Data Mining and Big Data Science Summer Project - 2016

Predicting Final Grade : In Portuguese Schools.

The educational level of the Portuguese population has improved in the last decades, however the statistics keep Portugal at Europe's tail end due to its high student failure rates. In particular, lack of success in the core classes of Mathematics and the Portuguese language is extremely serious. The present work intends to approach student achievement – in Mathematics only, in secondary education using DM techniques. A model is thought to predict the scores in Mathematics for the high school students of Portugal. For the analysis we are presented data with 33 characteristics and 395 observations.

Data preprocessing and missing values

We drop two variables namely G1 and G2, first and second period grade as they have strong correlation with G3 and is also part of research question. The final grade G3 is scaled between 0 to 20 and we are left with 30 predictors. No missing values were detected. On the continous variables outliers were detected using MAD[1]. For the 'age' variable above 20 years it turned out to be outlier values and on detecting 2 such values we removed those rows. We also, observe large number of outliers in 'abscences' and 'G3', but are retained. In G3 we observe large number of 0s in the score.

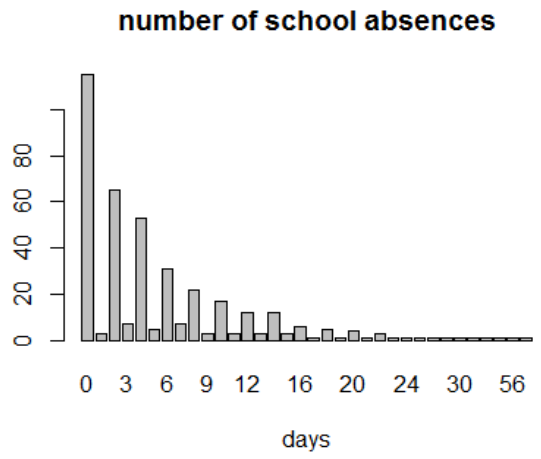

Figure 1

Models were built on 85% (to provide maximum power to build the models) of the randomly selected training data and validity was assessed on predictive performance on the remaining 15% test data. When deemed necessary, 10 fold cross validation was used for parameter tuning.

Data Exploration

In Fig1 the plot of "number of absences" is rather peculiar. The absences on 'Odd number of days' is drastically low, which raises doubts on the data collection and is highlighted as a risk. On exploring, G3 is not normally distributed, which

is confirmed by QQ plots and also Shapiro test. For further analysis the continuous variables are centered and scaled.

As we have 'mixed' type of data we used factor analysis of mixed data (FAMD) for exploration. For correlation involving both categorical and continuous variables, R^2 can be used as a crude indicator[3]

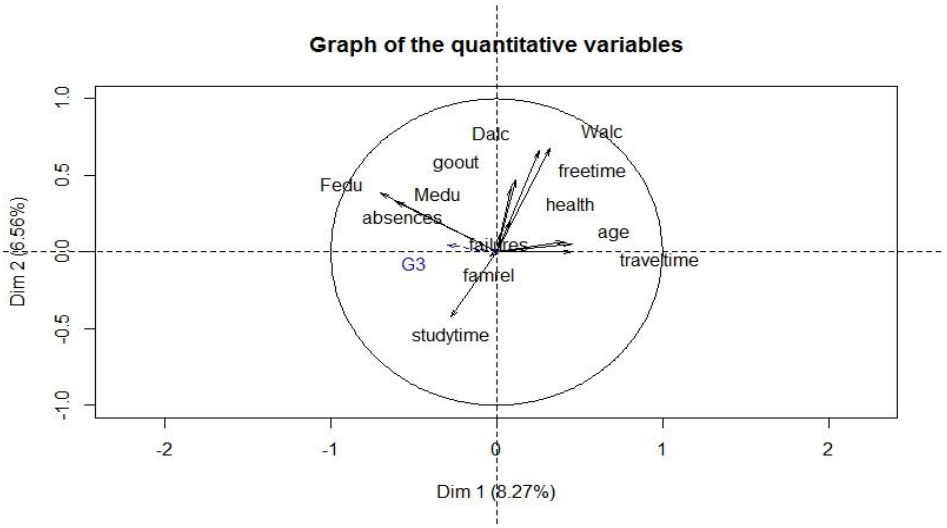

Figure 2: PCA of continious variables

Principle component analysis (PCA) was performed on the continuous predictors. No single predictor came out as being more important to explain variability in the predictor space than any other predictors. We see absences, parent's education, and health to be the slightly important contributors for the 'final score', as presented in fig2. Though this does not capture the categorical values, we do not take it at face value.

The table plot is a powerful visualization method to explore and analyze large multivariate datasets. In figure 3, we look at the table plot[4] and visualize the correlation of G3 with absences, failures, and mother's job quite clearly. Not all variables are considered in the figure though. For large number of rows, table plot can be generated along with parallel processing to overcome computing constraints.

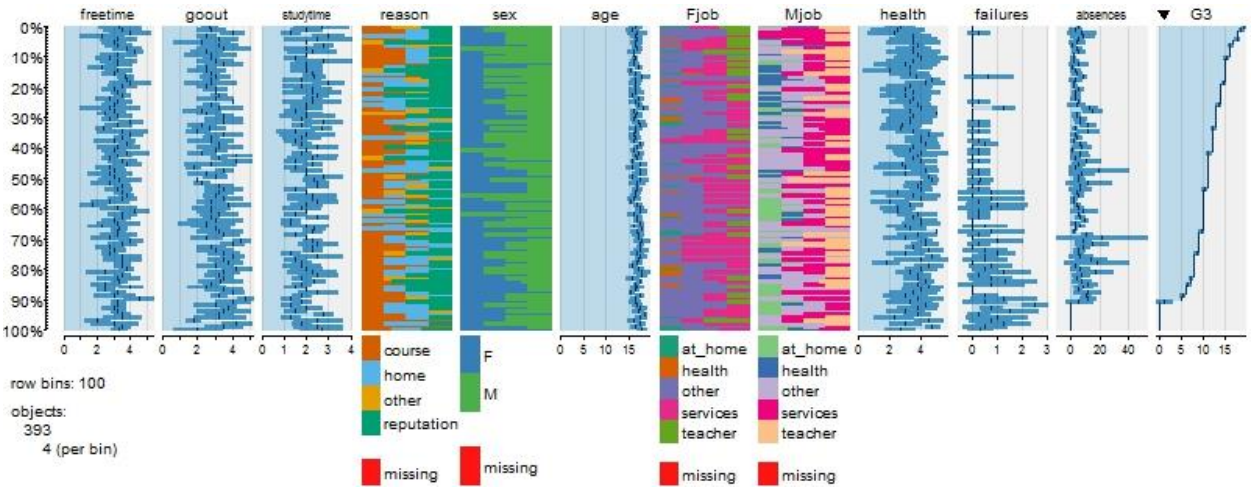

Figure 3 : Table plot. Mjob(teacher) vs G3 ; failures vs G3 ; absences vs G3 all these relations can be seen.

Performance of different prediction methods

Different predictive models were tried out. The general approach was to build the model on the training dataset and to test it on the test dataset. Not just the *Mean Squared Error (MSE)* but also *average of the absolute value of the residual (MAPE)* was observed. As we had many predictors regularized regression was also tried. For regressing with L1 norm and L2 norm for Lasso and Ridge regression respectively, we used 10 fold cross validation to get optimum parameter values, $\lambda = 1.1 \; and \; \gamma = 0.23$ respectively. We also tried the Elastic net. However, the performance of them was not exceptionally good with MSE = 21. 5 for Lasso and MSE=19.4 for Ridge regression.

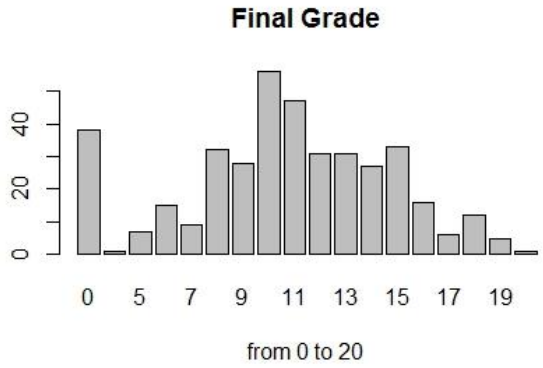

Figure 4 :histogram of mathematics grade

On looking at the distribution of the Mathematics grade as in fig 3, we also tried the Zero-inflated Poisson model (ZIPM), and Zero-inflated negative binomial model. The histogram of G3 seemed like a zero inflated Poisson distribution with a large lambda close to 10. On selecting the most significant predictors they give a decent prediction. Hence a model explaining inflated zeros is also of importance. ZIPM gives a MSE of 16.59 which is very close to our best model. Moreover, the factors effecting the zero inflation can be of greater interest.

Our next exploration was in the aggregated decision trees, these models do not require a specification of the predictor-outcome relationship and are therefore not burdened by linearity. The predictive power of three such models were tested, namely bagging, boosting and random forest.andom forests had larger predictive power than the ones mentioned before. A RF model with 400 trees gave us an MSE=15.8, and this by faris our best model. Inspection of the importance of the predictos in the model suggested, again, that number of school absences, number of past class failures, mother's job, current health status, going out with friends, father's job were most important for a student to score well in mathematics.

Conclusion

Extensive testing of several models showed, based on prediction error (MSE), the aggregated decision tress had superior predictive performance over other regression models. This advancement suggests a proper nonlinear relationships and possible interactions between predictors might be of importance in modelling mathematics scores among the school students of Portugal.

References

[1] . Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median .

[2] . https://cran.r-project.org/web/packages/tabplot/vignettes/tabplot-vignette.html

[3].http://stats.stackexchange.com/questions/119835/correlation-between-a-nominal-iv-and-a-continuous-dv-variable

[4]. https://cran.r-project.org/web/packages/tabplot/vignettes/tabplot-vignette.html

[5]. http://www.h2o.ai/

[6]. http://factominer.free.fr/

Big Data Analysis

The big V's of the challenge of big data analysis are: Volume, Velocity, Variety, Veracity, Visibility, Visually, Viability, Value and hence considering what the bottle neck is it has been handled.

For instance : For Big Data visualizations we have many techniques such as using maps in Choropleth map in combination with heat maps to high light the geographical importance, or radial node link diagrams to know the patterns in links etc. Or even tableplot.

To deal with data mining algorithms on big data, parallel processing is greatly useful. There are different technologies to deal with it like Hadoop or the latest Spark. 'Spark' is an open source cluster computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

Implementation: H20.io

In our set up, we try H2O.io[5], it is a Java Virtual Machine that is optimized for doing "in memory" processing of distributed, parallel machine learning algorithms on clusters. Because of H2O's rapid in-memory distributed parallel processing scaling is not an issue. H2O leverages the Open Source products like Apache Hadoop and Spark internally.

We created a program with a training and a test dataset. We used two methods: the 'classical' Random Forest, and the H2O distributed Random Forest algorithm.

Model name	**MSPE**
Random Forest without h2o	16.14
Random Forest with h2o	16.47

We obtain similar results with both, with a difference of less than 1%. As there is the element of randomness we do not get the same results all the time. Moreover the dataset that we are working with isn't really that big. This small code is shown just for illustration purposes, its power can be harnessed by using it on cloud or in a network, with multiple cores and more optimized usage of memory. And using h2o setting these parameters is relatively easy.

One of the advantages of it is, it can be used from the browser also by visiting http://localhost:54321/ once h20 is initialized in R or from other platform such as python or JAVA.

Conclusion:

Distributed framework is one of the most sought after ways of building models today. There are many frameworks available with their pros and cons.

Appendix :

	set	method	MSPE	R2	MAPE
16	test	tree 1	16.10490928	34.00528654	3.064524923
24	test	Random Forest.400	16.14577856	33.83781235	2.927355226
111	test	Random Forest- h2o.io	16.47634228	32.483228	2.967984435
28	test	ZeroInflatedPOisson.optimized	16.59795904	31.9848667	3.169247671
34	test	student.hurdle	16.8053226	31.1351321	3.214367888
32	test	ZeroInflatedNegBinom.optimized	16.8078	31.12498023	3.214606024
22	test	boosting	17.15925661	29.68478097	3.13537301
20	test	bagging	17.38785983	28.74801047	3.067381872
2	test	lm	18.51642088	24.12339186	3.253693204
14	test	PLS 4	18.71038052	23.32858385	3.268842586
12	test	PCR 16	19.43852131	20.34480777	3.372254624
6	test	ridge	19.47109542	20.21132558	3.333966041
4	test	lm2	20.55255779	15.77970799	3.531312599
18	test	tree 2	21.17979108	13.20943077	3.51659794
10	test	Elastic	21.4954123	11.91607783	3.517662436
8	test	Lasso	21.54813641	11.70002492	3.532172445
26	test	ZeroInflatedPOisson.overfit	26.0991957	-6.949310426	3.39378201
30	test	ZeroInflatedNegBinom	26.10293208	-6.964621372	3.394049518
110	train	Random Forest- h2o.io	2.16741279	89.37340373	1.124181743
23	train	Random Forest.400	3.152109387	84.5455402	1.340367408
21	train	boosting	6.746537978	66.922436	2.023165836
19	train	bagging	9.215311236	54.81830115	2.322803183
29	train	ZeroInflatedNegBinom	9.356007761	54.12848093	2.286185174
25	train	ZeroInflatedPOisson.overfit	9.356016179	54.12843966	2.286203684
15	train	tree 1	10.49776516	48.53056484	2.505930201
27	train	ZeroInflatedPOisson.optimized	12.92936711	36.60867703	2.753938363

33	train	student.hurdle	13.94102067	31.64864634	2.891068162
31	train	ZeroInflatedNegBinom.optimized	13.94148063	31.64639119	2.891125792
1	train	lm	14.92485182	26.825026	2.934126761
17	train	tree 2	15.05342647	26.19463806	2.921954838
13	train	PLS 4	15.26438957	25.16030823	2.981605983
5	train	ridge	15.80188355	22.52503191	2.995908924
9	train	Elastic	16.31027811	20.03242702	3.011633204
7	train	Lasso	16.35990677	19.78910296	3.020230688
3	train	lm2	16.66777431	18.27966089	3.132912322
11	train	PCR 16	16.69998258	18.12174712	3.129383848

R Code :

```
################################################################################
# Author : Omkar Kulkarni
# Topic  : Data Mining and Big Data Science:Examination Project – Resit
# Date   : 31/08/2016
################################################################################


##### Import Data #####
rm(list=ls()) # remove all variables
cat("\014") # clear console
#setwd("C:/Users/Omkar/OneDrive/Big Data and Data Mining/Summer_Project")

student <- read.csv("student-mat.csv", sep = ";", header = TRUE,
                    na.strings = "NA",fill = TRUE, stringsAsFactors = TRUE )

#### Flags ####
showInfo = FALSE # do not show intermediate results
showPlots = TRUE # show plots
saveMainResults = TRUE # save some results to a file
TRAIN.TEST.RATIO = 0.85

# Check for missing data
sapply(student, function(x) sum(is.na(x)))
# No missing data
# visvulize missing data
library(Amelia)
missmap(student, main = "Missing values vs observed")
# Strange, no Missing values !!
#fix(student)
```

```
#Manually skim through the data to look for unexpected values such as '?', 'NAA' etc

## As we already know, G1 and G2 are not to be part of the analysis.
## hence we remove them and make a new dataset student.n
student.n <- student[,]
drops <- c("G1","G2")
student.n <- student[ , !(names(student) %in% drops)]
rm(drops)
##### End Of Import Data #####



##### Pre define functions #####
# Detecting outliers: Do not use standard deviation around the mean,
# use absolute deviation around the median

DoubleMAD <- function(x, zero.mad.action="warn"){
  # The zero.mad.action determines the action in the event of an MAD of zero.
  # Possible values: "stop", "warn", "na" and "warn and na".
  x       <- x[!is.na(x)]
  m       <- median(x)
  abs.dev <- abs(x - m)
  left.mad  <- median(abs.dev[x<=m])
  right.mad <- median(abs.dev[x>=m])
  if (left.mad == 0 || right.mad == 0){
    if (zero.mad.action == "stop") stop("MAD is 0")
    if (zero.mad.action %in% c("warn", "warn and na")) warning("MAD is 0")
    if (zero.mad.action %in% c(  "na", "warn and na")){
      if (left.mad  == 0) left.mad  <- NA
      if (right.mad == 0) right.mad <- NA
    }
  }
  return(c(left.mad, right.mad))
}

DoubleMADsFromMedian <- function(x, zero.mad.action="warn"){
  # The zero.mad.action determines the action in the event of an MAD of zero.
  # Possible values: "stop", "warn", "na" and "warn and na".
  two.sided.mad <- DoubleMAD(x, zero.mad.action)
  m <- median(x, na.rm=TRUE)
  x.mad <- rep(two.sided.mad[1], length(x))
  x.mad[x > m] <- two.sided.mad[2]
  mad.distance <- abs(x - m) / x.mad
  mad.distance[x==m] <- 0
  return(mad.distance)
}

# DEFINE FUNCTIONS to calculate prediction statistics

CalculatePredictionStatistics2 <- function(set, method, response.values, predictions) {
```

```
  MSPE = mean((response.values - predictions)^2)
  SSE = sum((response.values - predictions)^2)
  SSTO = sum((response.values - mean(response.values))^2)
  R2 = (1 - SSE/SSTO) * 100
  MAPE = mean(abs(response.values - predictions)) # Mean absolute prediction error

  if (!exists("PredictionStatistics")) PredictionStatistics = NULL

  # remove statistics with the same method, but old data
  PredictionStatistics <<- PredictionStatistics[!(PredictionStatistics$set == set &
                                    PredictionStatistics$method == method),]

  NewRow = data.frame(set=set, method=method, MSPE=MSPE, R2=R2, MAPE=MAPE, stringsAsFactors =
FALSE)
  PredictionStatistics <<- rbind(PredictionStatistics, NewRow)

}


# this function receives predictions as input and calculates some statistics like
# MSPE, R2
CalculatePredictionStatistics <- function(method, predictions.train, predictions.test) {

  response.values.train = student.train$G3
  response.values.test = student.test$G3

  CalculatePredictionStatistics2("train", method, response.values.train, predictions.train)
  CalculatePredictionStatistics2("test",  method, response.values.test,  predictions.test)

  predictions.train <<- NULL; predictions.test <<- NULL
  rm(predictions.train); rm(predictions.test)
  return(PredictionStatistics)
}

##### End Of Pre define functions #####


##### Univariate Data Exploration #####
#### Data Exploration
plot(student.n$school, col = "blue", ylim=c(0,500), main="Schools")
plot(student.n$sex, col = "blue", ylim=c(0,500), main="Gender")
hist(student$age, col = "grey", main = "Age distribution", breaks = 8)
# The distribution of age is rather a bit odd,
# we see some large values for age, as students more than 19 years old is
# Odd we check for outliers in Age. We use MAD to check for outliers
table(student.n$age)
barplot(table(student.n$age), main = "Age distribution", xlab = "age",
      ylab = "frequency")

TEMP = student.n$G3[student.n$age>18]
```

```
barplot(table(TEMP), ylim = c(0,60), main = "G3 of students with Age>18")
rm(TEMP)
# we see nothing surprising with AGE>18 vs G3

## MAD for Age
#OUTLIERS IN AGE
print(student$age[DoubleMADsFromMedian(student$age) > 3])
plot(student$age,student$G3, main = "Age Vs G3")
## Observe the G3 value for the age>20 namely for the outliers 21 and 22 years of age.

#DROP THE ROWS WITH OUTLIERS IN AGE
student.n <- student.n[!(student.n$age>20),]

boxplot(student$age, col = "grey", main = "Age distribution before removing outliers")
boxplot(student.n$age, col = "grey", main = "Age distribution after removing outliers")

plot(student.n$address, col = "blue", ylim=c(0,500), main="Address",
    xlab="'U' - urban or 'R' - rural")
plot(student.n$famsize, col = "blue", ylim=c(0,500), main="Family Size",
    xlab="(LE3 - less or equal to 3 or GT3 - greater than 3")
plot(student.n$Pstatus, col = "blue", ylim=c(0,500), main="Parent's cohabitation status",
    xlab="T' - living together or 'A' - apart")
hist(student.n$Medu, col = "grey", breaks = 6, main = "Mother's Education",
    xlab = "0 - none, 1 - primary education (4th grade), 2 - 5th to 9th
    grade, 3 - secondary education or 4 - higher education")
hist(student.n$Fedu, col = "grey", breaks = 6, main = "Father's Education",
    xlab = "0 - none, 1 - primary education (4th grade), 2 - 5th to 9th
    grade, 3 - secondary education or 4 - higher education")
barplot(table(student.n$Mjob), main="Mother's job", ylim = c(0,200))
barplot(table(student.n$Fjob), main="Father's job", ylim = c(0,200))
barplot(table(student.n$reason), main="Reason to choose school", ylim = c(0,180))
barplot(table(student.n$guardian), main="Guardian")
barplot(table(student.n$traveltime), main="Travel time to school",
      xlab = "1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1
      hour, or 4 - >1 hour")
barplot(table(student.n$studytime), main="Study Time",
      xlab = "1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10
      hours")

cor(student.n$studytime, student.n$G3)
## strangely, correlation of study time and G3 is quite low= 0.10

barplot(table(student.n$failures), main="Number of past class failures",
      xlab = "n if 1<= n <3, else 4")
cor(student.n$failures, student.n$G3)
## As expected, a negative correlation and relatively OK of -0.36

plot(student.n$schoolsup, col="blue", main="Extra educational support")
table(student.n$schoolsup, student.n$G3)
```

```
## now we see 8 Binary variables, YES and NO!! not plotting all of them.
barplot(table(student.n$famrel), main="quality of family relationships",
      xlab = "1 - very bad to 5 - excellent")

## distribution of final grade
hist(student.n$G3, main = "Distribution of Final Grades")
barplot(table(student.n$G3), main = "Final Grade", xlab = "from 0 to 20")
summary(student$G3)
sd(student$G3)
boxplot(student.n$G3)

#OUTLIERS IN G3
print(student$G3[DoubleMADsFromMedian(student$G3) > 3])

## Zeros are indeed part of dataset and they are almost 10% of them. cannot possibly remove them.

## test for normality ; which obviously does not look normal.
shapiro.test(student.n$G3)
qqnorm(student.n$G3)
qqline(student.n$G3)
## certainly, G3 is NOT "Normally distributed", even Pvalue = 0.001 for Shapiro test.
## in the histogram we see a large number of 0's.

t = student.n$G3[student.n$G3>0]
hist(t, main = "Non Zero scores")
shapiro.test(t)
qqnorm(t, main = "non zero scores")
qqline(t)
rm(t)
## even without the 0's it is not normal.

##absences
hist(student.n$absences)
table(student.n$absences)
barplot(table(student.n$absences), main = "number of school absences", xlab = "days")
cor(student.n$absences, student.n$G3)

# The plot of "number of absences" is rather peculiar. The absences on 'Odd number of days'
# is very low, it it does not follow the over all exponential pattern. The plot induces doubts
# if the data collection for number of absences was correct
# it has weak correlation with G3.

#OUTLIERS IN absences
print(student$absences[DoubleMADsFromMedian(student$absences) > 3])
#outliers for exponential distributions make sense???
#no action to be taken on these outliers.

#Scaled age and absences.
student.s <- student.n
student.s$age <- (student.s$age - mean(student.s$age)) / sd(student.s$age)
```

```
student.s$absences <- (student.s$absences - mean(student.s$absences)) / sd(student.s$absences)

##### End of Univariate Data Exploration #####


##### Table plots #####
## check later,
# explore entire data set as whole.
require(tabplot)
tableplot(student,  sortCol = "G3")
# as expected we see strong corelation between G1,G2 and G3

#now plotting without G1 and G2 we see :
tableplot(student.n,  sortCol = "G3")
## Plot for the most significant variables
tableplot(student.n, select = c(freetime, goout,studytime,reason
                        ,sex,age,Fjob,Mjob,health,failures,absences,31) , sortCol = "G3")
##### End of Table plots #####



##### Factor Analysis #####

# The term mixed refers to the simultaneous presence, as active elements,
# of quantitative and qualitative variables. Roughly, we can say that FAMD works
# as a principal components analysis (PCA) for quantitative variables and as a
# multiple correspondence analysis (MCA) for qualitative variables.
library(FactoMineR)

#?FAMD()
Factor.analysis = FAMD(student.n, graph = TRUE, sup.var =31)
res.pca = PCA(student.n[,c(3,31,7,8,13,14,15,24,25,26,27,28,29,30)], graph = TRUE)
plot(Factor.analysis$quanti.var)
plot(Factor.analysis,habillage = 9)  # 9 is for Mjob
boxplot(student.n$G3~student.n$Mjob)

##### End Of Factor Analysis #####


####  scale and Split the data####
## 90% of the sample size

smp_size <- floor(TRAIN.TEST.RATIO * nrow(student.s))
## set the seed to make your partition reproductible
set.seed(123)
train_ind <- sample(seq_len(nrow(student.s)), size = smp_size)
student.train <- student.s[train_ind, ]
student.test <- student.s[-train_ind, ]

rm(smp_size, TRAIN.TEST.RATIO, train_ind)
```

```
rm(student,student.n,student.s)
####  End of scale and Split the data####

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

############################ CREATE PREDICTION MODELS   ##########################


#### OLS ####
student.model.lm = lm(G3~., data=student.train)
if (showInfo) summary(student.model.lm)
# calculate predictions
predictions.train = predict(student.model.lm, newdata=student.train)
predictions.test =  predict(student.model.lm, newdata=student.test)
CalculatePredictionStatistics("lm", predictions.train, predictions.test)
# this model overfits the training data

## tests
# Assessing Outliers
outlierTest(student.model.lm) # Bonferonni p-value for most extreme obs
qqPlot(student.model.lm, main="QQ Plot") #qq plot for studentized resid

# Evaluate homoscedasticity
# non-constant error variance test
ncvTest(student.model.lm)
# plot studentized residuals vs. fitted values
spreadLevelPlot(student.model.lm)


## clearly the model assumptions are violated.

keepAllVariables = TRUE
# OLS - selection of most significant variables
if (keepAllVariables)
{
  student.model.lm = lm(G3~sex+failures+famsup+activities+higher+
                        romantic+goout+Dalc+absences, data=student.train)
  # calculate predictions
  predictions.train = predict(student.model.lm, newdata=student.train)
  predictions.test = predict(student.model.lm, newdata=student.test)
  CalculatePredictionStatistics("lm2", predictions.train, predictions.test)
}

# clean up variables
rm(student.model.lm)


#####Ridge ####
```

```
#glmnet standardizes the y variable and uses the mean squared errors
#instead of sum of squared errors

library(glmnet)
x = model.matrix(G3~.,student.train)[,-31]
x.test = model.matrix(G3~.,student.test)[,-31]
# mode.matrix it also transforms any qualitative variables into dummy variables
# apart from creating X matrix.
y = student.train$G3

mcv_10fold <- cv.glmnet(x,y,alpha = 0 , nfolds = 10)
plot(mcv_10fold, xlab = "gamma", main="Ridge")
rm(mcv_10fold)

iteration=50
lambdas = NULL
for (i in 1:iteration)
{
  fit <- cv.glmnet(x,y,alpha = 0 , nfolds = 10)
  errors = data.frame(fit$lambda,fit$cvm, fit$lambda.1se)
  lambdas <- rbind(lambdas,errors)
}

# take mean cvm for each lambda
lambdas <- aggregate(lambdas[, 2:3], list(lambdas$fit.lambda), mean)

# select the best one
bestindex = which(lambdas[2]==min(lambdas[2]))
bestlambda = lambdas[bestindex,1]
# and now run glmnet once more with bestlambda


ridge.final.model <- glmnet(x,y,lambda=bestlambda, alpha = 0)

# calculate predictions
predictions.train = predict(ridge.final.model, newx = x)
predictions.test = predict(ridge.final.model, newx = x.test)
CalculatePredictionStatistics("ridge", predictions.train, predictions.test)

rm(iteration,lambdas,bestlambda,bestindex,predictions.test,predictions.train)
rm(ridge.final.model)




##### Lasso  ####

Lasso_10fold <- cv.glmnet(x,y,alpha = 1 , nfolds = 10)
plot(Lasso_10fold, xlab = "gamma", main="Lasso")
rm(mcv_10fold)
```

```
iteration=50
lambdas = NULL
for (i in 1:iteration)
{
  fit <- cv.glmnet(x,y,alpha = 1 , nfolds = 10)
  errors = data.frame(fit$lambda,fit$cvm, fit$lambda.1se)
  lambdas <- rbind(lambdas,errors)
}

# take mean cvm for each lambda
lambdas <- aggregate(lambdas[, 2:3], list(lambdas$fit.lambda), mean)

# select the best one
bestindex = which(lambdas[2]==min(lambdas[2]))
bestlambda = lambdas[bestindex,1]
# and now run glmnet once more with bestlambda


lasso.final.model <- glmnet(x,y,lambda=bestlambda, alpha = 1)

# calculate predictions
predictions.train = predict(lasso.final.model, newx = x)
predictions.test = predict(lasso.final.model, newx = x.test)
CalculatePredictionStatistics("Lasso", predictions.train, predictions.test)

rm(iteration,lambdas,bestlambda,bestindex,predictions.test,predictions.train)
rm(lasso.final.model)




##### Elastic Net  ####

Elastic_10fold <- cv.glmnet(x,y,alpha = 0.5 , nfolds = 10)
plot(Elastic_10fold, xlab = "gamma", main="Elastic")
rm(Elastic_10fold)

iteration=50
lambdas = NULL
for (i in 1:iteration)
{
  fit <- cv.glmnet(x,y,alpha = 0.5 , nfolds = 10)
  errors = data.frame(fit$lambda,fit$cvm, fit$lambda.1se)
  lambdas <- rbind(lambdas,errors)
}

# take mean cvm for each lambda
lambdas <- aggregate(lambdas[, 2:3], list(lambdas$fit.lambda), mean)
```

```
# select the best one
bestindex = which(lambdas[2]==min(lambdas[2]))
bestlambda = lambdas[bestindex,1]
# and now run glmnet once more with bestlambda


elastic.final.model <- glmnet(x,y,lambda=bestlambda, alpha = 0.5)

# calculate predictions
predictions.train = predict(elastic.final.model, newx = x)
predictions.test = predict(elastic.final.model, newx = x.test)
CalculatePredictionStatistics("Elastic", predictions.train, predictions.test)

# clean up variables
rm(iteration,lambdas,bestlambda,bestindex,predictions.test,predictions.train)
rm(elastic.final.model)
rm(x,x.test)




##### PCR ####
library(pls)
#help(pcr)
set.seed(15)
student.model.pcr = pcr(G3~., data=student.train, validation = "CV",
                        segments=10, segments.type="consecutive",scale=TRUE)
if (showInfo) summary(student.model.pcr)
if (showPlots) validationplot(student.model.pcr, val.type="MSEP")
# Calculate predictions for test set
ncomp = 16
predictions.train = predict(student.model.pcr, newdata=student.train, ncomp=ncomp)
predictions.test  = predict(student.model.pcr, newdata=student.test, ncomp=ncomp)
CalculatePredictionStatistics(paste("PCR", ncomp), predictions.train, predictions.test)

# clean up variables no longer needed
rm(student.model.pcr,ncomp)




##### PLS  #####
#library(pls)
#help(plsr)
set.seed(13)
student.model.pls = plsr(G3~., data=student.train, validation = "CV",
                         segments=10, segments.type="consecutive")
if (showInfo) summary(student.model.pls)
if (showPlots) plot(student.model.pls,plottype="validation")
ncomp = 4
```

```
predictions.train = predict(student.model.pls, newdata=student.train, ncomp=ncomp)
predictions.test  = predict(student.model.pls, newdata=student.test, ncomp=ncomp)
CalculatePredictionStatistics(paste("PLS", ncomp), predictions.train, predictions.test)

# clean up variables no longer needed
rm(student.model.pls,ncomp)



###### DECISION TREES #####

##### Regression tree####
library(rpart)
#help(package=rpart)
#help(rpart)
#help(summary.rpart)
#help(rpart.control)
student.model.tree1 = rpart(G3~., data=student.train, control=rpart.control(minsplit=10))

par(mar=c(0.5, 0.5, 0.5, 0.5))
if (showInfo) summary(student.model.tree1)
if (showPlots) plot(student.model.tree1)
if (showPlots) text(student.model.tree1)
if (showPlots) plotcp(student.model.tree1, upper=c("splits"))
predictions.train = as.vector(predict(student.model.tree1, newdata=student.train, type="vector"))
predictions.test  = as.vector(predict(student.model.tree1, newdata=student.test,  type="vector"))
CalculatePredictionStatistics("tree 1", predictions.train, predictions.test)

#install.packages("rattle")
#install.packages("rpart.plot")
library(rattle)
# rattle()
library(rpart.plot)
library(RColorBrewer)
fancyRpartPlot(student.model.tree1)

printcp(student.model.tree1)

cp.optimal =  student.model.tree1$cptable[which.min(student.model.tree1$cptable[,"xerror"]),"CP"]

##### prune the tree 1 CD above minimal ####
#help(prune.rpart)
student.model.tree2 = prune(student.model.tree1, cp=cp.optimal)
if (showPlots) plot(student.model.tree2)
if (showPlots) text(student.model.tree2)
if (showPlots) plotcp(student.model.tree2)
if (showInfo) summary(student.model.tree2)
predictions.train = as.vector(predict(student.model.tree2, newdata=student.train, type="vector"))
predictions.test  = as.vector(predict(student.model.tree2, newdata=student.test,  type="vector"))
CalculatePredictionStatistics("tree 2", predictions.train, predictions.test)
```

```
fancyRpartPlot(student.model.tree2, uniform=TRUE, main="Pruned Classification Tree")


# clean up variables no longer needed
rm(student.model.tree1); rm(student.model.tree2); rm(cp.optimal)



##### Bagging 700 ####
library(ipred)
#help(bagging)
nbagg =  700
student.model.bagging = bagging(G3~., data=student.train, coob=TRUE, nbagg=nbagg)
if (showInfo)  student.model.bagging
#summary(student.model.bagging)
predictions.train = predict(student.model.bagging, newdata=student.train)
predictions.test  = predict(student.model.bagging, newdata=student.test)
CalculatePredictionStatistics("bagging", predictions.train, predictions.test)

# clean up variables no longer needed
rm(student.model.bagging); rm(nbagg)



##### Boosting #####
library(gbm)
#help(gbm)
n.trees = 7000
student.model.boosting = gbm(G3~., data=student.train, distribution="gaussian",cv.folds=10, bag.fraction=1,
n.trees=n.trees, interaction.depth = 5)
gbm.perf(student.model.boosting, method="cv")
predictions.train = predict(student.model.boosting, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.model.boosting, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("boosting", predictions.train, predictions.test)

summary(student.model.boosting)
par(mfrow=c(1,2))
plot(student.model.boosting, i="absences")
plot(student.model.boosting, i="failures")
par(mfrow=c(1,1))


# clean up variables no longer needed
rm(student.model.boosting); rm(n.trees)



###### Random forest #####
library(randomForest)
```

```
#help(randomForest)
n.tree =  400
student.model.RandomForest = randomForest(G3~., data=student.train, ntree=n.tree)
student.model.RandomForest
if (showPlots) plot(student.model.RandomForest)
if (showInfo) round(importance(student.model.RandomForest), 2)
predictions.train <- predict(student.model.RandomForest, newdata=student.train, type="response")
predictions.test  = predict(student.model.RandomForest, newdata=student.test,  type="response")
CalculatePredictionStatistics("Random Forest.400", predictions.train, predictions.test)

importance(student.model.RandomForest)
varImpPlot(student.model.RandomForest)
plot(student.model.RandomForest)

if (saveMainResults) write.table(round(importance(student.model.RandomForest), 2), sep=";", file =
"VarImportance.csv")

# clean up variables no longer needed
rm(student.model.RandomForest)

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


#### ZINB ####
library(pscl)

student.model.zip.full <- zeroinfl(G3~ .|.,data = student.train, link = "logit", dist = "poisson")
summary(student.model.zip.full)

Anova(student.model.zip.full)
predictions.train = predict(student.model.zip.full, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.model.zip.full, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("ZeroInflatedPOisson.overfit", predictions.train, predictions.test)

#### ZINB 2 ####
student.model.zip <- zeroinfl(G3~ absences+failures+Mjob+health+Fjob+age+goout+schoolsup|
                       absences+failures+Mjob+health+Fjob+age+goout+schoolsup,
                      data = student.train,
                      link = "logit", dist = "poisson")
summary(student.model.zip)

predictions.train = predict(student.model.zip, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.model.zip, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("ZeroInflatedPOisson.optimized", predictions.train, predictions.test)


#### Negative Binomial ####
library(car)
student.model.ZINB <- zeroinfl(G3~ .|.,data = student.train, link = "logit", dist = "negbin")
summary(student.model.ZINB)
```

```
predictions.train = predict(student.model.ZINB, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.model.ZINB, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("ZeroInflatedNegBinom", predictions.train, predictions.test)
Anova(student.model.ZINB)
#### Negative Binomial 2 ####

student.model.ZINB.optimized <- zeroinfl(G3~ absences+failures+Mjob+health+Fjob+age|
                                absences+failures+Mjob+health+Fjob+age,
                               data = student.train,
                               link = "logit", dist = "negbin")

summary(student.model.ZINB.optimized)
predictions.train = predict(student.model.ZINB.optimized, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.model.ZINB.optimized, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("ZeroInflatedNegBinom.optimized", predictions.train, predictions.test)

cbind(AIC(student.model.zip.full),AIC(student.model.zip),
    AIC(student.model.ZINB),AIC(student.model.ZINB.optimized))


#### hurdle ####

student.hurdle <-hurdle(G3~ absences+failures+Mjob+health+Fjob+age|
              absences+failures+Mjob+health+Fjob+age,
             data = student.train,
             zero.dist="binomial",link="logit",dist="negbin")
summary(student.hurdle)
predictions.train = predict(student.hurdle, newdata=student.train, n.trees=n.trees)
predictions.test  = predict(student.hurdle, newdata=student.test,  n.trees=n.trees)
CalculatePredictionStatistics("student.hurdle", predictions.train, predictions.test)


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

PredictionStatistics <- PredictionStatistics[
  order( PredictionStatistics[,1],PredictionStatistics[,3] ) , ]
PredictionStatistics
if (saveMainResults) write.table(PredictionStatistics,sep = ",",
                         file = "PredictionStatistics.csv")

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

##############################################################################
#################### BIG DATA IMPLEMENTATION ###################################
##############################################################################



####### h20.io random forest demo ####
cat("\014") # clear console
```

```
# Model 1 : Random forest H2O versions
library(h2o)
## Start a local cluster with 2GB RAM
localH2O = h2o.init(ip = "localhost", port = 54321, startH2O = TRUE,
                max_mem_size = '2g',min_mem_size='1g')

#   http://localhost:54321/    should be active in browser now.
h2o.clusterInfo()
n.tree = 1000


student.train.h = as.h2o(student.train)
X.predictors = colnames(student.train)
X.predictors = X.predictors[-match("G3", X.predictors)]

student.h20 = h2o.randomForest(y="G3", x=X.predictors,
                           training_frame = student.train.h,
                           ntrees = n.tree)
predictions.train = as.data.frame(h2o.predict(student.h20,
                                  newdata = as.h2o(student.train)))$predict
predictions.test = as.data.frame(h2o.predict(student.h20,
                                 newdata = as.h2o(student.test)))$predict
CalculatePredictionStatistics("Random Forest- h2o.io", predictions.train, predictions.test)

PredictionStatistics <- PredictionStatistics[
  order( PredictionStatistics[,1],PredictionStatistics[,3] ) , ]
PredictionStatistics
if (saveMainResults) write.table(PredictionStatistics,sep = ",",
                         file = "PredictionStatistics.csv")
######## end of h20.io random forest demo ####
```