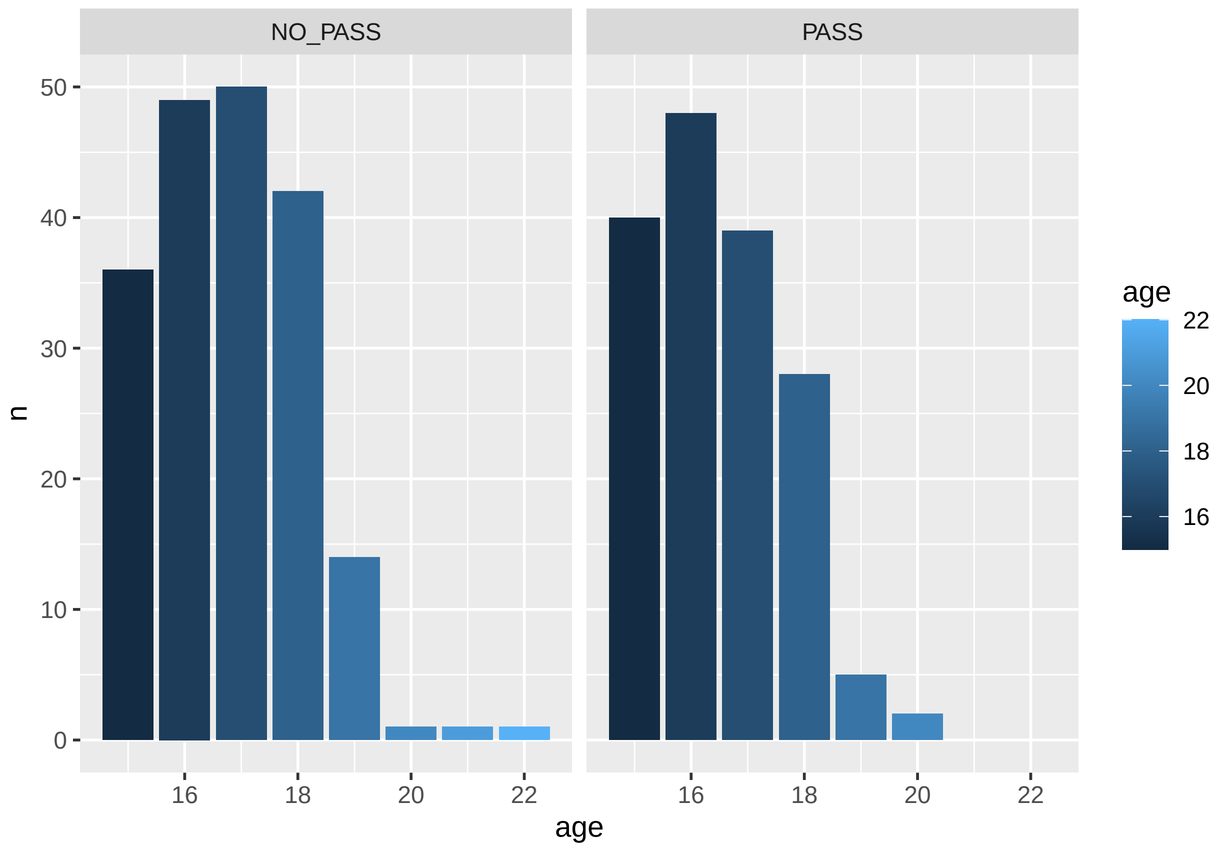
```
## `summarise()` has grouped output by 'RESULT'. You can override using the `.groups` argument.

## # A tibble: 14 x 4
## # Groups:   RESULT [2]
##    RESULT    age     n    freq
##    <fct>   <int> <int>   <dbl>
##  1 NO_PASS    15    36 0.186
##  2 NO_PASS    16    49 0.253
##  3 NO_PASS    17    50 0.258
##  4 NO_PASS    18    42 0.216
##  5 NO_PASS    19    14 0.0722
##  6 NO_PASS    20     1 0.00515
##  7 NO_PASS    21     1 0.00515
##  8 NO_PASS    22     1 0.00515
##  9 PASS       15    40 0.247
## 10 PASS       16    48 0.296
## 11 PASS       17    39 0.241
## 12 PASS       18    28 0.173
## 13 PASS       19     5 0.0309
## 14 PASS       20     2 0.0123
```

```r
# %>% arrange(desc(freq))

student3 %>%
  group_by(RESULT, age) %>%
  tally() %>%
  arrange(desc(n))
```

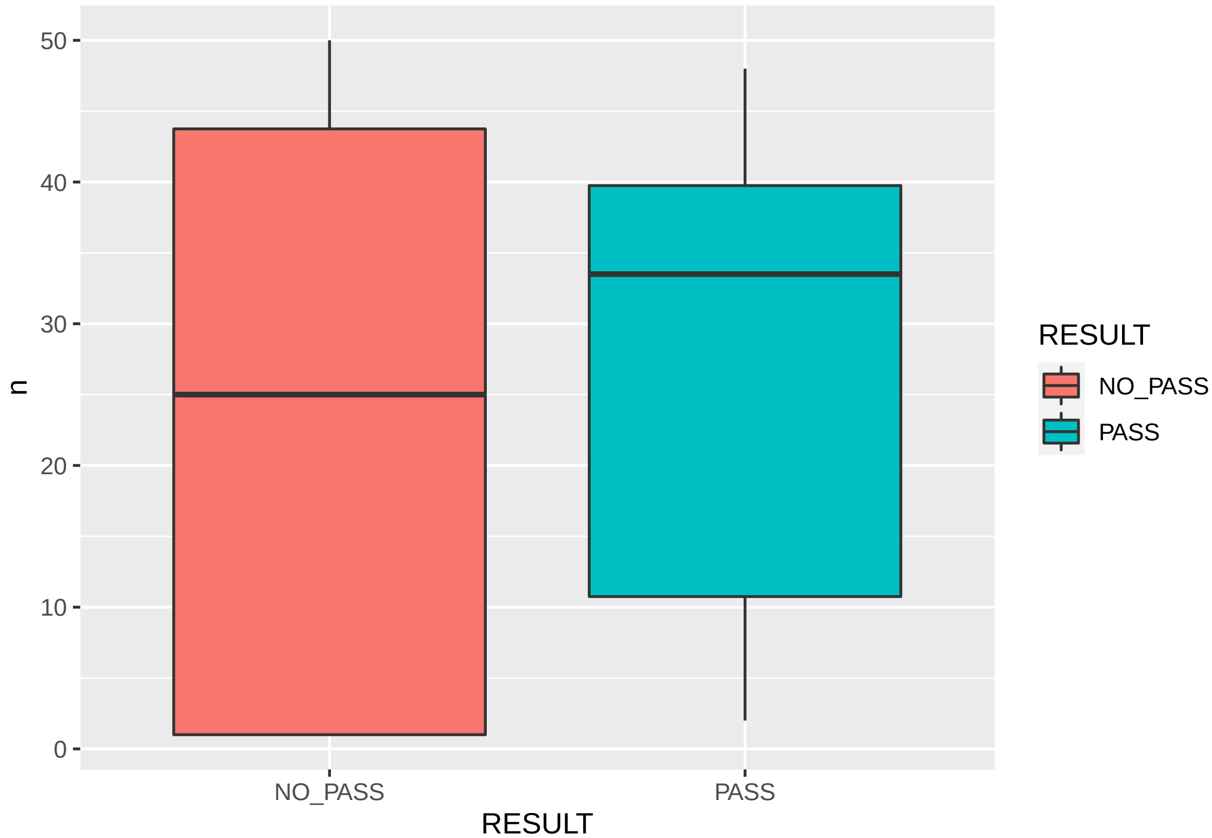
```
## # A tibble: 14 x 3
## # Groups:   RESULT [2]
##    RESULT    age     n
##    <fct>   <int> <int>
##  1 NO_PASS    17    50
##  2 NO_PASS    16    49
##  3 PASS       16    48
##  4 NO_PASS    18    42
##  5 PASS       15    40
##  6 PASS       17    39
##  7 NO_PASS    15    36
##  8 PASS       18    28
##  9 NO_PASS    19    14
```

```
## 10 PASS       19      5
## 11 PASS       20      2
## 12 NO_PASS    20      1
## 13 NO_PASS    21      1
## 14 NO_PASS    22      1
```

```
student3 %>%
  group_by(RESULT, age) %>%
  tally() %>%
  arrange(desc(n)) %>%
ggplot(aes(x = age, y=n)) +
       geom_bar(stat="identity", aes(fill=age)) +
       facet_wrap(~RESULT)
```




```
student3 %>%
  group_by(RESULT, age) %>%
  tally() %>%
  # arrange(desc(n)) %>%
ggplot(aes(x = RESULT, y=n)) +
       geom_boxplot(aes(fill=RESULT))
```



50
40
30
20
10
0
n
NO_PASS
PASS
RESULT
RESULT
NO_PASS
PASS

4. DATA SUMMARY and VISUALIZATION

We display the GRADE G3 (NO PASS/PASS), function of ABSENCES

```
## after we REMOVE the RECORDS where the GRADE G3 is > 2 ;
## we add a new piece of R code where we display the GRADE G3, function of ABSENCES

student3 %>%
  group_by(RESULT, absences) %>%
  summarise (n = n()) %>%
  mutate(freq = n / sum(n))
```

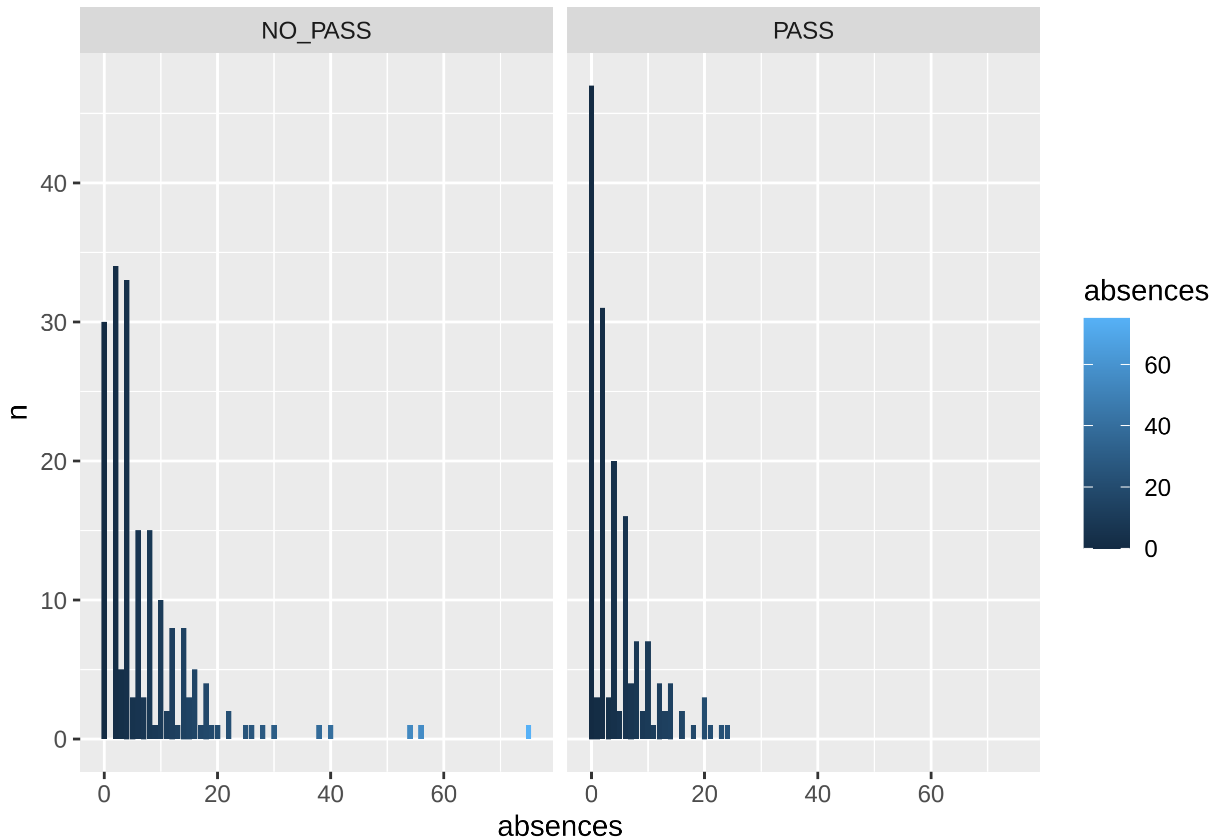
```
## `summarise()` has grouped output by 'RESULT'. You can override using the `.groups` argument.
```

```
## # A tibble: 51 x 4
## # Groups:   RESULT [2]
##    RESULT  absences     n    freq
##    <fct>      <int> <int>   <dbl>
##  1 NO_PASS        0    30 0.155
##  2 NO_PASS        2    34 0.175
##  3 NO_PASS        3     5 0.0258
##  4 NO_PASS        4    33 0.170
##  5 NO_PASS        5     3 0.0155
##  6 NO_PASS        6    15 0.0773
##  7 NO_PASS        7     3 0.0155
##  8 NO_PASS        8    15 0.0773
##  9 NO_PASS        9     1 0.00515
## 10 NO_PASS       10    10 0.0515
## # ... with 41 more rows
```

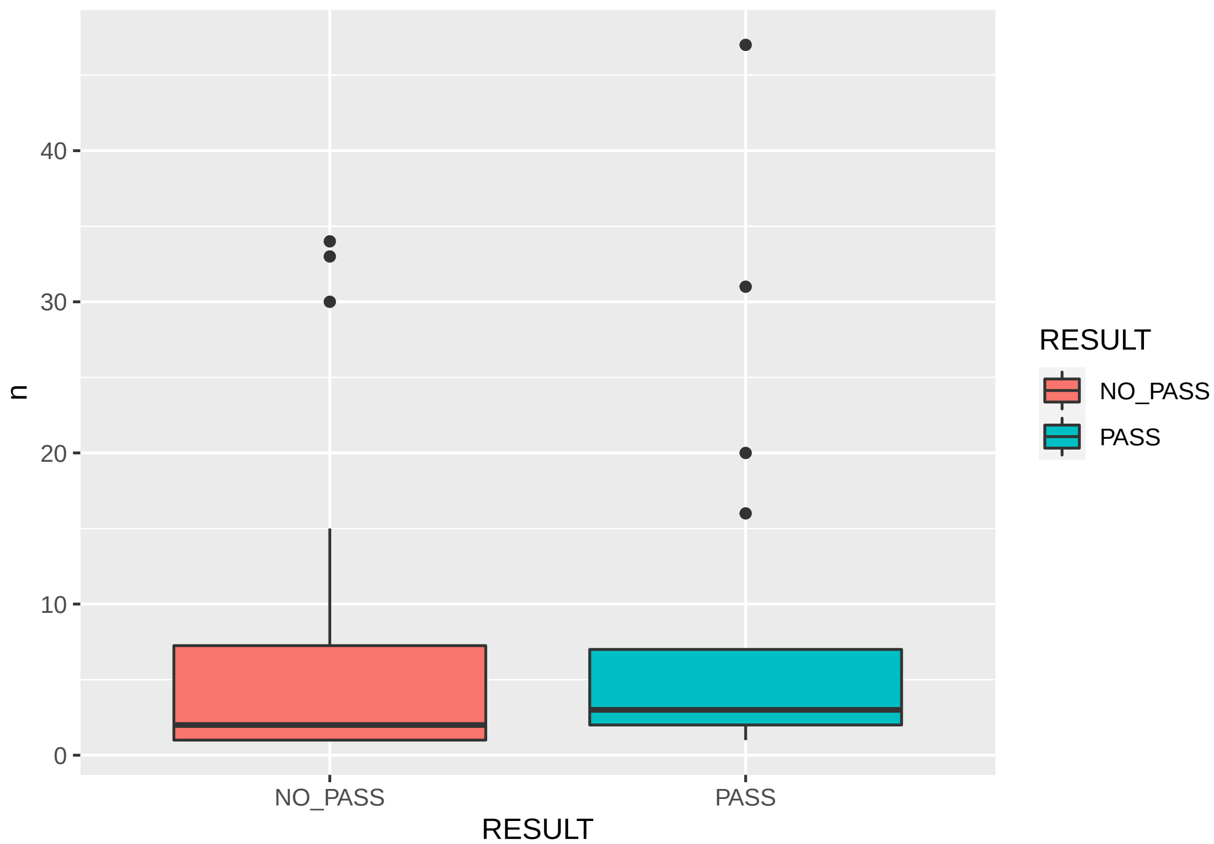
```
# %>% arrange(desc(freq))

student3 %>%
  group_by(RESULT, absences) %>%
  tally() %>%
  arrange(desc(n))
```

```
## # A tibble: 51 x 3
## # Groups:   RESULT [2]
##    RESULT  absences     n
##    <fct>      <int> <int>
##  1 PASS           0    47
##  2 NO_PASS        2    34
##  3 NO_PASS        4    33
##  4 PASS           2    31
##  5 NO_PASS        0    30
##  6 PASS           4    20
##  7 PASS           6    16
##  8 NO_PASS        6    15
##  9 NO_PASS        8    15
## 10 NO_PASS       10    10
## # ... with 41 more rows
```

```r
student3 %>%
  group_by(RESULT, absences) %>%
  tally() %>%
  arrange(desc(n)) %>%
ggplot(aes(x = absences, y=n)) +
       geom_bar(stat="identity", aes(fill=absences)) +
      facet_wrap(~RESULT)
```




```r
student3 %>%
  group_by(RESULT, absences) %>%
  tally() %>%
  # arrange(desc(n)) %>%
ggplot(aes(x = RESULT, y=n)) +
       geom_boxplot(aes(fill=RESULT))
```



40
30
20
10
0
n
NO_PASS
PASS
RESULT
RESULT
NO_PASS
PASS

4. DATA SUMMARY and VISUALIZATION : the CORRELATION PLOTS

We aim to address the following Q2 from the course

STEP 2 Correlation and Regression Analysis

although the data that we have chosen and the numerical features does not allow us a classical regression analysis. We will do it, just to set up the code in R (for other datasets).

Q2. Among the quantitative variables generate Relationships and Associations.

Correlation and Regression:

a) Identify two or more quantitative variables that might be correlated.

b) Find the correlation coefficient.

c) Create the scatter diagram under graphs.

d) Provide your rationale and justify your findings regarding the correlation between two quantitative variables of interest.

Here, we aim to answer also the question Q3 :

Q3.Prepare data by using the following preprocessing transformation and plots:

a) Please standardize the data.

b) Check for null values

c) Check for outliers

d) Check for Regression assumptions generate regression diagnostic plots.

4. DATA SUMMARY and VISUALIZATION : the CORRELATION PLOTS

We display the SCATTER PLOTS between the numerical features that we have the dataset i.e. AGE and ABSENCES (although the SCATTER PLOTS looks atypical for the data that we have chosen).

```
# library(Hmisc)
suppressMessages(library(Hmisc))

# computing the CORRELATION COEFFICIENT between AGE and ABSENCES ;
# we find a SMALL CORRELATION COEFFICIENT (< 0.3)

cor(student3$age, student3$absences)
```

```
## [1] 0.2152499
```

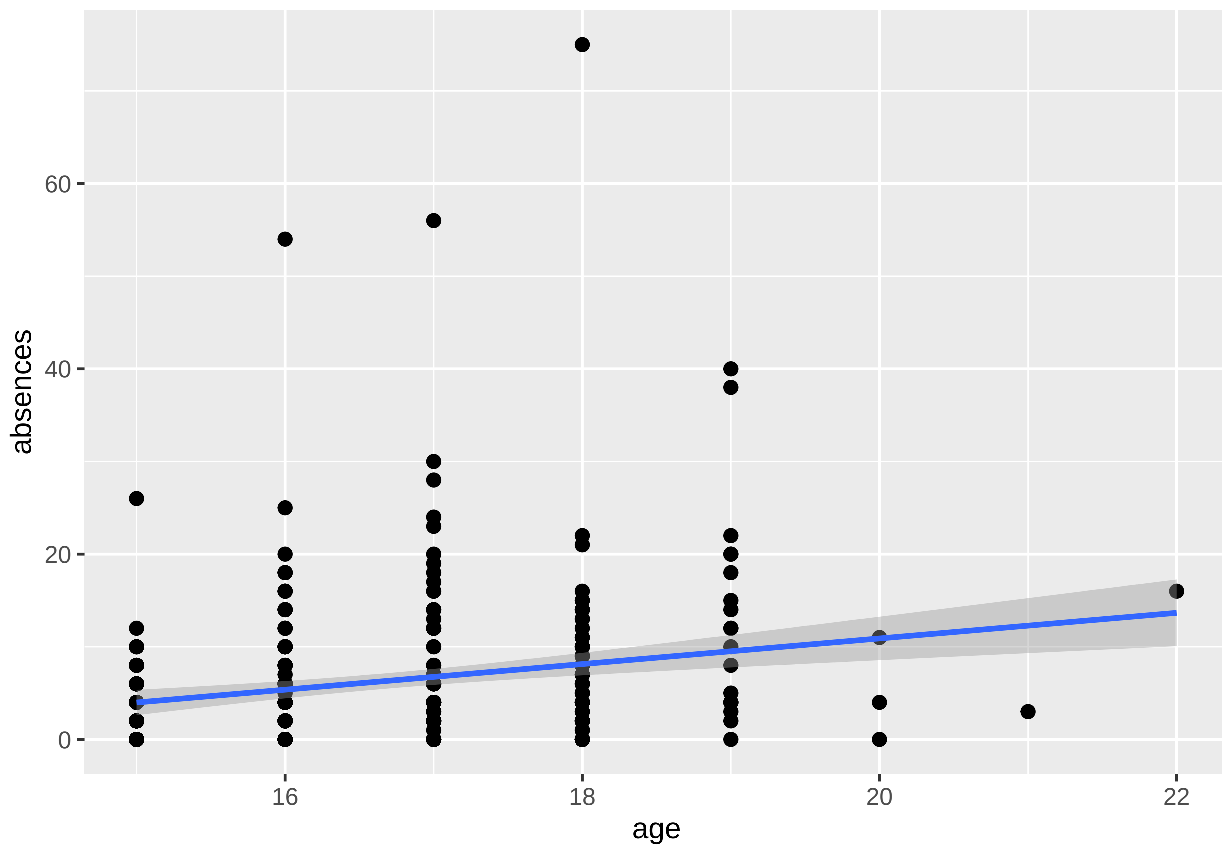
As the CORRELATION COEFFICIENT is small (<0.3), we can keep both AGE and ABSENCES in the model, as **INDEPENDENT FEATURES** (we know that some ML approaches are sensitive to features that are highly correlated).

```
cov(student3$age, student3$absences)
```

```
## [1] 2.229617
```

```
# a SCATTER PLOT using geom_smooth()

# ggplot(student3, aes(x=age, y=absences)) +
#       geom_point(size=2) +
#       geom_smooth()

# a SCATTER PLOT using geom_smooth(method=lm)

ggplot(student3, aes(x=age, y=absences)) +
       geom_point(size=2) +
       geom_smooth(method=lm)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



As an exercise in this section, as we look at the correlation between **AGE** and **ABSENCES**, we also perform a more formal linear regression analysis and compute the **DIAGNOSTIC PLOTS**.

```
library(broom) ### in order to add : AUGMENT

## A LM approach :

reg_model <- lm(absences~age, data = student3)

reg_model
```

```
##
## Call:
## lm(formula = absences ~ age, data = student3)
##
## Coefficients:
## (Intercept)          age
##     -16.753        1.383
```

```
## Listing R.squared in the LM approach :

summary(reg_model)$r.squared
```

```
## [1] 0.04633253
```

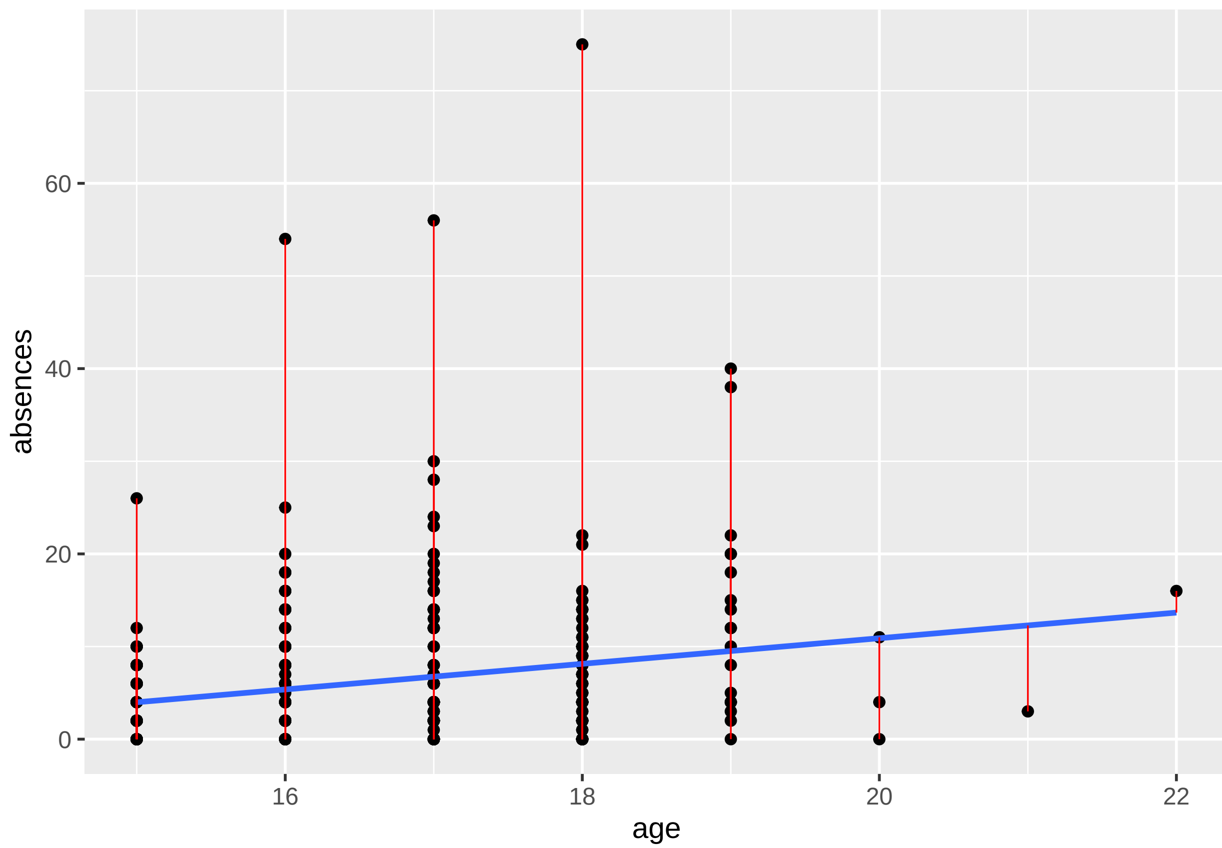
```
## Making the Diagnostic Plots:

reg_model.diagnostics <- augment(reg_model)
```

```r
head(reg_model.diagnostics)
```

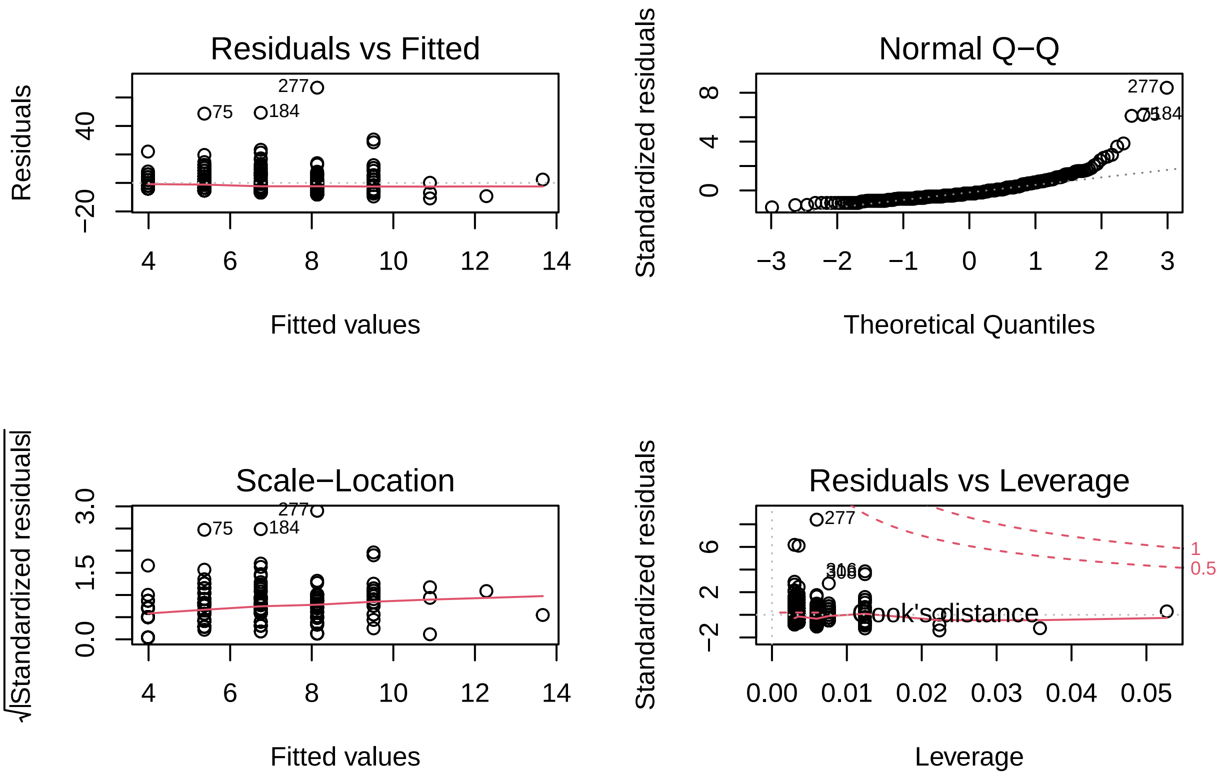
```
## # A tibble: 6 x 9
##   .rownames absences   age .fitted .resid    .hat .sigma   .cooksd .std.resid
##   <chr>        <int> <int>   <dbl>  <dbl>   <dbl>  <dbl>     <dbl>      <dbl>
## 1 1                6    18    8.13  -2.13 0.00597   7.99 0.000216      -0.268
## 2 2                4    17    6.75  -2.75 0.00302   7.99 0.000180      -0.345
## 3 3               10    15    3.99   6.01 0.00759   7.98 0.00219        0.757
## 4 4                2    15    3.99  -1.99 0.00759   7.99 0.000239      -0.250
## 5 5                4    16    5.37  -1.37 0.00356   7.99 0.0000527     -0.172
## 6 6               10    16    5.37   4.63 0.00356   7.98 0.000604       0.582
```

```r
## Another view at the data ;
## potentially to identify the outlier values :

ggplot(reg_model.diagnostics, aes(age, absences)) +
geom_point() +
stat_smooth(method = lm, se = FALSE) +
geom_segment(aes(xend = age, yend = .fitted), color = "red", size = 0.3)
```

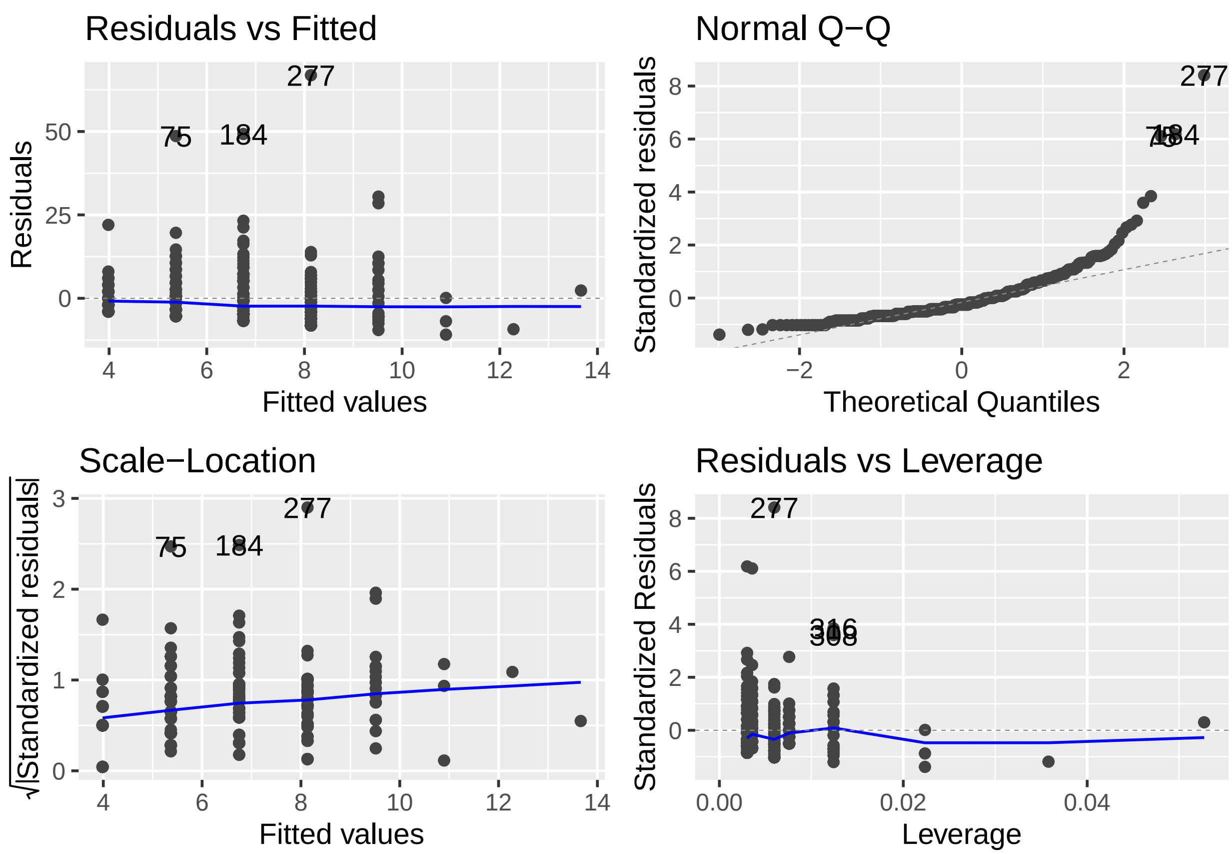
```
## `geom_smooth()` using formula 'y ~ x'
```




```r
## A view at the LINEAR REGRESSION RESULTS :

par(mfrow = c(2, 2))
plot(reg_model)
```




```r
## Another view at the LINEAR REGRESSION RESULTS :

library(ggfortify)
autoplot(reg_model)
```



As we can see in the **Q-Q PLOT**, the distribution of the data is not GAUSSIAN, and there are 3 OUTLIER POINTS that have the INDEXES 75, 184, and 277 (below).

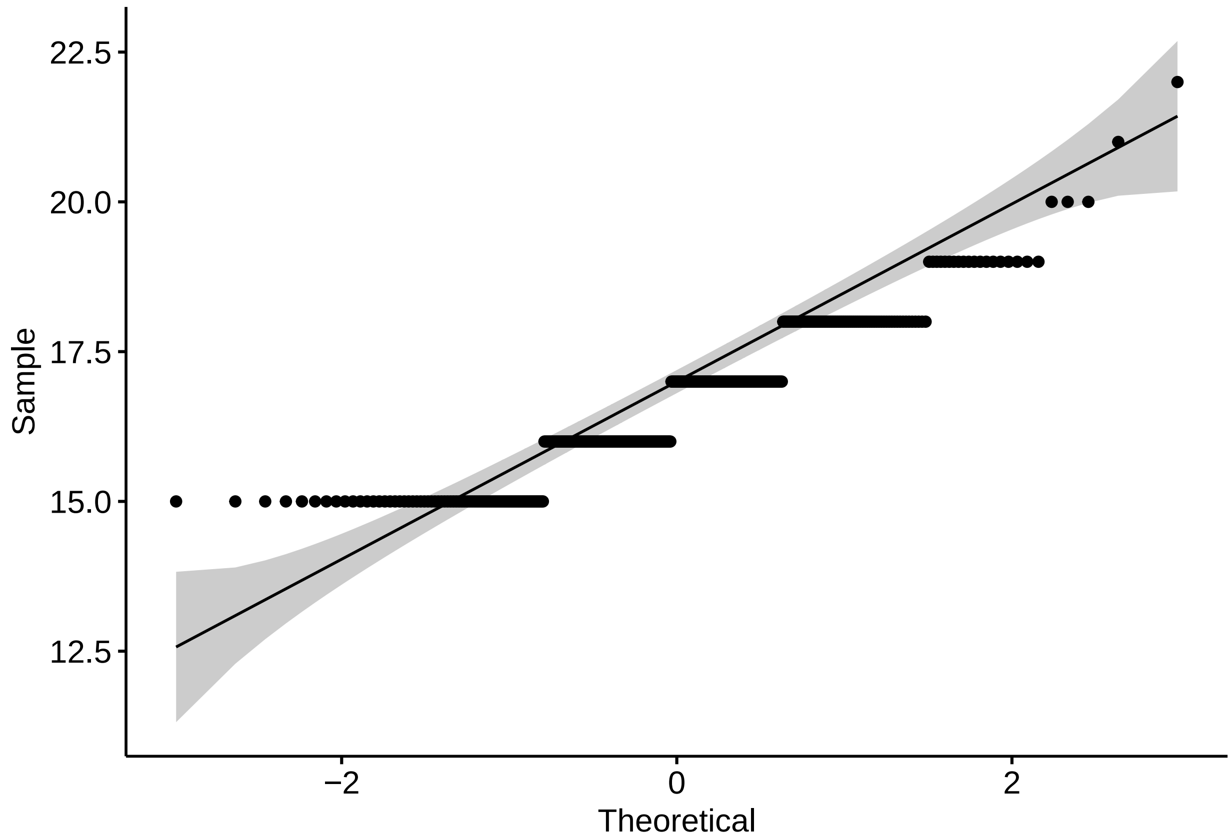
```
## Indeed, the DATA POINTS on ABSENCES that have the INDEXES 75, 184, 277,
## are the TOP OULIERS, and we may wanna remove these OUTLIERS from the data.

student3[75,]
```

```
##    age traveltime studytime failures absences  RESULT
## 75  16          1         2        0       54 NO_PASS
```

```
# student3[184,]
# student3[277,]
```

We also use several methods for normality testing such as **Kolmogorov-Smirnov (K-S) normality test** and **Shapiro-Wilk's test**. As we see below, the hypothesis of "normality" is rejected for both features "age" and "absences".

```
library(ggpubr)
ggqqplot(student3$age)
```



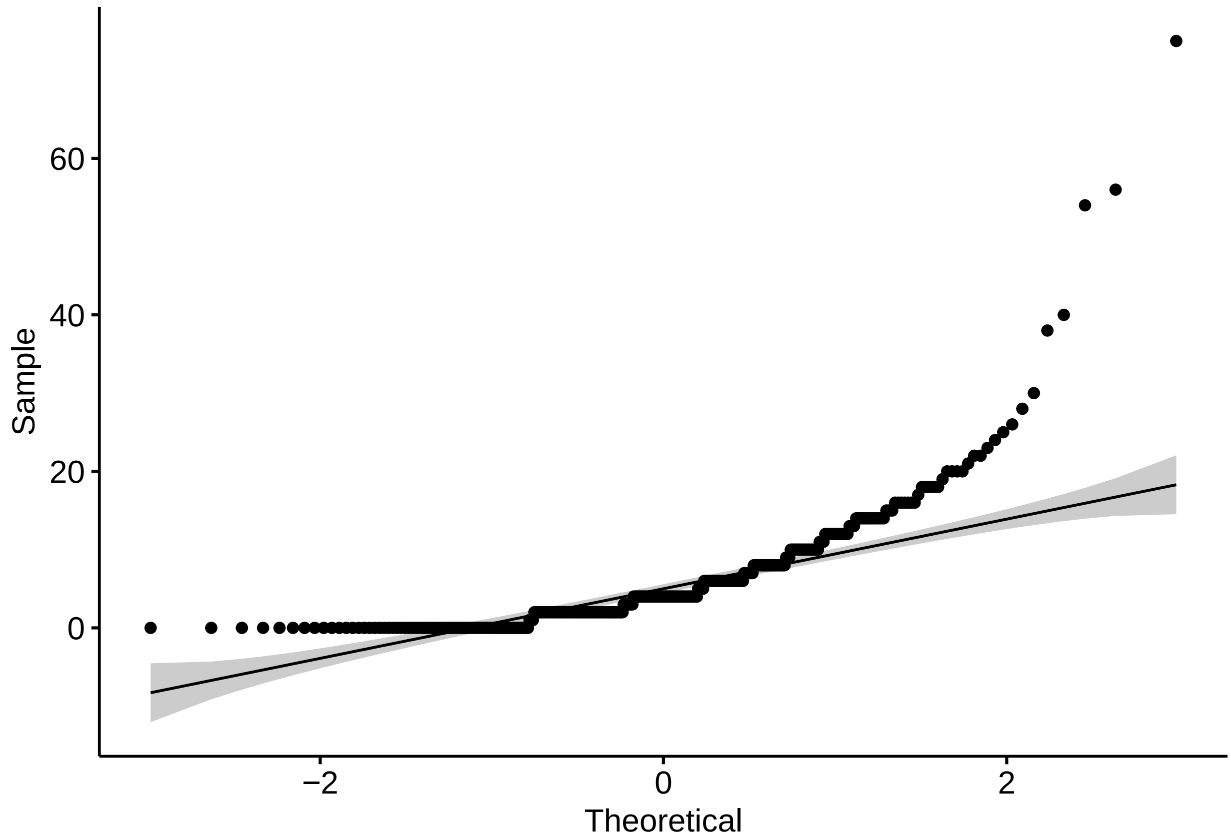

```r
shapiro.test(student3$age)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  student3$age
## W = 0.90721, p-value = 5.763e-14
```

```r
ks.test(student3$age, "pnorm")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  student3$age
## D = 1, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
library(ggpubr)
ggqqplot(student3$absences)
```




```r
shapiro.test(student3$absences)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  student3$absences
## W = 0.67768, p-value < 2.2e-16
```

```r
ks.test(student3$absences, "pnorm")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  student3$absences
## D = 0.75253, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

5. TRAINING AND TEST SETS

```r
## CHOOSING the TRAINING and the TESTING SETS

indxTrain <- createDataPartition(student3$RESULT,
                                 p = .70,
                                 list = FALSE)

training <- student3[indxTrain,]
# training

testing <- student3[-indxTrain,]
# testing

dim(student3)
```

```
## [1] 356   6
```

```r
dim(training)
```

```
## [1] 250   6
```

```r
dim(testing)
```

```
## [1] 106   6
```

6. PRE-PROCESSING THE DATA

We can pre-process the data in a manner that is shown below, by using the COMMAND "preProcess" and "method = c("center","scale")", although it is likely easier to do the pre-processing by using the option "preProcess = c("center","scale")" in *train()*.

```
## PRE-PROCESSING the DATA

trainX        <- training[, names(training) != "RESULT"]

preProcValues <- preProcess(x = trainX, method = c("center", "scale"))

# preProcValues

names(trainX)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
```

```
dim(trainX)
```

```
## [1] 250   5
```

```
names(training)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
```

```
names(testing)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
```

```
scaledTrain <- predict(preProcValues, trainX)
```

7. PERFORMING THE TRAINING

We cover in the following sections the following :

Step 4 Implement Regression and Decision Trees

Conduct Regression and answer the following questions:

Q4 Implement Regression and Decision Tree.

a) Objective and rationale of using the specific algorithm to achieve the objective.

b) Steps of implementing the algorithm with regards to the context.

c) Interpretation of the results and prediction accuracy achieved.

d) Performance improvement techniques and improved accuracy achieved.

Use feature selection, variable importance, compare RMSE(Regression) across models and Information gain (Decision Trees), K-fold cross validation, grid search etc.

e) Implement the two algorithms and state the insights obtained from the implemented project.

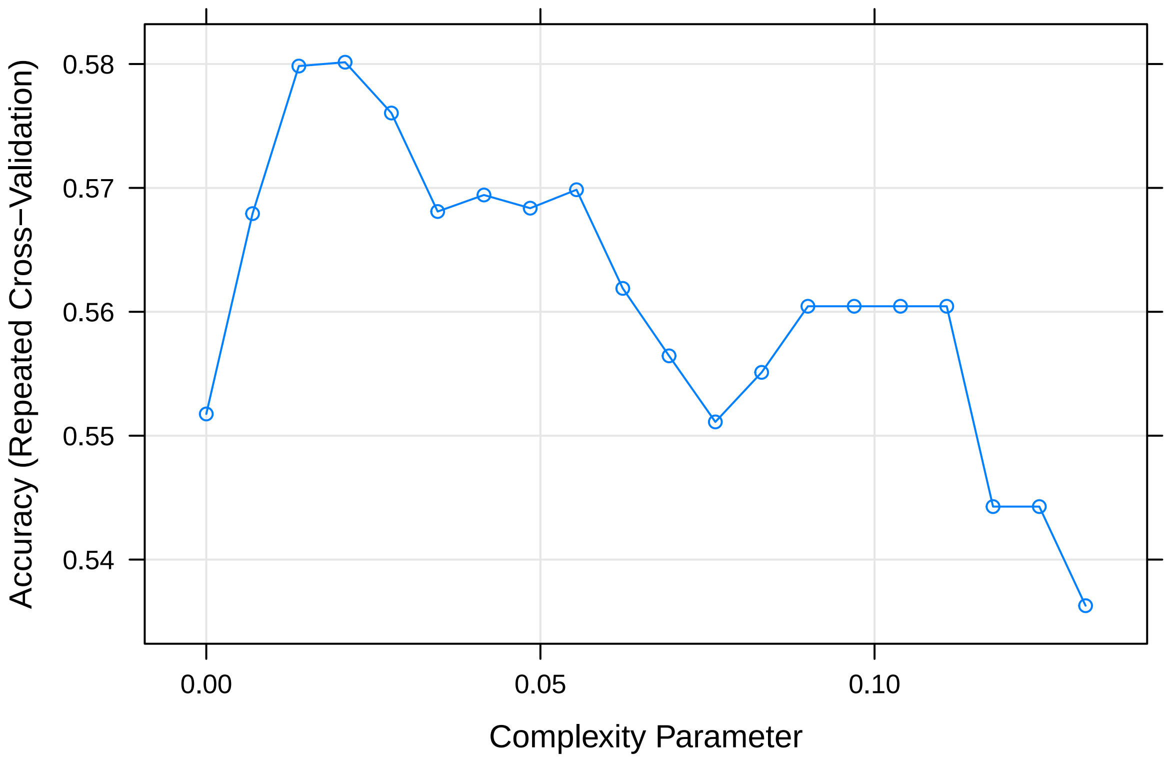
7. PERFORMING THE TRAINING

```r
## PERFORMING the TRAINING

set.seed(400)
ctrl <- trainControl(method="repeatedcv", repeats = 3)

rpartFit <- train( RESULT~ .,
                 data = training,
                 method = "rpart",
                 trControl = ctrl,
                 preProcess = c("center","scale"), tuneLength = 20)

## The output of rpartFit

rpartFit
```

```
## CART
##
## 250 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 225, 226, 224, 225, 225, 224, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.000000000  0.5517521  0.09111987
##   0.006925208  0.5679231  0.12158064
##   0.013850416  0.5798333  0.14335333
##   0.020775623  0.5801453  0.14150887
##   0.027700831  0.5760427  0.13158938
##   0.034626039  0.5680940  0.11066517
##   0.041551247  0.5694274  0.11383183
##   0.048476454  0.5683632  0.10881730
##   0.055401662  0.5698504  0.11040251
##   0.062326870  0.5618889  0.08830228
##   0.069252078  0.5564444  0.06493201
##   0.076177285  0.5511111  0.04817775
##   0.083102493  0.5551111  0.05451288
##   0.090027701  0.5604444  0.06384166
##   0.096952909  0.5604444  0.06384166
##   0.103878116  0.5604444  0.06384166
##   0.110803324  0.5604444  0.06384166
##   0.117728532  0.5442778  0.02430480
##   0.124653740  0.5442778  0.02430480
##   0.131578947  0.5362778  0.00291619
##
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final value used for the model was cp = 0.02077562.
```

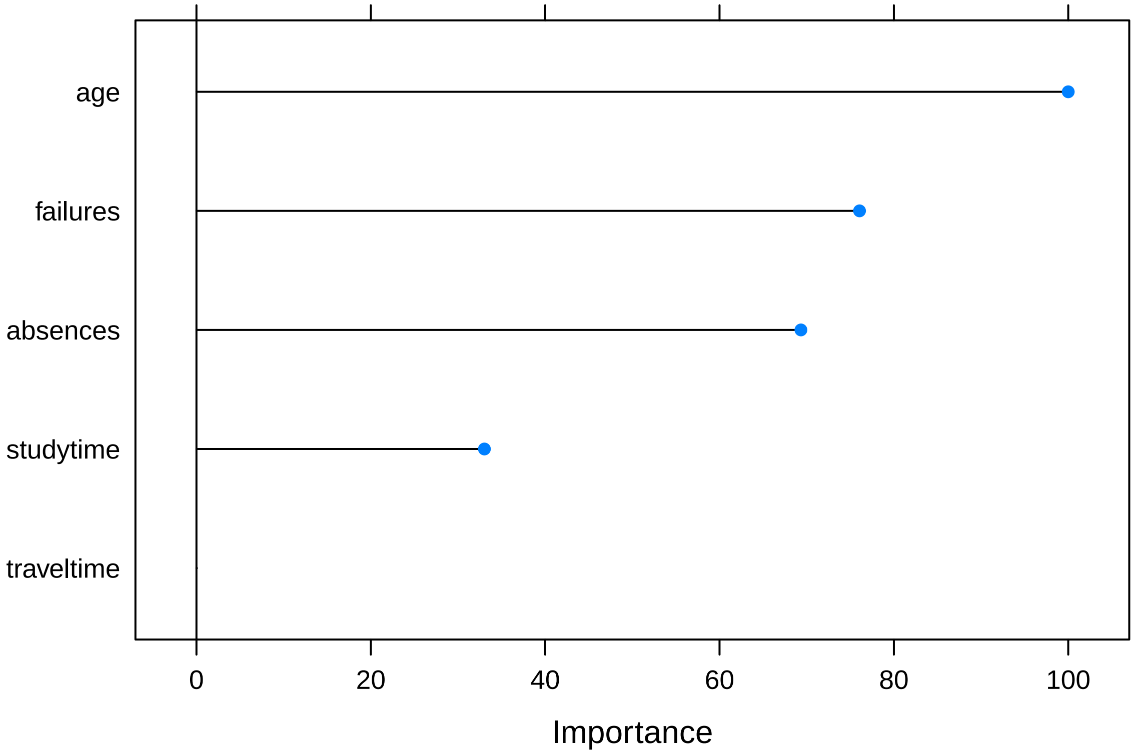
```r
## summary(rpartFit$finalModel)
## it outputs a very long summary

## The plot of rpartFit

plot(rpartFit)
```




```r
png("the.results.rpartFIT.png")
plot(rpartFit)
dev.off()
```

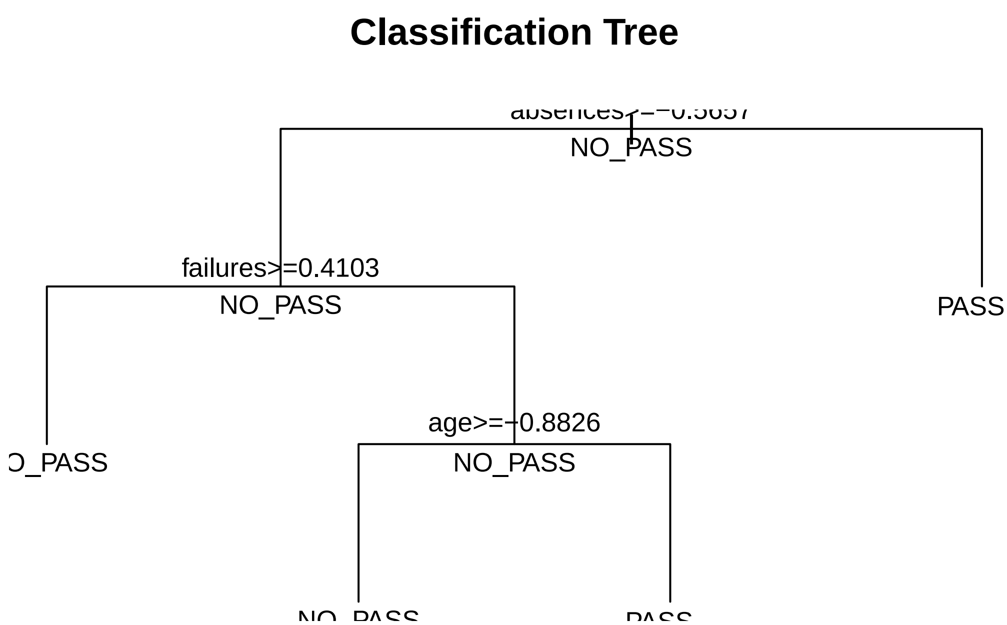
```
## pdf
##   2
```

```r
## To look at the VARIABLE IMPORTANCE

X <- varImp(rpartFit)
plot(X)
```



As we can note, in the current model, **more important FEATURES are AGE, ABSENCES, and STUDY TIME.**

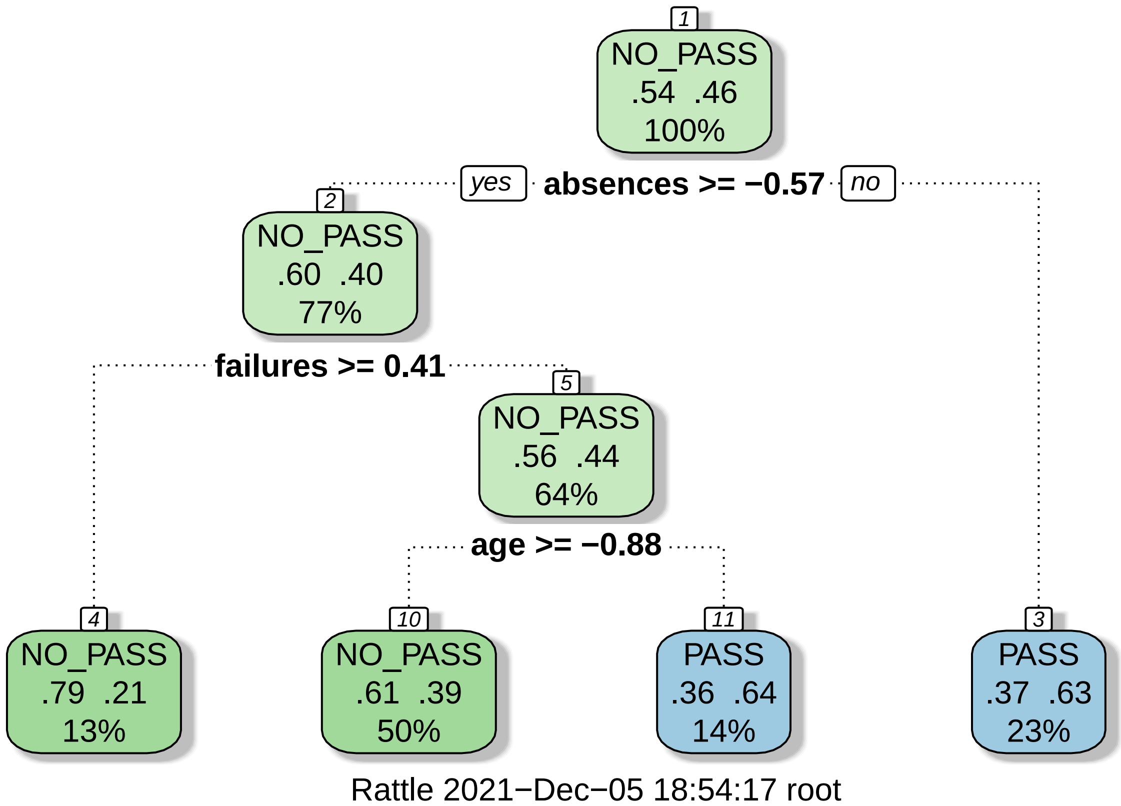
```
### DISPLAYING THE TREE

plot(rpartFit$finalModel,
    uniform=TRUE,
    main="Classification Tree")
text(rpartFit$finalModel, use.n.=TRUE, all=TRUE, cex=.8)
```




```r
png("the.results.rpartFIT.finalModel.png")
plot(rpartFit$finalModel,
    uniform=TRUE,
    main="Classification Tree")
text(rpartFit$finalModel, use.n.=TRUE, all=TRUE, cex=.8)
dev.off()
```

```
## pdf
##   2
```

```r
# library(rattle)
suppressMessages(library(rattle))

fancyRpartPlot(rpartFit$finalModel)
```




```
png("the.results.rpartFIT.fancyR.png")
fancyRpartPlot(rpartFit$finalModel)
dev.off()
```

```
## pdf
##   2
```

8. MAKING THE PREDICTIONS

```r
## Making the PREDICTIONS :

rpartPredict <- predict(rpartFit, newdata = testing)

# rpartPredict
```

We may aim to optimize the model by FEATURE SELECTION or by including NEW FEATURES from the data that is available (we have excluded at the beginning many fetures).

9. THE CONFUSION MATRIX

```r
## COMPUTING the CONFUSION MATRIX :

confusionMatrix(rpartPredict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      43   30
##    PASS         15   18
##
##                Accuracy : 0.5755
##                  95% CI : (0.4757, 0.6709)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : 0.31378
##
##                   Kappa : 0.1196
##
##  Mcnemar's Test P-Value : 0.03689
##
##             Sensitivity : 0.7414
##             Specificity : 0.3750
##          Pos Pred Value : 0.5890
##          Neg Pred Value : 0.5455
##              Prevalence : 0.5472
##          Detection Rate : 0.4057
##    Detection Prevalence : 0.6887
##       Balanced Accuracy : 0.5582
##
##        'Positive' Class : NO_PASS
##
```

```r
mean(rpartPredict == testing$RESULT)
```

```
## [1] 0.5754717
```

```r
dim(student3)
```

```
## [1] 356   6
```

The ACCURACY of the MODEL based on DECISION TREES is :

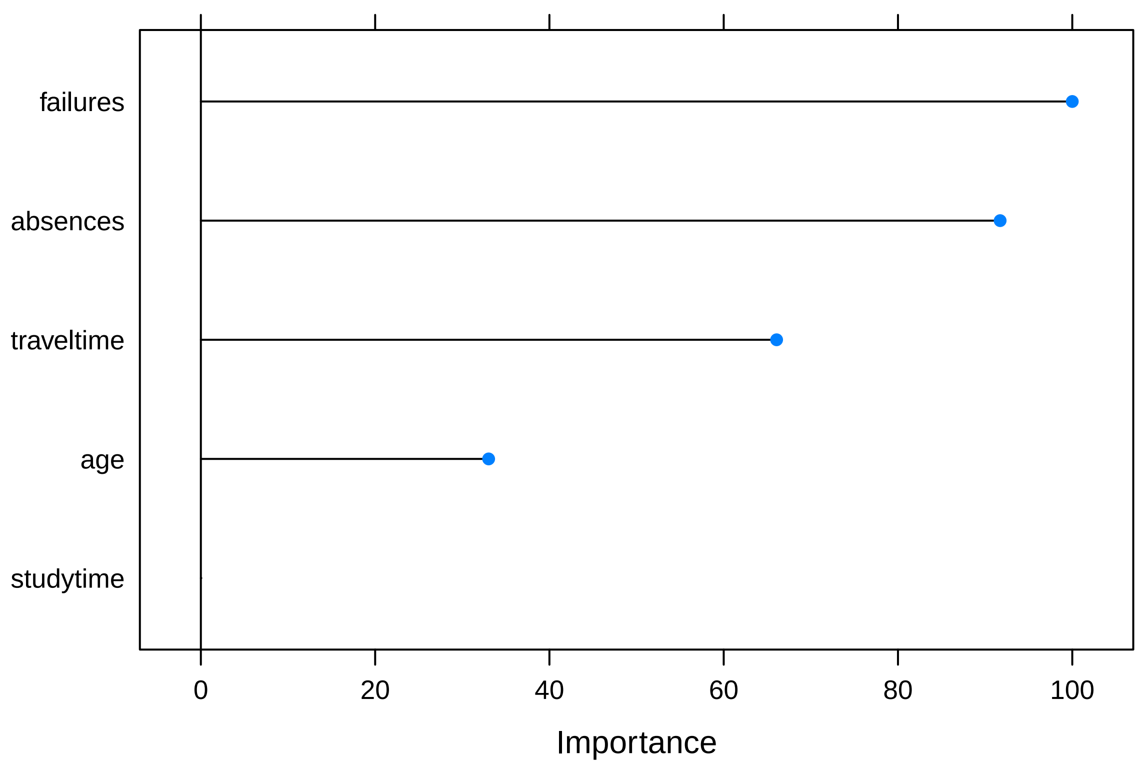
```r
mean(rpartPredict == testing$RESULT)
```

```
## [1] 0.5754717
```

10. CONCLUSIONS AND OTHER MODELS

Because the LABEL is BINARY (PASS/NO PASS), we can compare the performance of the model above with the performance of a model that is based on **LOGISTIC REGRESSION**.

```
logisticFit = train( RESULT ~ .,
  data = training,
  trControl = ctrl,
  method = "glm",
  family = "binomial",
  preProcess = c("center","scale"), tuneLength = 20)

## To look at the VARIABLE IMPORTANCE

X <- varImp(logisticFit)
plot(X)
```




```
## To compute the PREDICTIONS
logisticPredict <- predict(logisticFit, newdata = testing)

## To display the CONFUSION MATRIX
confusionMatrix(logisticPredict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
```

```
##
##             Reference
## Prediction NO_PASS PASS
##    NO_PASS      37   24
##    PASS         21   24
##
##                Accuracy : 0.5755
##                  95% CI : (0.4757, 0.6709)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : 0.3138
##
##                   Kappa : 0.1387
##
##  Mcnemar's Test P-Value : 0.7656
##
##             Sensitivity : 0.6379
##             Specificity : 0.5000
##          Pos Pred Value : 0.6066
##          Neg Pred Value : 0.5333
##              Prevalence : 0.5472
##          Detection Rate : 0.3491
##    Detection Prevalence : 0.5755
##       Balanced Accuracy : 0.5690
##
##        'Positive' Class : NO_PASS
##
```

```r
mean(logisticPredict == testing$RESULT)
```

```
## [1] 0.5754717
```

The ACCURACY of the MODEL based on LOGISTIC REGRESSION is :

```r
mean(logisticPredict == testing$RESULT)
```

```
## [1] 0.5754717
```

As we can see, by comparing the **ACCURACY**, the ML model that is based on **DECISION TREES** performs better than the ML model that is based on **LOGISTIC REGRESSION**.

Also the **FEATURES** that are considered as important differ between these two models : the LOGISTIC REGRESSION model emphasizes more on "failures", "absences", and "traveltime", and less on "studytime" and on "age", in sharp contrast with the model based on DECISION TREES.

A note to add about the model based on DECISION TREES, we do not have to standardize the data..

```r
## CHOOSING the TRAINING and TESTING SETS

indxTrain <- createDataPartition(student3$RESULT,
                                 p = .75,
                                 list = FALSE)

training <- student3[indxTrain,]
# head(training)

testing <- student3[-indxTrain,]
# head(testing)

dim(student3)
```

```
## [1] 356   6
```

```r
dim(training)
```

```
## [1] 268   6
```

```r
dim(testing)
```

```
## [1] 88  6
```

6. PRE-PROCESSING THE DATA

```r
### PRE-PROCESSING the DATA

trainX        <- training[, names(training) != "RESULT"]

# for NB we may not need to CENTER and SCALE the data :)
# preProcValues <- preProcess(x = trainX, method = c("center", "scale"))
# preProcValues

names(trainX)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
```

```r
dim(trainX)
```

```
## [1] 268   5
```

```r
names(training)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
```

```r
### THE BALANCE of the DATA in TRAINING and TESTING SETS

prop.table(table(training$RESULT)) * 100
```

```
##
##  NO_PASS     PASS
## 54.47761 45.52239
```

```r
prop.table(table(testing$RESULT)) * 100
```

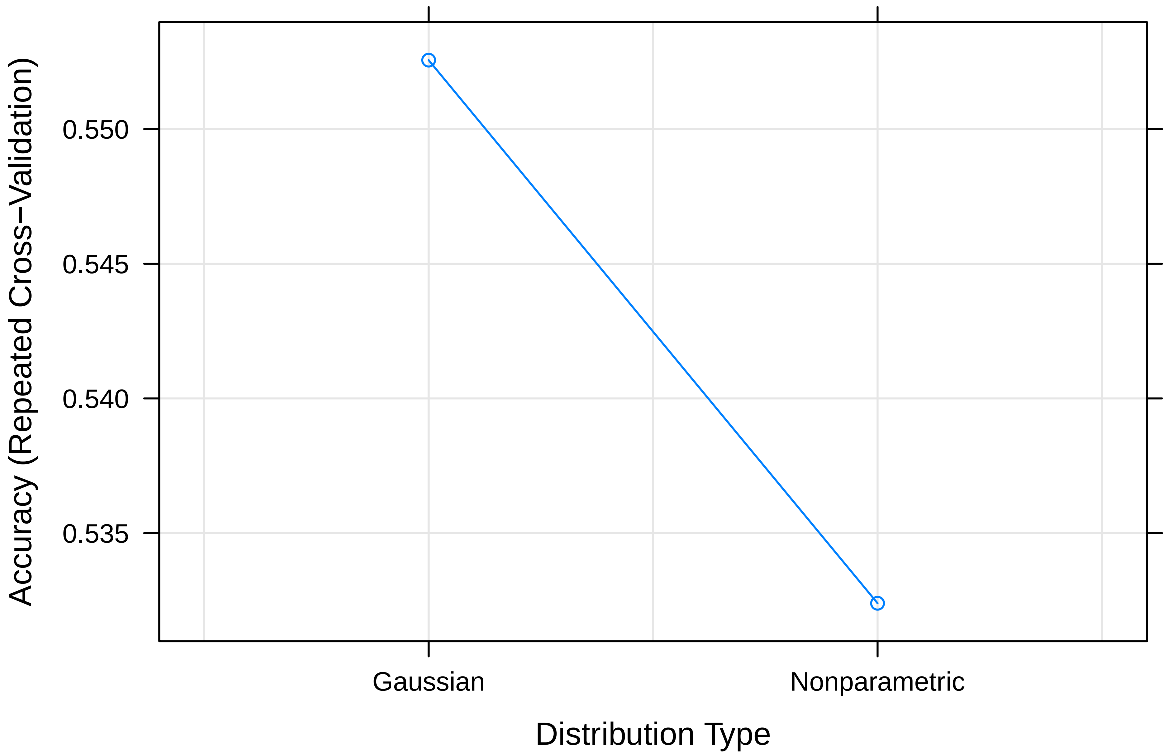
```
##
##  NO_PASS     PASS
## 54.54545 45.45455
```

7. PERFORMING THE TRAINING

```r
### PERFORMING the TRAINING

set.seed(400)
ctrl <- trainControl(method="repeatedcv", repeats = 10)

nbFit = train( RESULT~ .,
                 data = training,
                 method = "nb",
                 trControl = ctrl)

## The output of nbFit fit

nbFit
```

```
## Naive Bayes
##
## 268 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 241, 242, 240, 241, 242, 241, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.5525590  0.1417325
##    TRUE      0.5323962  0.1006722
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adjust
##  = 1.
```

```r
plot(nbFit)
```




```
png("the.results.nb.FIT.png")
plot(nbFit)
dev.off()
```

```
## pdf
##   2
```

8. MAKING THE PREDICTIONS

```
### Making the PREDICTIONS :

nbPredict <- predict(nbFit, newdata = testing)
```

9. THE CONFUSION MATRIX (caret package)

```
### COMPUTING the CONFUSION MATRIX :

confusionMatrix(nbPredict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      18    3
##    PASS         30   37
##
##                Accuracy : 0.625
##                  95% CI : (0.5153, 0.726)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.08137
##
##                   Kappa : 0.284
##
##  Mcnemar's Test P-Value : 6.011e-06
##
##             Sensitivity : 0.3750
##             Specificity : 0.9250
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.5522
##              Prevalence : 0.5455
##          Detection Rate : 0.2045
##    Detection Prevalence : 0.2386
##       Balanced Accuracy : 0.6500
##
##        'Positive' Class : NO_PASS
##
```

```r
mean(nbPredict == testing$RESULT)
```

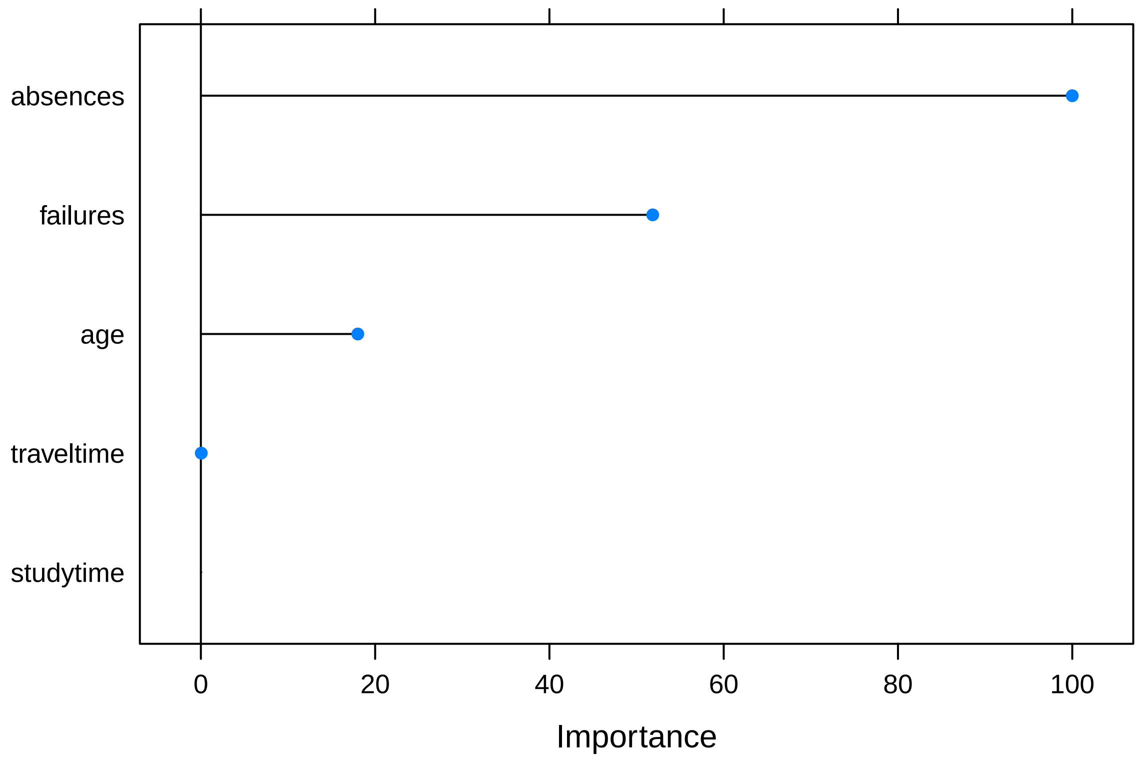
```
## [1] 0.625
```

```r
dim(student3)
```

```
## [1] 356   6
```

```r
# We implement the NB model also in other packages ("klaR", "e1071").
# here only another version of the R code

# library(e1071)

# x = training[, -6]
# y = training$RESULT

# model = train(x, y, 'nb', trControl=trainControl(method='cv',number=10))

# predict(model$finalModel,x)
# head(predict(model$finalModel,x)$class)

# table(predict(model$finalModel,x)$class,y)

# Predict <- predict(model, newdata = testing )

# We draw a plot that shows how each predictor variable is independently
# responsible for predicting the outcome.
```

```r
# to display Variable Performance
# X <- varImp(model)
# plot(X)

# the confusion matrix to see accuracy value and other parameter values
# confusionMatrix(Predict, testing$RESULT)

X <- varImp(nbFit)
plot(X)
```

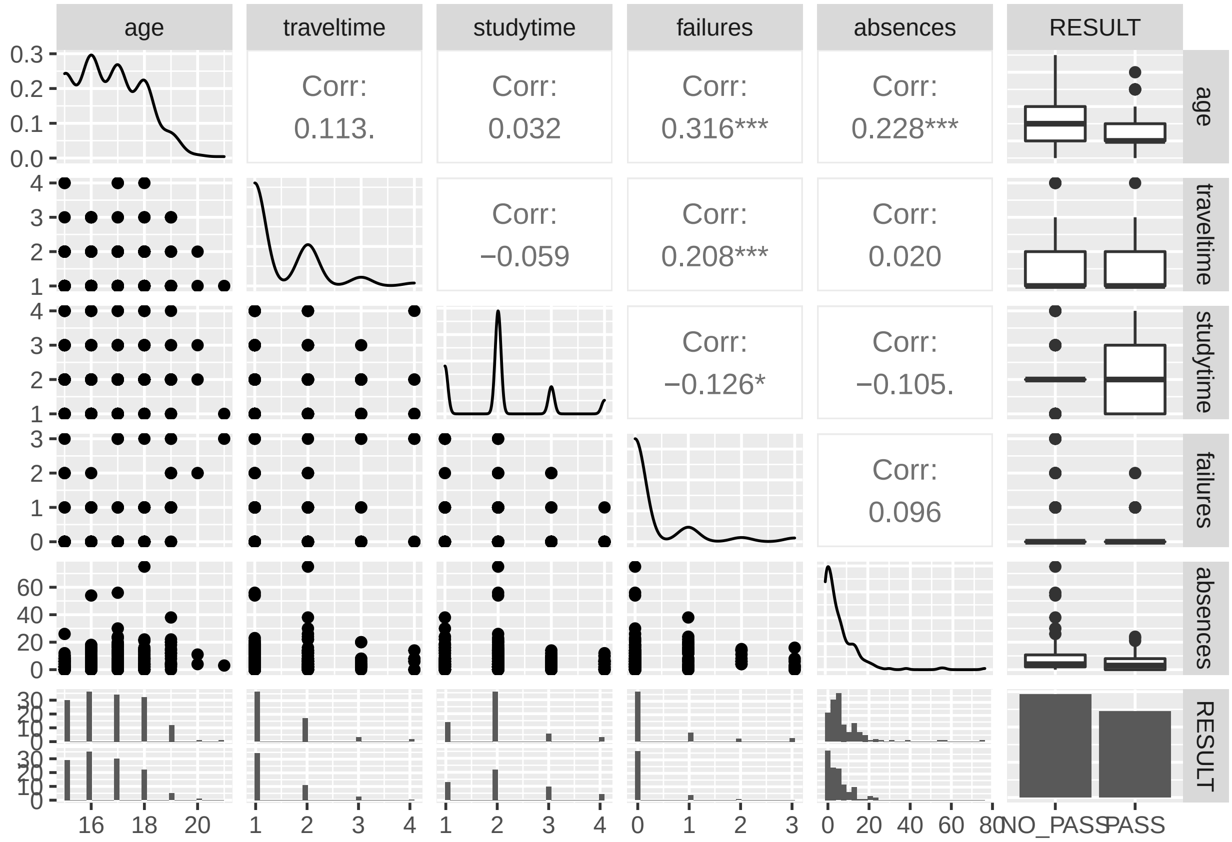


10. THE RESULTS (klaR package)

```r
### looking at the CORRELATIONS between the FEATURES
library(GGally)

ggpairs(training)
```

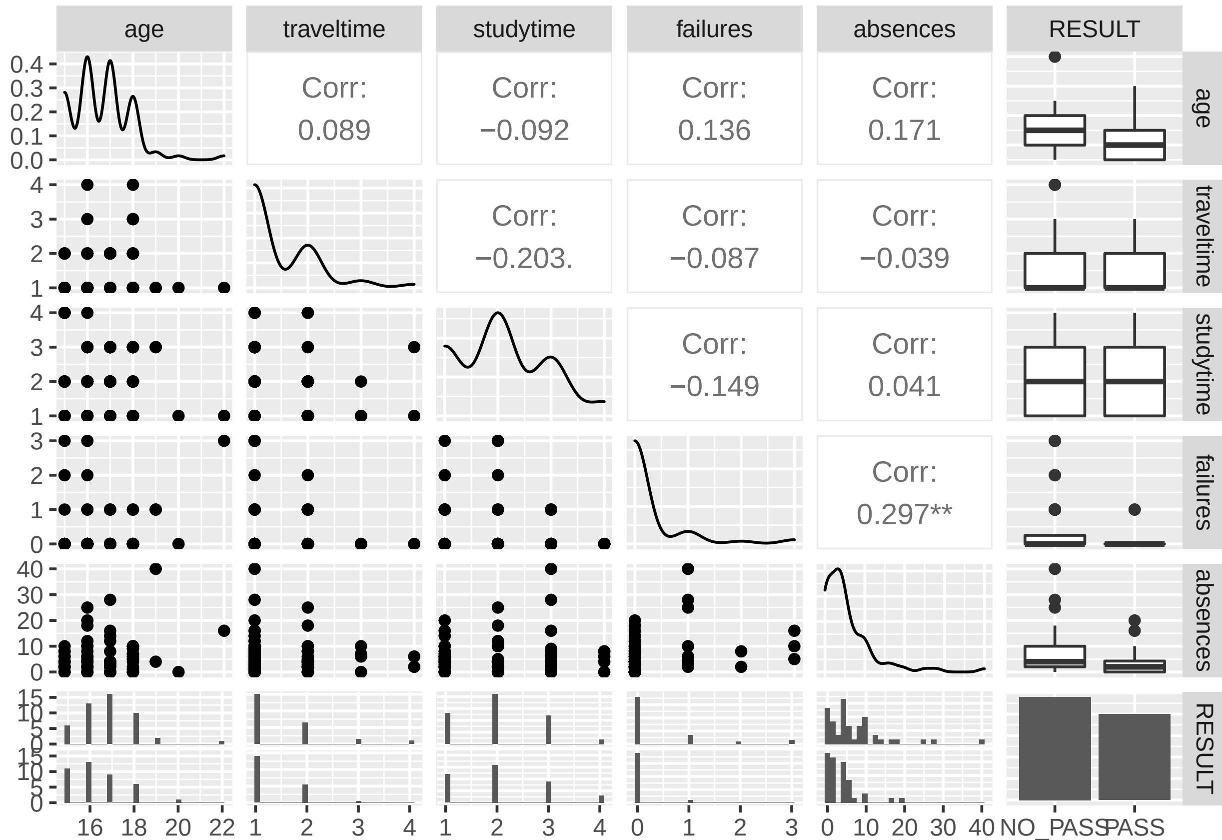
```
##  plot: [1,1] [>-----------------------------------------------]  3% est: 0s  plot: [1,2] [==>-----
##  plot: [6,2] [==========================================>------] 89% est: 0s `stat_bin()` using `bi
##  plot: [6,3] [============================================>----] 92% est: 0s `stat_bin()` using `bi
##  plot: [6,4] [=============================================>---] 94% est: 0s `stat_bin()` using `bi
```

```
## plot: [6,5] [================================================>-] 97% est: 0s `stat_bin()` using `bi
## plot: [6,6] [==================================================]100% est: 0s
```




```r
ggpairs(testing)
```

```
## plot: [1,1] [>-------------------------------------------------]  3% est: 0s  plot: [1,2] [==>-----
## plot: [6,2] [=========================================>------] 89% est: 0s `stat_bin()` using `bi
## plot: [6,3] [=============================================>----] 92% est: 0s `stat_bin()` using `bi
## plot: [6,4] [==============================================>---] 94% est: 0s `stat_bin()` using `bi
## plot: [6,5] [================================================>-] 97% est: 0s `stat_bin()` using `bi
## plot: [6,6] [==================================================]100% est: 0s
```




```r
### using KLAR PACKAGE
library(klaR)

model = NaiveBayes( RESULT~ ., data = training)

predictions <- model %>% predict(testing)

# The ACCURACY
mean(predictions$class == testing$RESULT)
```

```
## [1] 0.625
```

As we can see, shall we set up the ML approach with NB, the accuracies of our models are almost equal and not too great.

5. TRAINING AND TEST SETS

```r
## CHOOSING the TRAINING and TESTING SETS

indxTrain <- createDataPartition(student3$RESULT,
                                 p = .75,
                                 list = FALSE)
```

```r
training <- student3[indxTrain,]
# training

testing <- student3[-indxTrain,]
# testing

dim(student3)
```

```
## [1] 356   6
```

```r
dim(training)
```

```
## [1] 268   6
```

```r
dim(testing)
```

```
## [1] 88  6
```

6. PRE-PROCESSING THE DATA

```r
## PRE-PROCESSING the DATA

trainX        <- training[, names(training) != "RESULT"]

preProcValues <- preProcess(x = trainX, method = c("center", "scale"))

preProcValues
```

```
## Created from 268 samples and 5 variables
##
## Pre-processing:
##   - centered (5)
##   - ignored (0)
##   - scaled (5)
```

```r
names(trainX)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
```

```r
dim(trainX)
```

```
## [1] 268   5
```

```r
names(training)
```

```
## [1] "age"        "traveltime" "studytime"  "failures"   "absences"
## [6] "RESULT"
```

7. PERFORMING THE TRAINING

```r
## PERFORMING the TRAINING

set.seed(400)
```

```r
ctrl <- trainControl(method="repeatedcv",repeats = 3)

knnFit <- train( RESULT~ .,
                 data = training,
                 method = "knn",
                 trControl = ctrl,
                 preProcess = c("center","scale"), tuneLength = 20)

## The output of kNN fit

knnFit
```

```
## k-Nearest Neighbors
##
## 268 samples
##   5 predictor
##   2 classes: 'NO_PASS', 'PASS'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 241, 242, 240, 241, 242, 241, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.6069326  0.2081785
##    7  0.6241249  0.2395012
##    9  0.6388957  0.2700239
##   11  0.6352869  0.2622706
##   13  0.6442579  0.2834988
##   15  0.6390788  0.2733702
##   17  0.6328958  0.2590676
##   19  0.6127086  0.2187786
##   21  0.6342118  0.2607925
##   23  0.6377866  0.2668033
##   25  0.6440612  0.2774770
##   27  0.6404558  0.2682230
##   29  0.6292939  0.2456728
##   31  0.6230634  0.2317282
##   33  0.6241114  0.2331955
##   35  0.6214998  0.2279499
##   37  0.6065832  0.1947966
##   39  0.6092830  0.2019650
##   41  0.6192104  0.2214491
##   43  0.6168363  0.2178759
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```r
png("the.results.knn.FIT.png")
plot(knnFit)
dev.off()
```

```
## pdf
##   2
```

8. MAKING THE PREDICTIONS

```r
## Making the PREDICTIONS :

knnPredict <- predict(knnFit, newdata = testing)
```

9. THE CONFUSION MATRIX

```r
## COMPUTING the CONFUSION MATRIX :

confusionMatrix(knnPredict, testing$RESULT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO_PASS PASS
##    NO_PASS      30   21
##    PASS         18   19
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.4587
##
##                   Kappa : 0.1006
##
##  Mcnemar's Test P-Value : 0.7488
##
##             Sensitivity : 0.6250
##             Specificity : 0.4750
##          Pos Pred Value : 0.5882
##          Neg Pred Value : 0.5135
##              Prevalence : 0.5455
##          Detection Rate : 0.3409
##    Detection Prevalence : 0.5795
##       Balanced Accuracy : 0.5500
##
##        'Positive' Class : NO_PASS
##
```

```r
mean(knnPredict == testing$RESULT)
```

```
## [1] 0.5568182
```

```r
dim(student3)
```

```
## [1] 356   6
```

We may aim to optimize the model by feature selection or by including new features from the data that is available.

IN THE NEXT SECTIONS to ADD the ENSEMBLE METHODS and the SUPER LEARNERS ..