15.077 Final Project

Shi Wang

2015.5

Table of Contents

1. Source of dataset

student performance dataset from online resource

https://archive.ics.uci.edu/ml/datasets/Student+Performance

2. Main objective

This data approach student achievement in secondary education of two Portuguese schools. There are totally 649 observations and 32 explanatory variables (see Appendix 1). The data attributes include student grades, demographic, social and school related features) and it was collected by using school reports and questionnaires.

A 20-point grading scale is used in this evaluation system. The dependent variable is G3 (final year grade, which is a continuous, integer variable ranging from 0 to 20). G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful for education research. The tentative task of this dataset is to predict the student performance (G3) .

3. Descriptive analysis

The distribution of 32 attributes in this dataset are shown in Appendix 2. Figure 1 shows the distribution of dependent variable G3. The range is from 0 to 20. Table 1 shows the basic statistics about G3.

We may find the distribution of G3 is centered around 10. There are nearly 40 students achieving 0 score for the exam. Since we expect the first period grade (G1) and second period grade (G2) to have high correlation with final grade (G3), correlation coefficient is calculated as in Table 2.

4. Methodology

4.1 Partition of datasets

The whole dataset is randomly divided into two parts: 70% for training/validation and 30% for testing. For the models developed in following sections, training set would be used to determine the best model specification and calculate model parameters. Testing set would use the derived model to calculate evaluation metrics and present predictive performance.

4.2 Evaluation metric

The coefficient of determination (R square) and Root Mean Squared Error (RMSE) are used.

$$\textbf{(1)}\quad RMSE=\sqrt{\left(\sum_{i=1}^{N}(y_i-\hat{y}_i)^2/N\right)} \qquad \textbf{(2)}\quad R^2=1-\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}$$

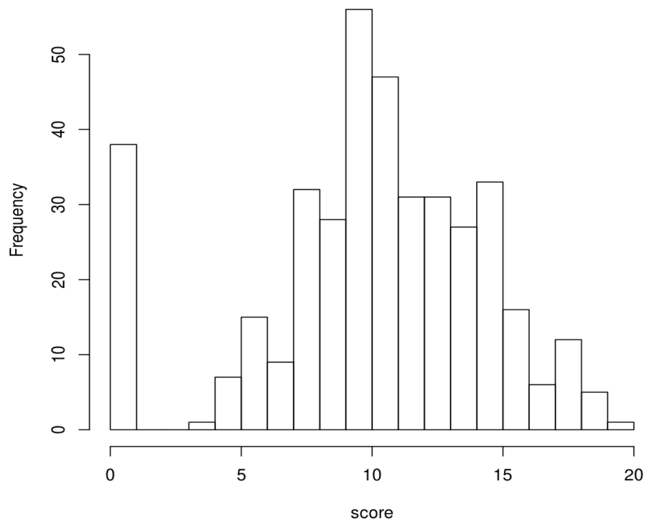


Figure 1 Distribution of G3

Mean	10.42
Median	11
Standard deviation	4.58
Min	0
Max	20

Table 1 G3 statistical summary

G1 and G3	0.801
G2 and G3	0.905

Table 2 Correlation between G1/G2 and G3

Where $\hat{y}_i$ denotes the predicted value for i-th instance, and y_i denotes the actual value for i-th instance. Lower value for RMSE or higher value for R square represents better predictive performance.

4.3 Input configuration

In this study, we want to predict the final grade G3 without considering the mid-term grades G2. Because predicting academic performance without knowing past grades could be more helpful for educational research. Therefore, only G1 (first period exam grade) and all other variables are used in analysis.

4.4 Modeling approach

(I) *Typical regression methods*

- Stepwise linear regression
- Principle Component Regression (PCR)
- Partial Least Square Regression (PLSR)
- Ridge regression

(II) *Regression by classification tools*

- Decision Tree (DT)
- Neutral Network (NN)
- Support Vector Machine (SVM)
- *k* Nearest Neighbor (kNN)
- Random Forest (RF)

4.5 Ideas on extended models

1. Use random forest to perform variable selection for other methods

Inclusion of irrelevant variables may cause the problem of low accuracy or overfittness for classification methods like SVM, *k*NN, NNet, resulting in sometimes poor predictive performance. Also for decision tree, the accuracy may be improved by feature pre-selection so as to avoid too much influence from noise variables.

The idea of random forest is to build full trees but the number of variables searched for each split is a random subset of *p*. Random forest itself can generate good predictive performance without the need for feature pre-selection. Also it can provide results on variable importance.

Therefore, I want to use the importance of variables extracted from results of random forests to pick the variable subset for other methods.

The process can be described as the following steps:

(1) Build random forest, and extract variable importance (use increase% in MSE).

(2) To further decide which set of variables to use, pick three subsets consisting 10%, 25%, 50% and 100% of the most useful attributes.

(3) For decision tree, I would use 10%, 25%, 50% and 100% subset to build decision tree, and I would compare the performance of the resulting pruned trees to determine which size of subset generates the best result.

(4) For simplicity, I would use the 10% most useful set for SVM, kNN and NNet, and compare the predictive performance with full variable subset.

2. Combination of decision tree and linear regression (call it Piece-wise linear models)

Although decision tree can performance regression, the value at each leaf node is the simple average of all instances classified under this node. It seems some useful information is lost during the process.

For linear regression, we use the same coefficients for all instances. It might be that we should use different set of coefficients for different groups of student. For example, final grade of students with higher past period grade may be less sensitive to other influencing factors

The idea is to first use decision tree to classify the instances into homogeneous groups (without going too deep), and for each group, perform ordinary linear regression. In this way, the advantages of both methods are demonstrated to some extent.

The process can be described as the following steps:

(1) Develop a regression tree using "*rpart*" function in *R*.

(2) For each leaf node, create a linear regression model by calling "*lm*" function in *R*, with the training set for this linear regression model restricted to instances from the corresponding leaf. This process is

realized by writing a new function "*lmrpart*".

5. Model application

5.1 Typical Regression methods

(1) Stepwise linear regression

Stepwise linear regression is used by "*stepAIC*" function, with the mode of stepwise search set to be "*both*", which is a combination of forward selection and backward elimination.

(2) Principle Component Regression and Partial Least Square Regression

These two regression methods are performed in a similar way using "*pcr*" and "*plsr*" function. For both of the methods, the number of components *ncomp* is determined by selecting the minimum bias-corrected mean squared error of prediction (MSEP).

(3) Ridge regression

Ridge regression is performed using "*lm.ridge*" function. The main parameter in ridge regression is *lambda*, which is the penalty parameter. The generalized cross validation score (GCV) is used as criteria for choosing the best lambda.

5.2 Regression by classification tools

(1) Decision tree (or call Regression tree)

The decision trees are built using "*rpart*" function. First idea is to build the full tree with the predetermined complexity parameter *cp* to be zero. Considering the relatively small size of the dataset in this project, the minimum number of samples at each node *minbucket* is set as 10. Then two pruning approaches are used:

-- *cp* achieving the minimum 10 fold cross-validation

-- *cp* with minimum tree size within one standard deviation from the minimum 10 fold cross-validation error

The two selection principles are written in two functions "*cpmin*" and "*cp1sd*".

(2) Support Vector machine

The SVM model is build using "*ksvm"* function. One important parameter to be tuned is the penalty parameter *C*. 10 fold cross validation are used to find the *C* value achieving minimum cross validation error. For the kernel function, typically there are linear, polynomial, radial basis and sigmoid, they would be compared and best one would be selected.

(3) Neural Network

The NNet model is built using "*nnet"* function. One hidden layer is used to restrict model complexity. Maximum iteration is set to be 1000 to ensure convergence. The number of nodes are determined via 10-fold cross validation using "*tune.nnet*" function.

(4) *k* nearest neighbor

The *k*NN model is built using "*kNN*" function. The main parameter to be tuned is the number of neighbors *k*. Similarly, cross validation is applied to select the best *k* value.

(5) Random forest

Random forest is realized using "*randomForest*" function. The minimum size of terminal nodes is set as 5, and the number of variables randomly sampled as candidates at each split is set as *p*/3, where *p* is the number of variables. The number of trees to grow is set as 500.

6. Results

6.1 Stepwise Linear Regression

The resulting coefficients are listed as below. Totally 9 variables are included in the final regression model. Then this regression model is used for prediction on test dataset, and the R square for prediction set is **0.603** (RMSE = **3.121**).

Coefficients for Stepwise regression

(Intercept)	age	address	failures	schoolsup
2.7558707559	-0.2831701061	0.5469268513	-0.5990008241	0.8350337991
romantic	Dalc	absences	G1	reason.fother
-0.6791180265	0.2369541916	0.0334135239	1.0158563823	1.1099766999

6.2 PCR/ PLSR

Regression	Best # of components	RMSE	R-square
PCR	40	3.11	0.606
PLSR	12	3.11	0.605

Table 3 Output of PCR and PLSR

Generally, PCR and PLSR can give the similar predictive performance as stepwise regression. However, they result in quite different optimal number of components. The coefficients for these two types of regression are listed in Appendix C.

6.3 Ridge regression

When the minimum GCV value is reached, the value of *lambda* is 20. The R-square for prediction is **0.594** (RMSE = **3.154**), which is very close to previous regression method. The coefficients for ridge regression are listed in Appendix C.

For illustration purpose, Figure 2 shows the plot of predicted value for G3 and the actual value from regression. In summary, the four regression methods used in this project (Stepwise regression, PCR, PLSR, Ridge) give us the similar prediction power. We find that for those students with actual score to be zero, all of the methods tend to give poor prediction results.

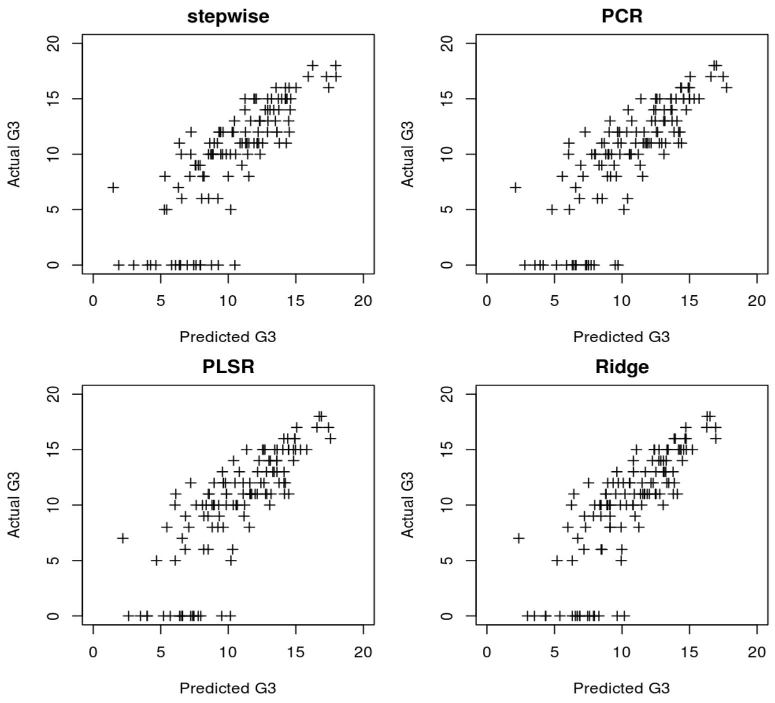


Figure 2 Prediction from four regression methods

6.4 Decision tree

We may find that under the full tree (Figure 3), the main influencing factor is G1, which is consistent with our expectation, because of the high correlation between G1 and G3. There are also some other factors worth attention. The number of absence (*absences*), going out with friends (*goout*), student age (*age*) and weekend extracurricular activities (*activities*) may also influence students' final grade.

After pruning the tree based on cross-validation error, we find G1, *absences* and *age* now dominants the influence (See Figure 4). The R square for testing using the 1-std pruned tree is **0.629** (RMSE = **3.102).**

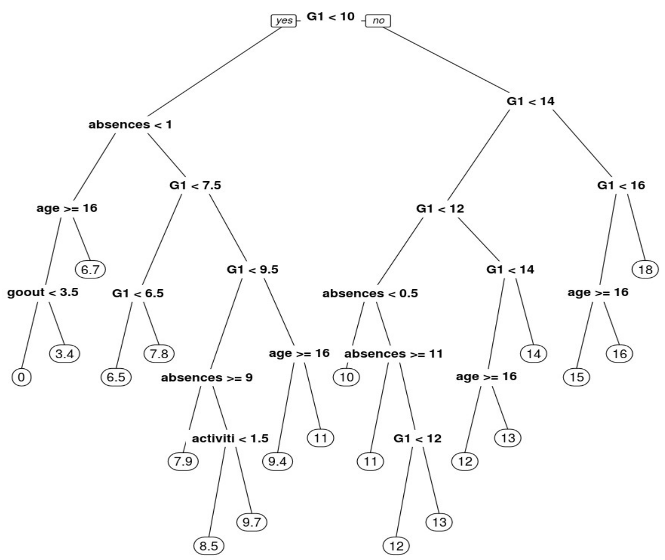


Figure 3 Unpruned tree

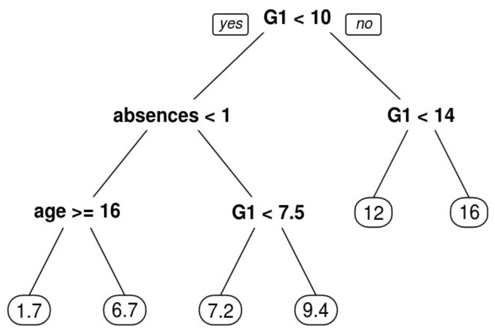


Figure 4 Pruned tree (1-std)

6.5 Random forest

After running random forest using the setting described in 5.2(5), We get the result as R-square=**0.652** and RMSE=**2.954** we can get the variable importance output. We can find G1, *absences* and *failures* have the most important influence. Here we can find actually in our dataset, there is only a few variables have significant influence on G3. The quality of the dataset is worse than expectation.

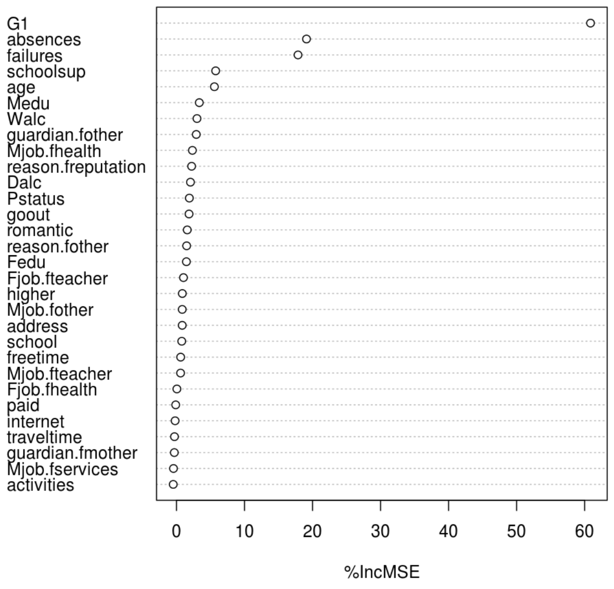


Figure 5 Variable importance plot

Output of 10/20/50 % variable set from random forest

```
$`10 percent`
[1] "G1"   "absences"  "failures"  "schoolsup"

$`25 percent`
 [1] "G1"                "absences"          "failures"
 [4] "schoolsup"         "age"               "Medu"
 [7] "Walc"              "guardian.fother"   "Mjob.fhealth"
[10] "reason.freputation"

$`50 percent`
 [1] "G1"                "absences"          "failures"
 [4] "schoolsup"         "age"               "Medu"
 [7] "Walc"              "guardian.fother"   "Mjob.fhealth"
[10] "reason.freputation" "Dalc"             "Pstatus"
[13] "goout"             "romantic"          "reason.fother"
[16] "Fedu"              "Fjob.fteacher"     "higher"
[19] "Mjob.fother"       "address"
```

By observing the best 10% variable subsets, one interesting fact is that when we are building the unpruned/pruned decision tree in previous section, "*failure*" and "*schoolup*" are included in none of the nodes. However, their importance is demonstrated here by random forest. Conversely, while age is included in the final pruned tree, it does not have very high importance here.

6.6 Decision tree with feature pre-selection

We then use different subset size to build decision tree, and prune the resulting tree using minimum CV error and minimum tree within 1 standard error of min CV error (as described in 5.2(1)). The following table shows the result for R square on testing set.

	10 percent	**25 percent**	**50 percent**	**100 percent**
R sqaure (pmin)	0.664	0.651	0.627	0.625
R square (p1sd)	0.646	0.601	0.608	0.608

Table 4 R-square with different variable subset

Generally, we want to use the within 1 std pruned tree instead of the minimum CV error tree. We can find that the maximum R square on 2nd line is obtained by using the 10 percent subset (R square=**0.646**, RMSE=**3.016**). When we are using bigger subset to build the decision tree, there is decreasing R square for pruned tree. Figure 6 shows pruned tree using 10% subset.

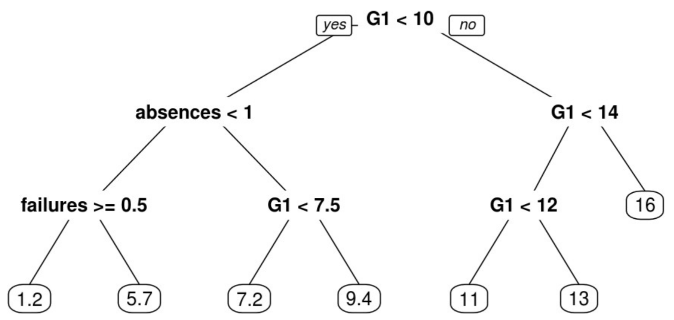


Figure 6 Pruned tree (1std) using 10% subset

6.7 Piece-wise linear models

1) First build regression tree with *cp* set as 0.1. Again using the 10% subset.

The reason for generating such a simple tree is that if I set the *cp* value to a smaller value, then it would be very close to the trees generated in previous parts. Since the predicted values at each leaf node would be close to each other, there is no much point in making regression again at each leaf node.

2) Figure 7 is the simple tree is generated, we can find the values at each leaf is 3.7, 8.7, and 14. Then regression is performed for each of the leaf as below. R square for prediction is **0.693** (RMSE = **2.795**). There is some improvement compared with decision tree and random forest in previous sections.

Coefficients of linear regression model at each leaf node

Leaf 1 $\boxed{3.7}$

```
Coefficients:
(Intercept)       G1    absences    failures   schoolsup
   -5.3808    0.4995          NA     -0.9369      5.5257
```

Leaf 2 $\boxed{8.7}$

```
Coefficients:
(Intercept)       G1    absences    failures   schoolsup
   2.68741   0.76767    -0.04111    -0.05921     0.05235
```

Leaf 3 $\boxed{14}$

```
Coefficients:
(Intercept)       G1    absences    failures   schoolsup
   0.82066   0.98280     0.01057    -1.90489    -0.57089
```

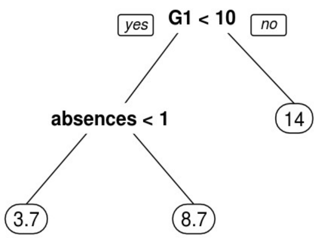


Figure 7 Pruned tree ($cp = 0.1$)

*Here the coefficient for "*absences*" at leaf 1 is not identifiable because all the samples have zero value for this variable. Also, we find different coefficients under different leaves, which may be explained by heterogeneity across students with different level of G1 and students with/without school absence.

6.8 SVM

Variables	Best C value	Kernel function	RMSE	R-square
all variables	3	Radial Basis	3.407	0.522
10% subset	6	Radial Basis	3.267	0.618

Table 5: SVM output

From the tests we find the "Radial Basis" to be the most proper kernel function. Cross validation shows the best cost parameter to be 3 and 6 respectively for all variables and 10% subset. The result in R-square shows improvement if only 10% subset is used for developing SVM model.

6.9 NNet

Variables	Number of hidden layer	Number of hidden nodes	RMSE	R-square
all variables	1	7	3.170	0.616
10% subset	1	5	2.821	0.704

Table 6: NNet output

From the tests we find when 10% subset is used, the optimal number of nodes in hidden layer is decreased from 7 to 5 to reach the minimum cross validation error. The visualized network with 10% subset is shown in Figure 8. The R-square for prediction is improved. We may conclude that feature pre-selection by random forest could be a good choice for neural network.

6.10 *k*NN

Variables	Distance measure	Best value of *k*	RMSE	R-square
all variables	Euclidean	13	2.889	0.660
10% subset	Euclidean	3	3.000	0.605

Table 7: kNN output

In this part, the traditional Euclidean distance is used as distance measure. Best *k* is selected to reach to minimum leave-one-out cross validation error (as in Figure 9). An interesting fact found here is that the 10% subset leads to poor predictive performance compared with all variables. But if I change the number of neighbor to be 13 as well for 10% subset, the R square increased to around 0.660. which is comparable with all variables. So there is no reason to overly trust the cross validation set performance.

Also, one thing worth attention is that *k*NN itself does not have the ability of weighing the input variables. Current *kNN* and *wkNN* package only provide weighing among instances instead of among variables. So actually in order to further improve predictive performance, proper variable weights together with instance weights need to be assigned, which is not covered in this project.

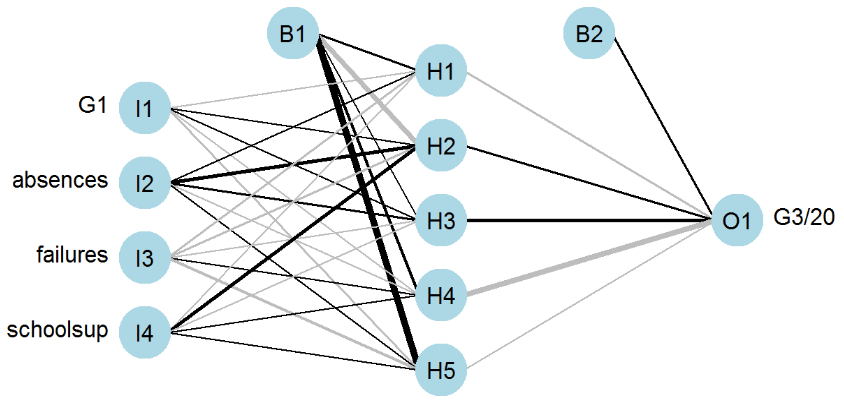


Figure 8 Neural network

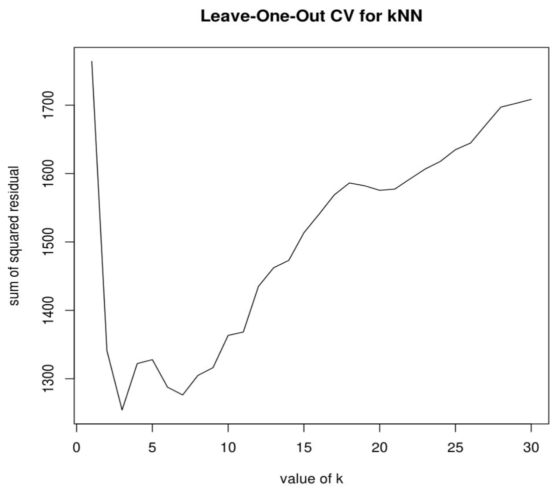


Figure 9 Selection of k

7. Conclusion

Method	RMSE	R-square
Step-wise	3.121	**0.603(-)**
PCR	3.110	0.606
PLSR	3.110	0.605
Ridge	**3.154(-)**	**0.594(-)**
Decision tree	3.016	0.646
Random Forest	2.954	0.652
Piece-wise	**2.795(+)**	**0.693(+)**
Neural Network	**2.821(+)**	**0.704(+)**
SVM	**3.267(-)**	0.618
KNN	3.000	0.605

Table 8 Summary of all the methods
* Results with variable selection via random forest is shown for decision tree, NNet, SVM and kNN

Table 8 summarizes the predictive performance for all the methods applied in this project. Method with annotated "+" are shown to be superior than others, while methods with "-" have relatively poor performance. Overall, piece-wise linear models and neural network are better than others in terms of RMSE and R-square.

There are some other interesting findings from analysis results:

(1) As discussed in 6.5, the number of useful input variables in this dataset is inadequate. This indicates even if we try different methods, or make combination of them, there is no much room for improvement. An important observation is that G1 is highly correlated with G3, and G1 is always the most important influencing factor in various methods. In contrast, although there are many attributes related to student demographic and family information, for most of them, the effects on final grade G3 is hardly detected.

(2) The results from these methods can fluctuate because the best model specification is determined based on the results of cross validation, which would not keep unchanged in different runs. Thus, it is not justifiable to assert that neural network and piece-wise linear models are superior than others. One

better solution, which is not implemented in this project, is to run cross validation for a number of times. For example, for SVM, run 10-fold cross validation for 20 times, and we may have 20 models with different values of best *C*. In this way, we can capture the variation of the best models.

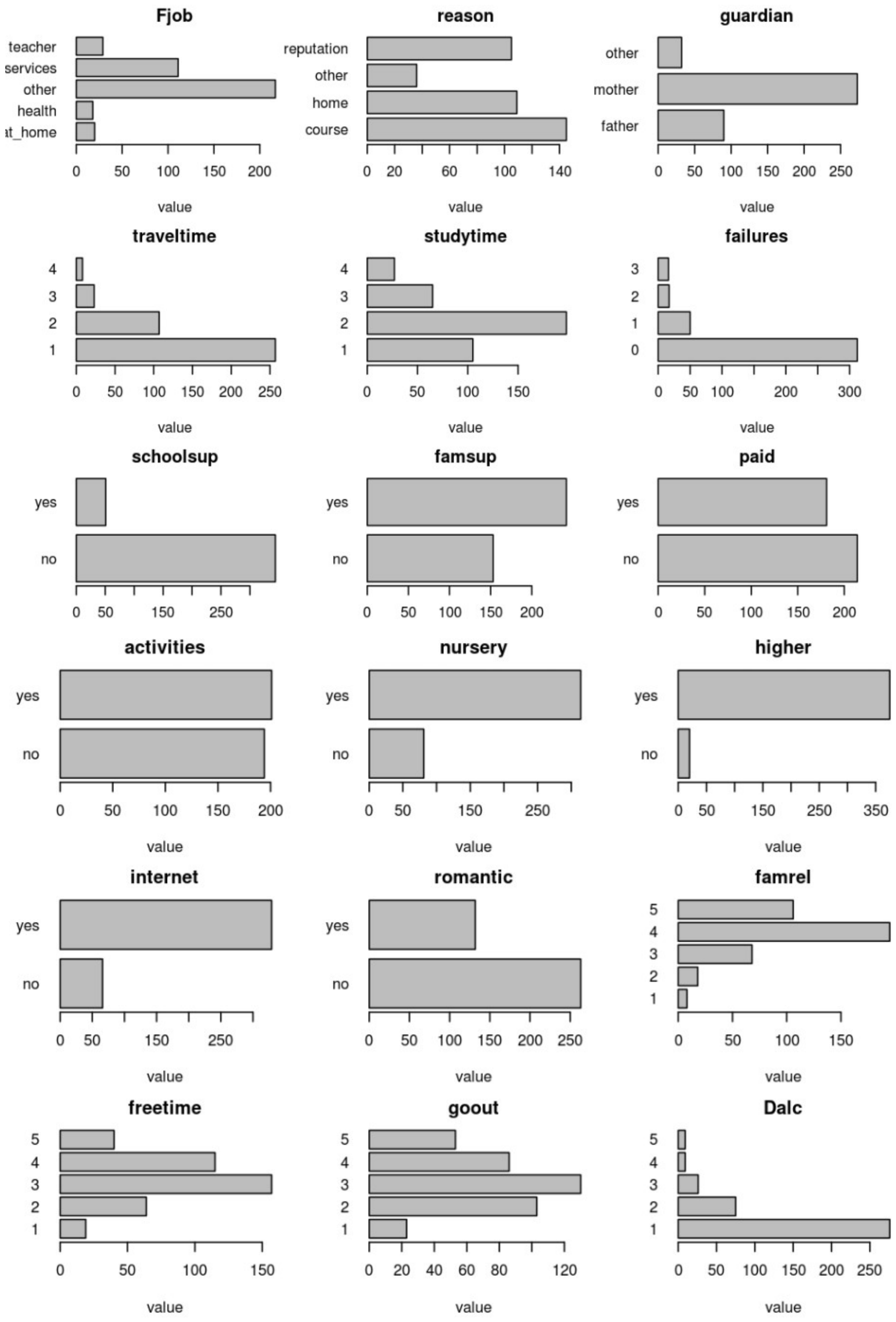
(3) There are a number of students receiving zero score for G3. They can be viewed as outliers to some extent. Maybe some effective techniques could be developed so as to detect these potential outliers and exclude them from regular prediction.

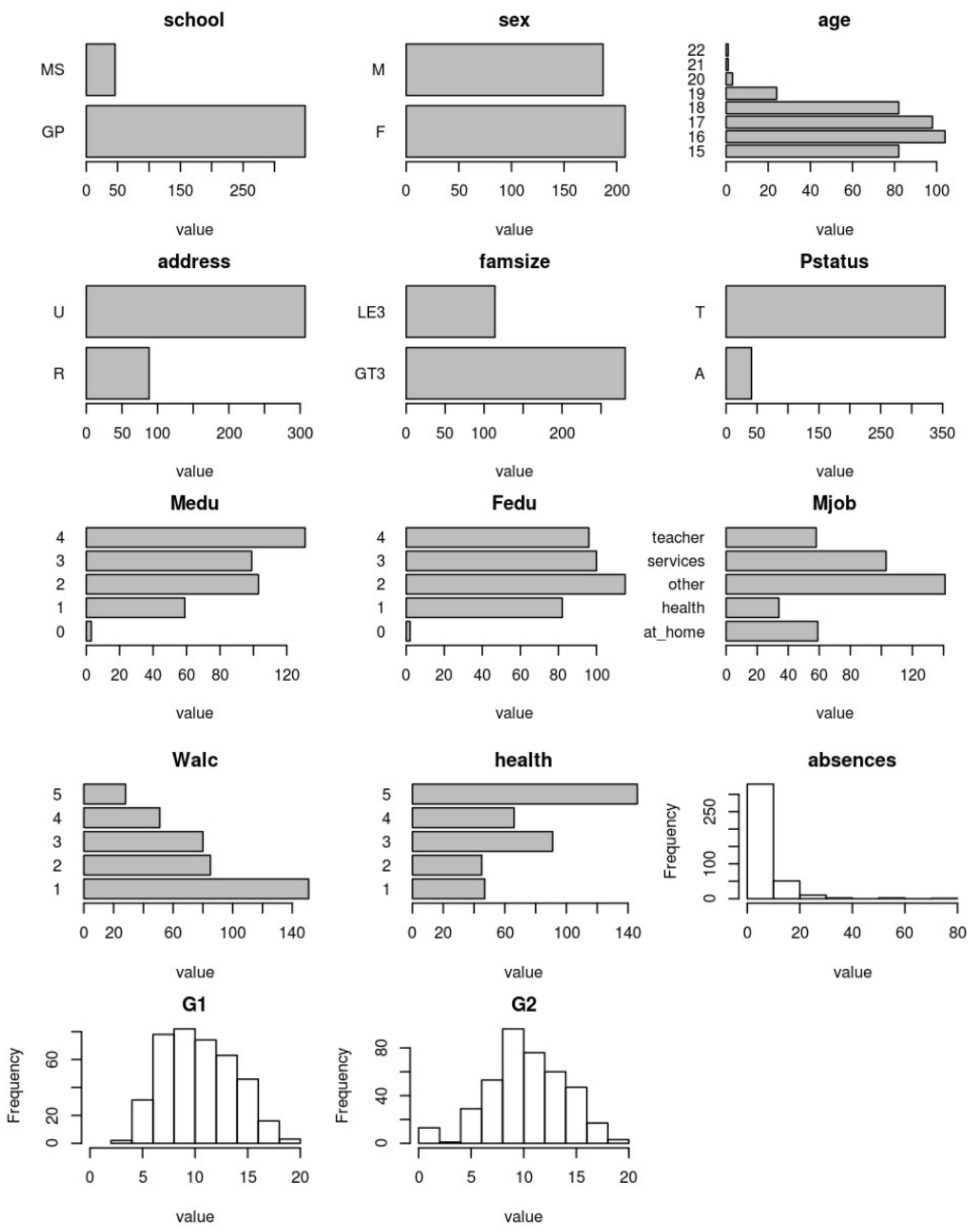
Appendix 1: Attribute Information

# Attributes for both student-mat.csv (Math course) datasets:
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)

# these grades are related with the course subject
31 G1 - first period grade (numeric: from 0 to 20)
31 G2 - second period grade (numeric: from 0 to 20)
32 G3 - final grade (numeric: from 0 to 20, output target)

Appendix 2: Barplot/histogram of attributes




school
MS
GP
sex
M
F
age
address
U
R
famsize
LE3
GT3
Pstatus
T
A
Medu
Fedu
Mjob
teacher
services
other
health
at_home
Walc
health
absences
Frequency
G1
G2
value

Appendix 3: Coefficients for PCR, PLSR and Ridge Regression

	PCR	PLSR	Ridge
Intercept	2.002	2.002	3.535
school	0.784	0.784	0.613
sex	0.240	0.240	0.235
age	-0.348	-0.348	-0.338
address	0.758	0.758	0.709
famsize	0.081	0.081	0.128
Pstatus	-0.366	-0.366	-0.336
Medu	0.295	0.295	0.263
Fedu	-0.187	-0.187	-0.154
traveltime	-0.046	-0.046	-0.018
studytime	0.009	0.009	0.047
failures	-0.538	-0.538	-0.626
schoolsup	1.142	1.142	0.768
famsup	0.249	0.249	0.137
paid	0.388	0.388	0.278
activities	-0.093	-0.093	-0.096
nursery	-0.063	-0.063	-0.034
higher	-0.395	-0.395	-0.295
internet	-0.271	-0.271	-0.227
romantic	-0.612	-0.612	-0.620
famrel	0.073	0.073	0.081
freetime	-0.184	-0.184	-0.133
goout	-0.006	-0.006	-0.053
Dalc	0.156	0.156	0.153
Walc	0.100	0.100	0.090
health	0.051	0.051	0.011
absences	0.028	0.028	0.028
G1	1.045	1.045	0.944
Mjob.fhealth	0.172	0.172	0.225
Mjob.fother	0.142	0.142	0.076
Mjob.fservices	-0.146	-0.146	-0.055
Mjob.fteacher	-0.543	-0.543	-0.502
Fjob.fhealth	0.404	0.404	0.601
Fjob.fother	0.367	0.367	0.398
Fjob.fservices	0.355	0.355	0.328
Fjob.fteacher	-0.333	-0.333	0.051
guardian.fmother	0.338	0.338	0.301
guardian.fother	0.618	0.618	0.597
reason.fhome	-0.138	-0.138	-0.099
reason.fother	0.858	0.858	0.761
reason.freputation	-0.171	-0.171	-0.124

Appendix 4: R codes

```
library(rpart)
library(Metrics)
library(randomForest)
library(rpart.plot)
library(FNN)
library(kknn)
library(pls)
library(Metrics)
library(MASS)
library(kernlab)
library(neuralnet)
library(MASS)
library(nnet)
library(e1071)

### Define function used in analysis

# R-sqaure
r2 <- function(pred.y, true.y)
{ 1 - length(true.y)*mse(pred.y, true.y)/((length(true.y)-1)*var(true.y)) }

# complexity parameter corresponding to the minimum cross-validated error
cpmin <- function(cptab) { cptab[which.min(cptab[,4])[1], 1] }

# complexity parameter corresponding to the smallest tree within 1-SD from the minimum cross-validated error
cp1sd <- function(cptab)
{ cptab[which(cptab[,4]<min(cptab[,4]) + cptab[which.min(cptab[,4]),5])[1], 1] }

# create a piecewise linear model represented by an rpart regression tree
lmrpart <- function(formula, data, skip.attr=FALSE, ...)
{
m.tree <- rpart(formula, data, ...)
m.leaves <- sort(unique(predict(m.tree, data)))
m.lm <- 'names<-'(lapply(m.leaves, function(l)
lm(G3~.,
data[predict(m.tree, data)==l,])),
m.leaves)
'class<-'(list(tree=m.tree, lm=m.lm), "lmrpart")
}

# prediction method for lmrpart
predict.lmrpart <- function(model, data)
{
leaves <- as.character(predict(model$tree, data))
sapply(1:nrow(data),
function(i) predict(model$lm[[leaves[i]]], data[i,]))
}

### Read in raw data

data <- read.table("student-mat.csv",sep=";",header=TRUE)

### Generate dummy variables for discrete atribute with more than two categories

attach(data)
Mjob.f = factor(Mjob)
Mjob.dum = model.matrix(~Mjob.f)[,2:5]

Fjob.f = factor(Fjob)
Fjob.dum = model.matrix(~Fjob.f)[,2:5]
```

```
guardian.f = factor(guardian)
guardian.dum = model.matrix(~guardian.f)[,2:3]

reason.f = factor(reason)
reason.dum = model.matrix(~reason.f)[,2:4]

### Bind dummay variables with dataset

detach(data)
data<-subset(data,select = -c(Mjob, Fjob, guardian, reason))
data<-cbind(data,Mjob.dum,Fjob.dum,guardian.dum , reason.dum )

data=data
for(i in 1:ncol(data)){
data[,i]<-as.numeric(data[,i])}

### Histograms

png(filename="Histogram of G3",width = 1000, height = 1000,res=150)
hist(data[["G3"]],breaks = 20, main = "Histogram of G3", xlab = "score")
dev.off()

### Barplot
png(filename="Barplot.png",width = 1000, height = 3000,res=150)
opar <- par(mfrow = c(12, 3), oma = c(0, 0, 1.1, 0),mar = c(4.1, 4.1, 2.1, 1.1))
for(i in 1:29){
barplot(table(data[,i]), main = colnames(data)[i], xlab = "value", col="grey",horiz = TRUE, las=1)}
for(i in 30:30){
hist(data[,i],breaks = 10, main = colnames(data)[i], xlab = "value", xlim=c(0,80))}
for(i in 31:32){
hist(data[,i],breaks = 10, main = colnames(data)[i], xlab = "value", xlim=c(0,20))}
par(opar)
dev.off()

png(filename="Correlation_G1_G3.png",width = 1000, height = 1000,res=150)
plot(data$G1, data$G3, type="p",pch=21,main="G1 and G3", xlab="G1", ylab="G3")
dev.off()
cor(data$G1, data$G3)

png(filename="Correlation_G2_G3.png",width = 1000, height = 1000,res=150)
plot(data$G2, data$G3, type="p",pch=21, main="G2 and G3",xlab="G2", ylab="G3")
dev.off()
cor(data$G2, data$G3)

### data partition 3:7

L<-nrow(data)
set.seed(12345)
L_train = 0.7 * L;
label<-sample(c(1:L),L_train)

### train and test split

train<-subset(data, rownames(data) %in% label)
test<-subset(data, !(rownames(data) %in% label))

### Run by different cases

case = 2; #Include G1 and others

if (case == 1){
ex <- names(train) %in% c("pass")
ex2 <- names(train) %in% c("G3")}
```

```
if (case == 2){
ex <- names(train) %in% c("G2")
ex2 <- names(train) %in% c("G2","G3")}

if (case == 3){
ex <- names(train) %in% c("G1","G2")
ex2 <- names(train) %in% c("G1","G2","G3")}

### LS

mod.ls <- lm(G3~., data = train[!ex])
p.ls <- predict(mod.ls,newdata = test)
r2.ls =r2 (p.ls , test$G3)
r2.ls

### Stepwise Regression

mod.ls.step <- stepAIC(mod.ls, direction="both")
p.ls.step <- predict(mod.ls.step,newdata = test)
B.ls.step <- mod.ls.step[["coefficients"]]
r2.ls.step = r2(predict(mod.ls.step, test), test$G3)
r2.ls.step

### PCR

mod.pcr <- pcr(G3~., validation="CV", data = train[!ex])
ncomp <- which.min(mod.pcr$validation$adj)
p.pcr <- predict(mod.pcr,newdata=test,ncomp=ncomp)
B.pcr <- as.numeric(coef(mod.pcr, ncomp = ncomp, intercept = TRUE))
r2.pcr =r2 (p.pcr , test$G3)
r2.pcr

### PLS

mod.plsr <- plsr(G3~., validation="CV", data = train[!ex])
ncomp <- which.min(mod.plsr$validation$adj) #  bias-corrected mean squared error of prediction (MSEP)
p.plsr <- predict(mod.plsr,test,ncomp=ncomp)
B.plsr <- as.numeric(coef(mod.plsr, ncomp = ncomp, intercept = TRUE))
r2.plsr =r2 (p.plsr , test$G3)
r2.plsr

### Ridge

m.ridge <- lm.ridge(G3 ~ .,data=train[!ex], lambda = seq(0,50,1))
plot(m.ridge$GCV)
p.ridge = scale(test[!ex2],center = T, scale = m.ridge$scales)%*% m.ridge$coef[,which.min(m.ridge$GCV)] + m.ridge$ym
B.ridge <- as.numeric(coef(lm.ridge(G3 ~ .,data=train[!ex], lambda = m.ridge$lambda[which.min(m.ridge$GCV)])))#Best lambda
r2.ridge =r2 (p.ridge , test$G3)
r2.ridge

### Plot for regression

png(filename="Four regression.png",width = 1000, height = 1000,res=150)
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),mar = c(4.1, 4.1, 2.1, 1.1))
plot(p.ls.step, test$G3,pch=3,main="stepwise",ylab="Actual G3",xlab="Predicted G3",xlim=c(0,20),ylim=c(0,20))
plot(p.pcr, test$G3,pch=3,main="PCR",ylab="Actual G3",xlab="Predicted G3",xlim=c(0,20),ylim=c(0,20))
plot(p.plsr, test$G3,pch=3,main="PLSR",ylab="Actual G3",xlab="Predicted G3",xlim=c(0,20),ylim=c(0,20))
plot(p.ridge, test$G3,pch=3,main="Ridge",ylab="Actual G3",xlab="Predicted G3",xlim=c(0,20),ylim=c(0,20))
dev.off()

### Decision Tree

control <- rpart.control(minsplit = 2,minbucket = 10,cp=0)
fit.dt<-rpart(G3 ~., data = train[!ex],control = control,method="anova")
fit.dt.pmin = prune(fit.dt, cpmin(fit.dt$cptable))
fit.dt.p1sd = prune(fit.dt, cp1sd(fit.dt$cptable))

r2.dt.pmin = r2(predict(fit.dt.pmin, test, type="vector"),test[["G3"]])
r2.dt.pmin
r2.dt.p1sd = r2(predict(fit.dt.p1sd, test, type="vector"),test[["G3"]])
r2.dt.p1sd

png(filename="Unpruned decision tree",width = 1000, height = 1000,res=150)
prp(fit.dt)
dev.off()

png(filename="Pruned decision tree",width = 1000, height = 1000,res=150)
prp(fit.dt.p1sd)
dev.off()

### Random forest

fit.rf <- randomForest(G3 ~., data=train[!ex], importance=TRUE)

p.rf<-predict(fit.rf, test, type="response")
r2.rf = r2(p.rf,test[["G3"]])
r2.rf

attr.utl<-sort(importance(fit.rf)[,1],decreasing = TRUE)
asets<-
'names<-'(lapply(c(10,25,50,100),
function(p)
names(attr.utl)[1:round(p*length(attr.utl)/100)]),
paste(c(10,25,50,100),"percent",sep=" "))

png(filename="Variable importance",width = 1000, height = 1000,res=150)
varImpPlot(fit.rf, type=1) #PLOT IMPORTANCE
dev.off()

### Decision tree with variable selection

fit.dt.group <-
lapply(asets,
function(aset)
{

tree.full <- rpart(make.formula("G3", aset), train[!ex],
minsplit=2, minbucket=10, cp=0)
tree.pmin <- prune(tree.full, cpmin(tree.full$cptable))
tree.p1sd <- prune(tree.full, cp1sd(tree.full$cptable))
list(
tree.pmin=tree.pmin,
r2.pmin=r2(predict(tree.pmin, test), test$G3),
tree.p1sd=tree.p1sd,
r2.p1sd=r2(predict(tree.p1sd, test), test$G3))
})

write.table(
sapply(fit.dt.group,
function(tree) c(r2.pmin=tree$r2.pmin, r2.p1sd=tree$r2.p1sd)),
file=paste("case ",case," Decision tree assets"))

png(filename="Decision-tree-best-subset",width = 1000, height = 1000,res=150)
prp(fit.dt.group$`10 percent`$tree.p1sd, varlen=0, faclen=0)
dev.off()

### Piece-wise linear regression
```

```
control <- rpart.control(minbucket = 10,cp=0.1,maxdepth=7,xval = 10)
lmtree<-lmrpart(make.formula("G3", asets$"10 percent"), train[,c("G3",asets$"10 percent")], control = control)
p.lmtree= predict.lmrpart(lmtree, test[,asets$"10 percent"])

r2.lmtree = r2(p.lmtree,test[["G3"]])
r2.lmtree

png(filename="LMTREE",width = 1000, height = 1000,res=150)
prp(lmtree$tree, varlen=0, faclen=0)
dev.off()

### kNN
for(x.sel in 0:1){
if(x.sel == 0) {train.knn = train[!ex2]; test.knn = test[!ex2]}
if(x.sel == 1) {train.knn = train[,c(asets$"10 percent")]; test.knn = test[,c(asets$"10 percent")]}

cv.err.knn = sapply(1:30,
function(k) knn.reg(train.knn,y=train[["G3"]],k=k)$PRESS)

k.best=which.min(cv.err.knn)

png(filename="kNN-selection_of_k",width = 1000, height = 1000,res=150)
plot(c(1:30),cv.err.knn,type="l",ylab="sum of squared residual", xlab="value of k", main="Leave-One-Out CV for kNN")
dev.off()

p.knn<-knn.reg(train.knn, test.knn, train[["G3"]], k=k.best,algorithm=c("kd_tree"))$pred
if (x.sel == 0) r2.knn = r2(p.knn,test[["G3"]])
if (x.sel == 1) r2.knn.xsel = r2(p.knn,test[["G3"]])
}

### NN

for(x.sel in 0:1){
if(x.sel == 0) {train.nn = train[!ex]; test.nn = test[!ex2]}
if(x.sel == 1) {train.nn = train[,c("G3",asets$"10 percent")]; test.nn = test[,c(asets$"10 percent")]}

fit.net<-best.nnet(G3/20~., data = train.nn, size = 1:10,maxit = 1000,tunecontrol=tune.control(cross=10))
p.net<-predict(fit.net,test.nn, type="raw")*20

if (x.sel == 0) r2.net<-r2 (p.net, test$G3)
if (x.sel == 1) r2.net.xsel <- r2 (p.net, test$G3)
}

### SVM
for(x.sel in 0:1){

if(x.sel == 0) train.svm = train[!ex];
if(x.sel == 1) train.svm = train[,c("G3",asets$"10 percent")];

cv.err.svm = sapply(1:20,
function(C) cross(ksvm(G3~.,data=train.svm,kernel="rbfdot",type="nu-svr",cross=10,C=C)))

C.best = which.min(cv.err.svm)
fit.svm<-svm(G3~.,data=train.svm,kernel="radial",cost=C.best)

p.svm<-predict(fit.svm,newdata=test,type="response")
if (x.sel == 0) r2.svm<-r2(p.svm, test$G3)
if (x.sel == 1) r2.svm.xsel<-r2(p.svm, test$G3)
}

### Report error rate
X = c(r2.ls.step,r2.pcr,r2.plsr,r2.ridge,r2.dt.pmin,r2.dt.p1sd,r2.rf,r2.lmtree,r2.net,r2.net.xsel,r2.svm,r2.svm.xsel,r2.knn,r2.knn.xsel)
write.table(X,file=paste("case ",case," Test-error"),row.names=FALSE, col.names=FALSE)
```