Final project - Big Data

Anna Kołacz, Reneeleonette Serota, Mariusz Zakrzewski, Kamil Stec

December 2019

Contents

1 Abstract

This work focuses on data gathered in a survey of a secondary school's students. The dataset contains information about students and their performance in school. Our work has two major goals: first to predict whether a student will pass an exam or not, second to predict the exact number of points student will score.

We begin by elaborating the used dataset as well as methods which we applied to clean the database. In the next chapter we implemented various classification methods to predict whether a student will have a good or bad exam's performance. For this task we used logistic regression, support vector machines, KNN and random forest. In the 3. chapter, we explored regression methods to predict the number of points students will score in their final exam. We implemented ... (TODO:which methods) At the end, we summarised as well as compared obtained Machine Learning algorithms scores.

2 Dataset

Data was obtained in a survey of a secondary school's students. It is available under the link https://archive.ics.uci.edu/ml/datasets/student+performance.

The dataset contains variables about social, personal and study aspects as well as information about obtained number of points in Maths and Portugese language. Because of major part of the results in Portugese language and our reluctance to combine grades from entirely different courses, we concentrated on Portugese language's student in our research.

What is more, we decided to exclude a few variables, because of a negligible relationship with the dependent variable from our project. Moreover, variables G1 and G2 (first and second exam results) were excluded because we want to simulate a situation where student has not taken any exam yet and predict if he will pass the exam or not.
Additionaly, we created a variable 'studentPerformance' which divides the students into those who passed the exam and who did not. The division was made by passing score of 60%. Maximum number of score on that exam was 20, therefore passing score amounts to 12.

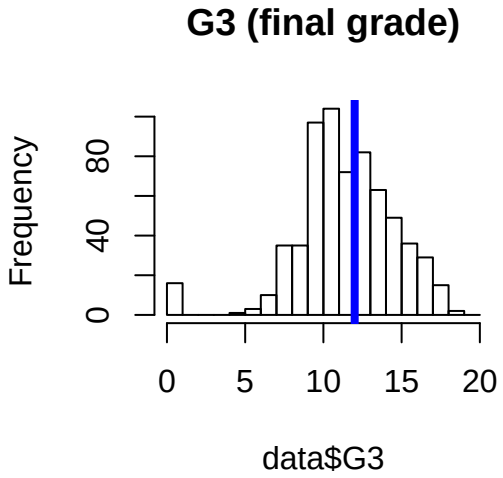
The following list shows the dependent variables used in the research:

- `school`: student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
- `sex`: tudent's sex (binary: "F" - female or "M" - male)
- `age`: student's age (numeric: from 15 to 22)
- `address`: student's home address type (binary: "U" - urban or "R" - rural)
- `famsize`: family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
- `Pstatus`: parent's cohabitation status (binary: "T" - living together or "A" - apart)
- `Medu`: mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- `Fedu`: father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- `reason`: reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
- `studytime`: weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- `failures`: number of past class failures (numeric: n if 1<=n<3, else 4)
- `schoolsup`: extra educational support (binary: yes or no)
- `famsup`: family educational support (binary: yes or no)
- `paid`: extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- `activities`: extra-curricular activities (binary: yes or no)
- `nursery`: attended nursery school (binary: yes or no)
- `higher`: wants to take higher education (binary: yes or no)
- `internet`: Internet access at home (binary: yes or no)
- `famrel`: quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- `freetime`: free time after school (numeric: from 1 - very low to 5 - very high)
- `Dalc`: workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- `Walc`: weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- `health`: current health status (numeric: from 1 - very bad to 5 - very good)
- `absences`: number of school absences (numeric: from 0 to 93)

The following list shows the atributes excluded from the research:

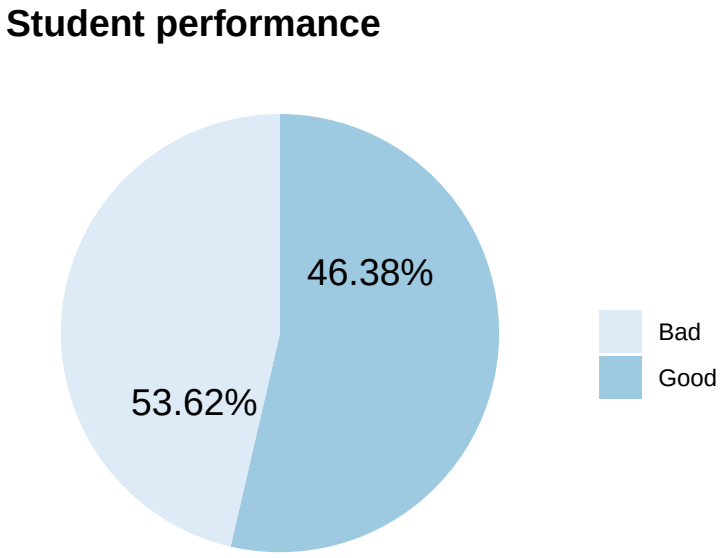
- `Mjob`: mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
- `Fjob`: father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
- `guardian`: student's guardian (nominal: "mother", "father" or "other")
- `romantic`: with a romantic relationship (binary: yes or no)
- `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)
- `goout`: going out with friends (numeric: from 1 - very low to 5 - very high)

In the classification task, G3 (final grade, numeric variable from 0 to 20) served as the dependent variable. In the regression task, the variable studentPerformance was our object of study. final grade (numeric: from 0 to 20, output target)



As can be seen on the histogram, there are few students with a number of points below 5 as well as few students with nearly maximum number of points. Most of learners obtained half of possible number of points.

The below plot represents a proportion of students whose passed and failed the exam.



As shown in the figure above, the proportion of passed and failed exams is nearly equal.

3 Classification

3.1 Model selection

3.2 Logistic regression

In this chapter, some results concerning Logistic regression were presented.
On the training test, the Logistic regression model achieved the success rate equal to 81.92%. However, because of frequent occurance of overfitting, it is better to compare algorithms' scores on the test set. Taking into account the test set, the model predicted correctly 17 students with Bad Performance and 80 students with Good Performance. It results in the correct prediction at the level of 75.19%. In order to determine if the obtained statistics is good or not, we compare the obtained result with the base model which is a

reflection of a random guessing. Its success rate is equal to 69,77%. It follows that the logistic regression model gives us an improvement of the success rate at the level of 5.4 percentage points.

As can be seen on the ROC curve, there is a large advantage of the true positive rate for a threshold's range of 0.0 - 0.7. ... The AUC for the logistic regression amounts to 78.86%.

```
dataLogit <- subset(data, select = -c(G3))

# split the data in the proportion 80%/20%
labelsLogit = sample.split(dataLogit$studentPerformance, SplitRatio = 4/5)

dataLogit_train = dataLogit[labelsLogit, ]
dataLogit_test = dataLogit[!labelsLogit, ]
```

We calculate confusion matrix for threshold equal to 0.5.

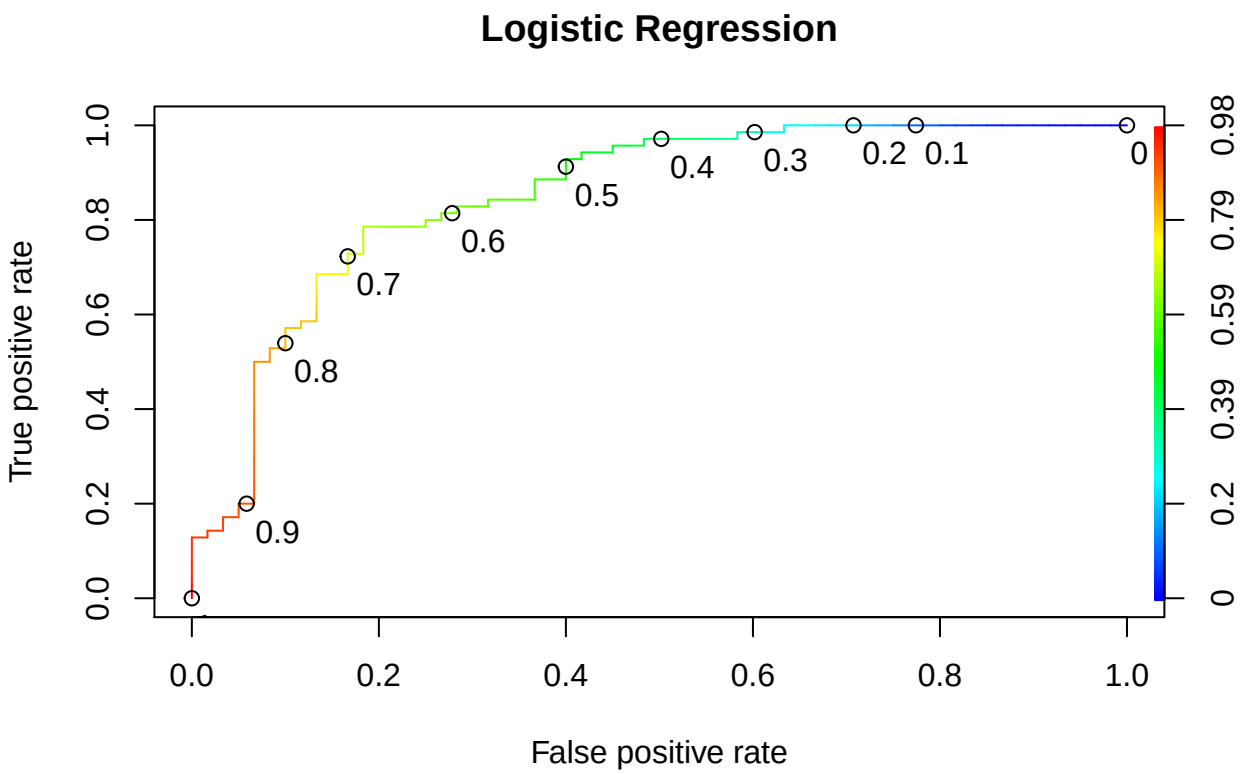
```
# fit the logistic regression model on the training sample only
glm.fit = glm(studentPerformance ~ ., data = dataLogit_train, family = binomial)

# predicted probabilities on the test sample
glm.probs = predict(glm.fit, newdata = dataLogit_test, type = "response")

# predicted signs on the test set
glm.pred = ifelse(glm.probs > 0.5, 1, 0)

# test sample confusion matrix
LogitConfMat <- table(true = dataLogit_test$studentPerformance, predict = glm.pred)
LogitConfMat
```

```
##       predict
## true    0  1
##   Bad  36 24
##   Good  7 63
```

```
# test sample total success rate
dataLogit_test$studentPerformance <- ifelse(dataLogit_test$studentPerformance=="Good",1,0)
LogitSuccessRate <- mean(glm.pred == dataLogit_test$studentPerformance)
LogitSuccessRate
```

```
## [1] 0.7615385
```

```
# test set ROC curve and AUC
predob = prediction(glm.probs, dataLogit_test$studentPerformance)
perf = performance(predob, "tpr", "fpr")
plot(perf, main = "Logistic Regression", colorize = TRUE, print.cutoffs.at = seq(0, 1, by = 0.1), text.a
```




```
LogitAUC <- as.numeric(performance(predob, "auc")@y.values)

# total test set success rate of the baseline model (with prediction 1 for every observation)
Basemodel <- mean(1 == dataLogit_test$studentPerformance)

# Confusion matrix and SuccessRate on training set
glm.probs_train = predict(glm.fit, newdata = dataLogit_train, type = "response")
glm.pred_train = ifelse(glm.probs_train > 0.5, 1, 0)
LogitConfMat_train <- table(true = dataLogit_train$studentPerformance, predict = glm.pred_train)
dataLogit_train$studentPerformance <- ifelse(dataLogit_train$studentPerformance=="Good",1,0)
LogitSuccessRate_train <- mean(glm.pred_train == dataLogit_train$studentPerformance)
LogitSuccessRate_train
```

```
## [1] 0.7668593
```

3.3 Support Vector Machines

In this chapter we use SVM for predicting if student will pass the final exam. 10-CV was used to find tuning parameters γ and λ (equal to $1/C$, C is the cost of a violation to the margin). As a kernel the "radial" function was used.

After assesing tuning parameters we measure the performance of SVM algorithm with the computed parameters.

SVM is not a probability classifier. The result of its prediction is simply a binary response. Therefore we cannot use ROC curve and AUC metric to measure its accuracy and compare it to other classification algorithms. That is why we used an accuracy metric error rate to check the SVM performance. What is more, we calculated also the confusion matrix.

```r
dataSVM <- subset(data, select = -c(G3))

# dividing our dataset into training and test sets (80%/20%)
train = sample(nrow(dataSVM), 0.8*nrow(dataSVM))
dataSVM_train = dataSVM[train, ]
dataSVM_test = dataSVM[-train, ]

dataSVM_test.response <- dataSVM_test$studentPerformance

### 10-CV using `tune()`
# cross-validating the tuning parameters gamma and lambda for the radial SVM om training data
tune.out = tune(svm, factor(studentPerformance) ~ ., data = dataSVM_train, kernel = "radial",
                ranges = list(cost = c(1e-10, 1e-5, 0.1 ,1 ,10 ,100 ,1000), gamma = c(0.5, 1, 2, 3, 4)))

# retrieve the best model
svmfit.opt = tune.out$best.model
summary(svmfit.opt)
```

```
##
## Call:
## best.tune(method = svm, train.x = factor(studentPerformance) ~ .,
##     data = dataSVM_train, ranges = list(cost = c(1e-10, 1e-05, 0.1,
##         1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  519
##
##  ( 244 275 )
##
##
## Number of Classes:  2
##
## Levels:
##  Bad Good
```

```r
# confusion matrix on the test set
table(true = dataSVM_test.response,  pred = predict(svmfit.opt, dataSVM_test))
```

```
##       pred
## true   Bad Good
##   Bad   35   22
##   Good  13   60
```

```r
SVMErrorRate <- (1 - mean(predict(svmfit.opt, dataSVM_test) == dataSVM_test.response)) # error rate

# TODO: in our dataset Number of Support Vectors is very large. How should we pick C to change it?
# TODO: how to pick kernel, cost, gamma
# TODO: parameter tuning with library(kernlab)
# TODO: probably because our dataset contains a lot of categorical variables, SVM method behavies very
```

As we can see in the results, the number of Support Vectors in our experiment is equal to 519 (much more than the $p = 23$). We assume a reason for such a high value can be the fact that the major part of predictors in our dataset are factorial and many of them are also binary.

The error rate of SVM classifier is equal to 0.2692308.

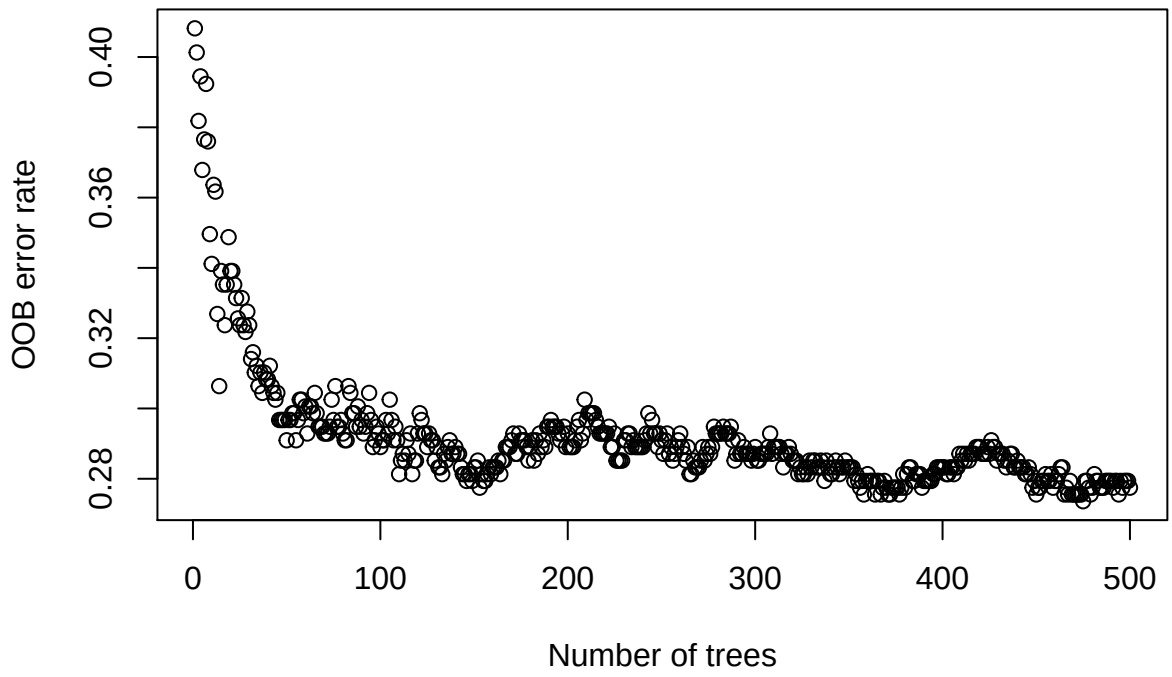
3.4 K-Nearest Neigbours

3.5 Linear Discriminant Analysis

3.6 Random Forests and Bagging

This chapter concentrates on predicting the students' perfomance using Random Forests and Bagging.
At the beginning, we checked wheter the default value for tuning parameter of Random Forests (500 decision trees in a model) is proper for our dataset. Later, we compare performance of Random Forests and Bagging method by measuring averaged AUC of models. Moreover, we created Importance Plots for applied algorithms.

```
dataRF <- subset(data, select = -c(G3))

labelsRF = sample.split(dataRF$studentPerformance, SplitRatio = 4/5)
dataRF_train = dataRF[labelsRF,]

# checking if 500 trees is enough for our dataset
checking = randomForest(studentPerformance ~ ., data = dataRF_train)
plot(checking$err.rate[,1], xlab="Number of trees", ylab="OOB error rate")
```



The OOB error rate decreases significantly to around 50 trees. For a larger number of trees the difference of the OOB error is slight. Therefore we conclude that $B = 500$ is sufficient.

After that, we built multiple models of Random Forests, tested its accuracy and averaged the results for obtaining a better estimation. Accuracy of model was measured by AUC (Area Under the Curve). While applying Random Forest we used default tuning parameters - 500 trees and $\lfloor sqrt(p) \rfloor = 4$ (for classification).

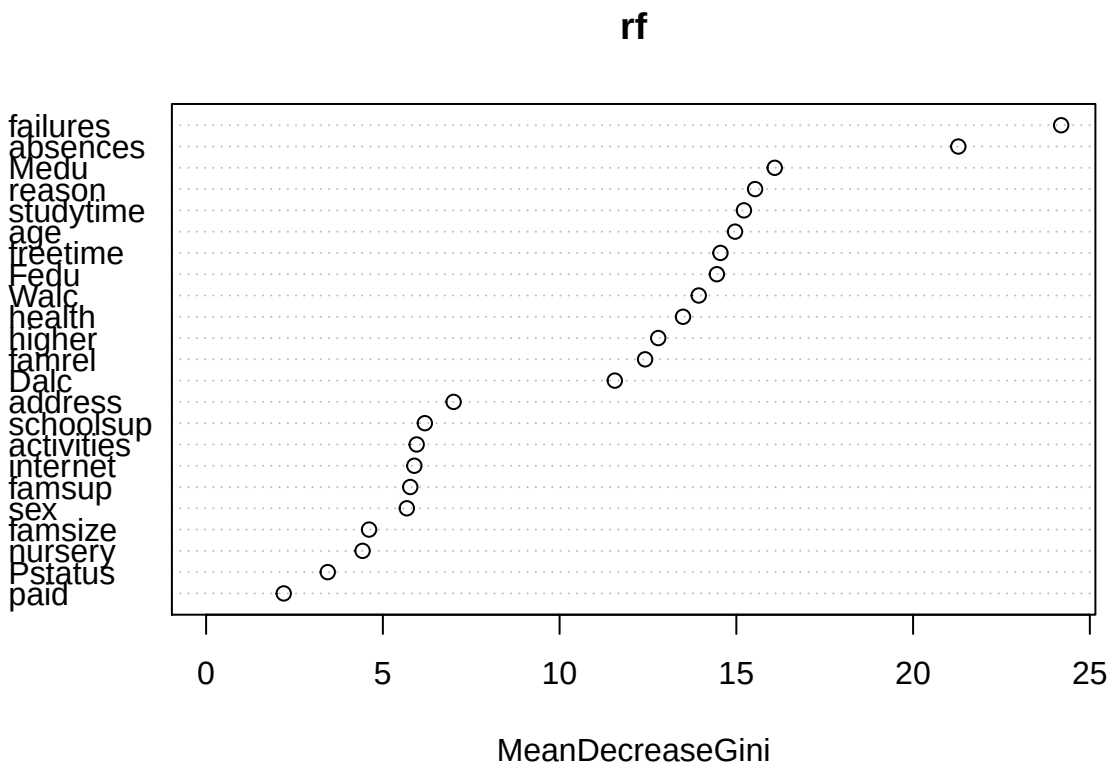
```
auc_value20 = 0.0
repeat_each = 20
for (i in 1:repeat_each) {

  labelsRF = sample.split(dataRF$studentPerformance, SplitRatio = 4/5)
  dataRF_train = dataRF[labelsRF,]
  dataRF_test = dataRF[!labelsRF,]

  randomForestResult = randomForest(studentPerformance ~ ., data = dataRF_train)

  rf.probs = predict(randomForestResult, dataRF_test, type = "prob")[, 2]
  predob.rf = prediction(rf.probs, dataRF_test$studentPerformance)
  perf = performance(predob.rf, "tpr", "fpr")

  auc_value <- as.numeric(performance(predob.rf, "auc")@y.values)
  auc_value20 = auc_value20 + auc_value
}
auc_value20 / repeat_each # mean auc
```

```
## [1] 0.8047381
```

The accuracy of Classification Random Forest measured by AUC is equal to 0.8011636.

Let us check importance of each variable with respect to predicting studentPerformance.

```
rf = randomForest(studentPerformance ~ ., data = dataRF_train)
varImpPlot(rf) # plot importance
```

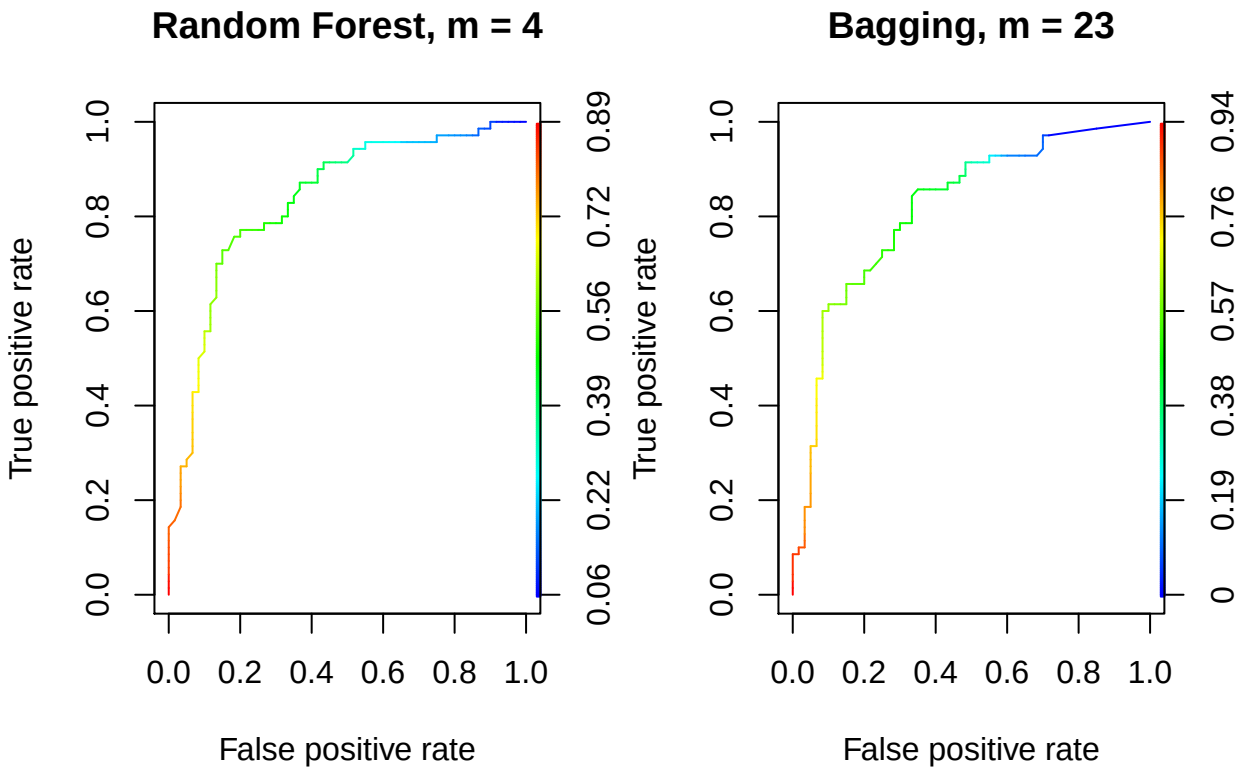


The importance of variables was measured for each tree by the improvement in the split-criterion. As a measure of improvement in split-criterion, Gini Index was served.
On the above plot we can see that the most important predictors of students success are: number of their

absences, mother education, measure of health, reason for school choice and study time.

Below, we plotted the ROC curve for Random Forest with default $mtry$ and with $mtry = p$ which is equivalent to bagging method.

```
# TODO: make sure if m = 23
# TODO: what about OOB?
BaggingResult = randomForest(studentPerformance ~ ., data = dataRF_train, mtry = 23, ntree = 500)
bagg.probs = predict(BaggingResult, dataRF_test, type = "prob")[, 2]
predob.bagg = prediction(bagg.probs, dataRF_test$studentPerformance)
perf.bagg = performance(predob.bagg, "tpr", "fpr")

par(mfrow = c(1, 2))
plot(perf, main = "Random Forest, m = 4", colorize = TRUE)
plot(perf.bagg, main = "Bagging, m = 23", colorize = TRUE)
```



Let's now compare averaged accuracy (measured by AUC) for default random forest and bagging method. Both errors were estimated by test data, and not by OOB error.

```
auc_value20 = 0.0
repeat_each = 20

for (i in 1:repeat_each) {
  labelsRF = sample.split(dataRF$studentPerformance, SplitRatio = 4/5)
  dataRF_train = dataRF[labelsRF,]
  dataRF_test = dataRF[!labelsRF,]

  BaggingResult = randomForest(studentPerformance ~ ., data = dataRF_train, mtry = 23, ntree = 500)
  bagg.probs = predict(BaggingResult, dataRF_test, type = "prob")[, 2]
  predob.bagg = prediction(bagg.probs, dataRF_test$studentPerformance)
```

```
  auc_value <- as.numeric(performance(predob.bagg, "auc")@y.values)
  auc_value20 = auc_value20 + auc_value
}

cat("RandomForest (m = 4): ", auc_value20 / repeat_each,
    "\nBagging      (m = 23): ", as.numeric(performance(predob.bagg, "auc")@y.values))
```

```
## RandomForest (m = 4):  0.7781488
## Bagging      (m = 23):  0.7630952
```

For random forest ($m = 4$) AUC curve is equal to 0.8011636.

For bagging ($m = 23$) AUC curve is equal to 0.7941455.

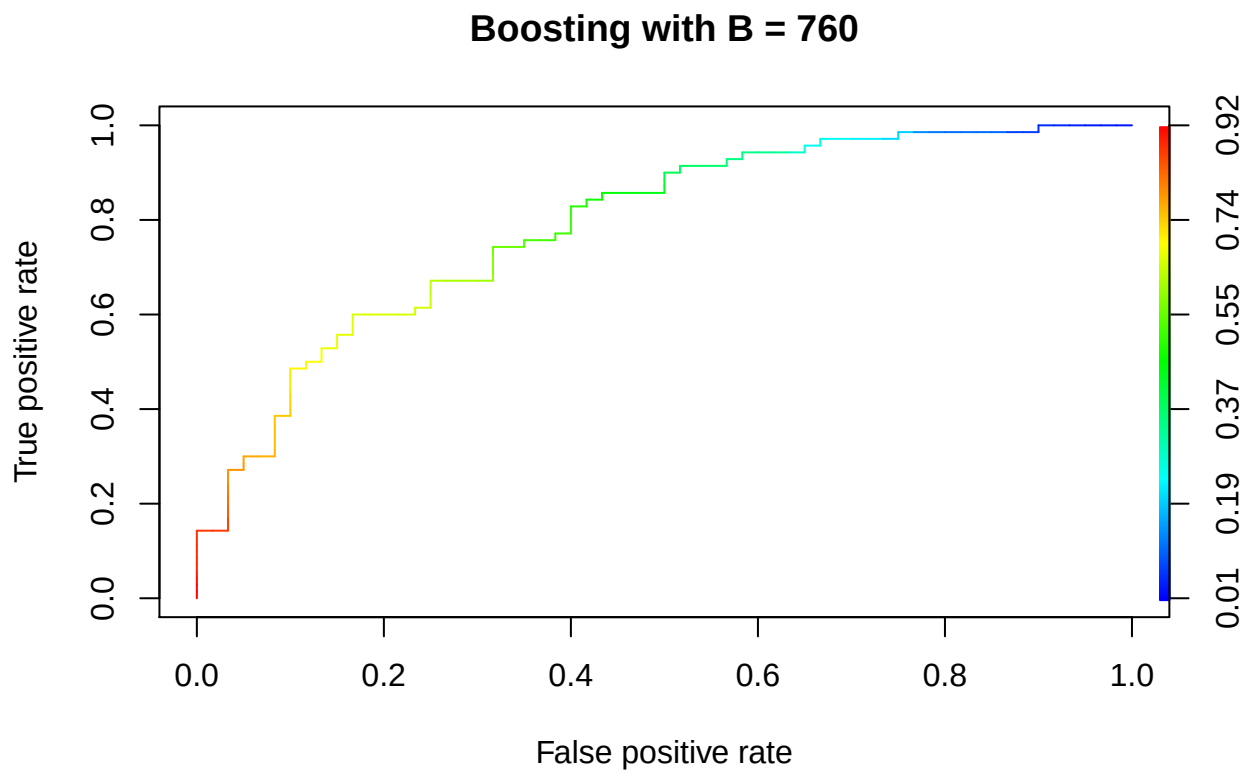
By applying bagging method we see a slight degradation of model accuracy.

3.6.1 Boosting

This chapter focuses on prediction the student performance by using Boosting methos.

At the beginning, we used Cross-Validation which enabled us to pick a proper number of trees equal to ???. According to this value, we built a Boosting model, plotted the ROC curve, calucalted the AUC and showed which variables were considered as the most important.

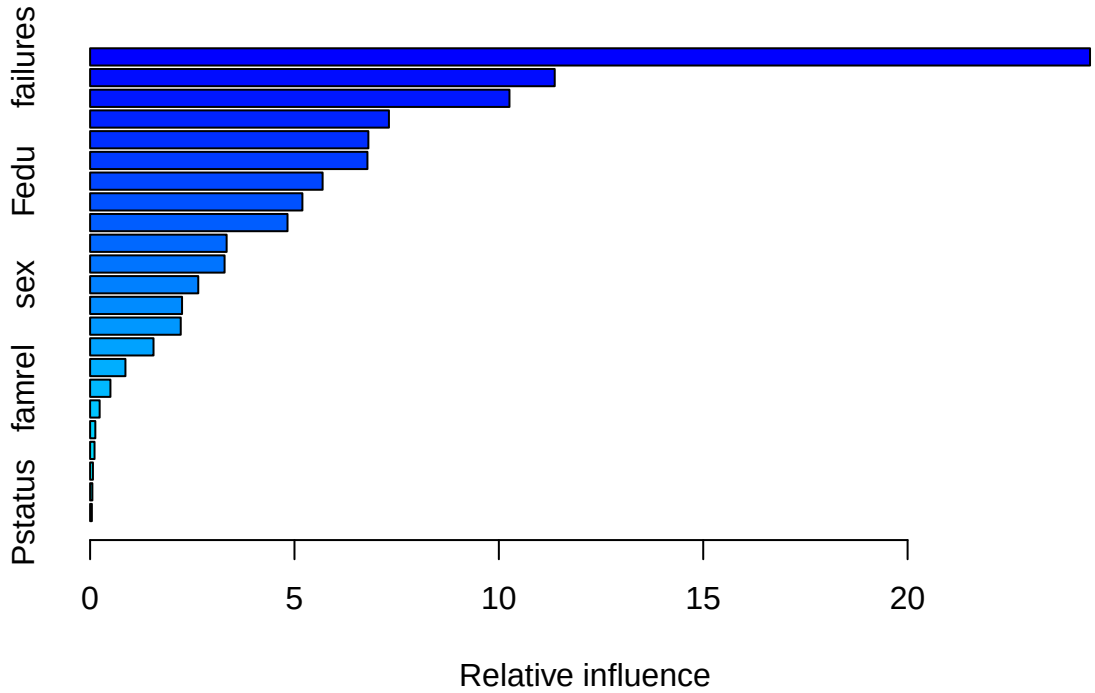
```
dataBoosting <- subset(data, select = -c(G3))
dataBoosting$studentPerformance <- as.integer(dataBoosting$studentPerformance)

labelsBoosting = sample.split(dataBoosting$studentPerformance, SplitRatio = 4/5)
dataBoosting_train = dataBoosting[labelsBoosting,]
dataBoosting_test = dataBoosting[!labelsBoosting,]

# We use CV to pick the number of trees.
boostingCV = gbm((studentPerformance-1) ~ ., data = dataBoosting_train, distribution = 'bernoulli', n.t
                 shrinkage = 0.01, cv.folds = 10)
# Bernoulli requires the response to be in {0,1}. That's why we use (studentPerformance-1)
best.iter = gbm.perf(boostingCV, method = "cv", plot.it = F)# Check the best iteration
# We fit the optimal boosted tree.
boostingBest = gbm((studentPerformance-1) ~ ., data = dataBoosting_train, distribution = 'bernoulli',
                   n.trees = best.iter, shrinkage = 0.01)

# test set ROC curve and the AUC for the best boosted tree.
boost.probs = predict(boostingBest, dataBoosting_test, n.trees = best.iter, type = "response")
predob = prediction(boost.probs, dataBoosting_test$studentPerformance)
perf = performance(predob, "tpr", "fpr")
plot(perf, main = "Boosting with B = 760", colorize = TRUE) #TODO: B
```




```
as.numeric(performance(predob, "auc")@y.values)
```

```
## [1] 0.7869048
```

```
# Variable importance plot
summary(boostingBest)
```




```
##                   var     rel.inf
```

```
## failures     failures 24.46305297
## higher         higher 11.36694819
## studytime   studytime 10.25990113
## address       address  7.31094635
## absences     absences  6.80752496
## Medu             Medu  6.78521763
## Fedu             Fedu  5.68807704
## Walc             Walc  5.19384299
## schoolsup   schoolsup  4.82993760
## reason         reason  3.34044957
## Dalc             Dalc  3.29009262
## sex               sex  2.64632432
## activities activities  2.24950121
## health         health  2.21719245
## freetime     freetime  1.55168799
## age               age  0.86429742
## famrel         famrel  0.49713639
## famsize       famsize  0.23351077
## paid             paid  0.12765944
## famsup         famsup  0.10845713
## nursery       nursery  0.06920045
## internet     internet  0.05566911
## Pstatus       Pstatus  0.04337230
```

```
#TODO: how to pick shrinkage, d, distribution
```

For Boosting method (with $B = 760$) the AUC metric is equal to 0.7820606. According to the obtained results of Boosting, the most relative predictor is 'number of failures'. A the next important variables the Boosting Algorithm indicated 'mother education', 'number of absences', 'reason to choose particular school'.

3.7 Classification summary

Random Forest AUC = 0.8011636 Random Forest importance variables: absences, Medu, health, reason, studytime.

Boosting AUC = 0.7820606 Boosting importance variables: failures, Medu,, absences, reason

SVM error rate = 0.2692308

Logistic regression AUC 0.7886

4 Regression

4.1 Model selection

4.2 Linear regression

4.3 Random Forests

4.4 ...

4.5 Regression summary