Predicting Students' Performance in Math

Wei-Chen (Eric) Wang

5/6/2020

Introduction

Education inequality has become a serious issue in recent years. Race, ethnicity, wealth, sex and many other factors play a significant role in the inequalities of the education system, in which privileged students tend to perform better academically. Students with less education resources receive less help, and have to allocate more time outside of school, such as having a part-time job. The difficulty of predicting students' performances makes it even harder for school faculties to identify and assist students who are falling behind.

This project explores potential factors that can impact students' academic performance in math. The dataset is obtained from the machine learning repository at the University of California, Irvine. The data are collected using school reports and questionnaires from two Portuguese secondary schools. Attributes include grades, demographic, social and other school related features.

In this report, we used multiple machine learning models, and we evaluate each model by plotting their predicted values against the true valuem and compare their mean squared errors. In particular, we run the following models:

- Ordinary least squares (OLS) linear regression
- Least absolute shrinkage and selection operator (LASSO) regression
- Ridge regression
- Decision tree
- Bootstrap aggregating (Bagging)
- Random Forest
- Boosting
- Extreme Gradient Boosting (XGboost)
- Neural Networks
- K-nearest neighbors (KNN)

Summary Statistics

Variables

The response variable in this study is the student final grade `G3` (numeric: from `0` to `20`). There are multiple potential predictor variables, but only the essential ones were listed below in alphabetical order.

Variables	Type	Description	Possible.Values
absences	numeric	number of school absences	from 0 to 93
activities	binary	extracurricular activities	yes or no
age	numeric	student's age	15 to 22
failures	numeric	number of past class failures	n for 1 <= n < 3, 3 for n >= 3
Fedu	numeric	father's education	0 for none, 1 for primary education (4th grade), 2 for 5th to 9th grade, 3 for secondary education, 4 for higher education
G1	numeric	first period grade	from 0 to 20
G2	numeric	second period grade	from 0 to 20
goout	numeric	going out with friends	from 1 for very low to 5 for very high
Medu	numeric	mother's education	0 for none, 1 for primary education (4th grade), 2 for 5th to 9th grade, 3 for secondary education, 4 for higher education
studytime	numeric	weekly study time	1 for less than 2 hours, 2 for 2 to 5 hours, 3 for 5 to 10 hours, 4 for more than 10 hours

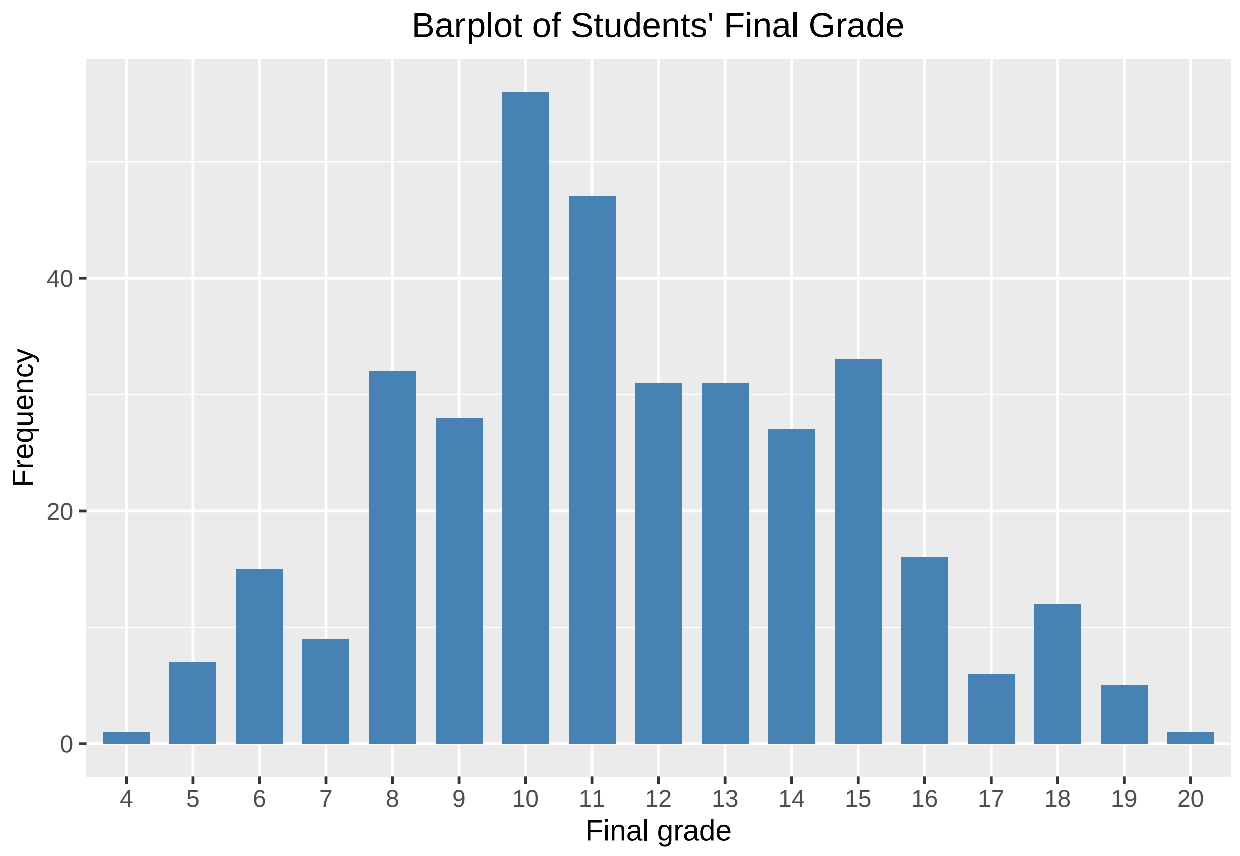
Statistics

The bar plots below show the distribution of the predictor variable `G3`, and four predictor variables `absences`, `failures`, `goout`, `studytime`, which are expected to have higher influence on the final grade.

```r
# Import data and extract required information
data = read.csv('student-mat.csv', sep = ';')
data = data[data$G3 > 0, ]
df = data.frame('G3' = data$G3,
                'absences' = data$absences,
                'failures' = data$failures,
                'goout' = data$goout,
                'studytime' = data$studytime)
```

Final Grade (G3)

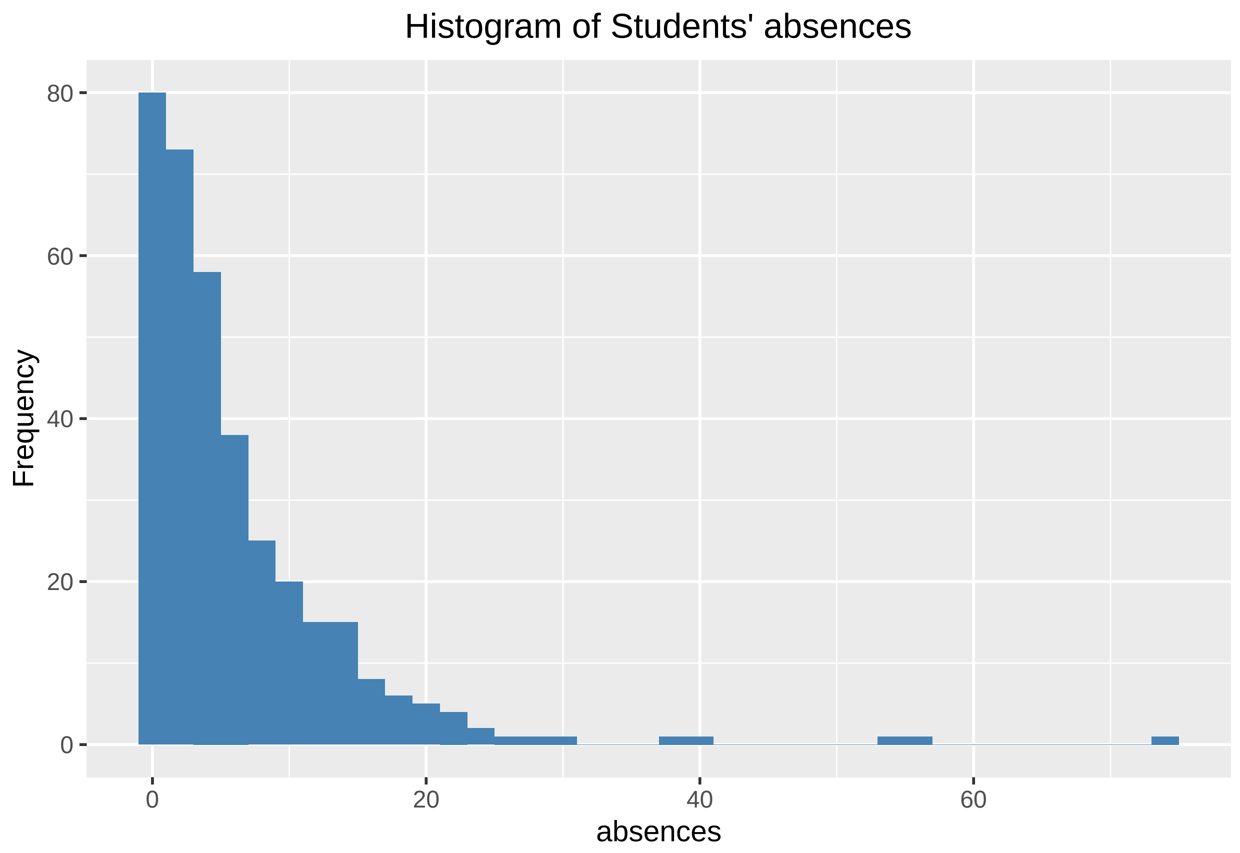
```r
ggplot(df, aes(x=factor(G3)))+
  geom_bar(stat="count", width=0.7, fill="steelblue") +
  ggtitle("Barplot of Students' Final Grade") +
  xlab('Final grade') +
  ylab("Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



Student final grades range from 4 to 20 and roughly follow a normal distribution.

Absences

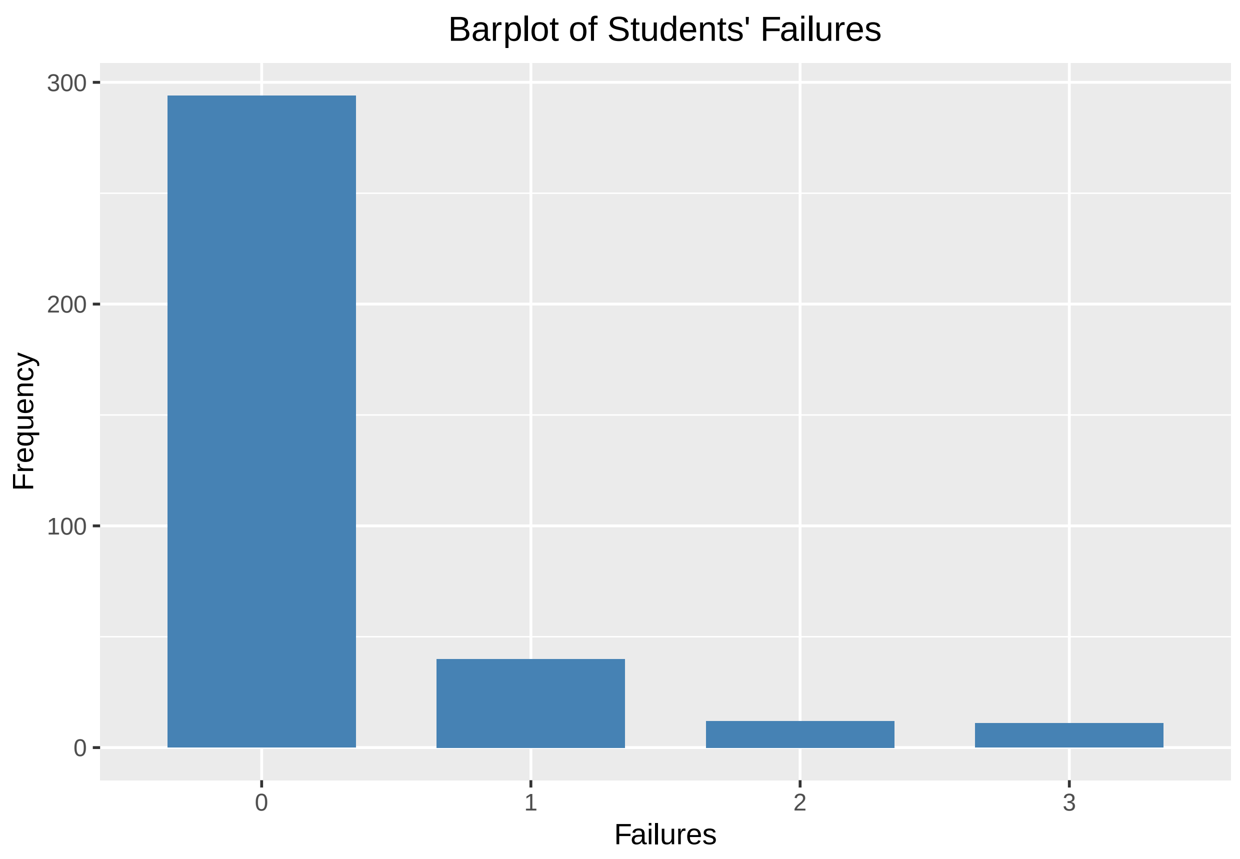
```r
ggplot(df, aes(x=absences)) +
  geom_histogram(binwidth = 2, fill='steelblue') +
  ggtitle("Histogram of Students' absences") +
  xlab('absences') +
  ylab("Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



While most students had less than 10 absences, there existed a considerable amount of outliers. Interestingly, the individual with the most absences (93) does not perform significantly worse than others.

Failures

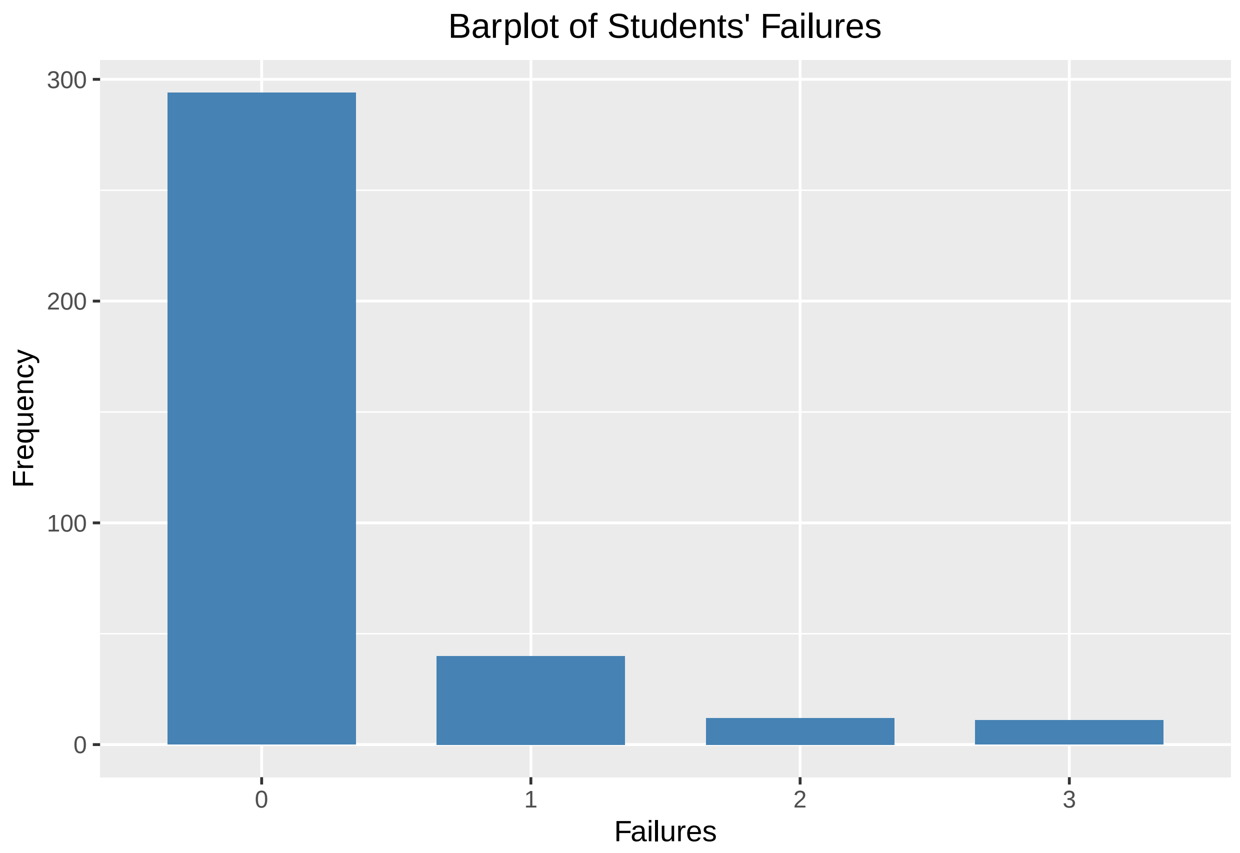
```r
ggplot(df, aes(x=factor(failures)))+
  geom_bar(stat="count", width=0.7, fill="steelblue") +
  ggtitle("Barplot of Students' Failures") +
  xlab('Failures') +
  ylab("Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



The majority of students did not fail even once in the past. Recall that `3` represents students who failed 3 or more classes in the past, so the number of failed classes are unknown.

Go outs

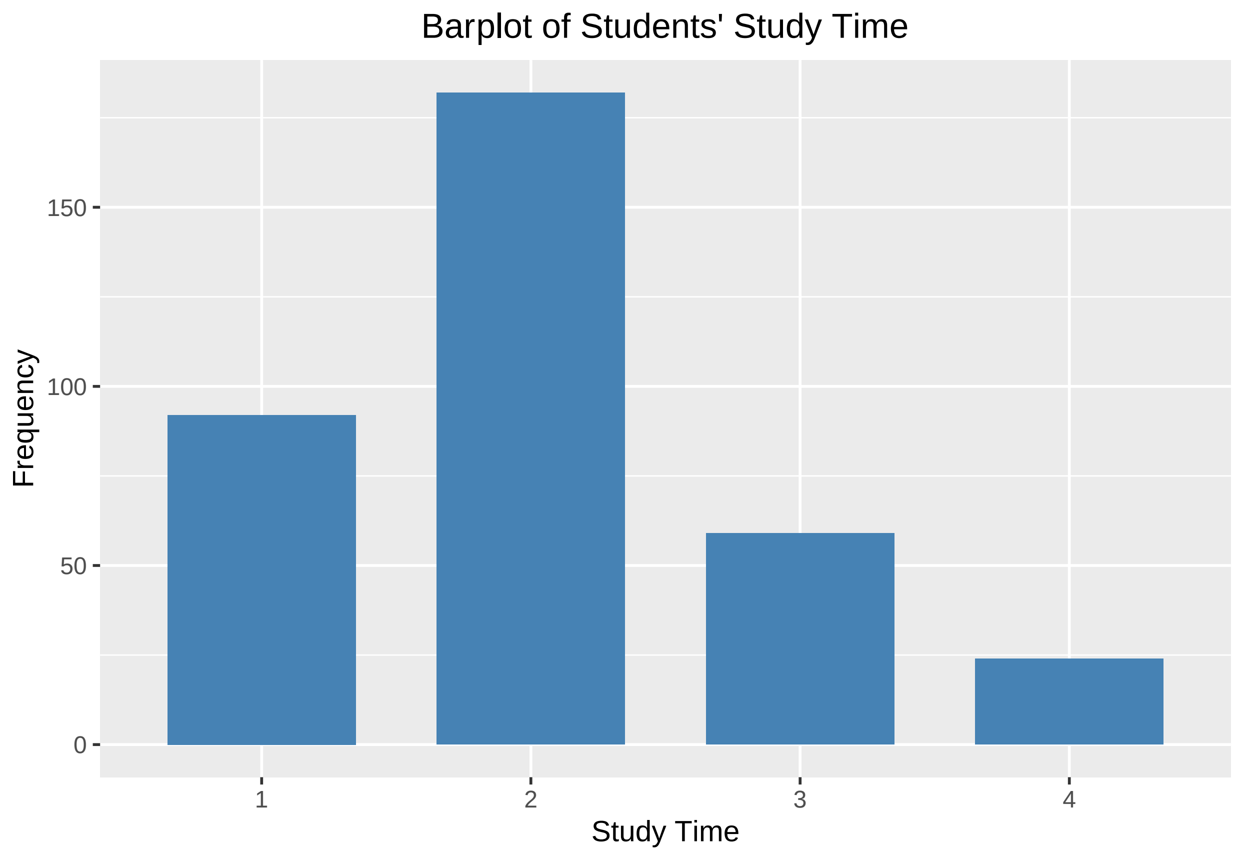
```r
ggplot(df, aes(x=factor(failures)))+
  geom_bar(stat="count", width=0.7, fill="steelblue") +
  ggtitle("Barplot of Students' Failures") +
  xlab('Failures') +
  ylab("Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



The frequency of going out seem to be normally distributed. Recall that `goout` ranges from `1` (very low) to `5` (very high).

Study Time

```r
ggplot(df, aes(x=factor(studytime)))+
  geom_bar(stat="count", width=0.7, fill="steelblue") +
  ggtitle("Barplot of Students' Study Time") +
  xlab('Study Time') +
  ylab("Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



Most students spend 2 to 5 hours weekly. Only about 20% of students spend more than 5 hours weekly.

Correlation

Variable	Correlation.with.G3
absences	-0.213
failures	-0.294
goout	-0.177
studytime	0.127

The above shows the correlation between the four variables and the final grade. None of the results should be surprising, as we learn that:

- More absences, failures in past courses, and time spent going out with friends, can all lead to a decrease in final grade.
- More time spent on studying can lead to an increase in final grade.

Note that the magnitudes of the four correlations are relatively small, which implies that they do not neccessarily have a huge impact on students' final grade.

Linear Regression

Model 1

Intuitively, students' grades in their first and second period grade have significant impact on their final grade. If a student performs well in the beginning and middle of the semester, he is likely to receive a good final grade. We run the model:

$$Y = \beta_0 + \beta_1 G_1 + \beta_2 G_2 + \epsilon$$

where Y is the final grade in math, G_1 and G_2 are the first and second period grades, ϵ is the error term, and β_i are the coefficients.

```
model1 = lm(G3 ~ G1 + G2, data = data)
summary(model1)$adj.r.squared
```

```
## [1] 0.9343245
```

Even though the adjusted R^2 is high, there exists a collineariy problem because the first and second period grades directly count towards part of the final grade. Furthermore, this project is more interested in how other student factors, such as the time they spent on studying, can impact their final grade. In many cases, the first and second period grades cannot be obtained before making a prediction. Therefore, we will try other models to see if a relatively good model can be built without using `G1` and `G2`.

Model 2

Among all the parameters, we believe that the time students spend on studying and the number of times they failed previous courses have the most significant impact on students' final grade. Therefore, we run the model:

$$Y = \beta_0 + \beta_1 S + \beta_2 F + \epsilon$$

where S is the time students spent on studying, and F is the number of times they failed the course.

```r
model2 = lm(G3 ~ studytime + failures, data = data)
summary(model2)$adj.r.squared
```

```
## [1] 0.08913735
```

Unfortunately, the adjusted R^2 indicates that the two variables do not explain much about the final grade. Therefore, we consider including more variables in the following models.

Model 3

Even though the second model yields a low adjusted R^2 squared, the second variable is significant at the level of $\alpha = 0.05$. Therefore, we try the model with only one predictor, namely:

$$Y = \beta_0 + \beta_2 F + \epsilon$$

```r
model3 = lm(G3 ~ failures, data = data)
summary(model3)$adj.r.squared
```

```
## [1] 0.0837629
```

Here, we see that the adjusted R^2 increased slightly. It implies an insignificant relationship between how much time students spend studying and their final grade. Therefore, we include F in future models, while excluding S in most cases.

Model 4

We include more variables to create a more complicated model:

$$Y = \beta_0 + \beta_1 F + \beta_2 ME + \beta_3 FE + \beta_4 G + \beta_5 T + \epsilon$$

where ME is the mother's education, FE is the father's education, G is the frequency of going out with friends, and T is the travel time from home to school.

```r
model4 = lm(G3 ~ failures + Medu + Fedu + goout + age + traveltime, data=data)
summary(model4)$adj.r.squared
```

```
## [1] 0.1152485
```

While the adjusted R^2 increases slightly, it is still far from desired. We continue to explore other variables that may have an impact on the final grade.

Model 5

Using a full model with all x variables (excluding $G1$, $G2$) and the step function, we try the following regression:

$$Y = \beta_0 + \beta_1 S + \beta_2 A + \beta_3 FS + \beta_4 ME + \beta_5 T + \beta_6 F + \beta_7 H + \beta_8 G + \epsilon$$

where S is the dummy variable for sex (1 for male, 0 for female), A for age, FS is the dummy variable for family size (1 for family size less than or equal to 3, 0 otherwise), H is the dummy variable for wanting to take higher education (1 for yes, 0 for no$).

```
model5 = lm(G3 ~ school + sex + Mjob + Fjob + studytime + failures +
    schoolsup + famsup + paid + internet + goout + health + absences, data = data)
summary(model5)$adj.r.squared
```

```
## [1] 0.2773179
```

With more variables, the adjusted R^2 increased to 0.277, which is the highest among the models we tried without including `G1` and `G2`. However, as the adjusted R^2 is still far from desired, we will consider adding `G1` and `G2` in future models if appropriate.

Best model

The results above show that it is nearly impossible to obtain a good model without including `G1` and `G2`. As a result, we first include some predictors, and use the step function obtain a more concise model.

```
model = lm(G3 ~ absences + activities + age + failures + Fedu +
             goout + Medu + studytime + G1 + G2, data)
model = step(model, trace = 0)
round(coef(model), 3)
```

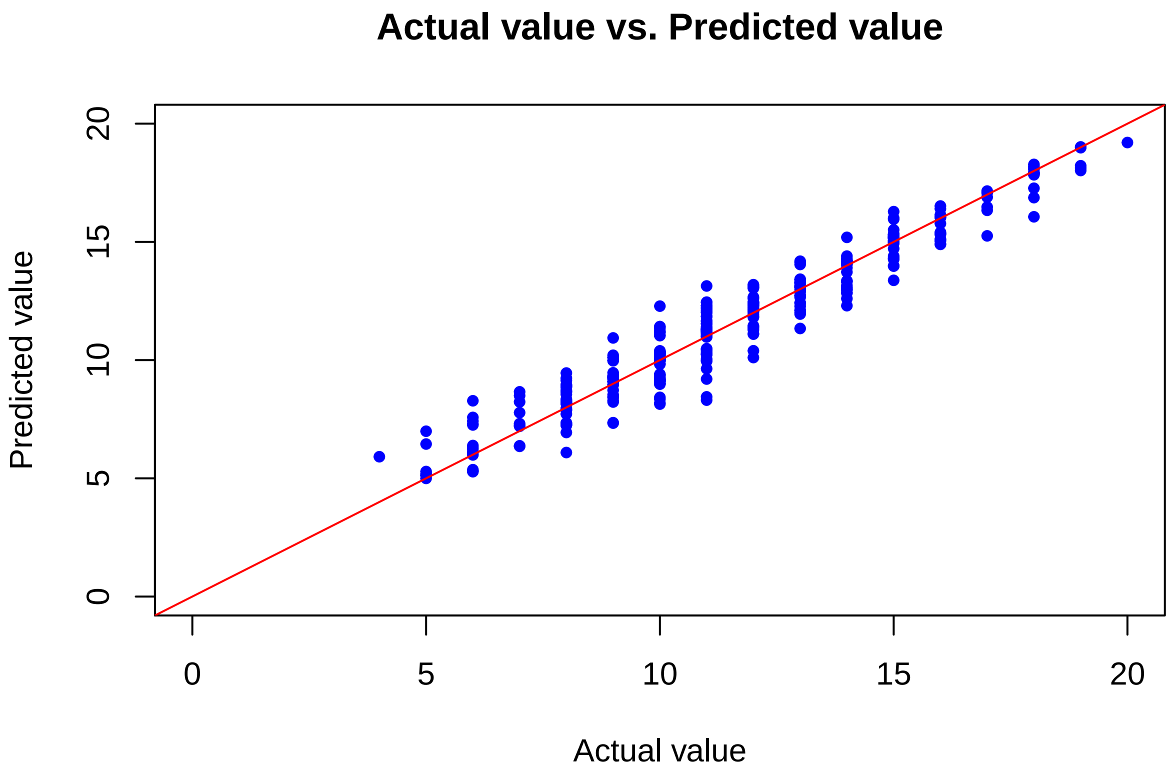
```
## (Intercept)    absences       goout          G1          G2
##       0.613      -0.011      -0.078       0.119       0.870
```

Here, we briefly discuss the effects and interpretations of each predictor:

- `absences`: For every additional absence, the student is expected to perform `0.011` points worse. This is not surprising because being absent from school is typically harmful for grades.
- `goout`: For every additional unit of going out with friends, the student is expected to perform `-0.078` points worse. This is not surprising because spending too much time outside of school may start to hurt grades.
- `G1` and `G2`: For every additional point scored in the first and second midterm, the student is expected to perform `0.119` and `0.870` points better. This is not surprising because students who performed better in midterms typically earn a better final grade. In addition, midterms grades also contribute to the final grade.

Predicted vs. Actual value

```
y_hat = predict(model)
y = data$G3
plot(y, y_hat, col = 'blue', pch = 20, xlab = 'Actual value', ylab = 'Predicted value',
     xlim = c(0, 20), ylim = c(0, 20), main = 'Actual value vs. Predicted value')
abline(0, 1, col = 'red')
```



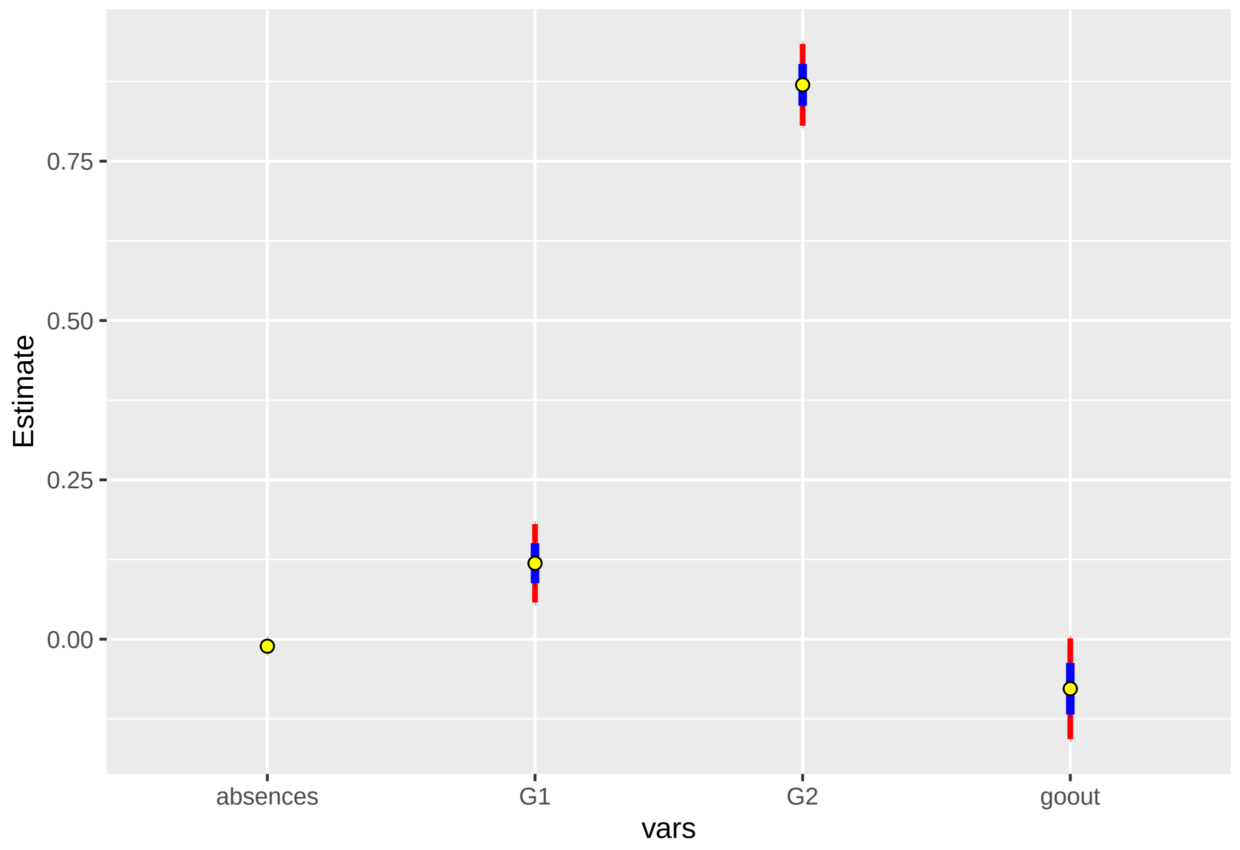
The plot shows a positive correlation between the predicted value and the actual value, and the model is doing a decent job in predicting the final grade. We will further examine the residuals to check if the model violates some statistical assumptions.

Estimated Coefficients and Standard Error

Variables	Est.Coefficients	Errors	t_value	p_value
absences	-0.0110	0.0055	-2.0178	0.0444
goout	-0.0777	0.0404	-1.9248	0.0551
G1	0.1191	0.0314	3.7942	0.0002
G2	0.8698	0.0328	26.5542	0.0000

At a significance level of $\alpha = 0.05$, variables `absences`, `G1`, `G2` are significant.

```r
coefs = as.data.frame(summary(model)$coefficients[-1, 1:2])
names(coefs)[2] = 'se'
coefs$vars = rownames(coefs)
ggplot(coefs, aes(vars, Estimate)) +
geom_errorbar(aes(ymin=Estimate - 1.96*se, ymax=Estimate + 1.96*se), lwd=1, colour="red", width=0) +
geom_errorbar(aes(ymin=Estimate - se, ymax=Estimate + se), lwd=1.5, colour="blue", width=0) +
geom_point(size=2, pch=21, fill="yellow")
```



The graph above shows that `G1` and `G2` have very high influence on the final grade, while `goout` is not significant at a $\alpha = 0.05$ level. Even though `absences` does not seem to be large, its influence can be greater when students are absent for multiple days.

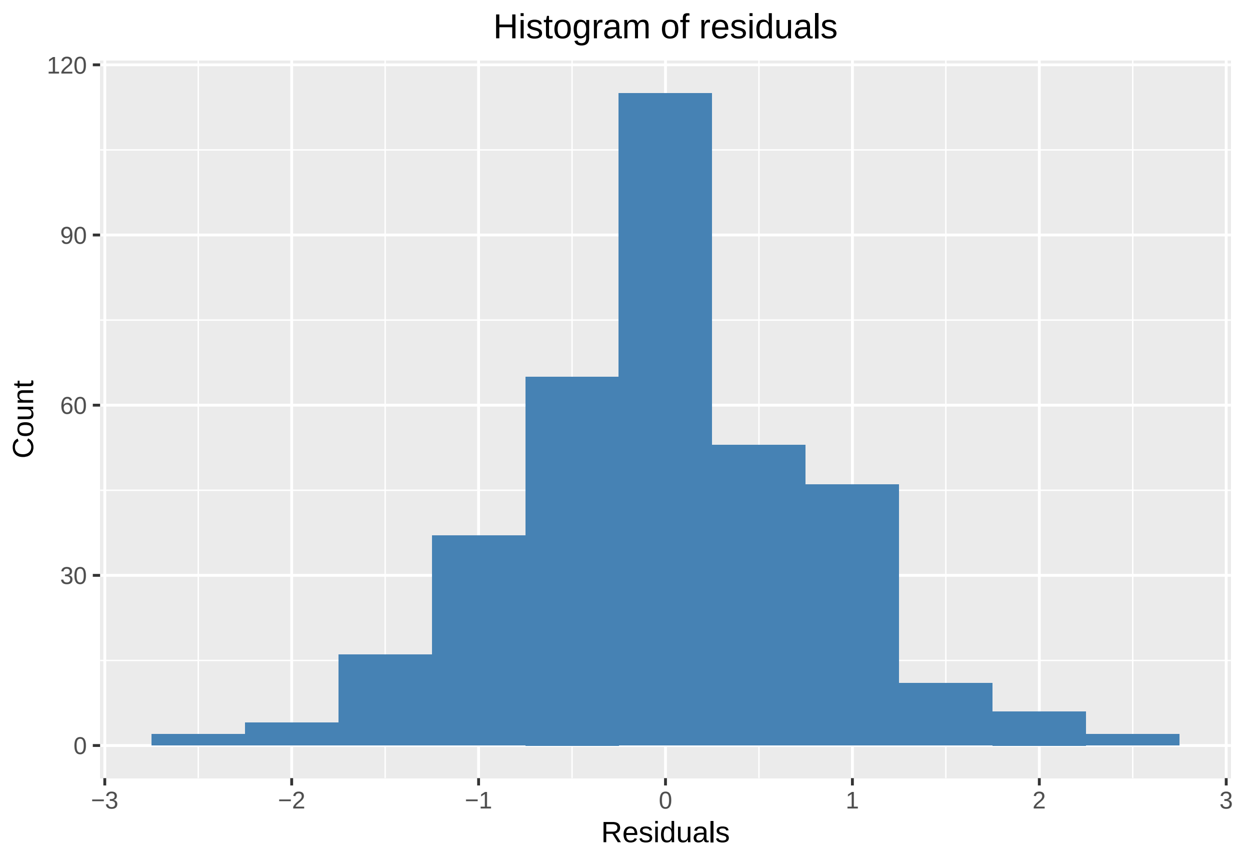
```r
round(mean((predict(model, data) - data$G3)^2), 3)
```

```
## [1] 0.663
```

The MSE of the best model is `0.663`.

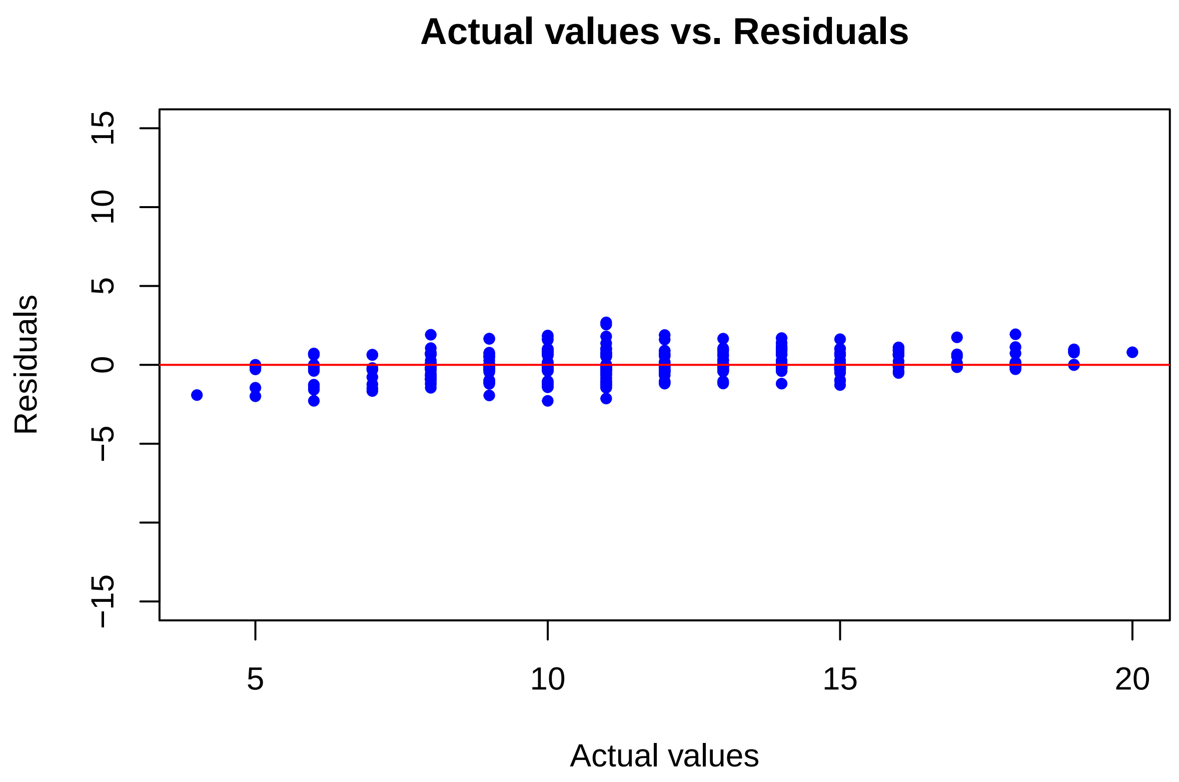
Residuals

```
resid = summary(model)$residuals
ggplot(as.data.frame(resid), aes(x = resid)) +
  geom_histogram(binwidth = 0.5, fill = "steelblue") +
  ggtitle("Histogram of residuals") +
  xlab('Residuals') +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5))
```



The residuals have an average of nearly zero, and roughly follow a normal distribution.

```r
plot(data$G3, resid, xlab = 'Actual values', ylab = 'Residuals', col = 'blue',
main = 'Actual values vs. Residuals', ylim = c(-15, 15), pch = 20)
abline(h=0, col = 'red')
```



The graph above shows that the residuals are generally uniform as the actual values increase. Even though the model slightly underestimates when the actual value is low, and slightly overestimates when the actual value is high, the heteroscedascity does not seems to be an issue as the residuals stay relatively close to the horizontal red line.

Lasso

Model fitting

We first partition the dataset into training (80%) and testing (20%) datasets, and fit the best model above to see its performance using cross-validation.

```r
set.seed(2020)
index = createDataPartition(data$G3, p = 0.8, list = F)
train = data[index, ]
test = data[-index, ]
model = lm(G3 ~ absences + goout + G1 + G2, train)
predictions = predict(model, test)
round(mean((test$G3 - predictions)^2), 3)
```

```
## [1] 0.694
```

We see that the MSE is relatively small. We will further use the same training and testing dataset to fit Ridge and Lasso regressions, and compare their MSEs. We first setup `x` and `y` to perform the Lasso regression. `x` includes all variables except for final grades, `y` includes final grades, and `grid` covers a wide range of λs. Then we use cross validation to find the flexibility of the model.

```r
x = model.matrix(G3 ~ ., data)[, -1]
y = data$G3
grid = 10^seq(10, -2, length = 100)
lasso_mod = glmnet(x, y, alpha = 1, lambda = grid)
```

To cross validate the model, we need to first set up the required variables, including training/testing predictors and final grades.

```r
x_train = model.matrix(G3 ~ ., train)[, -1]
x_test = model.matrix(G3 ~., test)[, -1]
y_train = train$G3
y_test = test$G3

set.seed(2020)
cv.out = cv.glmnet(x_train, y_train, alpha = 1)
bestlam = cv.out$lambda.min
round(bestlam, 3)
```

```
## [1] 0.069
```

Here we see that the best λ is `0.069`.

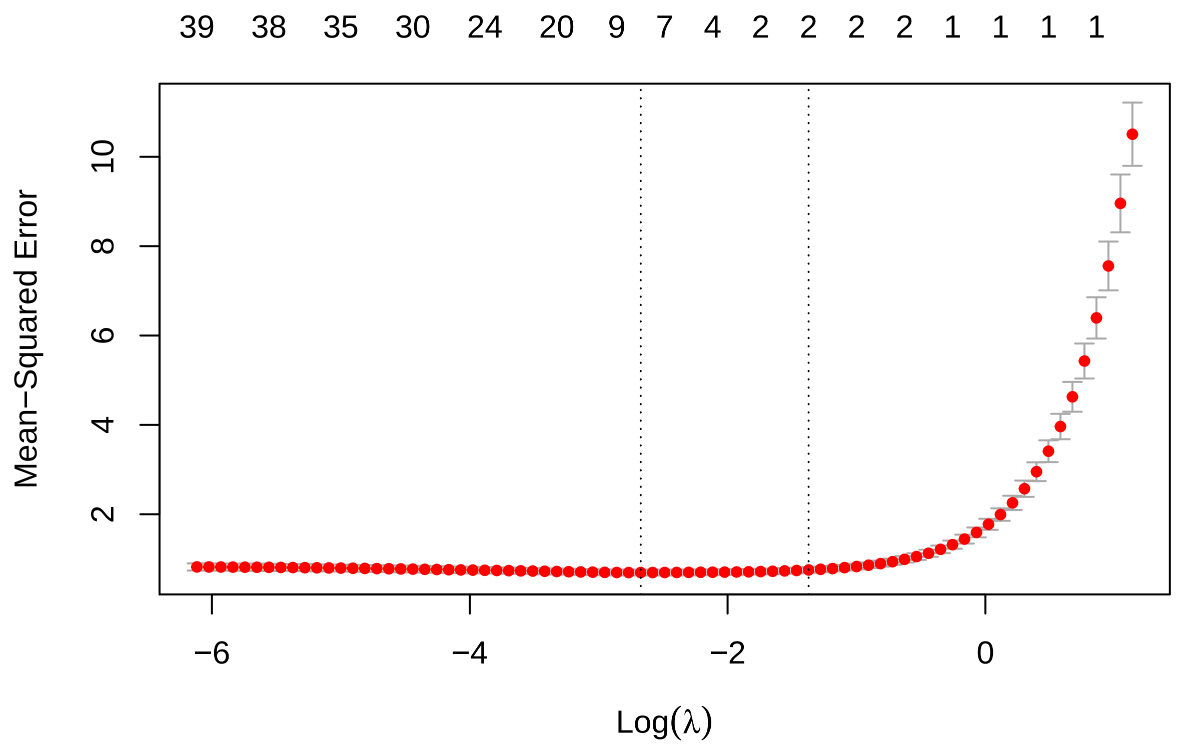
Model evaluation

```
cv.out
```

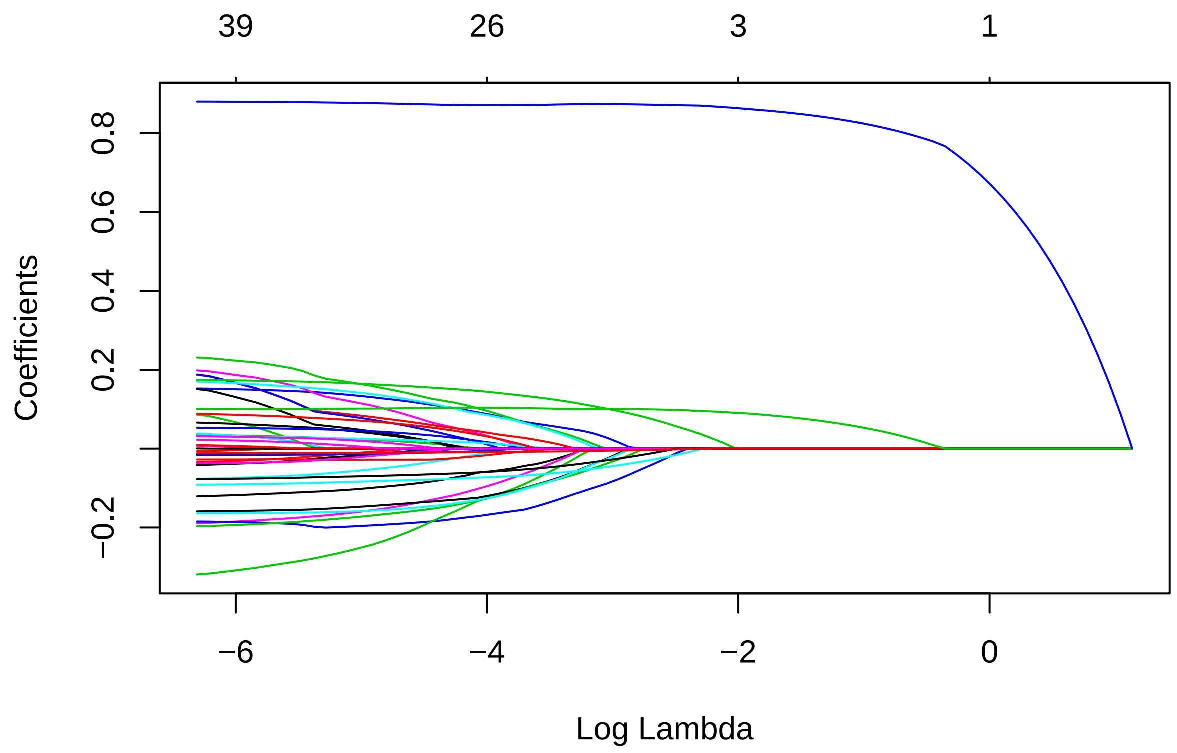
```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.06897  0.6946 0.06548       8
## 1se 0.25371  0.7530 0.06244       2
```

As shown above, λ has a standard error of `0.06548`. Hence, λ within one standard error is `(0.00349, 0.13445)`.

```
plot(cv.out)
```



The figure above shows the cross-validation error, in which the chosen λ is labeled as the vertical line at the right.



The plot above shows how the coefficients vary by different λs. As λs increase, the coefficients approach closer to zero. Notice that Lasso also performs feature selection, in which the coefficients of irrelavant predictors shrink to zero as λ increases.

At the chosen λ, most coefficients has shrunk to zero, except for the following:

	Coefficients
Mjobother	-0.0389938
famrel	0.0739174
goout	-0.0278291
health	-0.0109838
absences	-0.0032389
G1	0.0993672
G2	0.8720248

Finally, we compare the Lasso regression to the best linear model.

```r
predictions = predict(lasso_mod, s = cv.out$lambda.1se, newx = x_test)
round(mean((predictions - y_test)^2), 3)
```

```
## [1] 0.735
```

Compared to the best linear model, the Lasso model has a slightly higher MSE.

Ridge

Model fitting

The procedures of fitting a ridge model is similar to that of a lasso model.

```r
ridge_mod = glmnet(x, y, alpha = 0, lambda = grid)
set.seed(2020)
cv.out = cv.glmnet(x_train, y_train, alpha = 0)
bestlam = cv.out$lambda.min
round(bestlam, 3)
```

```
## [1] 0.313
```

Here we see that the best λ is `0.313`.

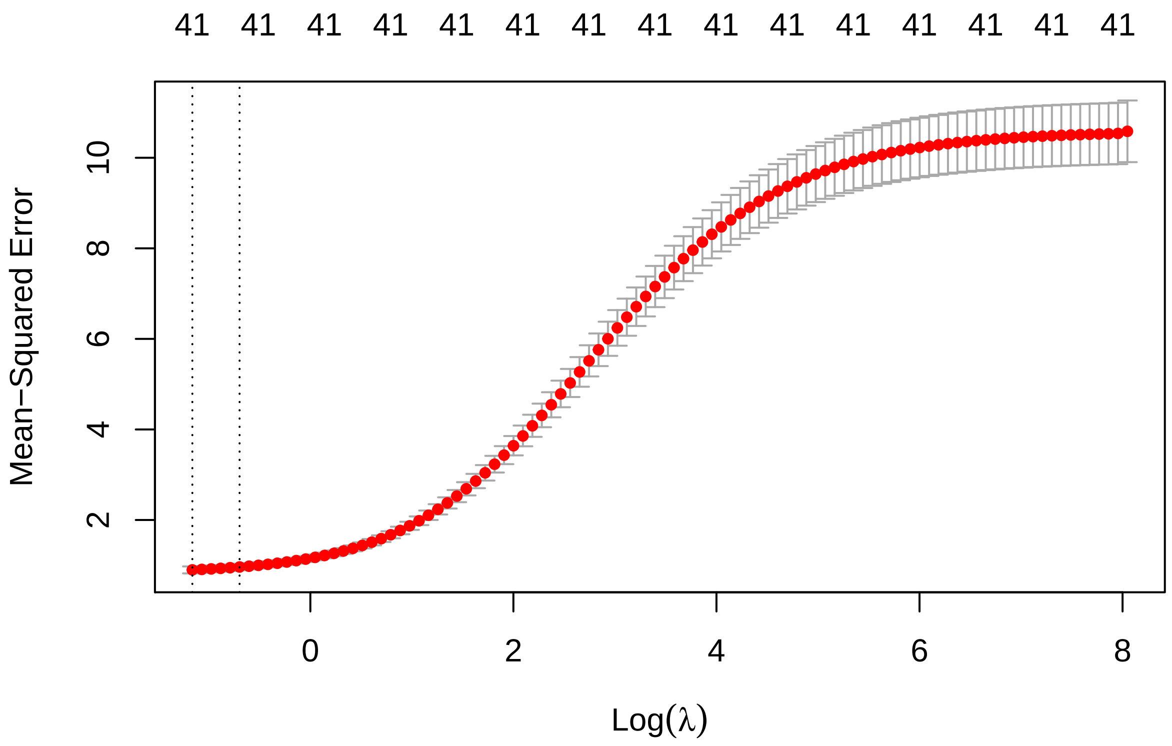
Model evaluation

```r
cv.out
```

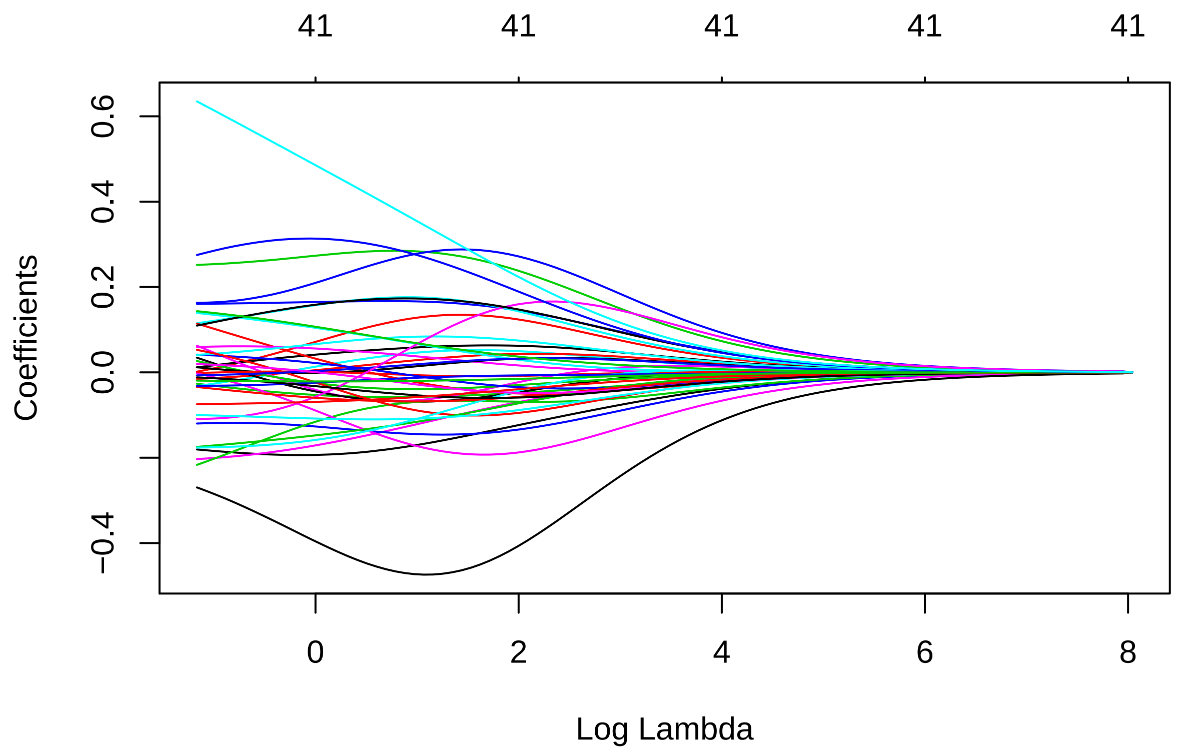
```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Measure      SE Nonzero
## min 0.3128  0.8981 0.07679      41
## 1se 0.4980  0.9617 0.07190      41
```

As shown above, λ has a standard error of `0.07679`. Hence, λ within one standard error is `(0.23601, 0.038959)`.

```r
plot(cv.out)
```



The figure above shows the cross-validation error, in which the chosen λ is labeled as the vertical line at the left.



The plot above shows how the coefficients vary by different λs. As λs increase, the coefficients approach closer to zero. Notice that Ridge models do not perform feature selection, so coefficients only shrink to close to but not equal to zero.

At the chosen λ, coefficients were not significantly shrunk. The following lists the coefficients greater than `0.2` in magnitude:

	Coefficients
PstatusT	-0.203
Mjobhealth	0.252
guardianother	-0.216
schoolsupyes	-0.270
G1	0.275
G2	0.634

Finally, we compare the Ridge regression to the best linear model.

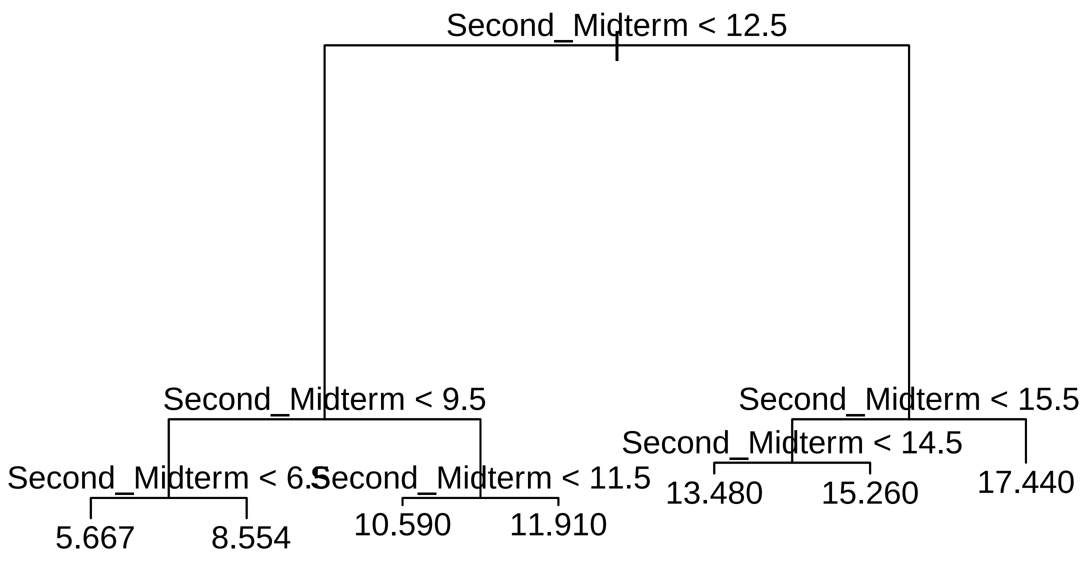
```r
predictions = predict(ridge_mod, s = cv.out$lambda.1se, newx = x_test)
round(mean((predictions - y_test)^2), 3)
```

```
## [1] 0.753
```

Compared to the best linear model and the Lasso model, the Ridge model has a slightly higher MSE.

Decision Tree

Big tree



The tree above indicates that the second midterm has a dominate influence on the final grade, which is an expected result. We now compare its MSE with previous models.

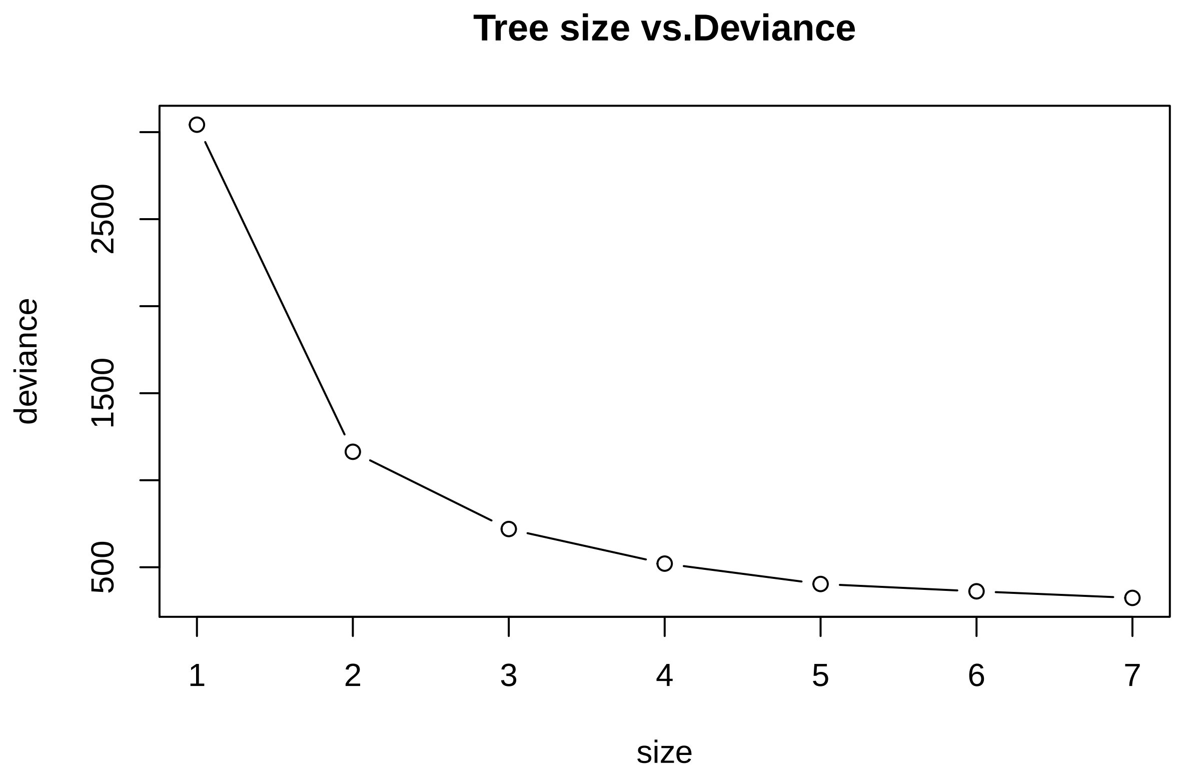
```
predictions = predict(tree, newdata = tree_test)
round(mean((predictions - tree_test$G3)^2), 3)
```

```
## [1] 1.102
```

As shown above, the testing MSE of the tree is higher than all previous models.

Tree purning

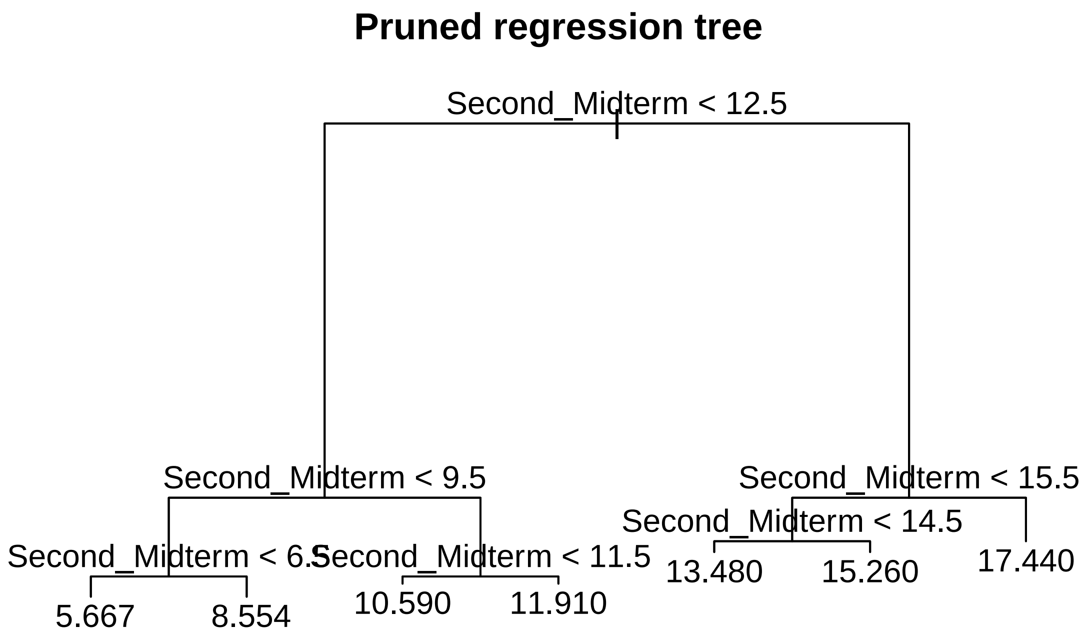
```
cv.performance = cv.tree(tree)
plot(cv.performance$size, cv.performance$dev, type = 'b',
     xlab = 'size', ylab = 'deviance', main = 'Tree size vs.Deviance')
```



The plot above shows the deviance with varying tree size. Using a cross-validation, a 7-node tree is selected.

Therefore, we can prune the tree as follows:

```
pruned = prune.tree(tree, best = 7)
plot(pruned)
text(pruned, pretty = 0)
title('Pruned regression tree')
```



The pruned tree follows the following splitting (In breadth-first search order):

0. Is `G2` less than `12.5`? If so, proceed to the left subtree (1); otherwise, proceed to the right subtree (2).
1. Is `G2` less than `9.5`? If so, proceed to the left subtree (3); otherwise, proceed to the right subtree (4).
2. Is `G2` less than `15.5`? If so, proceed to the left subtree (5); otherwise, the tree predicts a value of `17.44`.
3. Is `G2` less than `6.5`? If so, the tree predicts a value of `5.667`; otherwise, the tree predicts a value of `8.554`.
4. Is `G2` less than `11.5`? If so, the tree predicts a value of `10.59`; otherwise, the tree predicts a value of `11.91`.
5. Is `G2` less than `14.5`? If so, the tree predicts a value of `13.48`; otherwise, the tree predicts a value of `15.26`.

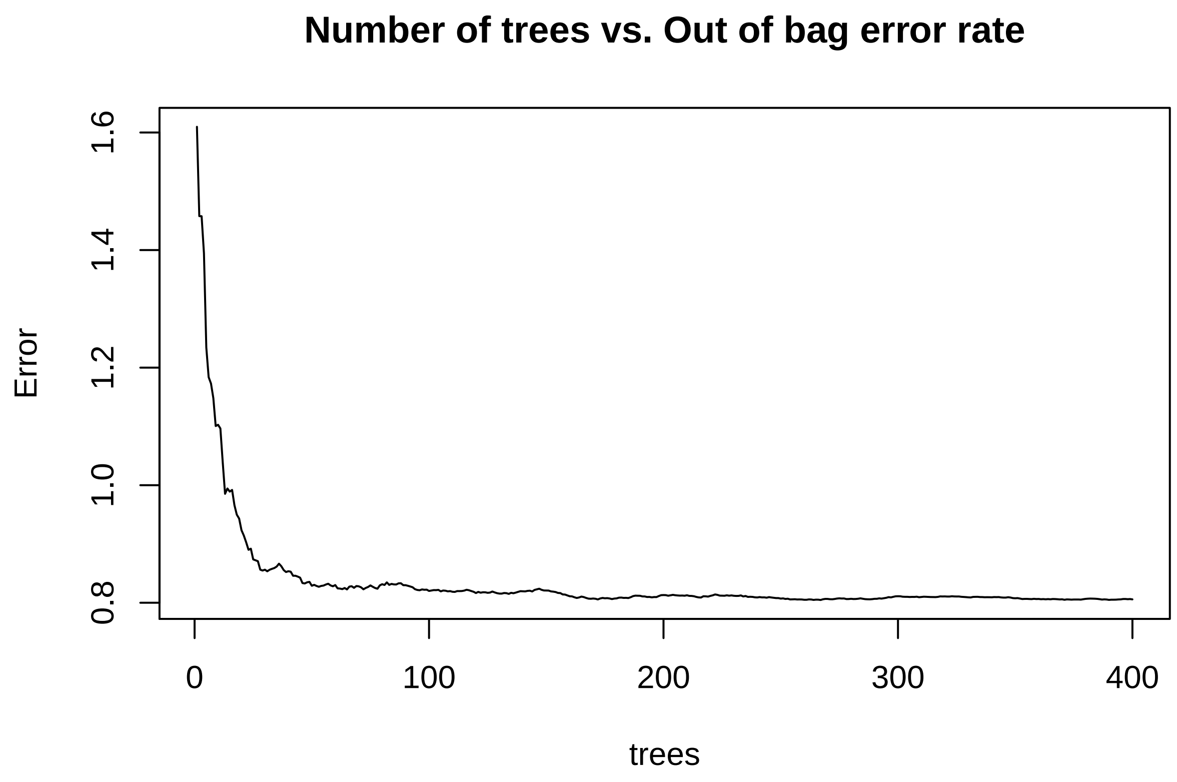
Interestingly, all splits are based on `G2`, which indicates that `G2` has a profound impact on the final grade, which is also an expected result. While the pruned tree is easier to interpret, it has a higher MSE compared to all previous models.

Bagging

We will be using the same training/testing partition above to evalutate the following models.

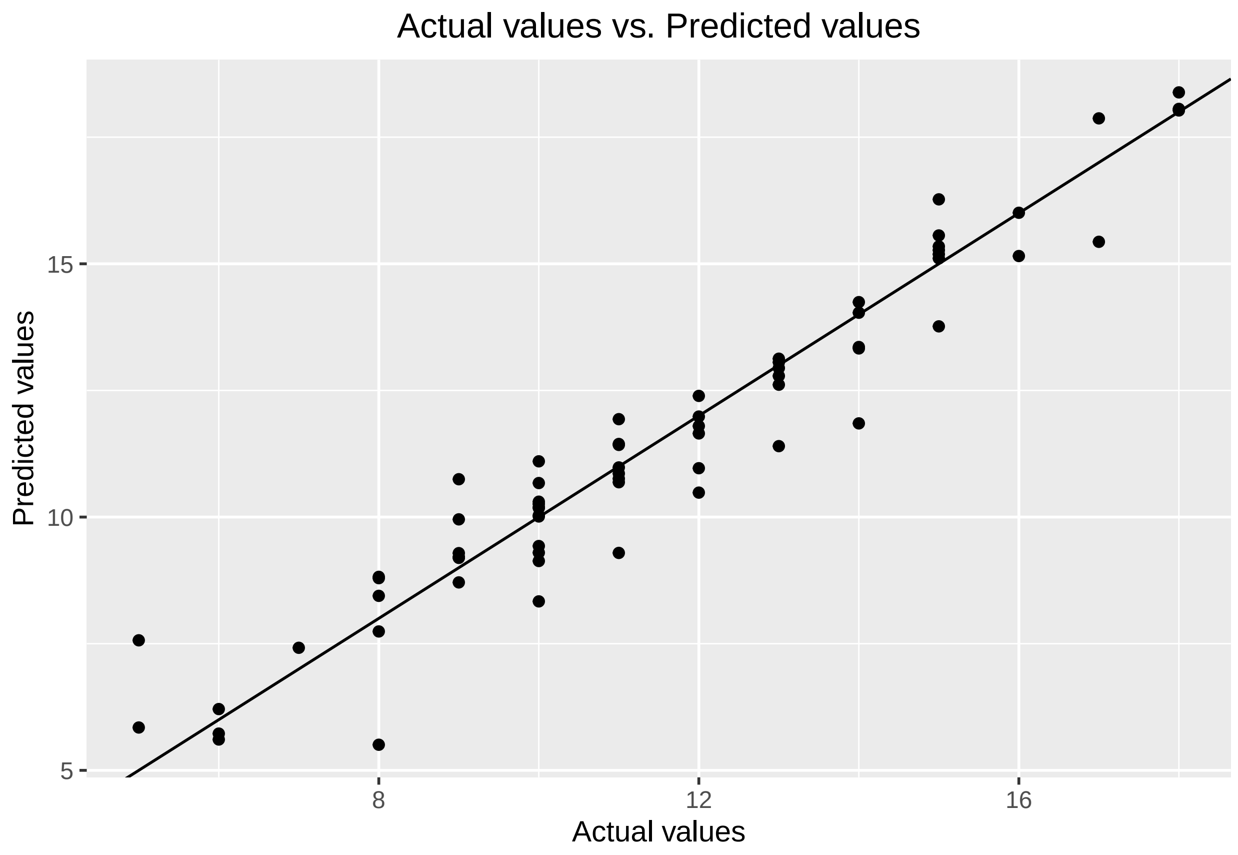
Error evaluation

```
model = randomForest(G3 ~ ., data = train, mtry = ncol(train) - 1, importance = TRUE, ntree = 400)
plot(model, main = "Number of trees vs. Out of bag error rate")
```



As shown above, the out-of-bag error decreases as the number of trees increase. We now investigate how the model performs against the actual values.

```
predictions = predict(model, newdata = test)
ggplot() +
  geom_point(aes(x = test$G3, y = predictions)) +
  geom_abline() +
  labs(x = "Actual values", y = "Predicted values", title = "Actual values vs. Predicted values") +
  theme(plot.title = element_text(hjust = 0.5))
```



As shown above, the model does a decent job in predicting the values.

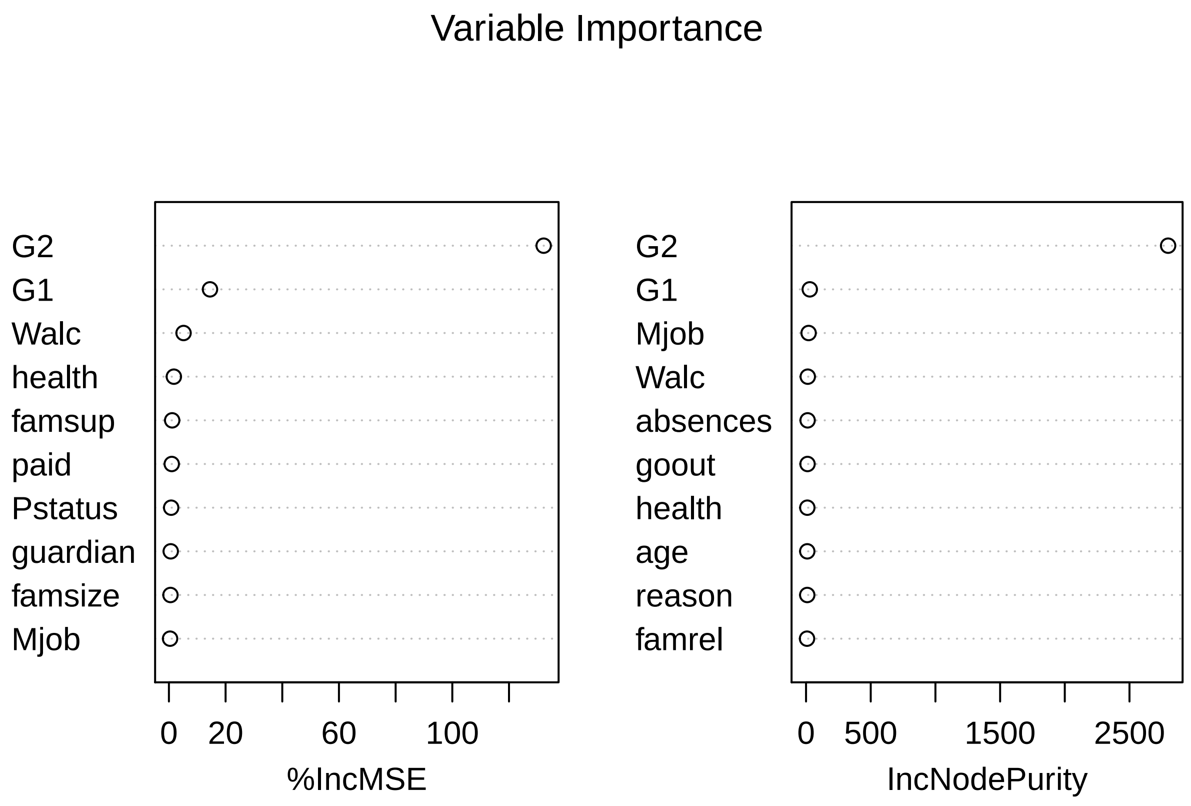
```r
round(mean((predictions - test$G3)^2), 3)
```

```
## [1] 0.727
```

Even though the MSE is relatively low, it is still slightly higher than that of the best OLS model.

Variable Importance

```r
varImpPlot(model, main = 'Variable Importance', n.var = 10)
```

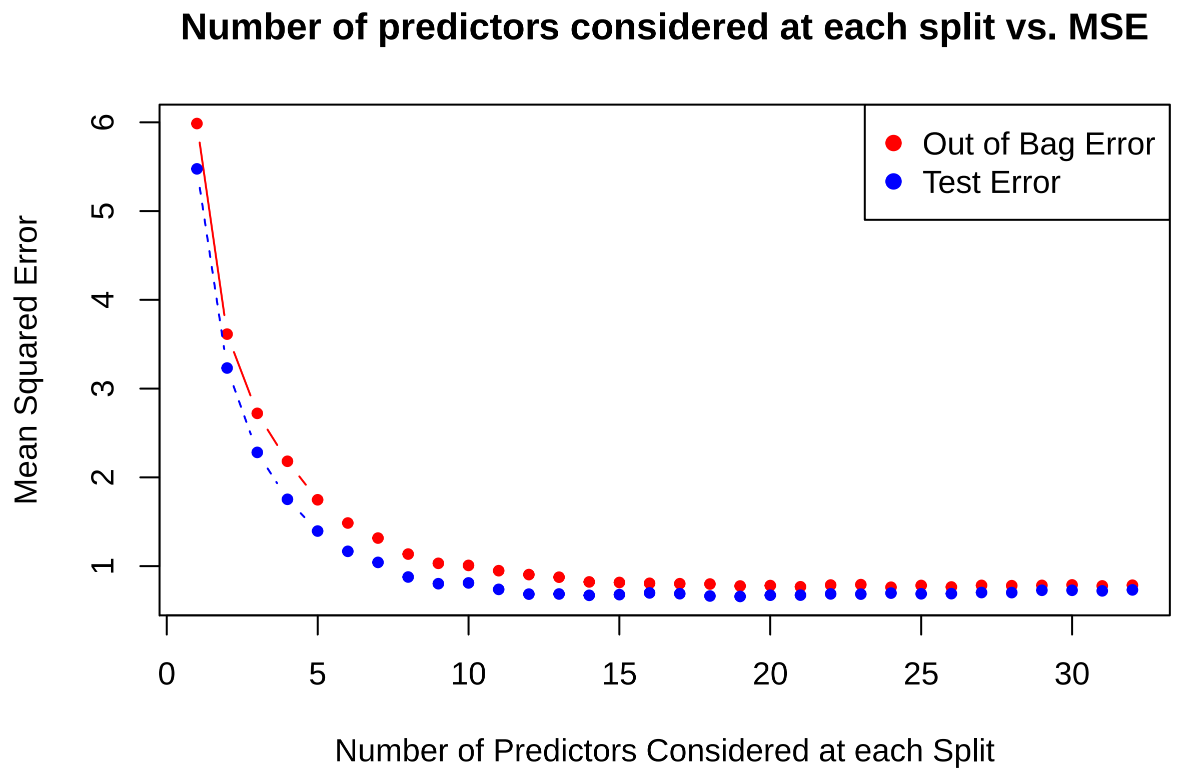


As shown above, `G2` has a very high importance in the model. This is an expected result, because previous models have shown that `G2` has a very high correlation with `G3`, and the second midterm grade unsurprisingly has a huge influence on the final grade. The variable importance is then followed by `G1`, which is the first midterm grade.

Random Forest

Error evaluation

```
k = ncol(train) - 1
oob_error = double(k)
test_error = double(k)
set.seed(2020)
for (mtry in 1:k) {
  model = randomForest(G3 ~ ., data = train, mtry = mtry, ntree = 400)
  oob_error[mtry] = model$mse[400]
  predictions = predict(model, test)
  test_error[mtry] = with(test, mean((test$G3 - predictions) ^ 2))
}
matplot(1:mtry , cbind(oob_error,test_error), pch=20 , col=c("red","blue"),type="b",ylab="Mean Squared E
="Number of Predictors Considered at each Split")
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19, col=c("red","blue"))
title('Number of predictors considered at each split vs. MSE')
```



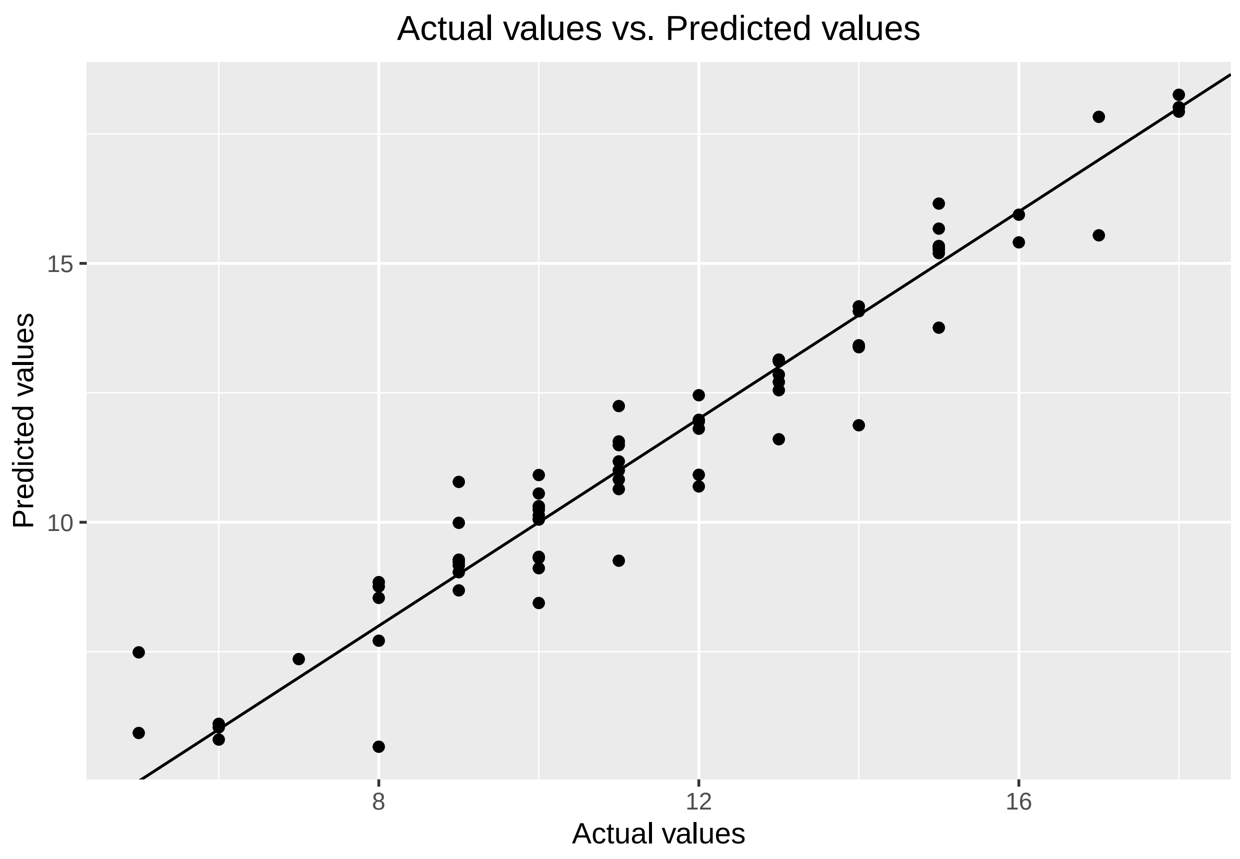
As the number of predictions increase, the out-of-bag error and test error both decrease. Both errors follow a similar pattern, even though the OOB error is slightly higher than the test error.

```
which.min(oob_error)
```

```
## [1] 24
```

The model reaches its minimum OOB error at 24 predictors. We know check how the model performs against actual values.

```
model = randomForest(G3 ~ ., data = train, mtry = 24, ntree = 400)
predictions = predict(model, newdata = test)
predictions = predict(model, newdata = test)
ggplot() +
  geom_point(aes(x = test$G3, y = predictions)) +
  geom_abline() +
  labs(x = "Actual values", y = "Predicted values", title = "Actual values vs. Predicted values") +
  theme(plot.title = element_text(hjust = 0.5))
```



The model did a decent job in predicting the values.

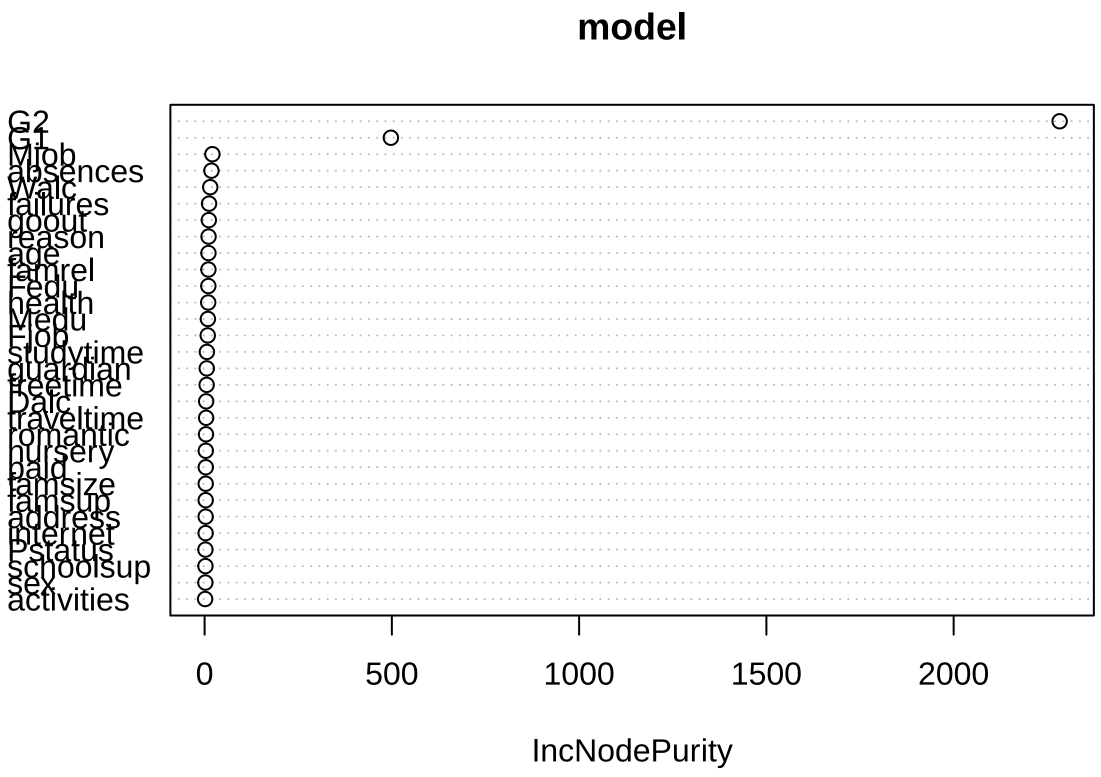
```
round(mean((predictions - test$G3)^2), 3)
```

```
## [1] 0.684
```

The MSE is a lot lower than the bagging MSE, even though it remains higher than that of the best OLS model.

Variable importance

```r
varImpPlot(model)
```



As shown above, `G2` still has a dominate impact on the final grade, followed by `G1`. The two midterm grades consistently have huge influence of all models we had so far.

Boosting

We will use the same training and testing datasets to fit and evaluate a boosting model.

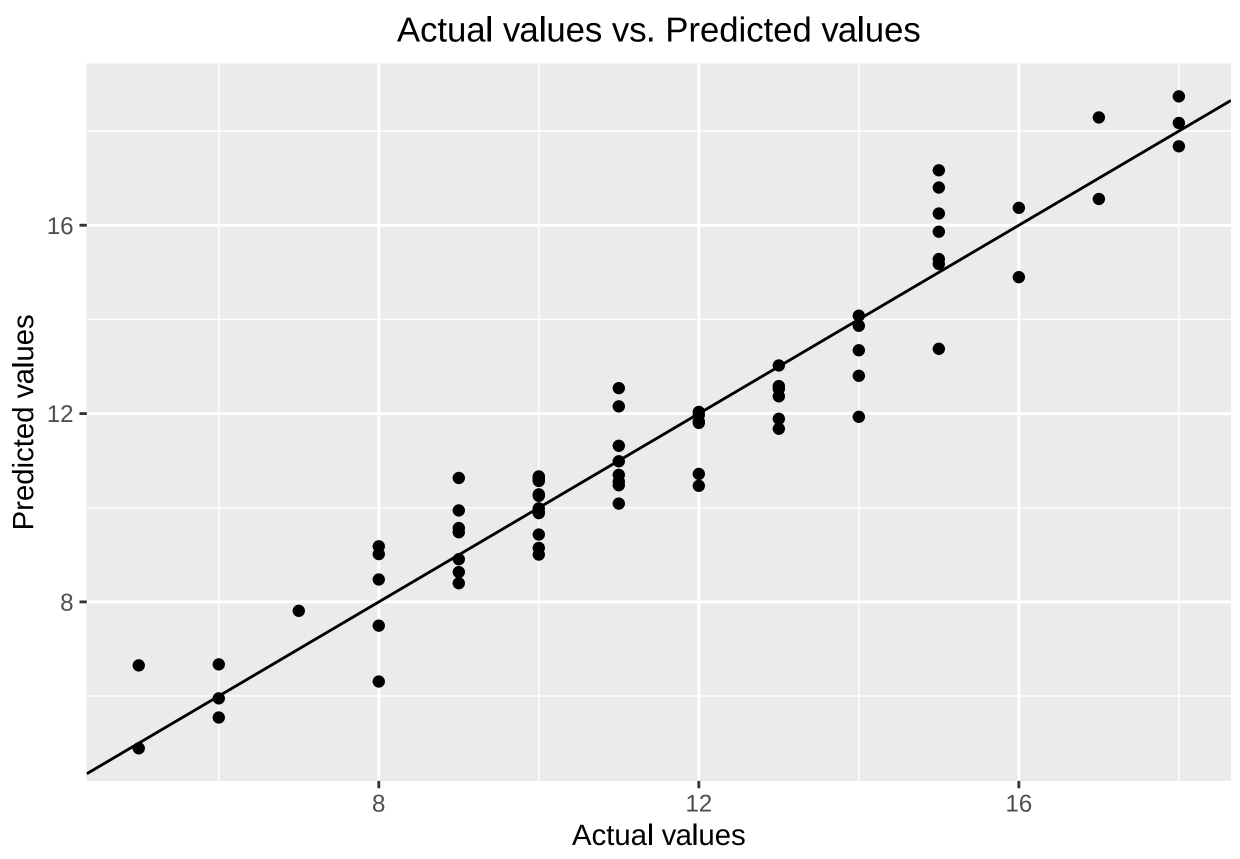
Error evaluation

We first fit the boosting model.

```r
set.seed(2020)
model = gbm(G3 ~ .,
            data = train,
            distribution = "gaussian",
            n.trees = 5000,
            interaction.depth = 4)
predictions = predict(model, newdata = test, n.trees = 5000)
```

Then we plot the predicted values against the actual values.

```r
ggplot() +
  geom_point(aes(x = test$G3, y = predictions)) +
  geom_abline() +
  labs(x = "Actual values", y = "Predicted values", title = "Actual values vs. Predicted values") +
  theme(plot.title = element_text(hjust = 0.5))
```



The model seems to perform worse compared to previous models.

```r
round(mean((predictions - test$G3)^2), 3)
```

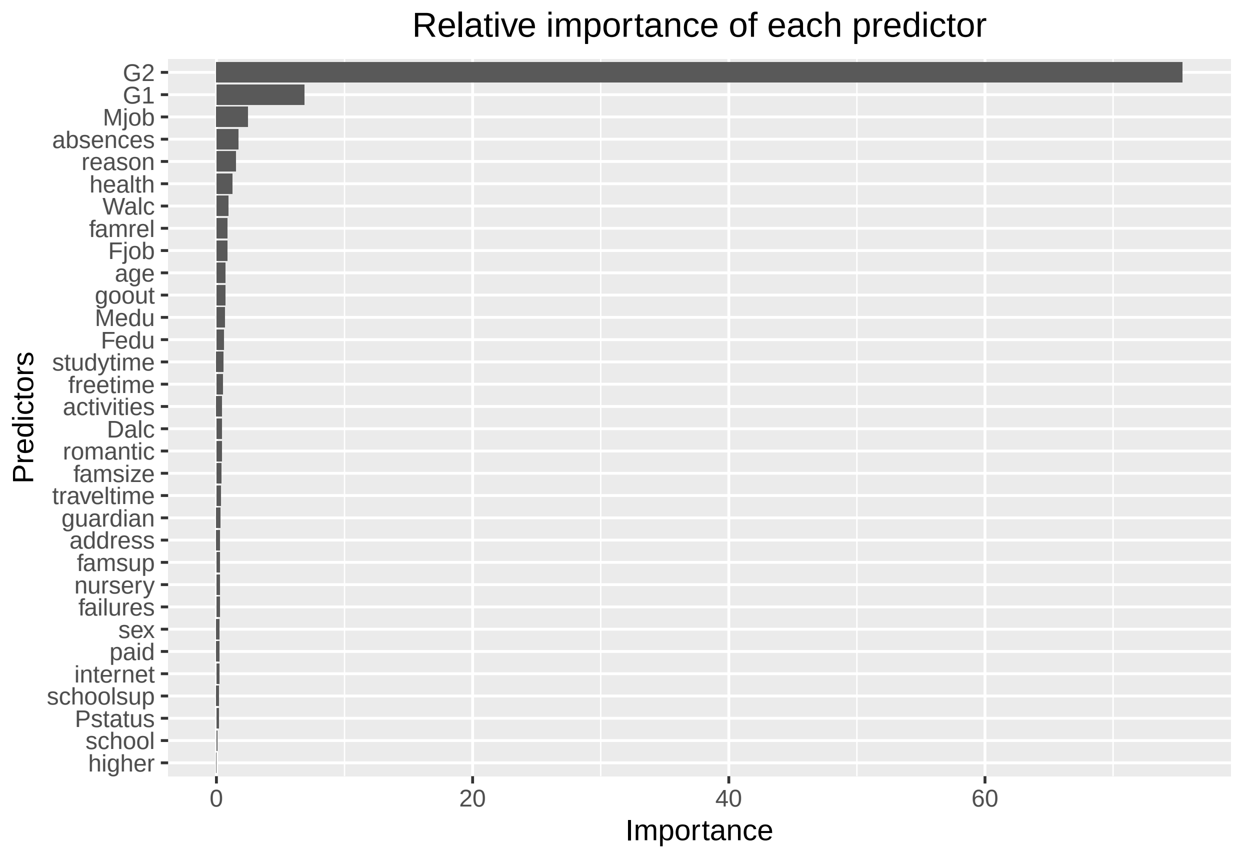
```
## [1] 0.787
```

The MSE of this model is also higher than most of the previous models.

Variable importance

```r
importance = summary.gbm(model, plotit = FALSE)
df = data.frame(
  name = importance$var,
  value = importance$rel.inf
)

ggplot(df, aes(x = reorder(name, value), y = value)) +
  geom_bar(stat = "identity") +
  labs(x = "Predictors", y = "Importance", title = "Relative importance of each predictor") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()
```



Same as all previous models, `G2` has a dominate effect on the final grade, followed by `G1`.

Parameters tuning

We can also tune the parameters using a grid search to optimize the model. Here, we tune the shrinkage parameter, total number of trees to fit, and the interaction depth.

```r
min_shrinkage = 0
min_trees = 0
min_interaction = 0
min_mse = 10000

set.seed(2020)
for (i in seq(0.001, 0.01, 0.001)) {
  for (j in seq(1000, 5000, 1000)) {
      for (k in seq(1, 5)) {
        model = gbm(G3 ~ .,
                data = train,
                distribution = "gaussian",
                shrinkage = i,
                n.trees = j,
                interaction.depth = k)

        predictions = predict(model, newdata = test, n.trees = j)
        mse = mean((predictions - test$G3)^2)
        if (mse < min_mse) {
        min_shrinkage = i
        min_trees = j
        min_interaction = k
        min_mse = mse
      }
    }
  }
}
```

After the grid search, we found that the shrinkage parameter, total number of trees to fit, and the interaction depth that yields an optimal model are:

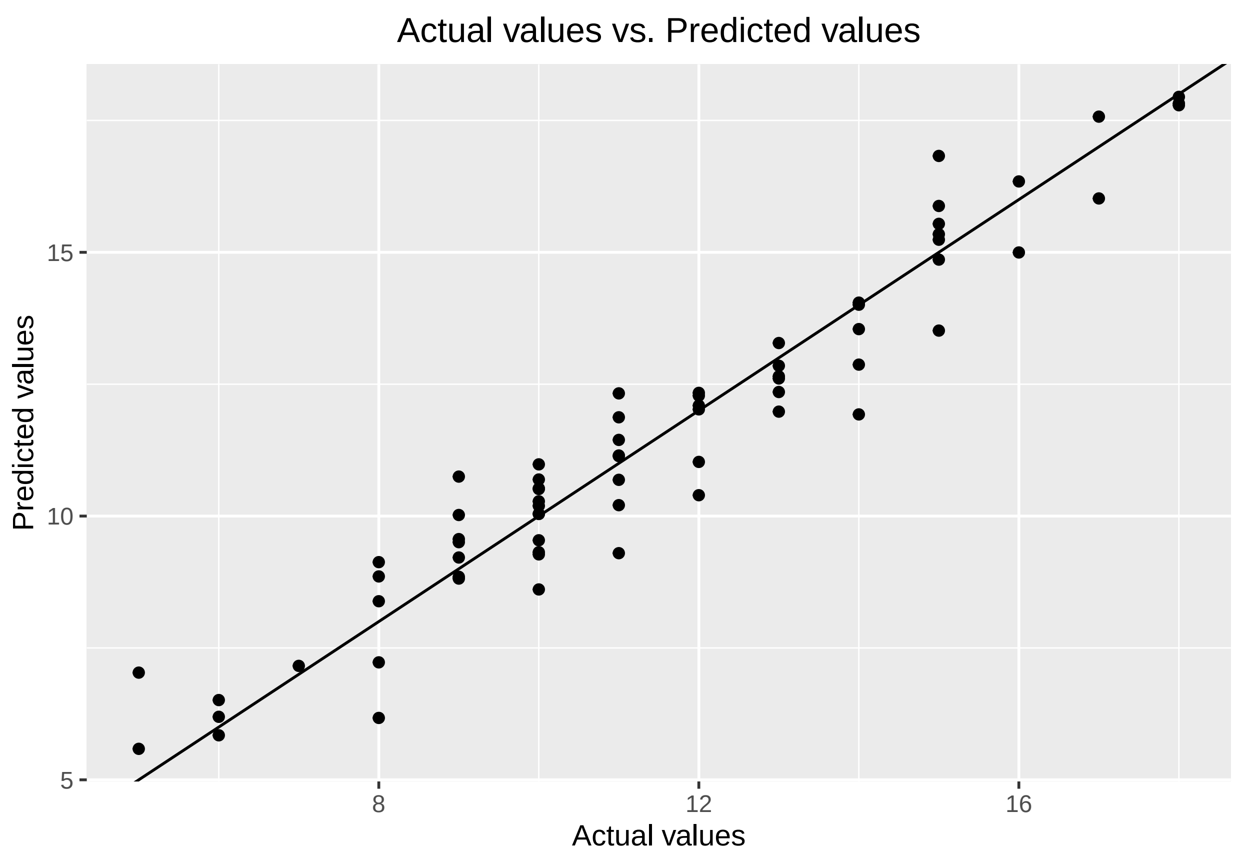
```r
c(min_shrinkage, min_trees, min_interaction)
```

```
## [1] 5e-03 1e+03 5e+00
```

We then fit the model using the tuned parameter, and plot the predicted values against the actual values.

```r
model = gbm(G3 ~ .,
        data = train,
        distribution = "gaussian",
        shrinkage = min_shrinkage,
        n.trees = min_trees,
        interaction.depth = min_interaction)
```

```r
predictions = predict(model, newdata = test, n.trees = min_trees)
ggplot() +
  geom_point(aes(x = test$G3, y = predictions)) +
  geom_abline() +
  labs(x = "Actual values", y = "Predicted values", title = "Actual values vs. Predicted values") +
  theme(plot.title = element_text(hjust = 0.5))
```



We can see that the new model did a better job than the untuned bagging model.

```r
round(mean((predictions - test$G3)^2), 3)
```

```
## [1] 0.695
```

Even though we were able to significantly reduce the MSE, the MSE is still higher than the best OLS model.

XGboost

Some of the columns were removed due to how XGB matrices accept data. However, this should have very little impact on the model given that `G2` has significant influence on the final grade, and is still in the model.

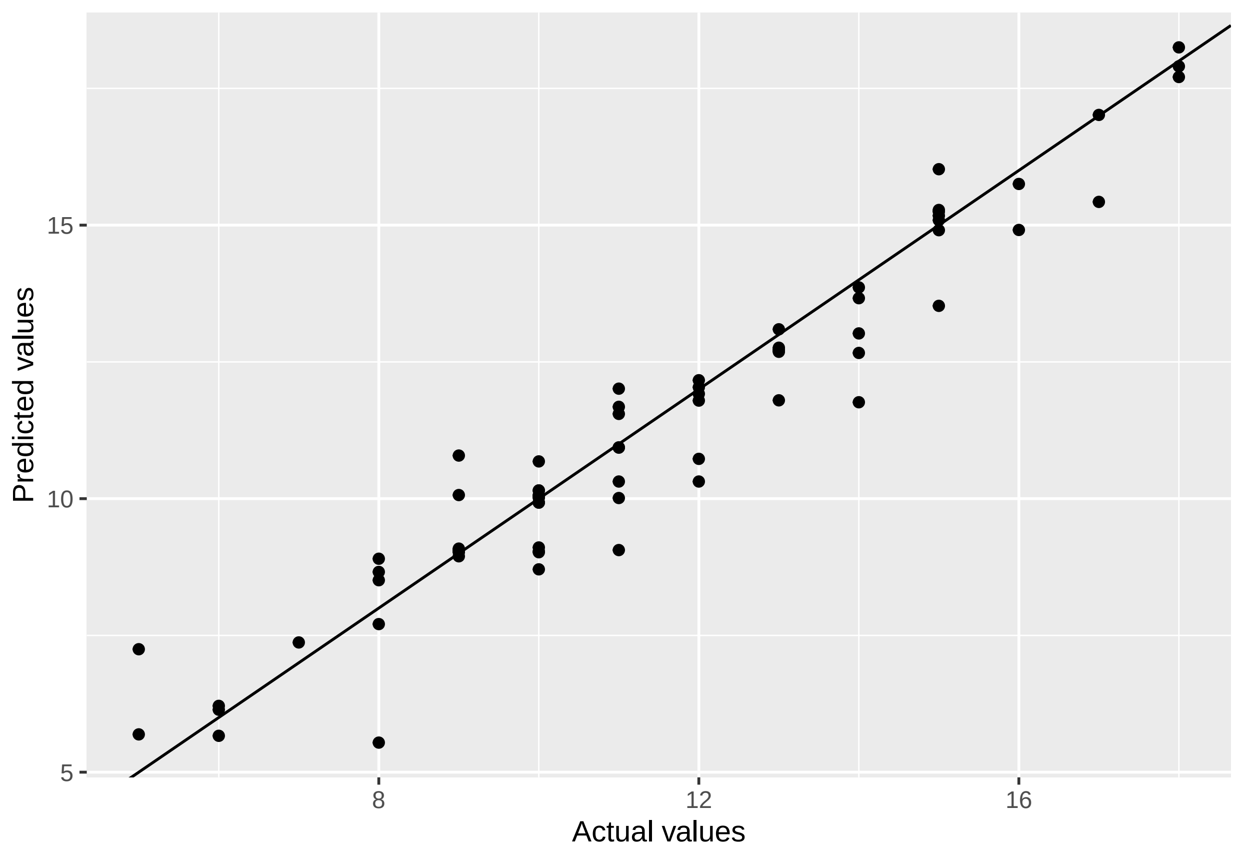
```
numerics = c("age", "Medu", "Fedu", "traveltime", "studytime", "failures", "famrel", "freetime", "goout
Y_train = as.matrix(train[, "G3"])
X_train = as.matrix(train[names(train) %in% numerics])
dtrain = xgb.DMatrix(data = X_train, label = Y_train)
X_test = as.matrix(test[names(train) %in% numerics])
```

Error evaluation

```
set.seed(2020)
model = xgboost(data = dtrain,
                max_depth = 2,
                eta = 0.1,
                nrounds = 40,
                lambda = 0,
                print_every_n = 10,
                objective = "reg:linear")
```

```
## [1]  train-rmse:10.368090
## [11] train-rmse:3.737932
## [21] train-rmse:1.524314
## [31] train-rmse:0.909952
## [40] train-rmse:0.784421
```

```
predictions = predict(model, X_test)
ggplot() +
  geom_point(aes(x = test$G3, y = predictions)) +
  geom_abline() +
  labs(x = "Actual values", y = "Predicted values")
```



The model seemed to perform worse than the tuned boosting model.

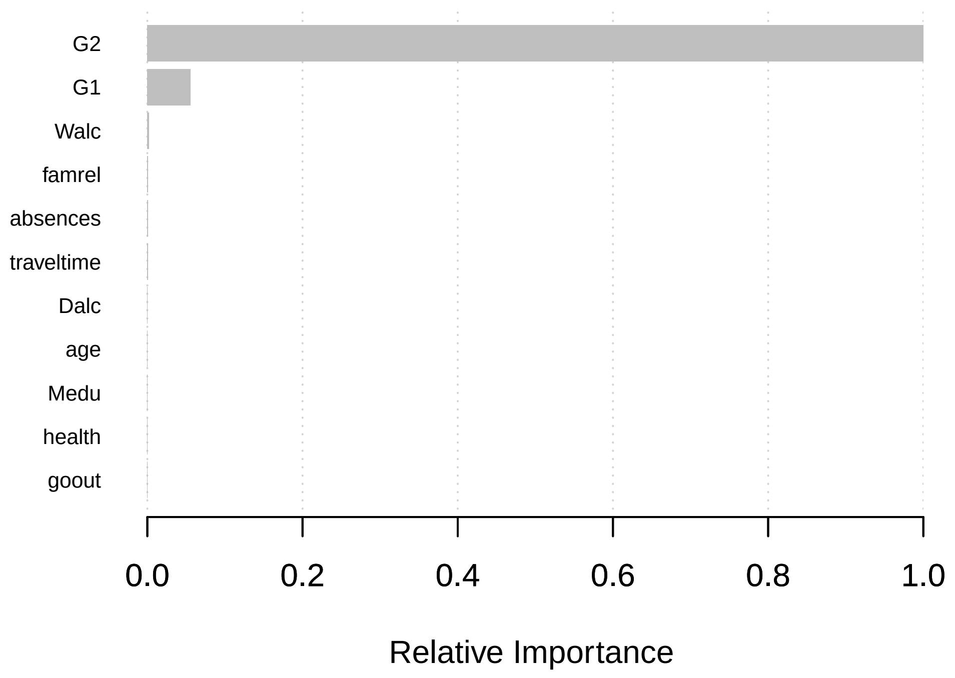
```r
round(mean((predictions - test$G3)^2), 3)
```

```
## [1] 0.742
```

The MSE is higher than most of the previous models.

Variable importance

```r
importance = xgb.importance(colnames(X_train), model = model)
xgb.plot.importance(importance, rel_to_first = TRUE, xlab = "Relative Importance")
```



G2 still had a dominate impact on the final grade, followed by G1. Other variables have very little impact.

Neural Network

We will be using the same training and testing datasets for the neural network model. First, we need to prepare the data in matrix form.

```
train_labels = as.matrix(train[, "G3"])
train_data = as.matrix(train[!names(train) %in% c("G3") & names(train) %in% numerics])
test_labels = as.matrix(test[, "G3"])
test_data = as.matrix(test[!names(test) %in% c("G3") & names(train) %in% numerics])

train_data <- scale(train_data)
test_data <- scale(test_data)
```

We define the keras model, which includes two densely-connected neural network layers. We also add an early stop callback in case that the training only requires a lower epoch.

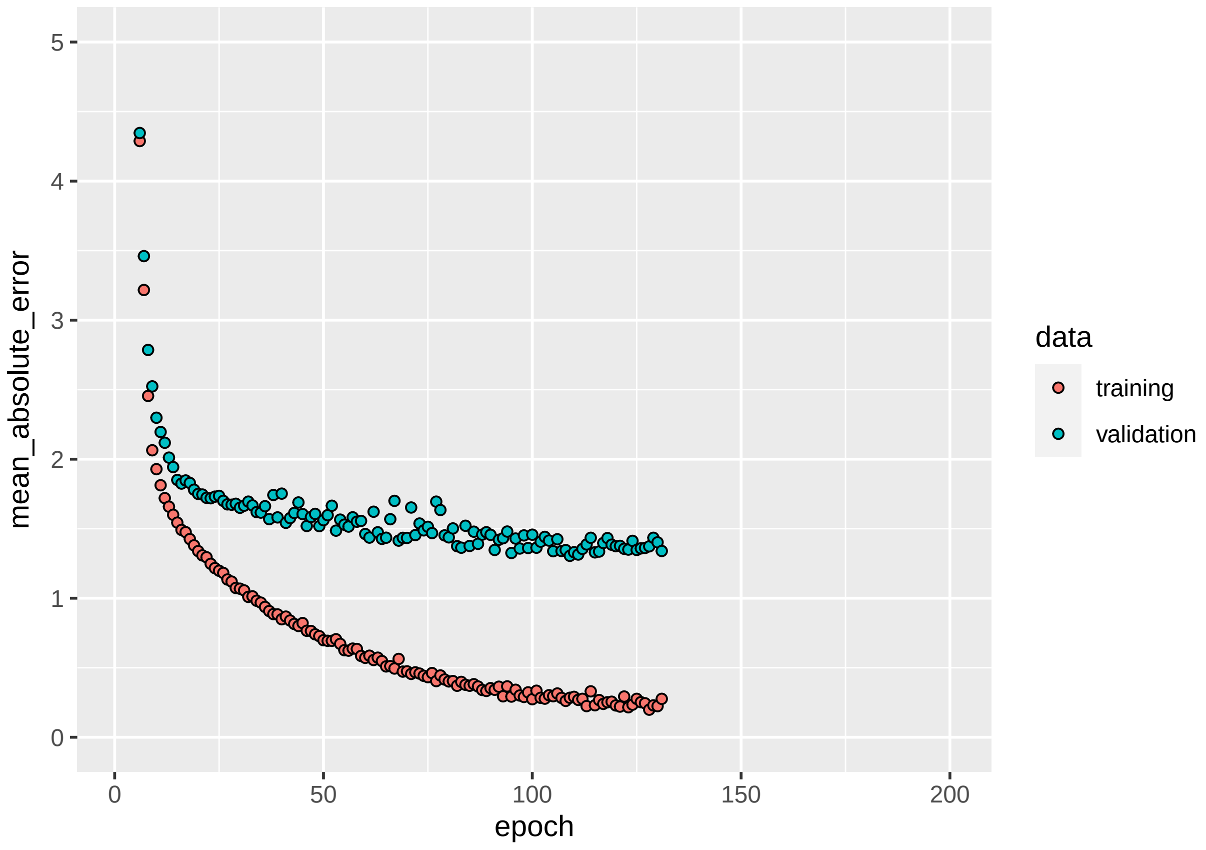
```
model = keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu",
              input_shape = dim(train_data)[2]) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 1)

model %>% compile(
  loss = "mse",
  optimizer = optimizer_rmsprop(),
  metrics = list("mean_absolute_error")
)

early_stop = callback_early_stopping(monitor = "val_loss", patience = 20)
```

Here, we plot the training and validation mean absolute error.

```r
library(ggplot2)
plot(history, metrics = "mean_absolute_error", smooth = FALSE) +
  coord_cartesian(ylim = c(0, 5))
```



As the training epoch increases, the mean absolute error decreases and eventually stabilizes. The early stop callback was revoked as the training did not reach the designated epoch.

```r
predictions = model %>% predict(test_data)
mse_nn = round(mean((test_labels - predictions) ^ 2), 3)
mse_nn
```

```
## [1] 2.817
```

Unfortunately, the MSE is significantly higher than all previous models.

KNN

We first partition the dataset into training (80%) and testing (20%) datasets with the same predictors

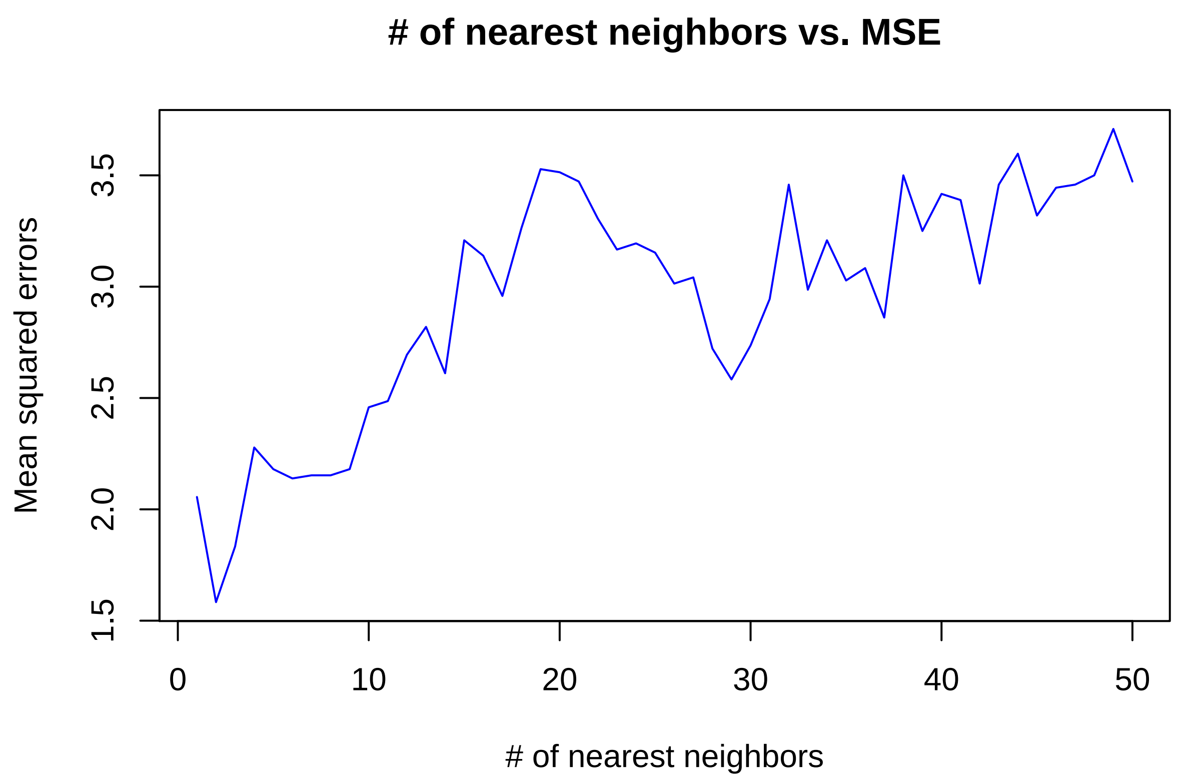
```
selected = c(26, 30, 31, 32)
set.seed(2020)
rand = sample(1:nrow(data), 0.8 * nrow(data))
train_dataset = data[rand, selected]
test_dataset = data[-rand, selected]
train_cl = data[rand, 33]
test_cl = data[-rand, 33]
```

We tune the `k` parameter between `1` and `50` to see which value will yield the optimal result.

```
mses = rep(0, 50)
set.seed(2020)
for (i in 1:50) {
  model = knn(train_dataset, test_dataset, cl = train_cl, k = i)
  model = as.numeric(levels(model))[model]
  mses[i] = mean((test_cl - model)^2)
}
```

We then get the following graph:

```
plot(mses, type = 'l', col = 'blue', xlab = '# of nearest neighbors', ylab = 'Mean squared errors',
     main = '# of nearest neighbors vs. MSE')
```



```r
which.min(mses)
```

```
## [1] 2
```

```r
round(mses[which.min(mses)], 3)
```

```
## [1] 1.583
```

KNN reached its optimal performance at $k = 2$, even though its MSE is still higher than that of the best OLS model.

Comparing All Models

One of the most straightforward methods to evaluate the accuracy of a model is to compare their mean squared errors (MSE). Here, we will list all the models, ranked by their MSE from low to high. Note that all models (if needed) were fit on the same training dataset, and tested on the same validation dataset.

Model	MSE
OLS	0.663
Random Forest	0.684
Boosting (tuned)	0.695
Bagging	0.727
Lasso	0.735
XGboost	0.742
Ridge	0.753
Boosting	0.787
Tree	1.102
KNN	1.583
Neural network	2.817

OLS had the least MSE, and is one of the easier models to interpret. The best linear model has a fairly low MSE of `0.663`, and includes only four predictors.

Conclusion

This paper faces a tradeoff of whether the two midterm grades should be used as predictors. If they were not used, the model becomes more meaningful because teachers can predict students' performance on the final grade even before the semester starts. Students with educational disadvantages can be identified in advance for teachers to make personalized adjustments, and help them succeed in the class. However, the project quickly identifies that it is infeasible to make a relatively accurate prediction of final grade without taking into account of midterm grades. Adding these two predictors limits teachers' ability to project students' performance early, and their decisions have to rely on past midterm grades.

This is perhaps an unsurprising result, because student performances are volatile, and do not heavily depend on students' background and past experience at a middle school level. It is not rare to see a middle school student perform well in one year, and bad in another. Data were one of the most challenging part of the research, as they were mostly collected from questionnaires answered by middle school students. In addition, many questions were subjective, as students can have wide range definition of "going out a lot with friends". The paper still provides meaningful feedback for teachers, parents and students because they can use a simple linear regression model to estimate their final grades. In particular, for students who are aiming for a particular letter grade, or those who simply want to pass the course, the model gives them a sense of whether their goals are achievable.

To summarize the best model, the first two midterm grades unsurprisingly have dominant impact, as they are a huge portion of the final grade. Being absent from classes and spending more time hanging out with friends also tend to decrease the final grade. Even though the linear regression yields the lowest MSE, other models have their own advantages, such as being easier to interpret, or performing feature selections. Furthermore, all other models depict that the first two midterm grades have tremendous impact on the final grade. The paper also shows that MSE can be significantly reduced by tuning the parameters.