Project 4

Filip Axelsson & Alexander Crompton

2021-12-25

Introduction

The group consists of Filip Axelsson and Alexander Crompton. We have chosen to work with the Student Performance Data Set, since it's a familiar and interesting topic. The data-set consists of 395 observations and 33 variables, and gives us information about the students performance in math. The response variable, final grade "G3" is a numeric vector from 0-20, but we have divided it into two groups, "Pass" or "Fail" since it will be easier to interpret and we wanted to work with a logistic outcome variable, the new variable is called "grade". Note that is also consists first period grade "G1" and second period grade "G2" which is included in the explanatory variables. More examples of the explanatory variables are school, which indicates which school the student attend to, "GP"- Gabriel Pereira or "MS" - Mousinho da Silveira. The student's sex, age, address family size are also example on the explanatory variables. We have also study time, failures, school support and much more, however all of the variables doesn't necessary need to be include in the model which we will see later on. In *Table 1* a glimpse of the the data set is listed. A more detalied overview can be seen in the appendix.

Table 1: Glimpse over 5 obesrvations for a few features in the dataset.

school	sex	age	famsize	G1	G2	grade
GP	F	18	GT3	5	6	Fail
GP	F	17	GT3	5	5	Fail
GP	F	15	LE3	7	8	Pass
GP	F	15	GT3	15	14	Pass
GP	F	16	GT3	6	10	Pass

Choice of methods

There exist no rule that states which methods will always perform better than another. Therefore no conclusion about which method is best suited can be made before making the analysis, however due to the underlying structure of the data set, one can have a hunch of which methods might be suitable for the specific data set and proceed with those. Since SVM tends to preform well when you have a smaller data set with many input variables, this is one of the methods we will use to fit the data. The data set consists of many categorical data variables, which makes decision tree's another potentially good method to use. Since this method has many advantages, such as it being highly interpretive, it's automatic handling of multicolliniarity. There for is this also chosen as a method.

Fitting the models

The packages "caret", "gbm" and "kernlab" is used in the project in order to fit the different models. In order to train a gradient boosting tree or SVM we use the function "train()", in *Table 2* we can see each tuning parameter for the given method. More details about other arguments that can be used in the function "train()"

can be seen inn the appendix, *Table 6.* Note that for SVM, the kernel that is used is either polynomial kernel or radial basis function kernel. If we start with gradient boosting trees, we are creating a function "cv_gbm()" that takes in the arguments n_fold, nodes, and max_trees. These arguments describe the amount of folds that are used in the cross-validation, the amount nodes of each split and the amount of trees that are grown. The distribution is set to "multinomial" since we have a factor as response variable, this is also set by default from the package "gbm". The function "cv_svm()" is created to perform cross-validation for a support vector machine, with either polynomial kernel or radial basis function kernel. The function takes the arguments cv_folds,c,s,m,p. The arguments decide the kernel that is used, the amount of fold for cross-validation, and the tuning grid for the parameters. These function can be found in the appendix.

Table 2: Each methods tuning parameters

Method	Tuning parameters
Gradient boosting trees	n.trees (Boosting iterations) interaction.depth (Max Tree Depth) shrinkage (Shrinkage) n.minobsinnode (Min. Terminal Node Size)
SVM, polynomial kernel	degree (Polynomial Degree) scale (Scale) C (Cost)
SVM, radial basis function kernel	sigma (Sigma) C (Cost)

Gradient boosting trees

By using the function "cv_gbm()", we can easily test different amounts of M (the total numbers of trees to fit) and m (the deapth of each split). Note that the "n.minobsinnode" which specify the minimum amount of observation in each node has been set to 5 instead of 10. This is done because of the total number of observation is 395, which doesn't consider as a large data set and therefore is more suitable to allow fewer observations for each node rather than the default value in the function. It is also worth to note that the shrinkage parameter ("Weak-learner" parameter) is set 0.05 instead of 0.1, that is, each step the model takes is now smaller than the default. The result from the cross-validation with 10 folds can be analyzed in *Figure 1* where $m \in \{1, 5, 10, 20\}$ and $M \in \{100, 500\}$, which generated that the lowest produced one standard error was given when $m = 1$ and $M = 500$ which generated a error rate around 8.04%. In *Figure 2* we can analyze the 10 variables with the highest importance, where G2 (grade in second period) clearly has the highest importance. However this wasn't a surprise since an individual that has a passing grade in the previous period, will also be likely to pass in the next term. This logic can also motivate the relatively high importance in G1. Other variables that has a relative high importance is absences, the student's age, how many past classes the student has failed. A common theme for most of these variables is that it's a time consuming activity, so a hypothesis would be the less time a student have to study the higher the chance it is for the student to fail. For a more detailed list of each parameters importance, see *Table 7* in the appendix.

Figure 1: Cross-validation with m = 1,5,10,20 and M = 100,500

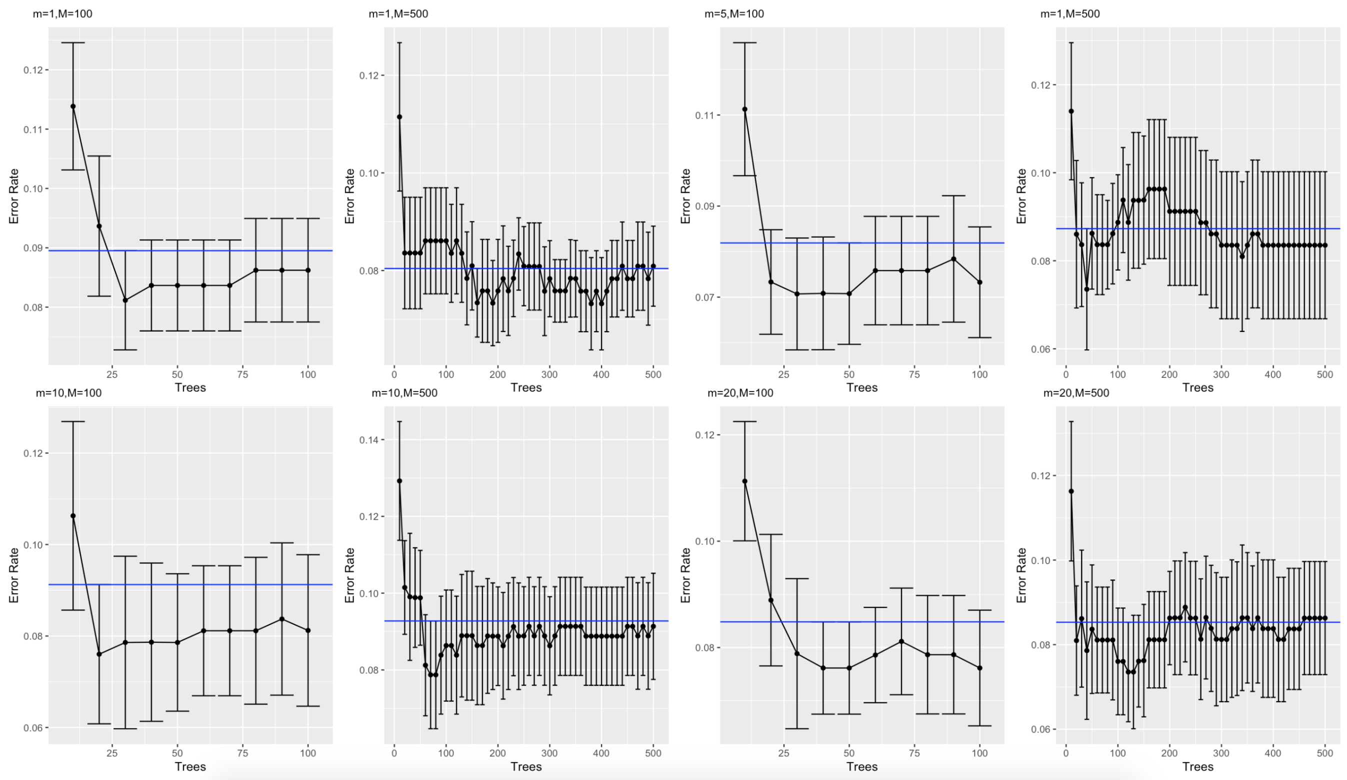
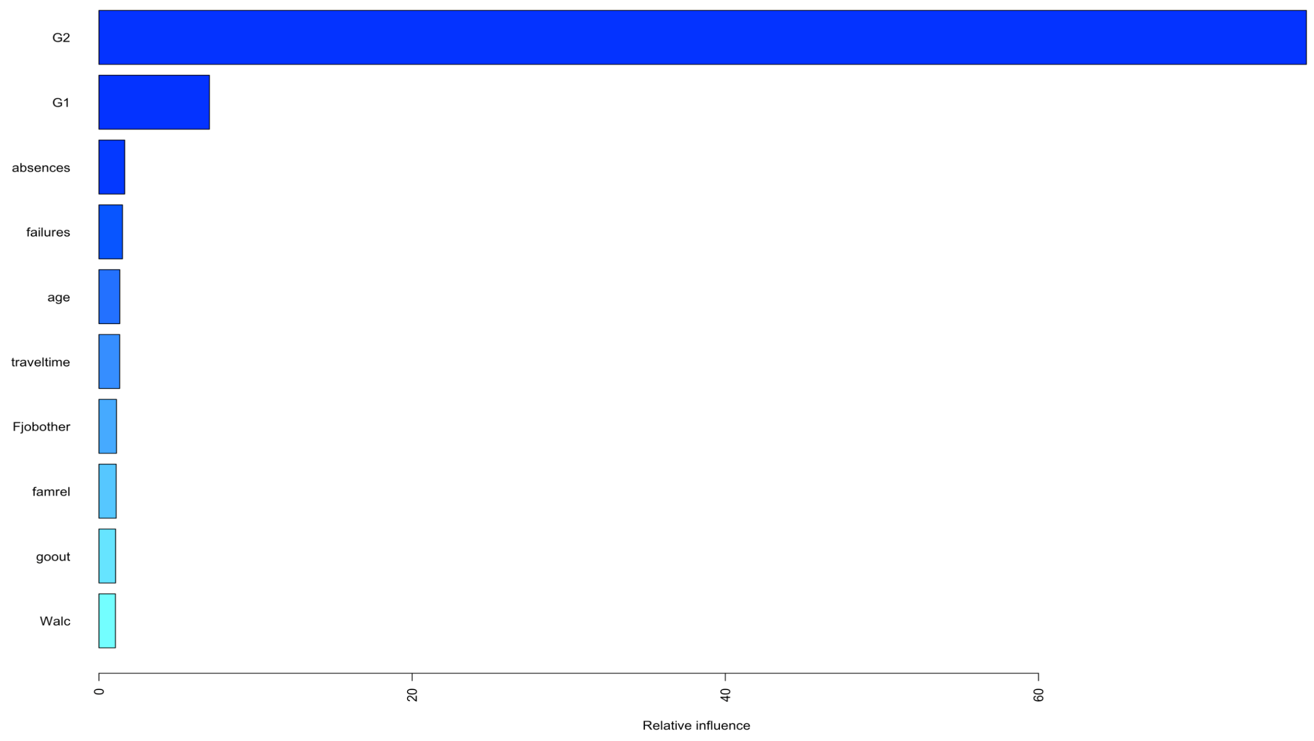


Figure 2: Variable importance, m = 1 and M = 500



Support vector machine

By using the function “cv__svm()” it’s easy tune SVM when using the polynomial kernel and radial basis function kernel. We can specify which kernel the function should use and then give a tuning grid which contains a range of scale,sigma and polynomail depending on the kernel and then preforme 10-fold cross-validation to optimize the model. The best model is then obtain by comparing each the one standard error. Analyzing *Figure 3* we can see for polynomial kernel, the best model according to the one standard error is obtained when using scale = 0.1, degree = 1, this generated a one standard error around 6.5%. In *Figure 4* we analyze SVM with radial basis function kernel and obtained the best model with sigma = 0.01, and that generated a one standard error around 11.6%. This gives us that when using the SVM with polynomial kernel we obtain the lowest error rate.

Figure 3: SVM, polynomial kernel

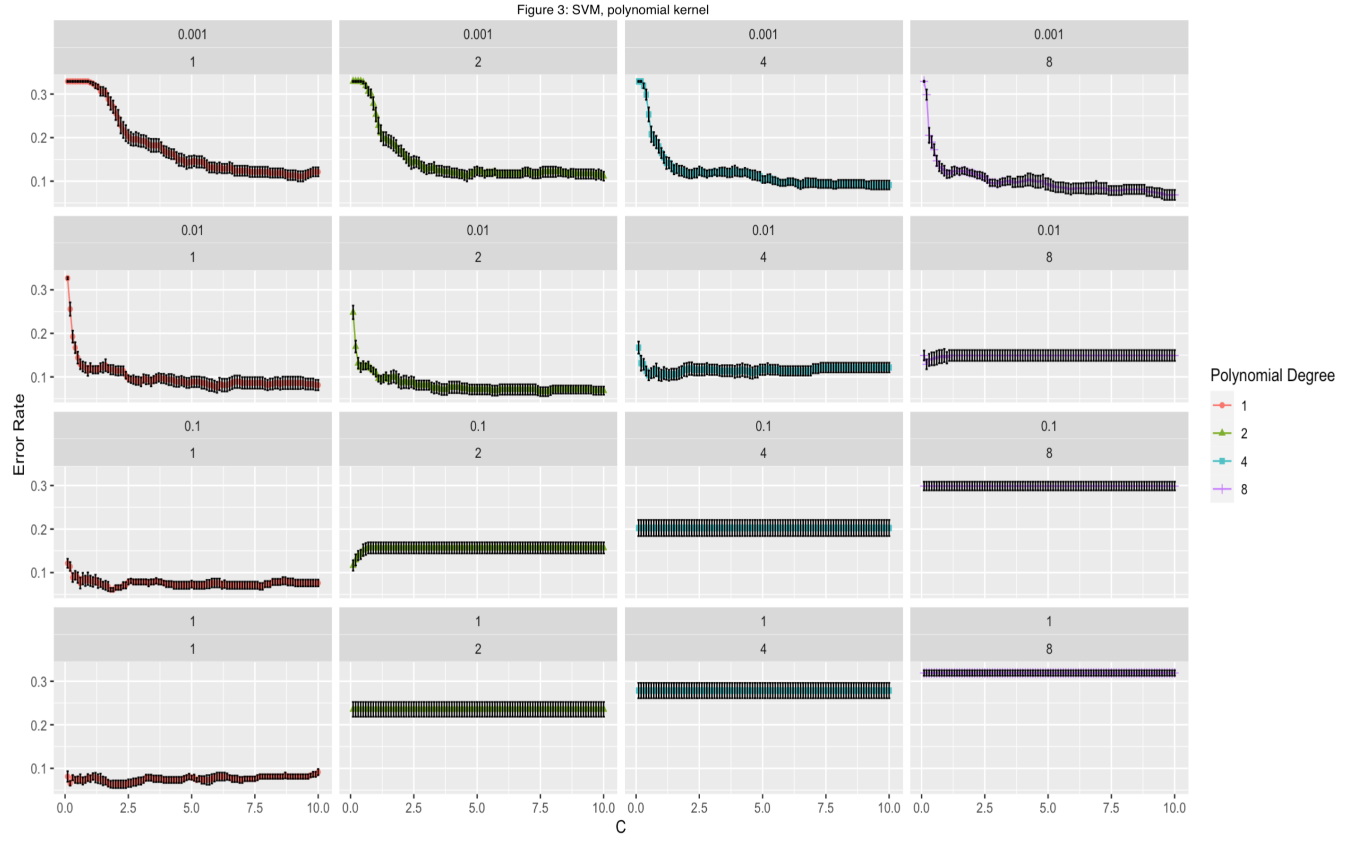
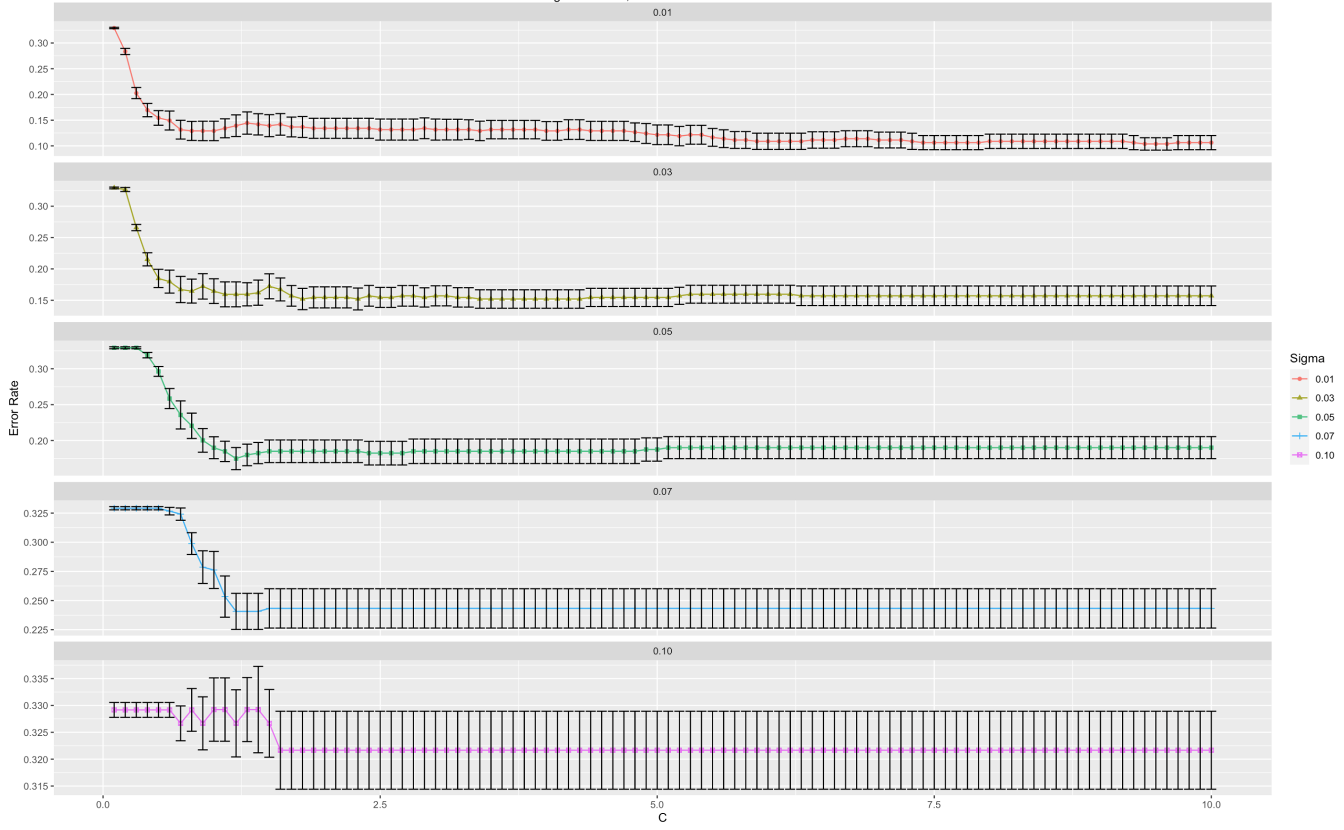


Figure 4: SVM, radial basis function kernel



Removing G1 & G2

Since G1 and G2 had very high variable importance, it's also in the intresset to see how the methods would preforme with out these variables. All the figures for the trained models can be find in the appendix, *Figure 6-8*, however the one standard error rate was significantly higher, it went from around 10% up to about 30%. Another difference was that SVM with polynomial kernel didn't generate the lowest error rate, in fact it had the highest, and gbm genereated the lowest.

Conclusion & Discussion

Intuitively the response variable G3 and the predictor variables G2,G1 are strongly correlating as they all measure student performance which are quite unlikely to differ greatly. When removing these variable from our models we could, rather unsurprisingly, find that our error rate increased significantly to about 30 per cent. However the reasoning behind building and using modeling without G2 and G1 is to find other variables that impacts grades besides previous grades. Interestingly we found that the gbm's importance for different is more balanced and focuses less on a single variable and more evenly balanced between a few variables.

The goout variable which measures the amount of time a student goes out with friends after school is one of the most important variables for the outcome model. A reason for it being included could be that it effects the amount of study time which can effect the final grade. Another important variable is the number of absences for a student. Obviously this variable indicates if a student has missed many occasions to learn in school but also this variable can indicate how committed a student is to pass a class. Age is another factor of importance. Usually older students are more committed to school which is an important factor to get a passing grade. Past failures is the most important measure for the model without G2 and G1. Previous failures obviously indicate struggles and difficulties in the subject which is an important factor to show how likely the student is to pass. Other factors that are important is famsup and schoolsup variable which indicates if the student gets educational support from the family or general support respectively which can help the student to pass if they receive help but also give the student additional motivation to pass.

As we can see from *Table 3* the SVM with the polynomial kernel is found to be the best performing method with the lowest rate of error for the data. However if we neglect the variables with previous grading for the students which we can see in *Table 4* we can see the gbm is the best method. So why did we get two different methods when studying the response with or without the G1 and G2 variables. A reason for this could be that SVM draws a boundary and can make a more fitting line when using highly influential and correlating variables for the response variable while GBM works to remove multicolliniarity. A motivation to why gbm is a better method when removing G1 and G2 is that it is a more robust method and it generally works well, it finds faults and improves the model step by step.

A general positive feature to the gbm compared to the SVM is that the run time is faster and results are provided faster. The reason for SVM's long run time could potentially be due to using expand.grid rather than tuneLength which depends on more parameters.

Instead of interpret this problem as a classification problem, we could have seen it as a regression problem by using the final grade "G3" as the numeric vector from 0-20 and used methods such as linear regression, lasso regression or ridge regression.

Table 3: Each methods best models one-standard-error

Method	One standard error
Gradient Boosting trees, m = 1, M = 500	0.0804145
SVM with polynomial kernel, degree = 1, scale = 0.1	0.0647334
SVM with radial basis kernel, sigma = 0.01	0.1158374

Table 4: Each methods best models one-standard-error, without G1 and G2

Method	One standard error
Gradient Boosting trees, m = 1, M = 500	0.2824796
SVM with polynomial kernel, degree = 1, scale = 0.01	0.3115991
SVM with radial basis kernel, sigma = 0.01	0.3029330

Appendix

Table 5: Feature description

Feature	Description
school	Student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
sex	Student's sex (binary: 'F' - female or 'M' - male)
age	Student's age (numeric: from 15 to 22)
adress	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 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
Fedu	Father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
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')
reason	Reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
guardian	Student's guardian (nominal: 'mother', 'father' or 'other')
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)
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 (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)
romantic	With a romantic relationship (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)
goout	Going out with friends (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)
G1	First period grade (numeric: from 0 to 20)
G2	Second period grade (numeric: from 0 to 20)
G3	Final grade (numeric: from 0 to 20, output target)
grade	Indicates if the student has passed or failed the course (Binary: Pass or Fail)

Table 6: List over parameters in the function train()

Parameter	Information
Method	Specifying which classification or regression model to use.
trControl	A list of values that define how this function acts
preProcess	A string vector that defines a pre-processing of the predictor data, eg "scale", and "center"
tuneGrid	A data frame with possible tuning values.

Figure 5: Cross-validation with m = 1,5,10,20 and M = 100,500, without G2 & G1

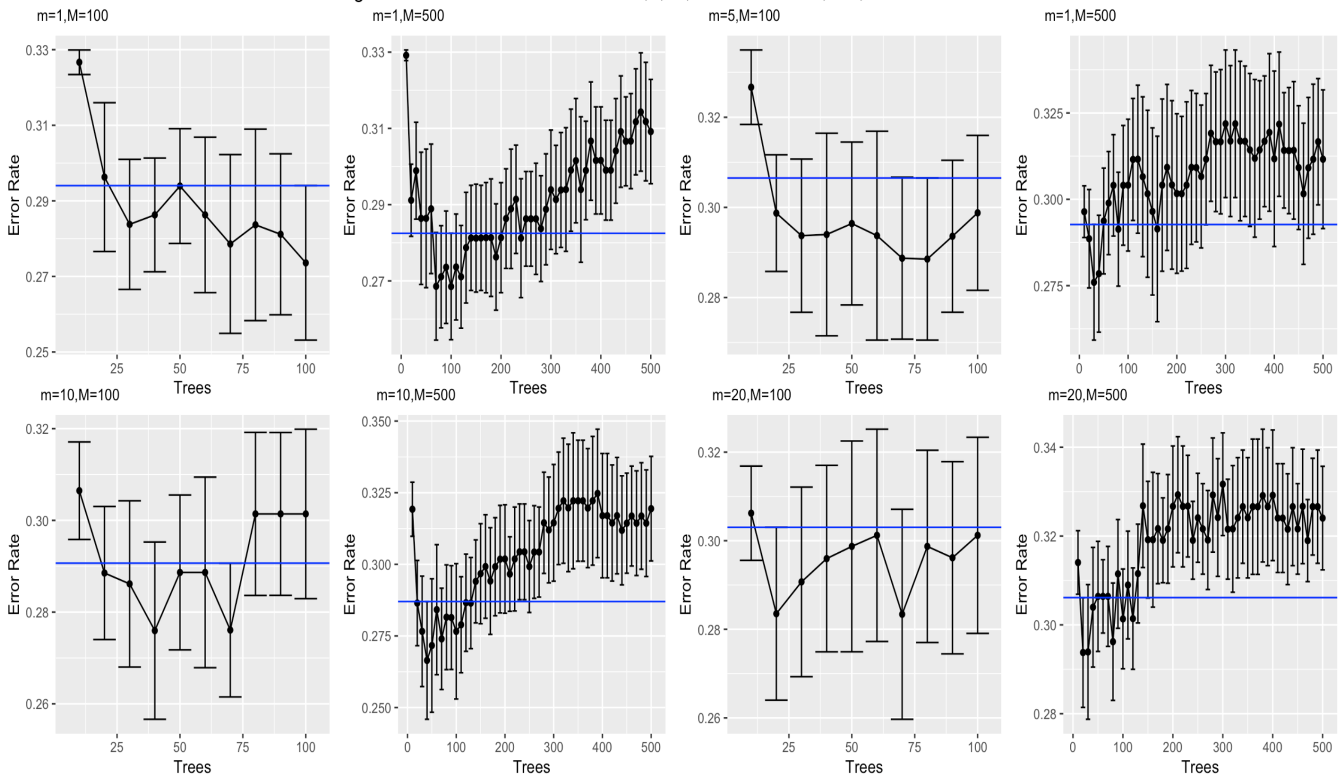


Figure 6: Variable importance, m = 1 and M = 500, without G1 & G2

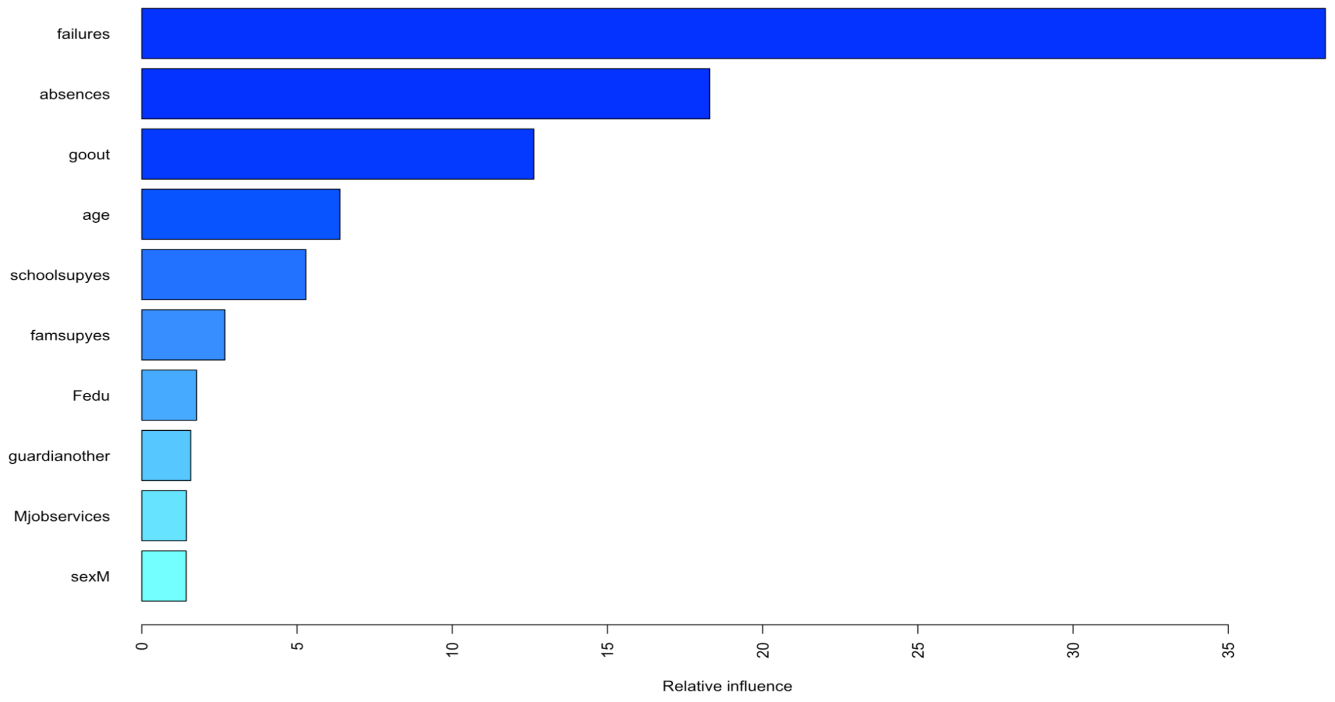


Table 7: List of each variable importance for gradient boosting trees with m = 1 and M = 500

	var	rel.inf
	<chr>	<dbl>
G2	G2	77.10919485
G1	G1	7.05726194
absences	absences	1.64516510
failures	failures	1.50551194
age	age	1.33259648
traveltime	traveltime	1.32654252
Fjobother	Fjobother	1.12124080
famrel	famrel	1.10167167
goout	goout	1.06280242
Walc	Walc	1.05334950
Fedu	Fedu	0.96897698
schoolsupyes	schoolsupyes	0.89471030
Mjobother	Mjobother	0.54252268
romanticyes	romanticyes	0.40214142
freetime	freetime	0.36833595
health	health	0.35240192
activitiesyes	activitiesyes	0.34804204
Medu	Medu	0.34082455
studytime	studytime	0.29395668
reasonother	reasonother	0.23152987
famsizeLE3	famsizeLE3	0.17207850
reasonhome	reasonhome	0.13154032
sexM	sexM	0.11009859
Mjobteacher	Mjobteacher	0.10471366
famsupyes	famsupyes	0.08699153
Dalc	Dalc	0.08685752
Fjobservices	Fjobservices	0.06111085
reasonreputation	reasonreputation	0.05673215
PstatusT	PstatusT	0.05552738
schoolMS	schoolMS	0.04099025
guardianmother	guardianmother	0.03457964
addressU	addressU	0.00000000
Mjobhealth	Mjobhealth	0.00000000
Mjobservices	Mjobservices	0.00000000
Fjobhealth	Fjobhealth	0.00000000
Fjobteacher	Fjobteacher	0.00000000
guardianother	guardianother	0.00000000
paidyes	paidyes	0.00000000
nurseryyes	nurseryyes	0.00000000
higheryes	higheryes	0.00000000
internetyes	internetyes	0.00000000

Figure 7: SVM with polynomial kernel, without G1 & G2

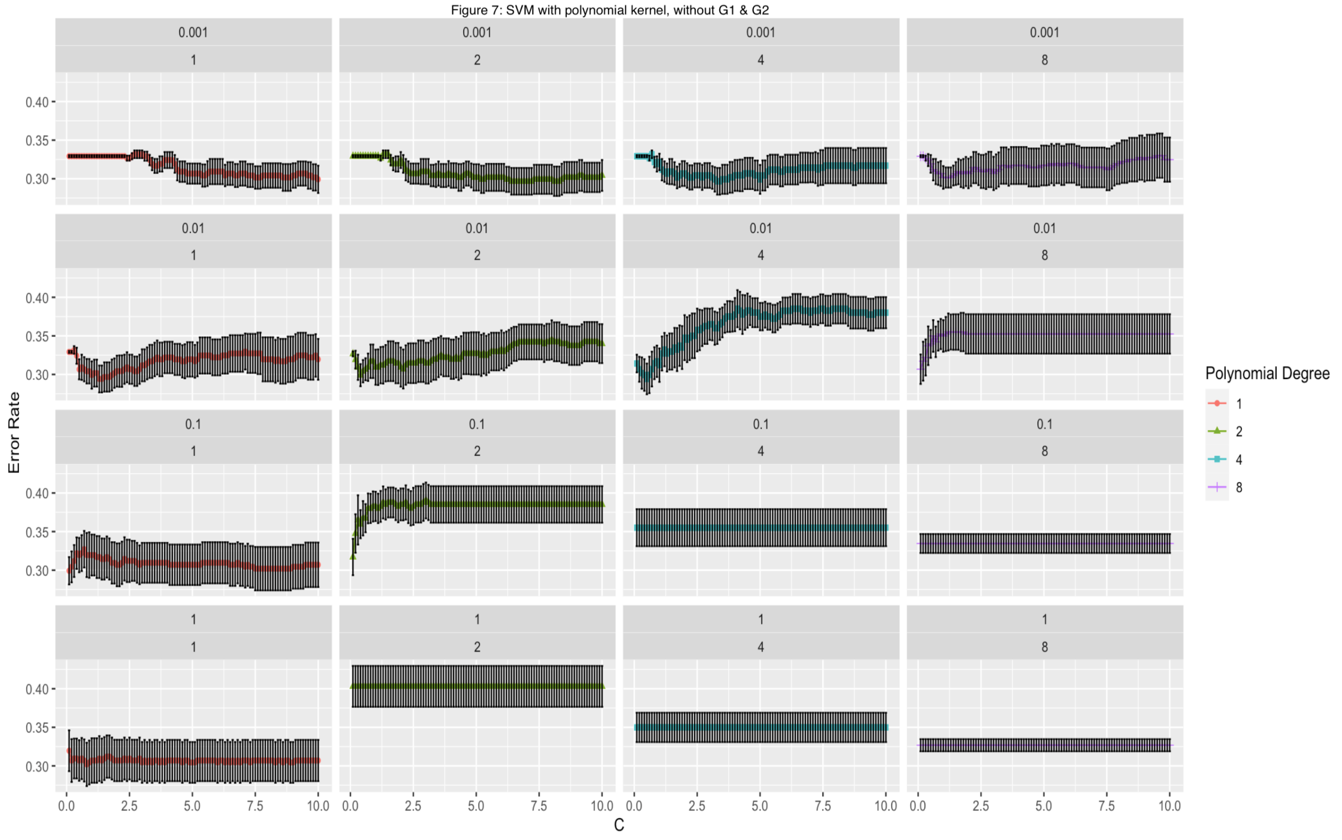
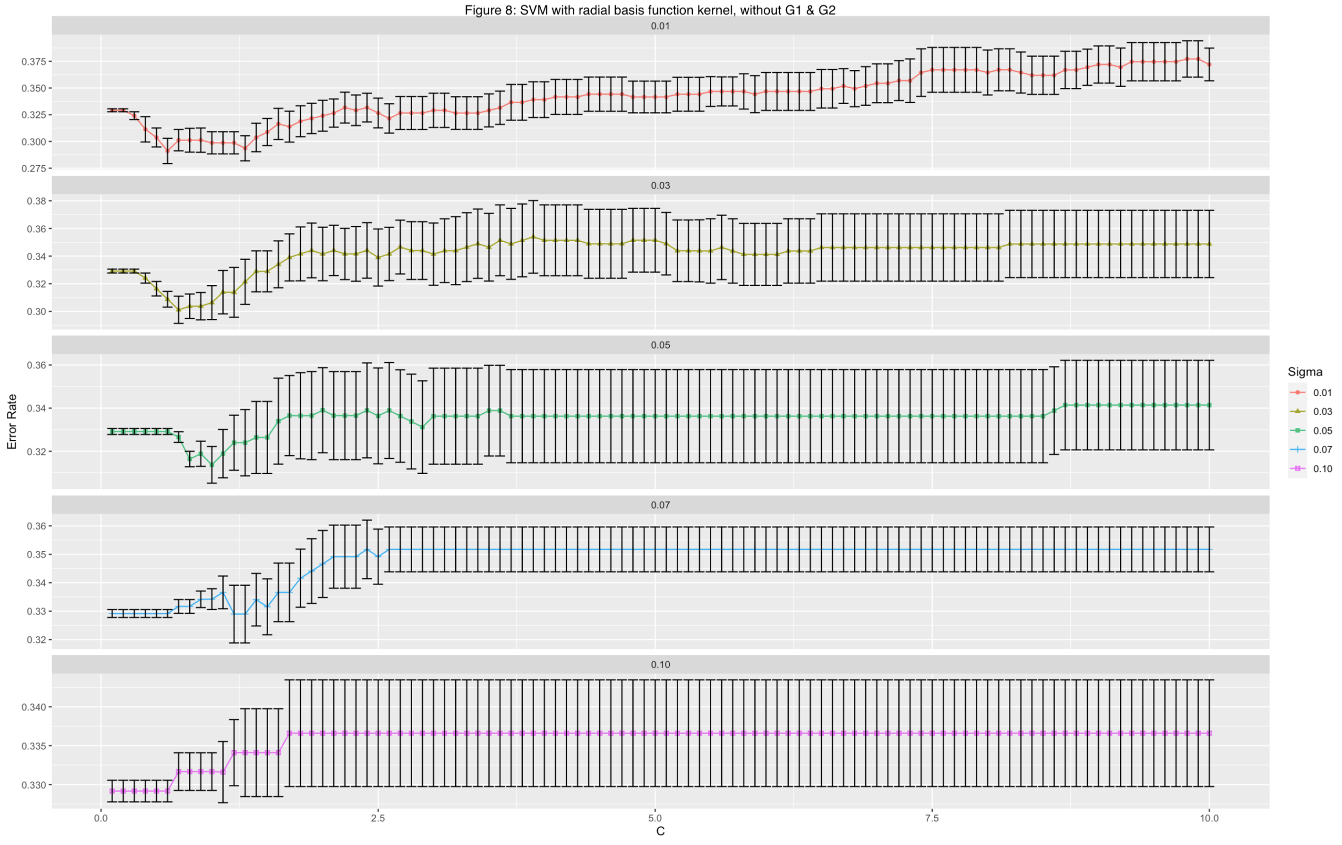


Figure 8: SVM with radial basis function kernel, without G1 & G2



Code

```r
# reading in all the neccessarry libraries
library(tidyverse) # type of coding
library(kableExtra) # making tables
library(cowplot) # gridplot
library(caret) # train()
library(gbm) # gradient boosting trees
library(kernlab) # svm
data=read.table("../Statistisk inlärning/student/student-mat.csv",sep=";",header=TRUE)
data <- data %>%
  mutate(grade = case_when(G3 < 10 ~ "Fail", # creating a variable that return pass or fail
                           G3 < 21 ~ "Pass",
                            )) %>%
  select(-G3)
```

```r
# creating a gradient boosting tree function,
# can specify, fold, nodes (m), amount of trees (M)
# and which dataset is used
cv_gbm <- function(cv_folds,nodes,max_trees, used_data){

# Decide how we should train the model, that is with cv.
train.control <- trainControl(method = "cv", number = cv_folds)

parameter_grid <- expand.grid(
 interaction.depth = nodes,
 n.trees = c(seq(from = 10 , to =max_trees, by = 10)), # Gives us M = 10,20,30,... etc
 shrinkage = 0.05, # Using 0.05 instead of 0.1 which is defualt in the function gbm()
 n.minobsinnode = 5 # Using 5 instead of 10 which is defualt in the function gbm()
                                        )
# Train the model
cv_model <- train(factor(grade) ~.,
                 data = used_data,
                 method = "gbm",
                 trControl = train.control,
                 verbose = FALSE, # Doenst print out every model
                 distribution = "bernoulli",
                 metric = "Accuracy",# Since we want the cv error rate we need to add the variable
                 tuneGrid = parameter_grid
                 )
return(cv_model)
}



# Creating a function for svm,
# SVM, m = svmRadial or m = svmPoly
# tuning parameters c = Cost, s = Sigma, d = degree, sc = Scale

cv_svm <- function(m,cv_folds,c,s,d,sc,used_data){

# Decide how we should train the model, that is with cv.
train_control <- trainControl(method = "cv", number = cv_folds)
#Traning the model for the choosen kernel

# Radial basis function kernel
if("svmRadial" == m){
svm <- train(factor(grade)~.,  # specifying which model, eg which variable is the respone
            data = used_data, # specifying the data
            method = m,  # specifying the method, we are using the radial basis kernel
            trControl = train_control, # Applying cv
            preProcess = c("center","scale"),
            tuneGrid = expand.grid(C = c, sigma = s)) # Specifying how the tuning parameters
}

# Polynonimial kernel
if("svmPoly" == m){
svm <- train(factor(grade)~.,  # specifying which model, eg which variable is the respone
            data = used_data, # specifying the data
            method = m,  # specifying the method, we are using the radial basis kernel
            trControl = train_control, # Applying cv
            preProcess = c("center","scale"),
            tuneGrid = expand.grid(degree = d,scale = sc,C = c)) # Specifying how the tuning parameters
}

return(svm)
}
```

```r
# creating a function to plot the error bars for the fitted trees

plot_cv_error_tree <- function(cv_file){
 cv_file[["results"]][["Accuracy"]] <- (1-cv_file[["results"]][["Accuracy"]]) # convert the Accuracy to
 cv_file %>%
   ggplot() +
   geom_errorbar(
   aes(ymin =cv_file$results$Accuracy-(cv_file$results$AccuracySD/sqrt(10)),
       ymax = cv_file$results$Accuracy+(cv_file$results$AccuracySD/sqrt(10)))) +
  scale_y_continuous("Error Rate") +
  scale_x_continuous("Trees") +
  geom_hline(yintercept = min(cv_file$results$Accuracy+(cv_file$results$AccuracySD/sqrt(10))), col ="bl
}


# Fitting the model to different M and m
# M = 100 and 500, m = 1
cv_1_100 <- cv_gbm(10,1,100,used_data = data)
cv_1_500 <- cv_gbm(10,1,500,used_data = data)

# M = 100 and 500, m = 5
cv_5_100<- cv_gbm(10,5,100,used_data = data)
cv_5_500<- cv_gbm(10,5,500,used_data = data)

# M = 100 and 500, m = 10
cv_10_100<- cv_gbm(10,10,100,used_data = data)
cv_10_500<- cv_gbm(10,10,500,used_data = data)

# M = 100 and 500, m = 20
```

```r
cv_20_100<- cv_gbm(10,20,100,used_data = data)
cv_20_500<- cv_gbm(10,20,500,used_data = data)


p1 <- plot_cv_error_tree(cv_1_100)
p2 <- plot_cv_error_tree(cv_1_500)

p3 <- plot_cv_error_tree(cv_5_100)
p4 <- plot_cv_error_tree(cv_5_500)

p5 <- plot_cv_error_tree(cv_10_100)
p6 <- plot_cv_error_tree(cv_10_500)

p7 <- plot_cv_error_tree(cv_20_100)
p8 <- plot_cv_error_tree(cv_20_500)
```

```r
# creates a function to plot the error bars for svm,
# differnt for each kernel
plot_cv_error_svm <- function(cv_file,m){
 cv_file[["results"]][["Accuracy"]] <- (1-cv_file[["results"]][["Accuracy"]]) # convert the Accuracy to

 if("svmPoly" == m){
   plot <- cv_file %>%
     ggplot() +
     geom_errorbar(
       aes(ymin =cv_file[["results"]][["Accuracy"]]-(cv_file[["results"]][["AccuracySD"]]/sqrt(10)),
           ymax =cv_file[["results"]][["Accuracy"]]+(cv_file[["results"]][["AccuracySD"]]/sqrt(10)))) +
   scale_y_continuous("Error Rate") +
   scale_x_continuous("C") +
   facet_wrap(.~scale+degree, ncol = 4) # split the graphs to seprat degrees
 }

 if("svmRadial" == m){
   plot <- cv_file %>%
     ggplot() +
     geom_errorbar(
       aes(ymin =cv_file[["results"]][["Accuracy"]]-(cv_file[["results"]][["AccuracySD"]]/sqrt(10)),
       ymax =cv_file[["results"]][["Accuracy"]]+(cv_file[["results"]][["AccuracySD"]]/sqrt(10)))) +
   scale_y_continuous("Error Rate") +
   scale_x_continuous("C") +
   facet_wrap(.~sigma, ncol = 1, scale = "free_y") # split the graphs to seprat sigmas.
 }
 return(plot)
}

radial <- cv_svm("svmRadial",10,seq(0.1,10,0.1),c(0.01,0.03,0.05,0.07,0.1),used_data=data)

poly <- cv_svm("svmPoly",10,seq(0.1,10,0.1),0,c(1,2,4,8),c(0.001,0.01,0.1,1),used_data=data)


plot_cv_error_svm(radial,"svmRadial")
plot_cv_error_svm(poly,"svmPoly")
```

```r
# Creating data without G1 and G2
# Repating the same process as before
data2 <- data %>%
  select(-G1,-G2)

radial2 <- cv_svm("svmRadial",10,seq(0.1,10,0.1),c(0.01,0.03,0.05,0.07,0.1),used_data=data2)
poly2 <- cv_svm("svmPoly",10,seq(0.1,10,0.1),0,c(1,2,4,8),c(0.001,0.01,0.1,1),used_data=data2)

plot_cv_error_svm(radial2,"svmRadial")
plot_cv_error_svm(poly2,"svmPoly")

# Fitting the model to different M and m
# M = 100 and 500, m = 1
cv2_1_100 <- cv_gbm(10,1,100,used_data = data2)
cv2_1_500 <- cv_gbm(10,1,500,used_data = data2)

# M = 100 and 500, m = 5
cv2_5_100<- cv_gbm(10,5,100,used_data = data2)
cv2_5_500<- cv_gbm(10,5,500,used_data = data2)

# M = 100 and 500, m = 10
cv2_10_100<- cv_gbm(10,10,100,used_data = data2)
cv2_10_500<- cv_gbm(10,10,500,used_data = data2)

# M = 100 and 500, m = 20
cv2_20_100<- cv_gbm(10,20,100,used_data = data2)
cv2_20_500<- cv_gbm(10,20,500,used_data = data2)


p12 <- plot_cv_error_tree(cv2_1_100)
p22 <- plot_cv_error_tree(cv2_1_500)

p32 <- plot_cv_error_tree(cv2_5_100)
p42 <- plot_cv_error_tree(cv2_5_500)

p52 <- plot_cv_error_tree(cv2_10_100)
p62 <- plot_cv_error_tree(cv2_10_500)

p72 <- plot_cv_error_tree(cv2_20_100)
p82 <- plot_cv_error_tree(cv2_20_500)
```

References

- "The Elements of Statistical Learning: Data Mining, Inference, and Prediction" 2nd Ed. by Hastie, Tibshirani & Friedman, Springer 2009
- "An introduction to statistical learning" Vol. 112. by James, G., Witten, D., Hastie, T., & Tibshirani, R. Springer 2013
- http://topepo.github.io/caret/train-models-by-tag.html - describes the function train()
- https://en.wikipedia.org/wiki/Academic_grading_in_Portugal - Gradingsystem in Portugal
- https://www.studyineurope.eu/study-in-portugal/grades - Gradingsystem in Portugal