Project Document

Part I: Exploratory Data Analysis

About the Data

Our data was obtained via UCI's Machine Learning Repository. The data is a multivariate set designed to explore student performance tied to various predictors during a collection period from 2005-2006. The data is split into two sets: Mathematics (student-mat.csv) and Portuguese (student-por.csv). These are the two subjects where records of student's attending two pubic school's from the Alentejo region of Portugal performance (our outcome y) were recorded. Predictor variables include a range of demographic, social, health, and school related attributes.

The data was utilized by a paper published in 2008 titled "Using data mining to predict secondary school performance". The study's goal was to use BI/DM techniques to build a model that accurately predicted student performance given predictor variables that provided the best accuracy. Below, we will conduct an EDA exploring and cleaning this data set prior to conducting a replication of their study while critiquing their process and adding/removing anything we deem necessary to result in the best models for our given data and prior proposed research goal.

Loading our Libraries

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(car)
```

```
Loading required package: carData

Attaching package: 'car'

The following object is masked from 'package:dplyr':
    recode

The following object is masked from 'package:purrr':

    some
```

Loading our Data

```
# loading in both of our data sets, we will title them by their
# subject
math <- read.csv("student-mat.csv", sep =  ";")
portuguese <- read.csv("student-por.csv", sep = ";") # our data was seperated by semi colons
# instead of the traditional comma
```

Understanding the Data Structure

Prior to inspecting and cleaning our data, it is important we fully encapsulate what each column, row, and value mean.

```r
head(math)
```

```
  school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob     reason
1     GP   F  18       U     GT3       A    4    4  at_home  teacher     course
2     GP   F  17       U     GT3       T    1    1  at_home    other     course
3     GP   F  15       U     LE3       T    1    1  at_home    other      other
4     GP   F  15       U     GT3       T    4    2   health services       home
5     GP   F  16       U     GT3       T    3    3    other    other       home
6     GP   M  16       U     LE3       T    4    3 services    other reputation
  guardian traveltime studytime failures schoolsup famsup paid activities
1   mother          2         2        0       yes     no   no         no
2   father          1         2        0        no    yes   no         no
3   mother          1         2        3       yes     no  yes         no
4   mother          1         3        0        no    yes  yes        yes
5   father          1         2        0        no    yes  yes         no
6   mother          1         2        0        no    yes  yes        yes
  nursery higher internet romantic famrel freetime goout Dalc Walc health
1     yes    yes       no       no      4        3     4    1    1      3
2      no    yes      yes       no      5        3     3    1    1      3
3     yes    yes      yes       no      4        3     2    2    3      3
4     yes    yes      yes      yes      3        2     2    1    1      5
5     yes    yes       no       no      4        3     2    1    2      5
6     yes    yes      yes       no      5        4     2    1    2      5
  absences G1 G2 G3
1        6  5  6  6
2        4  5  5  6
3       10  7  8 10
4        2 15 14 15
5        4  6 10 10
6       10 15 15 15
```

```r
head(portuguese)
```

```
  school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob     reason
1     GP   F  18       U     GT3       A    4    4  at_home  teacher     course
2     GP   F  17       U     GT3       T    1    1  at_home    other     course
```

```
3     GP   F  15        U     LE3       T    1    1  at_home    other      other
4     GP   F  15        U     GT3       T    4    2   health services       home
5     GP   F  16        U     GT3       T    3    3    other    other       home
6     GP   M  16        U     LE3       T    4    3 services    other reputation
  guardian traveltime studytime failures schoolsup famsup paid activities
1   mother          2         2        0       yes     no   no         no
2   father          1         2        0        no    yes   no         no
3   mother          1         2        0       yes     no   no         no
4   mother          1         3        0        no    yes   no        yes
5   father          1         2        0        no    yes   no         no
6   mother          1         2        0        no    yes   no        yes
  nursery higher internet romantic famrel freetime goout Dalc Walc health
1     yes    yes       no       no      4        3     4    1    1      3
2      no    yes      yes       no      5        3     3    1    1      3
3     yes    yes      yes       no      4        3     2    2    3      3
4     yes    yes      yes      yes      3        2     2    1    1      5
5     yes    yes       no       no      4        3     2    1    2      5
6     yes    yes      yes       no      5        4     2    1    2      5
  absences G1 G2 G3
1        4  0 11 11
2        2  9 11 11
3        6 12 13 12
4        0 14 14 14
5        0 11 13 13
6        6 12 12 13
```

Both of our data sets are structured the same with the same column and row variables as well as structured values. This will help make implementing any cleaning and modeling simpler.

Variables & Values

Referring back to the original paper, there are 33 columns of interest. Those variables are listed below alongside their values, with explanation and clarification as needed, below:

For a visual example, I have printed a random row to show how the values are presented as they are explained below

```r
math[3, ]
```

```
  school sex age address famsize Pstatus Medu Fedu    Mjob  Fjob reason
3     GP   F  15       U     LE3       T    1    1 at_home other  other
  guardian traveltime studytime failures schoolsup famsup paid activities
```

```
3   mother          1         2        3        yes     no  yes          no
  nursery higher internet romantic famrel freetime goout Dalc Walc health
3     yes    yes      yes       no      4        3     2    2    3      3
  absences G1 G2 G3
3       10  7  8 10
```

`school` - Binary values of either `GP` (Gabriel Pereira) or `MS` (Mousinho da Silveira) of which school a student attended.

`sex` - Binary values of either `F` (female) or `M` (male) regarding a students sex.

`age` - Numeric value of a students age from 15 - 22.

`address` - Binary values of either `U` (urban) or `R` (rural) regarding a students home address.

`famsize` - Binary values of either `LE3` (less than or equal to 3 family members) or `GT3` (greater than 3 family members).

`Pstatus` - Binary values of either `T` (parents are living together) or `A` (parents are living apart) for parents living status.

`Medu` - Leveled integer value of range 0-4 with 0 reflecting no education or below primary completion, 1 reflecting completion of primary education (up to 4th grade), 2 reflecting completion of 5-9th grade education, 3 reflecting completion of secondary education, and 4 reflecting higher education (college degree or higher) of a students' mother's education.

`Fedu` - Leveled integer value of range 0-4 with 0 reflecting no education or below primary completion, 1 reflecting completion of primary education (up to 4th grade), 2 reflecting completion of 5-9th grade education, 3 reflecting completion of secondary education, and 4 reflecting higher education (college degree or higher) of a students' father's education.

`Mjob` - Nominal values for a students' mother's job classified as `teacher`, `health` (any care related profession), `services` (any administrative or police related field), `at_home` (none), `other` (not stated).

`Fjob` - Nominal values for a students' father's job classified as `teacher`, `health` (any care related profession), `services` (any administrative or police related field), `at_home` (none), `other` (not stated).

`reason` - Nominal values for a student's reason for school selection as either `home` (close to home), `reputation`, `course` (valued courses provided), `other` (reason not stated).

`guardian` - Nominal values for who the primary caregiver of the student is as either `mother`, `father`, or `other`. Reason for why both parents cannot be listed is not stated.

`traveltime` - Leveled integer values representing travel time to school on a scale of 1-4, `1` reflecting <15 minutes, `2` reflecting 15-30 minutes, `3` reflecting 30 minutes to 1 hour, `4` reflecting >1 hour travel time.

`studytime` - Leveled integer values representing average weekly study time reported by the student on a scale of 1-4, `1` reflecting <2 hours, `2` reflecting 2-5 hours, `3` reflecting 5-10 hours, `4` reflecting >10 hours study time.

`failures` - Leveled integer values representing the number of classes a student has failed prior to enrolling in this course with a scale of 1-4, each reflecting the amount of courses failed, 4 being > or = 4 failed classes.

`schoolsup` - Binary value for either `yes` or `no` student receiving additional educational support outside of the course. Not specified if this is inclusive of in-school tutoring and/or support such as services for language gaps or speech development.

`famsup` - Binary value for either `yes` or `no` student receiving additional family educational support outside of the course (family members assist in helping the student with studying or homework). If `yes`, we are assuming a student receives help from family generally.

`paid` - Binary value for either `yes` or `no` student is paying for additional educational support for the course.

`activities` - Binary value for either `yes` or `no` student is participating in extra-curricular activities.

`nursery` - Binary value for either `yes` or `no` student attended nursery school in the past (equivalent to pre-school education in America).

`higher` - Binary value for either `yes` or `no` student wants to pursue higher education courses in the future.

`internet` - Binary value for either `yes` or `no` student has internet access at home.

`romantic` - Binary value for either `yes` or `no` student is currently in a romantic relationship.

`famrel` - Leveled integer values scaled from 1-5 for a students quality of family relationships, `1` being very bad and `5` being excellent.

`freetime` - Leveled integer values scaled from 1-5 for a students free time after school, `1` being very little free time and `5` being lots of free time.

`goout` - Leveled integer values scaled from 1-5 of how often a student goes out with freinds, `1` being not often and `5` being very often.

`Dalc` - Leveled integer values scaled from 1-5 of how often a student consumes alcohol on a weekday, `1` being not often and `5` being very often.

`Walc` - Leveled integer values scaled from 1-5 of how often a student consumes alcohol on a weekend, `1` being not often and `5` being very often.

`health` - Leveled integer values scaled from 1-5 of a students health status, `1` being bad and `5` being very good.

`absences` - Numeric values of the number of day absences the student has from the course so far, i.e. a value of `5` would mean the student has been absent from the class a total of 5 times.

`G1` - Leveled integer values scaled from 0-20 of a students first period grade in the course(period is a trimester in American equivalency).

`G2` - Leveled integer values scaled from 0-20 of a students second period grade in the course.

`G3` - Leveled integer values scaled from 0-20 of a students third period grade in the course.

Classes & Values

Given the review of our variables and their values, we should expect many integer columns and character columns, which we will change to factors.

```r
str(math)
```

```
'data.frame':	395 obs. of  33 variables:
 $ school    : chr  "GP" "GP" "GP" "GP" ...
 $ sex       : chr  "F" "F" "F" "F" ...
 $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
 $ address   : chr  "U" "U" "U" "U" ...
 $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
 $ Pstatus   : chr  "A" "T" "T" "T" ...
 $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
 $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
 $ Mjob      : chr  "at_home" "at_home" "at_home" "health" ...
 $ Fjob      : chr  "teacher" "other" "other" "services" ...
 $ reason    : chr  "course" "course" "other" "home" ...
 $ guardian  : chr  "mother" "father" "mother" "mother" ...
 $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
 $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
 $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
 $ schoolsup : chr  "yes" "no" "yes" "no" ...
 $ famsup    : chr  "no" "yes" "no" "yes" ...
 $ paid      : chr  "no" "no" "yes" "yes" ...
 $ activities: chr  "no" "no" "no" "yes" ...
 $ nursery   : chr  "yes" "no" "yes" "yes" ...
 $ higher    : chr  "yes" "yes" "yes" "yes" ...
 $ internet  : chr  "no" "yes" "yes" "yes" ...
 $ romantic  : chr  "no" "no" "no" "yes" ...
 $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
 $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
 $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
 $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
 $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
 $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
 $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
 $ G1        : int  5 5 7 15 6 15 12 6 16 14 ...
 $ G2        : int  6 5 8 14 10 15 12 5 18 15 ...
 $ G3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```r
str(portuguese)
```

```
'data.frame':   649 obs. of  33 variables:
 $ school    : chr  "GP" "GP" "GP" "GP" ...
 $ sex       : chr  "F" "F" "F" "F" ...
 $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
 $ address   : chr  "U" "U" "U" "U" ...
 $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
 $ Pstatus   : chr  "A" "T" "T" "T" ...
 $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
 $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
 $ Mjob      : chr  "at_home" "at_home" "at_home" "health" ...
 $ Fjob      : chr  "teacher" "other" "other" "services" ...
 $ reason    : chr  "course" "course" "other" "home" ...
 $ guardian  : chr  "mother" "father" "mother" "mother" ...
 $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
 $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
 $ failures  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ schoolsup : chr  "yes" "no" "yes" "no" ...
 $ famsup    : chr  "no" "yes" "no" "yes" ...
 $ paid      : chr  "no" "no" "no" "no" ...
 $ activities: chr  "no" "no" "no" "yes" ...
 $ nursery   : chr  "yes" "no" "yes" "yes" ...
 $ higher    : chr  "yes" "yes" "yes" "yes" ...
 $ internet  : chr  "no" "yes" "yes" "yes" ...
 $ romantic  : chr  "no" "no" "no" "yes" ...
 $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
 $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
 $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
 $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
 $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
 $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
 $ absences   : int  4 2 6 0 0 6 0 2 0 0 ...
 $ G1         : int  0 9 12 14 11 12 13 10 15 12 ...
 $ G2         : int  11 11 13 14 13 12 12 13 16 12 ...
 $ G3         : int  11 11 12 14 13 13 13 13 17 13 ...
```

This gives us a general idea of what each column look, and the class which is all integers and character columns.

Cleaning Up

```r
sum(is.na(math))
```

```
[1] 0
```

```r
sum(is.na(portuguese))
```

```
[1] 0
```

The original data from the survey was processed and certain variables were excluded by the author of the paper due to lack of discriminative value. To verify that our data sets are clean, we check to see if there are any missing values.

Our general approach to this project involves replicating some of the models used in the paper. The paper would predict student success using the G3 score, in one of three forums: binary classification, classification with five levels, and regression on the 0-20 scale. To ease the replication process we will create two new columns to represent the forum we want our output to be in:

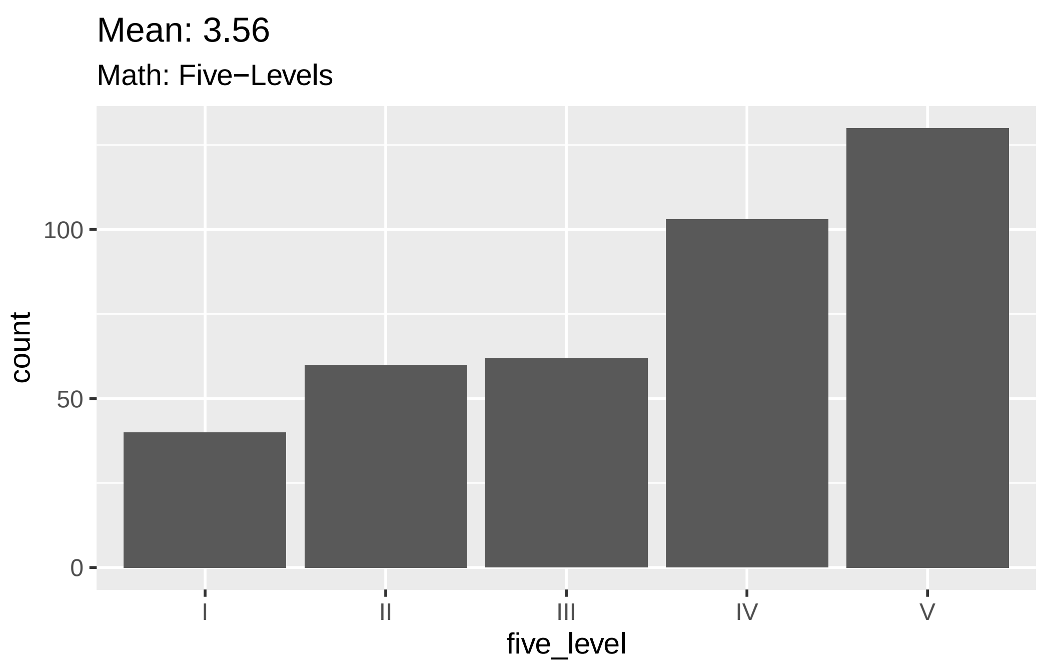
```r
math <- math |>
  mutate(five_level=case_when(
    G3 > 15 ~ "I",
    G3 >= 14 ~ "II",
    G3 >=12 ~ "III",
    G3 >=10  ~ "IV",
    G3 < 10 ~ "V"
  )) |>
    mutate(pass_fail=case_when(
      G3>=10 ~ "Pass",
      G3<10 ~ "Fail"
    )) -> math2


portuguese |>
  mutate(five_level=case_when(
    G3 > 15 ~ "I",
    G3 >= 14 ~ "II",
    G3 >=12 ~ "III",
    G3 >=10  ~ "IV",
    G3 < 10 ~ "V"
  )) |>
    mutate(pass_fail=case_when(
      G3>=10 ~ "Pass",
      G3<10 ~ "Fail"
    )) -> portuguese2

math2$five_level<-factor(math2$five_level)
portuguese2$five_level<-factor(portuguese2$five_level)

#Below we change characters to factors in the dataset we plan on using for our models
portuguese2 <- portuguese2 %>%
  mutate(across(c(pass_fail, romantic, internet, higher, nursery, activities, paid, famsup, s
```

Correlations & Distribution

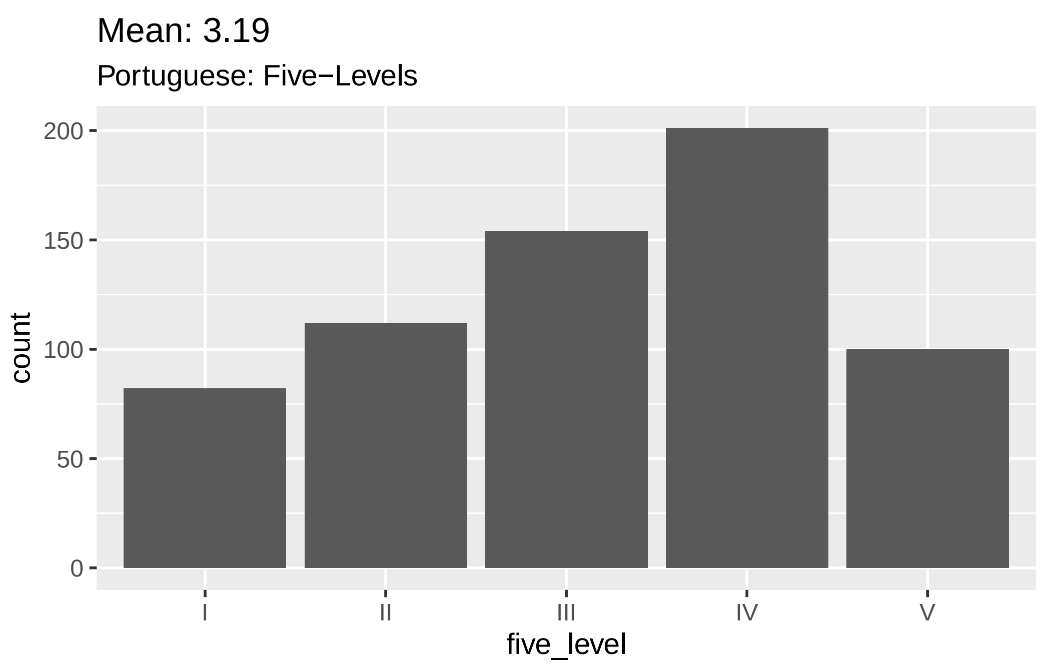
Means & Distributions

Here we exam visual distributions

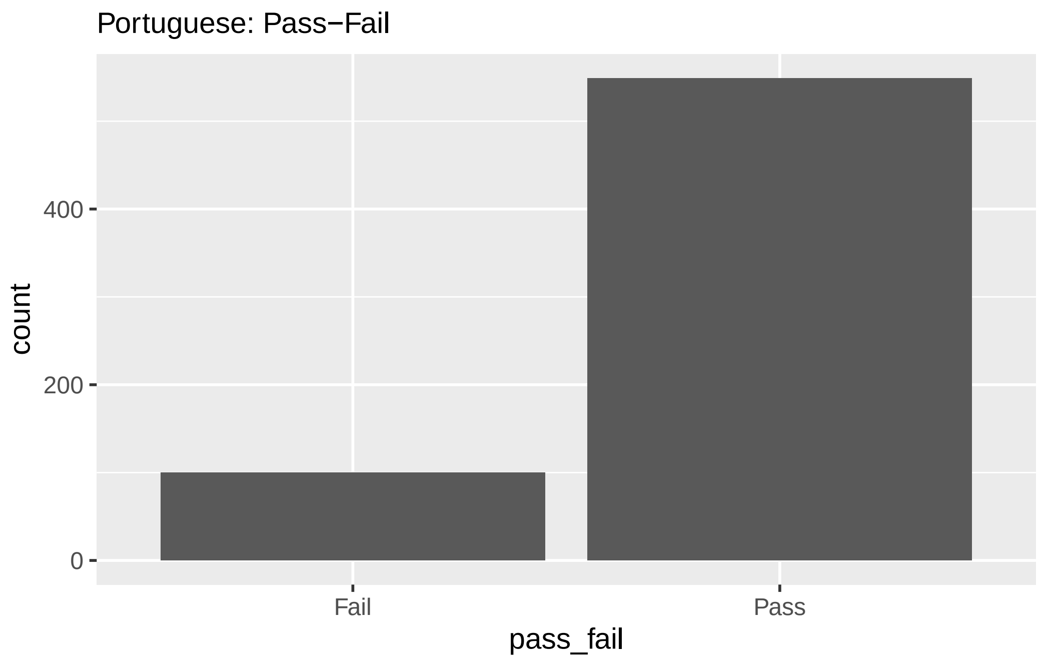
```r
library(ggplot2)
ggplot(data=math2, aes(x=five_level))+
  geom_bar() +
  ggtitle(paste("Mean:", round(mean(
    as.numeric(math2$five_level)), 2))) +
  labs(subtitle = "Math: Five-Levels")
```



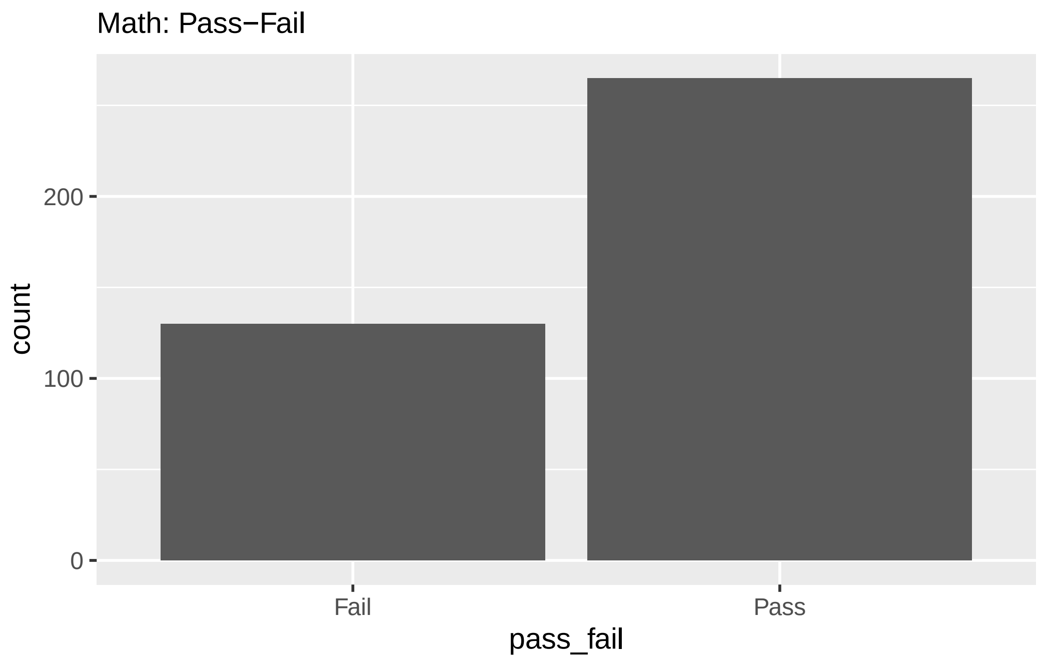
```r
ggplot(data=portuguese2, aes(x=five_level))+
  geom_bar() +
  ggtitle(paste("Mean:", round(mean(
    as.numeric(portuguese2$five_level)), 2))) +
  labs(subtitle = "Portuguese: Five-Levels")
```



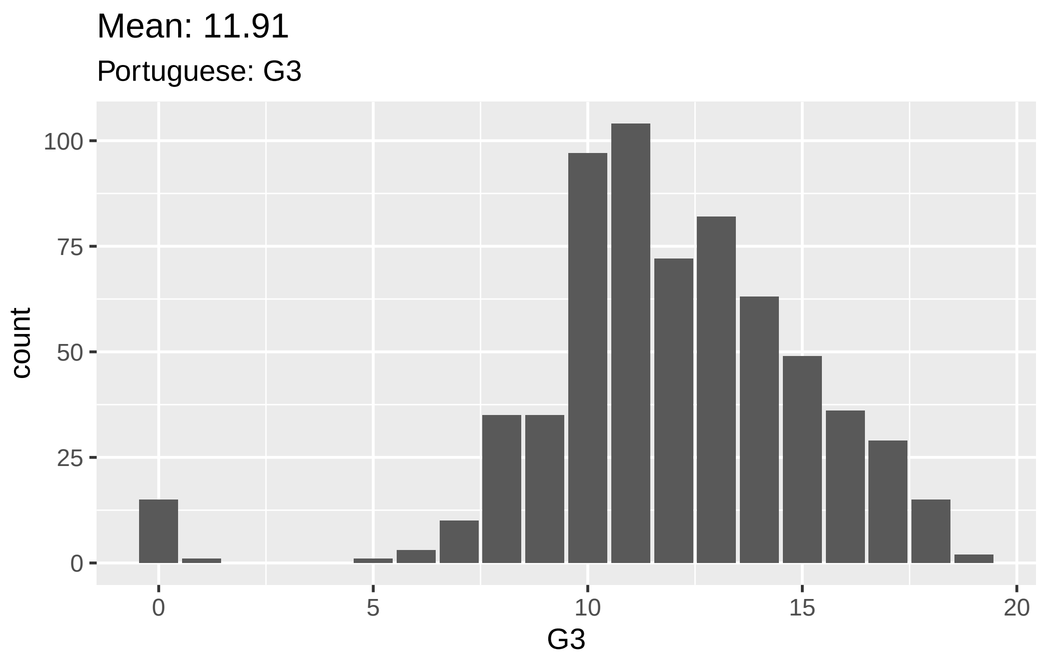
```
ggplot(data=portuguese2, aes(x=pass_fail))+
  geom_bar() +
  labs(subtitle = "Portuguese: Pass-Fail")
```



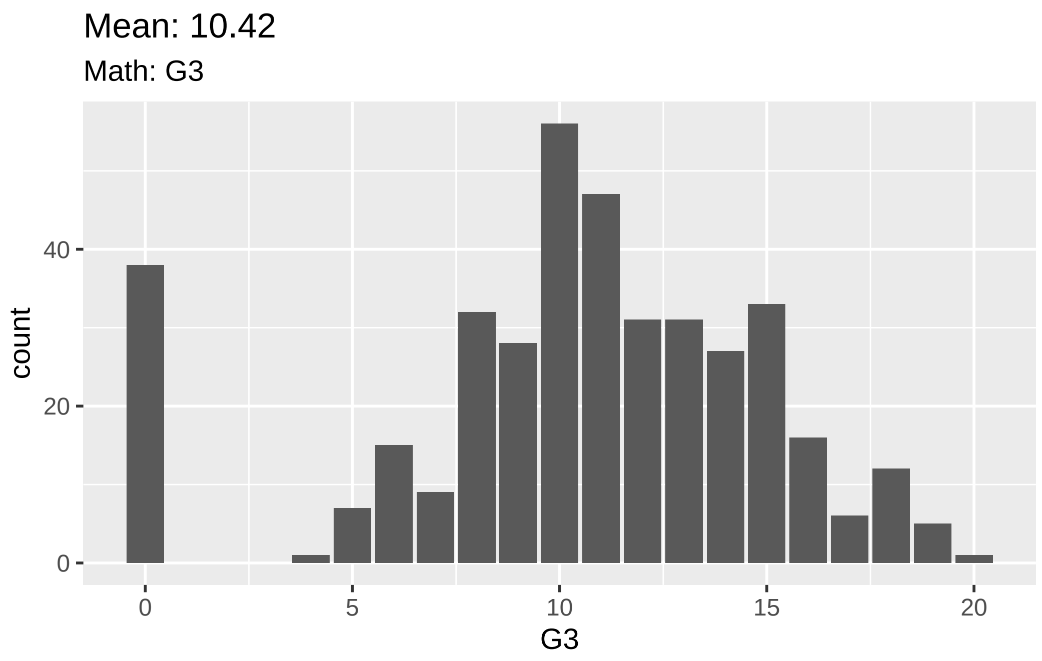
```
ggplot(data=math2, aes(x=pass_fail))+
  geom_bar() +
  labs(subtitle = "Math: Pass-Fail")
```



```
ggplot(data=portuguese2, aes(x=G3))+
  geom_bar() +
  ggtitle(paste("Mean:", round(mean(
    as.numeric(portuguese2$G3)), 2))) +
  labs(subtitle = "Portuguese: G3")
```

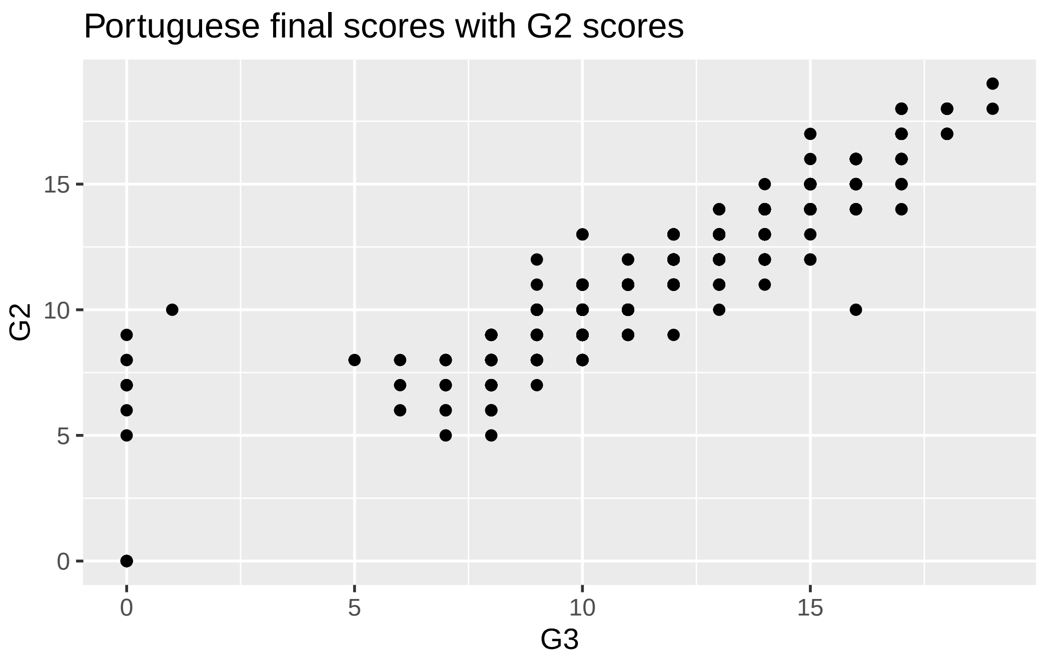


```
ggplot(data=math2, aes(x=G3))+
  geom_bar() +
  ggtitle(paste("Mean:", round(mean(
    as.numeric(math2$G3)), 2))) +
  labs(subtitle = "Math: G3")
```



Correlations & Plots

```r
ggplot(data=portuguese2, aes(x=G3, y=G2))+
  geom_point()+
  labs(title="Portuguese final scores with G2 scores")
```



Clear correlation between second period grades and final grade, shows the in-balance of models that use G2 as a predictor vs those that don't. We will dive into collinearity assumptions with tests in the statistical analysis section.

Statistical Analysis

Its important to note any patterns or anomalies with our data. We will look at possible outliers and quickly summarize G3 (our predicted variable).

```
summary(portuguese2$G3)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   10.00   12.00   11.91   14.00   19.00 
```

```
sd(portuguese2$G3)
```

```
[1] 3.230656
```

```
summary(math2$G3)
```

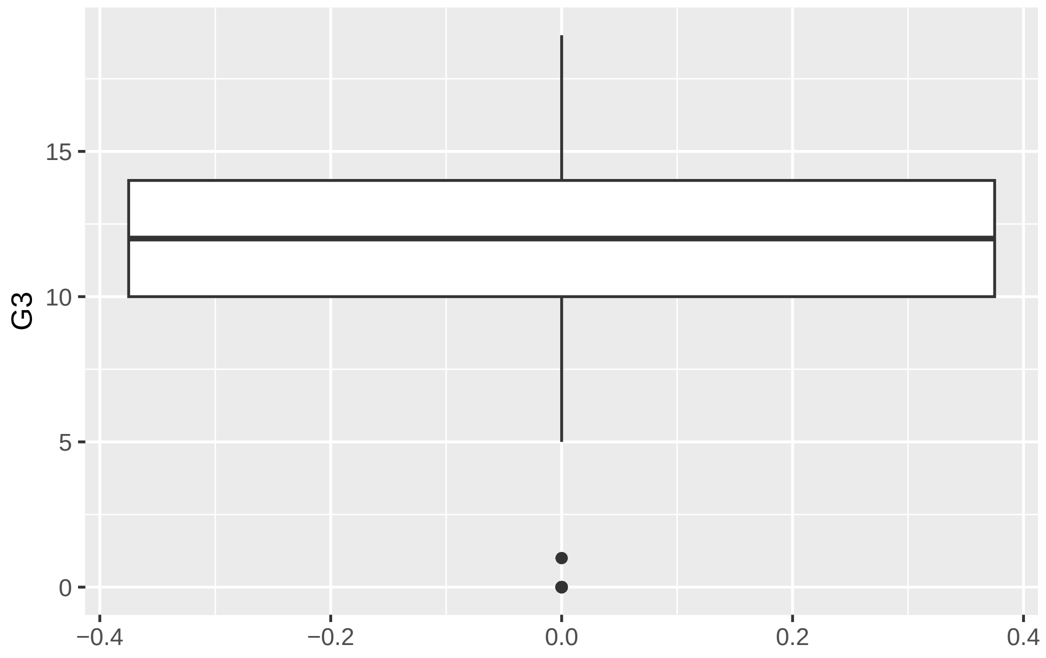
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00    8.00   11.00   10.42   14.00   20.00 
```

```
sd(math2$G3)
```

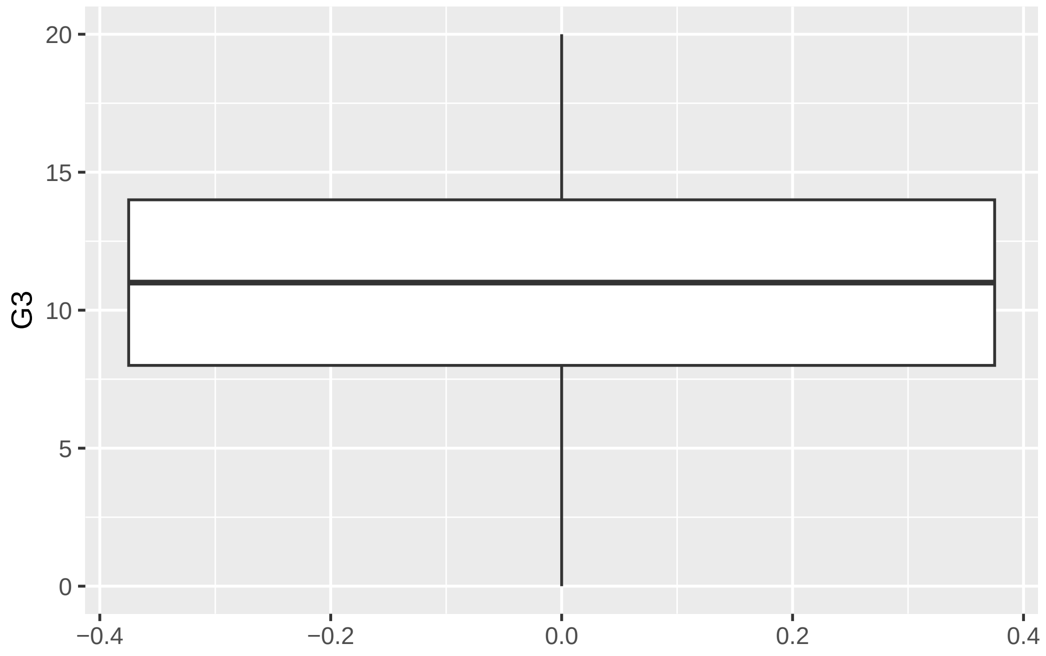
```
[1] 4.581443
```

It seems most students pass, with math scores being slightly lower on average.

```
ggplot(portuguese2, aes(y=G3)) + geom_boxplot()
```



```r
ggplot(math2, aes(y=G3)) + geom_boxplot()
```



It seems our Portuguese course has two values that are outliers, but we will not remove them as values due to their predictive ability for students who may fail a class. Also, tree-based models are not affected by outliers.

There are a few ways to test for collinearity with variables: VIF, visualization on a scatter plot, or using a pairwise approach and testing its correlation.

```r
# testing using VIF
lm_for_VIF <- lm(G3 ~ G1 + G2, data=portuguese2)

vif(lm_for_VIF)
```

```
      G1       G2
3.971299 3.971299
```

A VIF score of 1 is typically indicates no correlation with other predictors. A VIF of 10 is generally considered too high. However, its also important to consider what kind of model we are creating. We are creating prediction models, so we would consider a value of about ~3.97 to be relatively moderate. Essentially, utilization of both predictors G1 and G2 in our model is not likely to cause issues with predicting our outcome, G3.

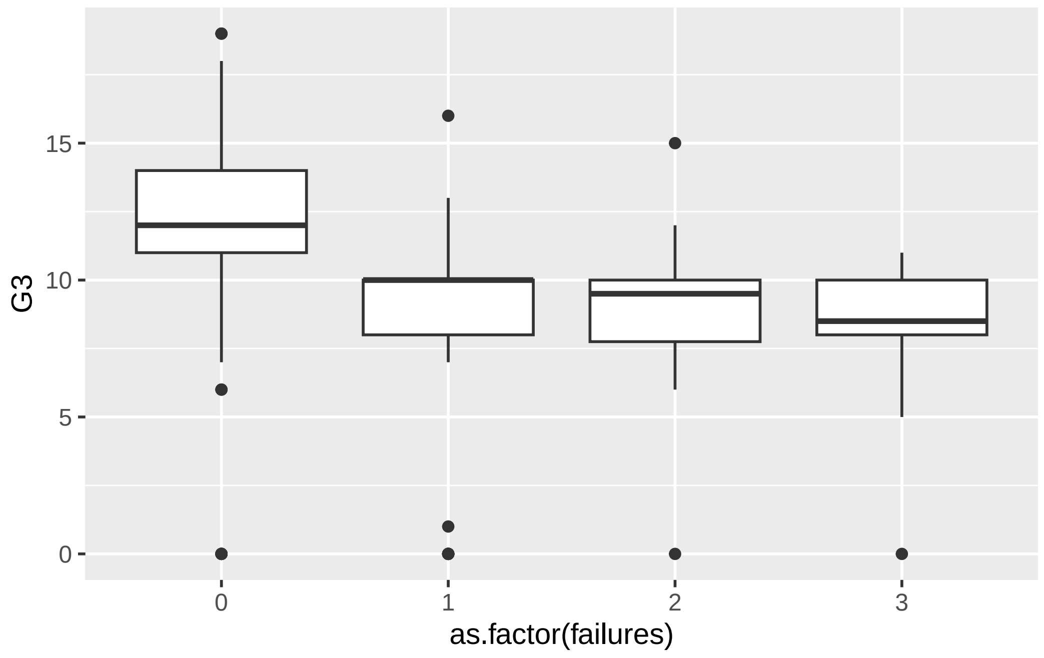
```r
grades <- portuguese2[, c("G1", "G2", "G3")]

cor_matrix <- cor(grades, use = "complete.obs")
print(cor_matrix)
```

```
          G1        G2        G3
G1 1.0000000 0.8649816 0.8263871
G2 0.8649816 1.0000000 0.9185480
G3 0.8263871 0.9185480 1.0000000
```

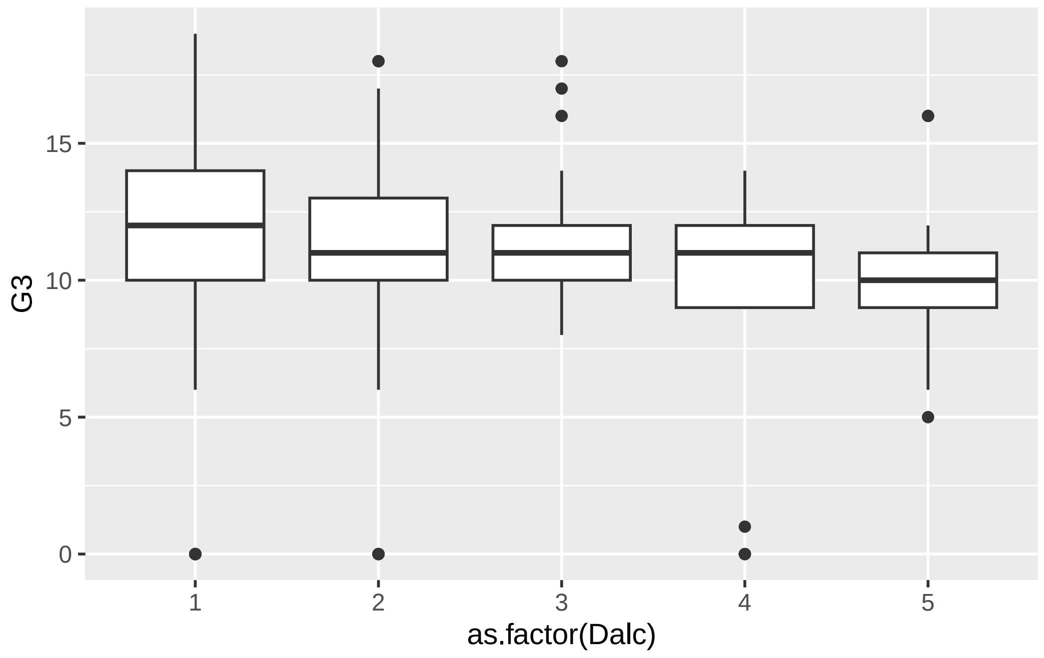
However, now that we have run a correlation matrix, it is displaying very strong correlation between our variables G1, G2, and G3. This confirms high collinearity among them, which would cause an increase in standard errors in our regression models.

Exploratory Graphs

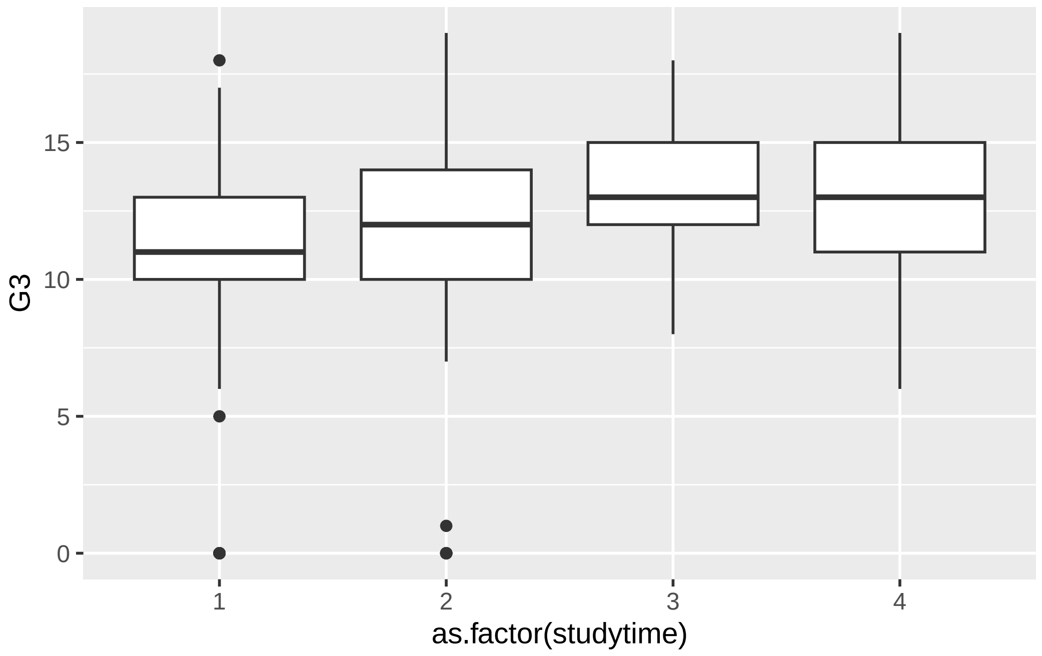
```r
ggplot(data=portuguese2, aes(x=as.factor(failures), y=G3))+
  geom_boxplot()
```



```
ggplot(data=portuguese2, aes(x=as.factor(Dalc), y=G3))+
  geom_boxplot()
```



```r
ggplot(data=portuguese2, aes(x=as.factor(studytime), y=G3))+
  geom_boxplot()
```



We wanted to look at the relationship between some predictor variables that we thought may have a strong relationship to G3 final grades. We do see small correlations like lower grades as alcohol consumption increases, and higher grades as study time increases.

Review of Plan to fit Models

Our general approach to this project will be to recreate some of the models created in the paper connected with this data set and additionally create some of our own. In the paper, the classification/regression methods tried to predict G3 (passing/failing Portuguese and Math) in 3 supervised approaches:

- Binary (Pass/Fail) - Pass is considered `G3`>/=10; else is Fail
- 5-level Classification
- Regression (as is current `G3` column, scale 0-20)

The data was then modeled using 5 data mining algorithm:

- Neural Networks (NN) - E = 100 training epochs utilizing BFGS algorithm

- Support Vector Machines (SVM) - SMO algorithm utilized
- Decision Trees (DT) - node splitting utilized to reduce sum of squares
- Random Forest (RF) - default parameters, T = 500
- Naive Predictor (NV) - (1) baseline of G3 = G2, (2) G3 = G1, (3) most common class or mean

These 4 DM's were compared against the baseline naive predictor (NV) model. Additionally it was noted that 20 runs of 10-fold cross validation were applied to each configuration.

Each model was run with each of 3 input setups. The setups included (A), all variables minus G3, (B) all variables minus G2 and G3, and (C) neither G2, G3, or G1. This means that model (A) is utilizing the prediction power of G1 and G2 grades in their model for accurate prediction of G3 grades, where setup (B) utilizes solely G1 grades to predict G3, and setup (C) uses none of the trimester grades as a predictor for G3.

The reason the authors made this decision was due to the likelihood of high collinearity between G1, G2, and G3. The usage of Naive Predictors as three input configurations is to account for this potential (and likely) collinearity.

Furthermore, more pre-processing was established with nominal variables as well. The authors decided to transform them into a *1-of-C* encoding with all attributes being standardized to a 0 mean and a one standard deviation.

Modeling Procedures

According to the paper that we are replicating, their goal was to "give a simple description that summarizes the best DM models". The authors used this model as it was collected, essentially creating a model that could be used to predict student outcomes once the student was in their third trimester of the class. We intend to work with the same desired prediction and usage of the variables.

In order to differentiate our model from theirs while still replicating part of the study, we wish to take the approach that the model is used prior to a students choice to enroll in a class. Essentially, our model see's to predict a students outcome in the class using variables that are known prior and during class enrollment so that a user could predict their grade before the third trimester. Essentially, this means removing G1 and G2 as predictive variables.

We understand the choice of the authors to use an A-B-C subset method, however, given time constraint, we decided to solely select subset (C). Not including G1 or G2 would indicate a model that can predict a students grade for trimester 3 without considering trimester 1 and 2.

We will prioritize replicating two of their original models - Decision Tree (DT) and Random Forest (RF). However, we plan to add three of our models: Support Vector Machine, LASSO Regression (LR), and Logistic Regression with PCA. We will utilize the binary (pass/fail) outcome on all models. We will predict outcomes for the Portuguese course (which has more observations, 649 v. 395) due to the time constraint, and only reproduce models with input C, as we want to view prediction power without the utilization of G1 and G2 grades.

Note on data splitting:

The paper states: "To access the predictive performances, 20 runs of a 10- fold cross-validation (Hastie et al. 2001) (in a total of 200 simulations) were applied to each configuration. Under such scheme, for a given run the data is randomly divided in 10 subsets of equal size. Sequentially, one different subset is tested (with 10% of the data) and the remaining data used to fit the DM technique. At the end of this process, the evaluated test set contains the whole dataset, although 10 variations of the same DM model are used to create the predictions."

We will do an initial 20-80 split and run cross-validation on most of our models.

In summary, we intend to replicate this study, with key differences: we intend to use solely setup (C), we intend to replicate Decision Tree and Random Forest from the original paper but create three models of our own. We will use a binary response variable of pass/fail.

Part 2: Model Fitting - Binary; Set Up C

Modeling Set Up - Creating Five Level, and Pass/Fail

```
portuguese |>
    mutate(pass_fail=case_when(
      G3>=10 ~ "Pass",
      G3<10 ~ "Fail"
    )) -> portuguese2

portuguese2 <- portuguese2 %>%
  mutate(across(c(pass_fail, romantic, internet, higher, nursery, activities, paid, famsup, s

# create training and testing data for PCA Logistic Regression, DT and RF
set.seed(627)
train.pct <- 0.8
Z <- sample(nrow(portuguese2), floor(train.pct*nrow(portuguese2)))
portuguese.data <- portuguese2[Z, ]
holdout.data <- portuguese2[-Z, ]

# standardize and one-hot-encoding for LASSO and SVM:
nominals <- c("Mjob", "Fjob", "reason", "guardian",
                  "address", "famsize", "Pstatus", "sex", "school")
numeric <- c("age", "absences")
neither <- c("Medu","Fedu","traveltime","studytime","failures",
                "famrel","freetime","goout","Dalc","Walc","health",
                "romantic","internet","higher","nursery","activities",
                "paid","famsup","schoolsup")

scaled_train <- scale(portuguese.data[, numeric])
scaled_holdout <- scale(holdout.data[, numeric],
                        center = attr(scaled_train, "scaled:center"),
                        scale  = attr(scaled_train, "scaled:scale"))
# scaling numerics, required outside research

# for one-hot-encoding, converst to 1-of-c using all nominals but removes our default interc
training_nominals <- model.matrix(~ . - 1, data = portuguese.data[, nominals])
holdout_nominals <- model.matrix(~ . - 1, data = holdout.data[, nominals])

# combines all of our 1 of c and numeric scaled variables together
portuguese.data_scaled <- cbind(pass_fail = portuguese.data$pass_fail,
  scaled_train, portuguese.data[, neither], training_nominals)
holdout.data_scaled <- cbind(pass_fail = holdout.data$pass_fail,
  scaled_holdout, holdout.data[, neither], holdout_nominals)
```

Note: we will use cross validation (as requested in the project instructions) and 20% testing 80% training data. As per the paper: "Before fitting the models, some preprocessing was required by the NN and SVM models. The nominal variables (e.g `Mjob`) were transformed into a 1-of-C encoding and all attributes were standardized to a zero mean and one standard deviation" (Hastie et al. 2001). So by doing so, we are replicating the paper more closely. This structure is only applied to SVM, but also applied to our LASSO as upon our research on how to compute 1-of-c, we discovered its best practice to also apply this to our LASSO because it is penalty-based model.

Model 1: Logistic Regression with PCA - Binary Outcome

Below is our model, followed by assumptions to assure reliability.

```
# remember we are removing G3 and five_level since those are for other outcomes
library(dplyr)
lrpca <- glm(pass_fail ~ . -G3 -G2 -G1, data = portuguese.data,
          family = binomial)
```

Lets check and see how it is performing pre-PCA:

```
summary(lrpca)
```

```
Call:
glm(formula = pass_fail ~ . - G3 - G2 - G1, family = binomial,
    data = portuguese.data)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.75513    2.81507  -0.623  0.53297
schoolMS          -2.24567    0.40537  -5.540 3.03e-08 ***
sexM              -0.24249    0.38527  -0.629  0.52908
age                0.32909    0.15802   2.083  0.03729 *
addressU           0.16536    0.34922   0.474  0.63584
famsizeLE3         0.43037    0.37651   1.143  0.25301
PstatusT           0.02377    0.50428   0.047  0.96241
```

```
Medu             -0.12384    0.20545  -0.603  0.54665
Fedu              0.42658    0.21305   2.002  0.04526 *
Mjobhealth       -0.37835    0.76013  -0.498  0.61866
Mjobother         0.02871    0.38531   0.075  0.94061
Mjobservices      0.29942    0.51563   0.581  0.56145
Mjobteacher       0.71178    0.86049   0.827  0.40814
Fjobhealth       -2.31779    1.09859  -2.110  0.03488 *
Fjobother        -0.90121    0.64050  -1.407  0.15941
Fjobservices     -1.46644    0.68504  -2.141  0.03230 *
Fjobteacher      -2.39491    1.18783  -2.016  0.04378 *
reasonhome        0.35360    0.43465   0.814  0.41593
reasonother       0.31678    0.45758   0.692  0.48875
reasonreputation  0.54396    0.51262   1.061  0.28863
guardianmother   -0.66137    0.40632  -1.628  0.10359
guardianother     0.19517    0.79965   0.244  0.80718
traveltime        0.25544    0.22084   1.157  0.24741
studytime         0.19951    0.21650   0.922  0.35677
failures         -1.07777    0.22975  -4.691 2.72e-06 ***
schoolsupyes     -0.86161    0.53200  -1.620  0.10532
famsupyes         0.41928    0.32732   1.281  0.20021
paidyes          -1.31506    0.57899  -2.271  0.02313 *
activitiesyes     0.39895    0.33580   1.188  0.23481
nurseryyes       -0.33692    0.40088  -0.840  0.40066
higheryes         1.39555    0.42529   3.281  0.00103 **
internetyes      -0.17765    0.36459  -0.487  0.62608
romanticyes      -0.50700    0.31781  -1.595  0.11064
famrel           -0.01271    0.15398  -0.083  0.93424
freetime         -0.11001    0.15707  -0.700  0.48369
goout             0.04354    0.15397   0.283  0.77735
Dalc              0.02728    0.20017   0.136  0.89158
Walc             -0.25789    0.16788  -1.536  0.12448
health           -0.07436    0.12111  -0.614  0.53921
absences         -0.10672    0.03618  -2.950  0.00318 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 459.55  on 518  degrees of freedom
Residual deviance: 305.82  on 479  degrees of freedom
AIC: 385.82

Number of Fisher Scoring iterations: 6
```

Our model is showing an residual deviance of 305.82, so our model is so and so on the portion of the variance in pass_fail. This was expected when removing G2 and G1.

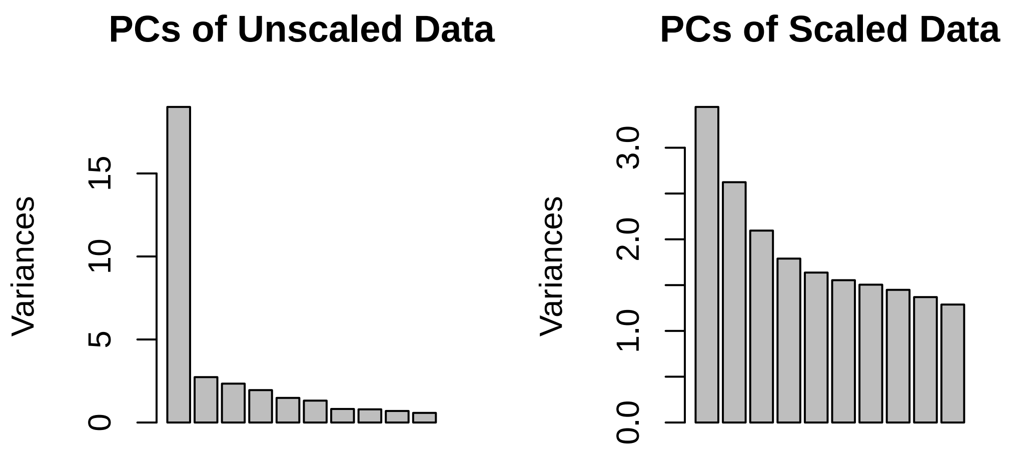
Now lets utilize PCA's to create a more tuned model:

```r
portuguese_X <- model.matrix(lrpca)[, -1]
pca <- prcomp(portuguese_X)
summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6    PC7
Standard deviation     4.3595 1.6531 1.53037 1.39669 1.21770 1.14836 0.9032
Proportion of Variance 0.5195 0.0747 0.06402 0.05332 0.04053 0.03605 0.0223
Cumulative Proportion  0.5195 0.5942 0.65825 0.71158 0.75211 0.78816 0.8105
                          PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     0.8909 0.83436 0.76149 0.69090 0.64606 0.62377 0.57140
Proportion of Variance 0.0217 0.01903 0.01585 0.01305 0.01141 0.01064 0.00893
Cumulative Proportion  0.8321 0.85118 0.86703 0.88008 0.89149 0.90213 0.91105
                          PC15    PC16    PC17    PC18    PC19   PC20    PC21
Standard deviation     0.51954 0.50800 0.49246 0.49097 0.47465 0.4686 0.44176
Proportion of Variance 0.00738 0.00705 0.00663 0.00659 0.00616 0.0060 0.00533
Cumulative Proportion  0.91843 0.92549 0.93212 0.93870 0.94486 0.9509 0.95620
                          PC22    PC23    PC24    PC25    PC26    PC27   PC28
Standard deviation     0.41365 0.40021 0.38525 0.37632 0.36369 0.35652 0.3258
Proportion of Variance 0.00468 0.00438 0.00406 0.00387 0.00362 0.00347 0.0029
Cumulative Proportion  0.96088 0.96526 0.96931 0.97318 0.97680 0.98027 0.9832
                          PC29    PC30    PC31    PC32    PC33    PC34    PC35
Standard deviation     0.29815 0.29328 0.28946 0.27588 0.25902 0.24547 0.22317
Proportion of Variance 0.00243 0.00235 0.00229 0.00208 0.00183 0.00165 0.00136
Cumulative Proportion  0.98561 0.98796 0.99025 0.99233 0.99416 0.99581 0.99717
                          PC36    PC37   PC38    PC39
Standard deviation     0.18951 0.18679 0.1487 0.10297
Proportion of Variance 0.00098 0.00095 0.0006 0.00029
Cumulative Proportion  0.99815 0.99911 0.9997 1.00000
```

I want to look at both scaled and unscaled data to see if unscaled fits better (likely, given our differing ranges in scaled data)

```r
portuguese_pcs <- prcomp(portuguese_X, scale=TRUE)

par(mfrow=c(1, 2))
plot(pca, main="PCs of Unscaled Data")
screeplot(portuguese_pcs, main="PCs of Scaled Data")
```



Based on the visual, scaling is better here. We will now view the variance of our pca's to decide how many to include in our model:

```
library(pls)
```

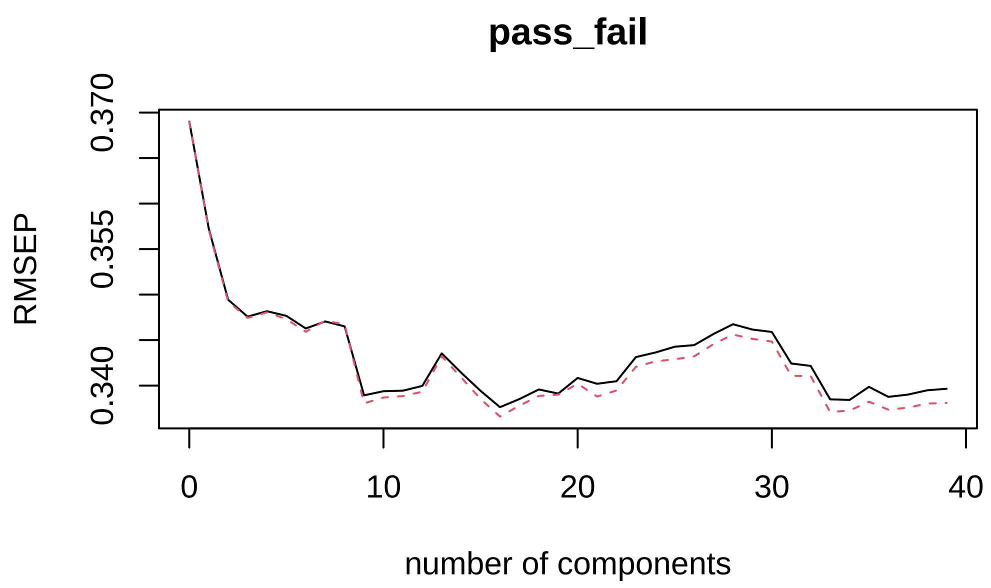
```


Attaching package: 'pls'


The following object is masked from 'package:stats':

    loadings
```

```
portuguese.data_01 <- portuguese.data |>
  mutate(pass_fail = ifelse(pass_fail == "Pass", 1, 0))

portguese_pcr <- pcr(pass_fail ~ . -G3 -G2 -G1,
                     data = portuguese.data_01, family = binomial,
                     scale = TRUE, validation = "CV")

summary(portguese_pcr)
```

```
Data:   X dimension: 519 39
    Y dimension: 519 1
Fit method: svdpc
Number of components considered: 39

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           0.369   0.3573   0.3494   0.3476   0.3482   0.3477   0.3463
adjCV        0.369   0.3572   0.3492   0.3474   0.3481   0.3473   0.3459
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV      0.3471   0.3465   0.3389    0.3394    0.3394    0.3400    0.3435
adjCV   0.3471   0.3467   0.3381    0.3387    0.3389    0.3393    0.3432
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV       0.3414    0.3394    0.3376    0.3385    0.3396    0.3391    0.3408
adjCV    0.3409    0.3385    0.3366    0.3378    0.3389    0.3390    0.3402
       21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
CV       0.3402    0.3405    0.3431    0.3436    0.3443    0.3444    0.3457
adjCV    0.3388    0.3395    0.3421    0.3427    0.3429    0.3432    0.3446
       28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
CV       0.3467    0.3462    0.3459    0.3424    0.3422    0.3385    0.3384
adjCV    0.3456    0.3451    0.3448    0.3411    0.3410    0.3371    0.3373
       35 comps  36 comps  37 comps  38 comps  39 comps
CV       0.3399    0.3388    0.3390    0.3395    0.3397
adjCV    0.3382    0.3374    0.3376    0.3380    0.3381

TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X            8.834    15.56    20.94    25.52    29.72    33.71    37.57
pass_fail    7.030    11.53    12.75    12.79    13.56    14.19    14.20
           8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X            41.28    44.79     48.09     51.23     54.28     57.25     59.97
pass_fail    14.53    18.42     18.51     18.52     18.67     18.75     19.55
           15 comps  16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
X             62.59     65.08     67.50     69.82     72.06     74.27     76.37
pass_fail     20.55     21.08     21.19     21.21     21.23     21.81     22.84
           22 comps  23 comps  24 comps  25 comps  26 comps  27 comps  28 comps
X             78.42     80.38     82.26     83.97     85.66     87.30     88.88
pass_fail     22.90     23.00     23.00     23.43     23.54     23.54     23.75
           29 comps  30 comps  31 comps  32 comps  33 comps  34 comps  35 comps
X             90.43     91.87     93.19     94.45     95.63     96.69     97.68
pass_fail     23.82     24.11     25.46     25.76     27.26     27.39     28.15
           36 comps  37 comps  38 comps  39 comps
```

```
X              98.63     99.31     99.74     100.0
pass_fail      28.21     28.37     28.39      29.1
```

```
validationplot(portguese_pcr, val.type = "RMSEP")
```



Based on the plot displaying our RMSEP, points at around 9, 16, and 33 components should give us the lowest RMSEP. When looking at the summary in depth, when comparing adjCV, 16 components had the lowest at .3366 followed by 33 at .3371 and 9 at .3381. We will select 16 components.

For measure, we will run a loss function and calculate the cross-validation error rate. By running through each possible glm() model for each given pca value, it provides us with insight into which principle component selection can provide the lowest cross-validation error rate, aside to our prior calculation where we selected pc's based on RMSEP - Root Mean Squared Error Prediction.

```
library(boot)
```

```

Attaching package: 'boot'

The following object is masked from 'package:car':

    logit
```

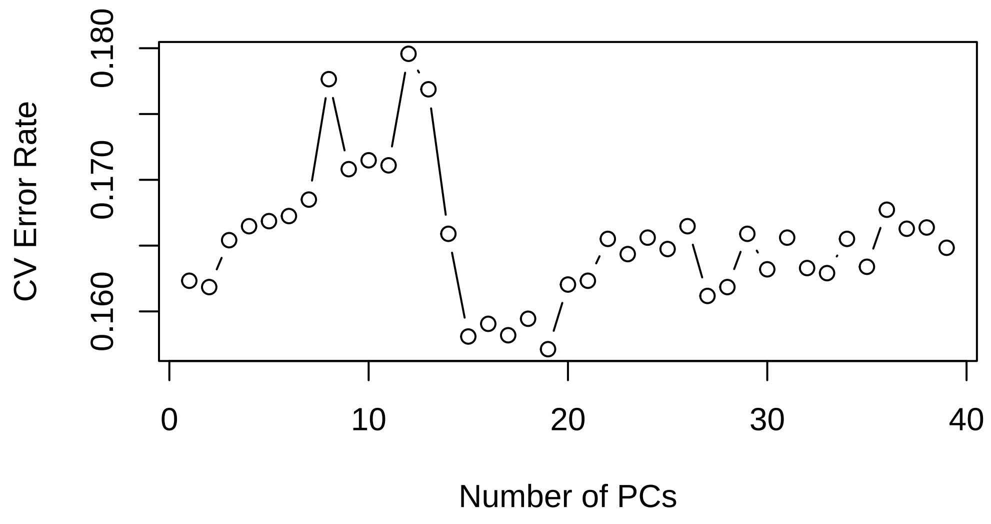
```
Y <- portuguese.data_scaled$pass_fail
loss <- function(Y, pred.p){return(mean((Y==1 & pred.p < 0.5) | # loss function from handout
                                          (Y==0 & pred.p >= 0.5)))}

cv_error <- function(k, repeats = 20){ # function using k (number of pc's)
  # paper used 20 funs of a 10 fold cv so we will do this as well
  pc <- pca$x[, 1:k] # all pc's considered
  data <- data.frame(pass_fail = Y, pc) # data frame with binary outcome
  # and pc's for fitting model
  errors <- numeric(repeats) # repeats 20 times in vector structure
  for(i in 1:repeats){
    errors[i] <- cv.glm(data, glm(pass_fail ~ ., family = binomial, data=data),
                        cost = loss, K = 10)$delta[1]} # cv for 10 fold and finds best cv.er
  mean(errors)} # averages of each of the 20 iterations

errors <- sapply(1:39, cv_error) # 39 because thats how many pc's calculated in rmsep format
errors
```

```
 [1] 0.1623314 0.1618497 0.1654143 0.1664740 0.1668593 0.1672447 0.1684971
 [8] 0.1776493 0.1708092 0.1714836 0.1710983 0.1795761 0.1768786 0.1658960
[15] 0.1580925 0.1590559 0.1581888 0.1594412 0.1571291 0.1620424 0.1623314
[22] 0.1655106 0.1643545 0.1656069 0.1647399 0.1664740 0.1611753 0.1618497
[29] 0.1658960 0.1631985 0.1656069 0.1632948 0.1629094 0.1655106 0.1633911
[36] 0.1677264 0.1662813 0.1663776 0.1648362
```

```
min(errors)
```

```
[1] 0.1571291
```

```
plot(1:39, errors, type = "b",
     xlab = "Number of PCs",
     ylab = "CV Error Rate")
```



Using this method, the lowest cross-validation error rate was .1571291, for a principal component selection of 19 components. Due to the reliability and closer replication to the paper using a cv.glm() model, we will likely select 19 components as our models final selection.

```
predicted_pcr <- predict(portguese_pcr, newdata = portuguese.data_01,
                         ncomp = 16)

# Accuracy based on RMSEP selection of 21 components
predictions <- predicted_pcr[,1, 1] # wouldnt work with just [,1] needed both
real_values <- portuguese.data_01$pass_fail

classification <- ifelse(predictions > 0.5, 1, 0) # funneling to 0 or 1
accuracy_rate <- mean(classification == real_values)
accuracy_rate
```

```
[1] 0.849711
```

```
table(Predicted = classification, Actual = real_values)
```

```
         Actual
Predicted   0   1
        0  16  10
        1  68 425
```

```
predicted_pcr_cv <- predict(portguese_pcr, newdata = portuguese.data_01,
                           ncomp = 19)

# Accuracyn based on cross-validation selection of 17 components
predictions <- predicted_pcr_cv[,1, 1] # wouldnt work with just [,1] needed both
real_values_cv <- portuguese.data_01$pass_fail

classification_cv <- ifelse(predictions > 0.5, 1, 0) # funneling to 0 or 1
accuracy_rate_cv <- mean(classification_cv == real_values_cv)
accuracy_rate_cv
```

```
[1] 0.8400771
```

```
table(Predicted = classification_cv, Actual = real_values_cv)
```

```
         Actual
Predicted   0   1
        0  14  13
        1  70 422
```

Our accuracy rate was actually pretty good, with our model accurately predicting 84.97% of our data correctly with 16 components and 84% with 19 components. I also added a matrix to look at specifically what the model is doing well or poorly. For both, it accurately categorized most of the passing students correctly, and seems to inaccurately predict those who failed the class overall, with 68 vs 70 of those who failed being classified by the models as passed, versus 16 vs 14 correct predictions of those who failed. Overall, our model seems to do well at predicting if you passed, when you actually do pass. It is not the best at predicting the failing students.

Given the slightly better correct classification rate of 19 components, we will utilize that as our final model.

Model 2: Lasso Regression - Binary

```
library(boot)
library(glmnet)
```

```
Loading required package: Matrix
```

```
Attaching package: 'Matrix'


The following objects are masked from 'package:tidyr':

    expand, pack, unpack


Loaded glmnet 4.1-8
```

```r
portuguese.data_01_scaled <- portuguese.data_scaled |>
  mutate(pass_fail = ifelse(pass_fail == "Pass", 1, 0)) # needed 0,1 to run

portuguese.glm <- glm(pass_fail ~ .,
                      data = portuguese.data_01_scaled)
X <- model.matrix(portuguese.glm) [,-1]

set.seed(627)
portuguese_lasso <- cv.glmnet(x = X, y = portuguese.data_01_scaled$pass_fail,
                              alpha = 1,
                              family = "binomial")
portuguese_lasso
```

```


Call:  cv.glmnet(x = X, y = portuguese.data_01_scaled$pass_fail, alpha = 1,      family = "b

Measure: Binomial Deviance

     Lambda Index Measure      SE Nonzero
min 0.02119    21  0.7334 0.03928       9
1se 0.06472     9  0.7722 0.03668       3
```

I see that our model for the minimum cv error (which was 0.7334) is selecting 9 nonzero predictors, the same amount of components we selected in our PCA model. For 1se, it selected 3 nonzero predictors with a .7722 error rate. Both of these are actually really good error rates for our model. For our model, I would like to select the min model with a 9 of our predictors. Its error rate is far smaller, and the deviance is lower. Even with more complexity as the tradeoff. Lets see which of the predictors were selected.

```r
coef(portuguese_lasso, s = "lambda.min")
```

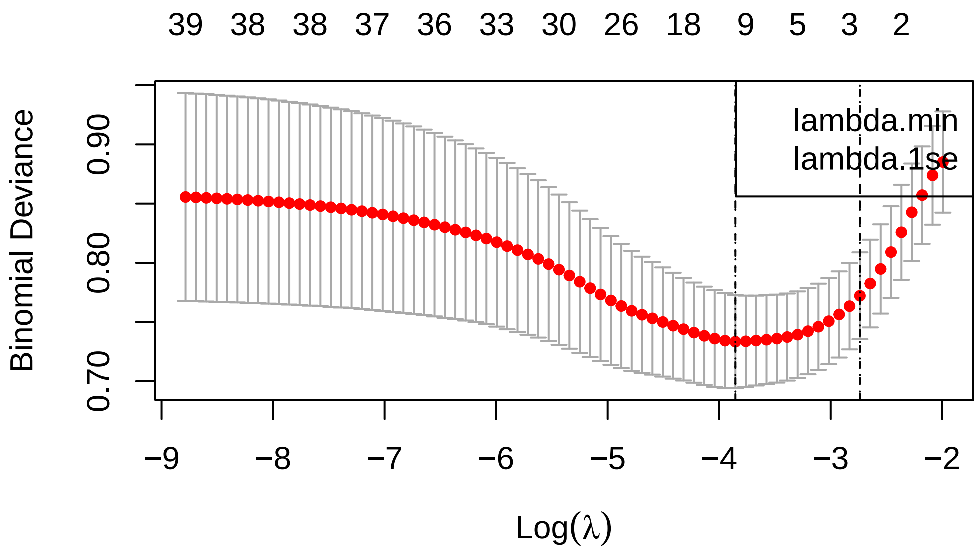
```
41 x 1 sparse Matrix of class "dgCMatrix"
                          s1
(Intercept)       2.04753309
age               .
absences         -0.03291491
Medu              .
Fedu              .
traveltime        .
studytime         0.11621992
failures         -0.81358896
famrel            .
freetime          .
goout             .
Dalc              .
Walc             -0.09256433
health            .
romanticyes      -0.03002587
internetyes       .
higheryes         0.76484860
nurseryyes        .
activitiesyes     .
paidyes          -0.25933936
famsupyes         .
schoolsupyes      .
Mjobat_home       .
Mjobhealth        .
Mjobother         .
Mjobservices      .
Mjobteacher       .
Fjobhealth        .
Fjobother         .
Fjobservices      .
Fjobteacher       .
reasonhome        .
reasonother       .
reasonreputation  .
guardianmother   -0.18469639
guardianother     .
addressU          .
famsizeLE3        .
PstatusT          .
sexM              .
schoolMS         -1.26219981
```

```
predictions <- predict(portuguese_lasso, newx = X, s = "lambda.min", type = "response")
classification <- ifelse(predictions > 0.5, 1, 0)
mean(classification == portuguese.data_01_scaled$pass_fail)
```

```
[1] 0.849711
```

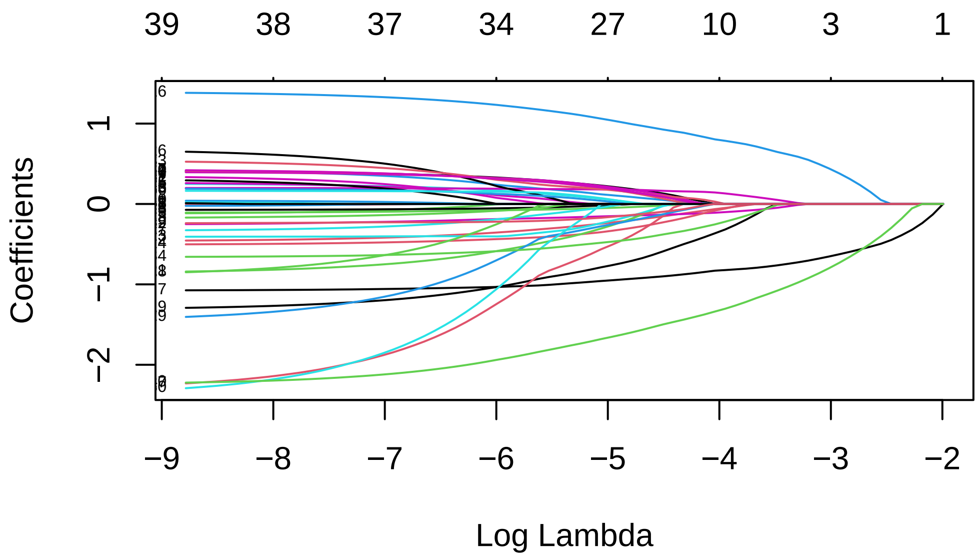
It predicted 84.97% of the training data correctly. There are 9 nonzero predictors. For caution, we will produce a model that displays error changes across differing lambda values:

```
plot(portuguese_lasso)
abline(v = log(portuguese_lasso$lambda.min), lty = 2)
abline(v = log(portuguese_lasso$lambda.1se), lty = 2)
legend("topright", legend = c("lambda.min", "lambda.1se"))
```



Our model is displaying that 9 variables was a good fit. Visualization of our predictors as they are shrunk:

```
plot(portuguese_lasso$glmnet.fit, xvar = "lambda", label = TRUE)
```

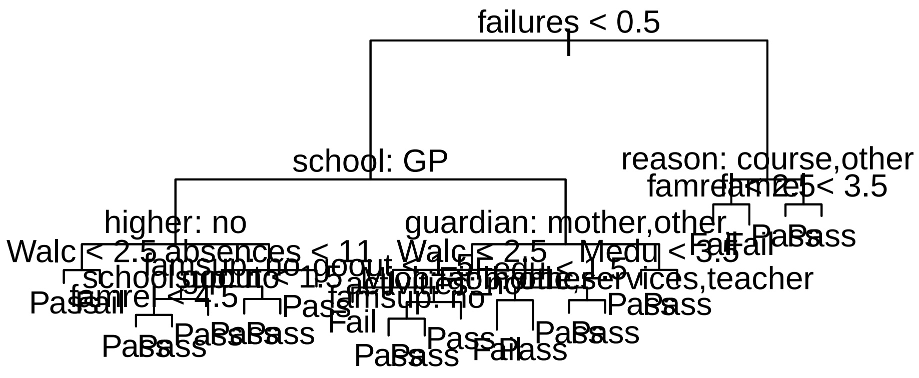


Model 3: Decision Tree

```
set.seed(627)
library(tree)
tree_initial<-tree(pass_fail ~ . -G3-G2-G1, data=portuguese.data)
summary(tree_initial)
```

```
Classification tree:
tree(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data)
Variables actually used in tree construction:
 [1] "failures"   "school"     "higher"     "Walc"       "absences"
 [6] "famsup"     "schoolsup"  "famrel"     "goout"      "guardian"
[11] "activities" "Fedu"       "Fjob"       "Mjob"       "Medu"
[16] "reason"
Number of terminal nodes:  22
Residual mean deviance:  0.4507 = 224 / 497
Misclassification error rate: 0.1175 = 61 / 519
```

```
plot(tree_initial)
text(tree_initial, pretty=0)
```



The initial tree uses 15 variables, has 22 terminal nodes, and is very busy; the model could possibly overfit our data, so we will now tune the tree.

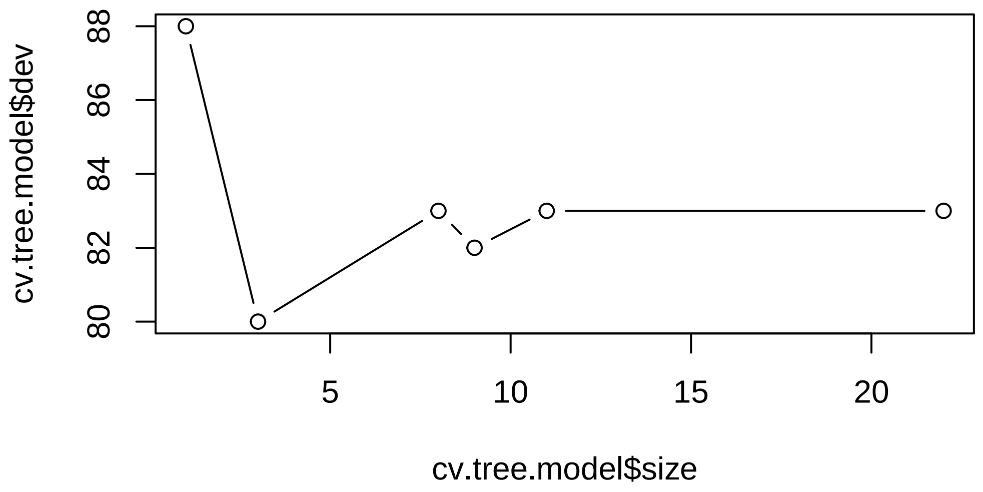
Tuning tree:

```
set.seed(627)
cv.tree.model<- cv.tree(tree_initial, FUN=prune.misclass, K=10)
cv.tree.model
```

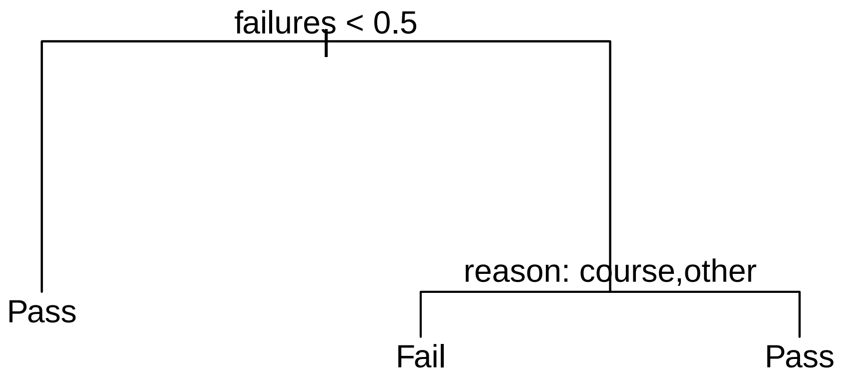
```
$size
[1] 22 11  9  8  3  1

$dev
[1] 83 83 82 83 80 88

$k
[1] -Inf  0.0  0.5  1.0  1.8  6.0

$method
[1] "misclass"
```

```
attr(,"class")
[1] "prune"         "tree.sequence"
```

```r
plot(cv.tree.model$size, cv.tree.model$dev, type="b")
```



```r
tuned_tree<-prune.misclass(tree_initial, best=3)
plot(tuned_tree)
text(tuned_tree, pretty=0)
```



The tuned model is best with 3 terminal nodes. It only uses two variables: failures and reason. Note that reason is a categorical variable with 4 options, course and other being two of the options.

Model 4: Random Forest

```r
library(randomForest)
```

```
randomForest 4.7-1.2

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```r
set.seed(627)
randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=6, importance=TRUE)->RF6
RF6
```

```

Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 6,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 15.41%
Confusion matrix:
     Fail Pass class.error
Fail   17   67  0.79761905
Pass   13  422  0.02988506
```

```r
randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=5, importance=TRUE)->RF5
RF5
```

```

Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 5,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 15.99%
Confusion matrix:
     Fail Pass class.error
Fail   11   73  0.86904762
Pass   10  425  0.02298851
```

```r
set.seed(627)
randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=4, importance=TRUE)->RF4
RF4
```

```

Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 4,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 15.61%
Confusion matrix:
     Fail Pass class.error
Fail    9   75   0.8928571
Pass    6  429   0.0137931
```

```r
randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=7, importance=TRUE)->RF7
RF7
```

```
Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 7,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 16.38%
Confusion matrix:
     Fail Pass class.error
Fail   13   71  0.84523810
Pass   14  421  0.03218391
```

Looking at the OOB estimate of error rate, and keeping in mind the ideal of m = sqrt(p), we tried making m 4, 5, 6, and 7 (we have 30 predictors in the model). The model with m=6 produces the smallest OOB error rate of 15.41.

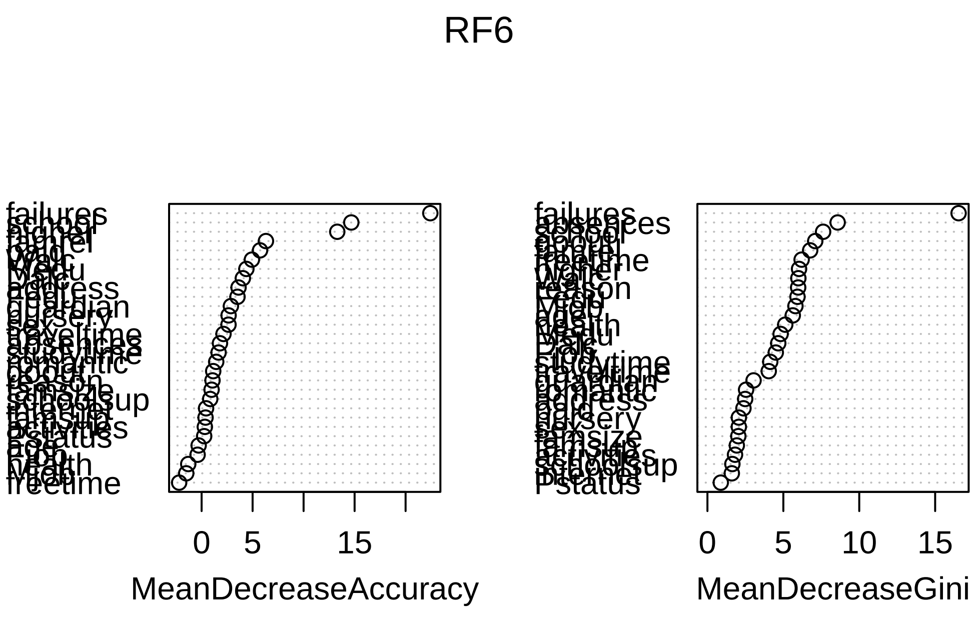
looking at RF variable importance:

```r
importance(RF6)
```

```
                 Fail        Pass MeanDecreaseAccuracy MeanDecreaseGini
school     13.7114807 10.30657356           14.6768506        7.6201262
sex         0.4362858  2.48407103            2.6385787        2.0689599
age        -2.3624385  1.13445705           -0.2885577        5.6223881
address     2.0912671  2.92826929            3.6210346        2.4850924
famsize     1.6805549  0.34198811            0.9800217        2.0347374
Pstatus     0.3449406  0.03588579            0.2507111        0.8778668
```

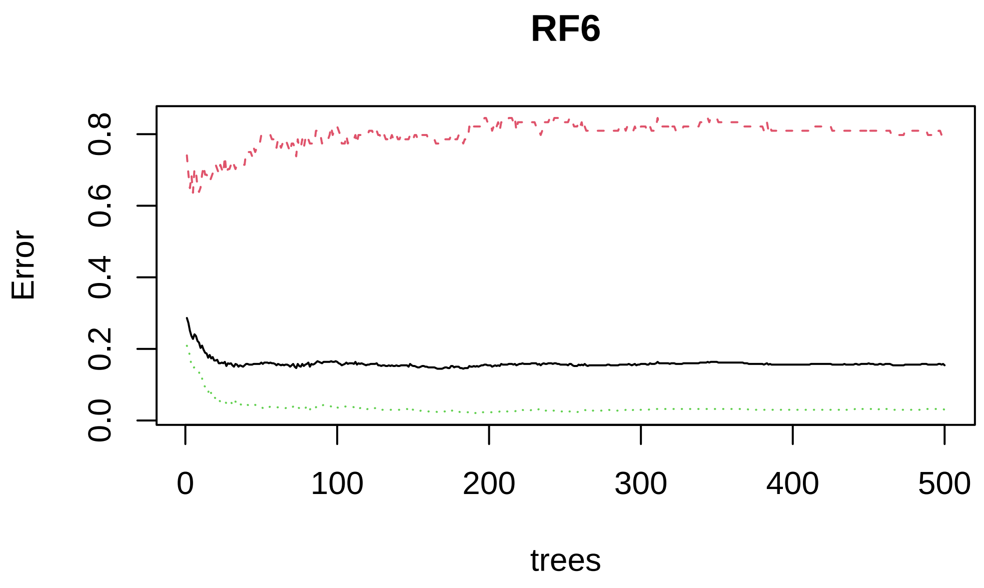
```
Medu        0.6789849  4.50527497                   4.3897110         4.8219339
Fedu        1.9519419  2.80316536                   3.5198297         5.9326023
Mjob       -2.0217749 -0.52867112                  -1.5057109         5.7963529
Fjob       -1.4502329  0.35206852                  -0.3909861         4.5030425
reason      4.9084283 -1.83286420                   1.0681412         5.9762324
guardian    0.8842241  2.96326664                   2.8719778         3.0386220
traveltime  0.8135458  1.91265051                   2.1449013         4.0386688
studytime   5.2450818 -1.13391799                   1.6635819         4.1361001
failures   20.2476080 16.29799434                  22.4263316        16.5449394
schoolsup   2.8855173 -0.20739444                   0.8438599         1.6424743
famsup      0.8250592  0.05192372                   0.3857237         1.9354232
paid        3.1860566  5.25500742                   5.7292298         2.3733610
activities  0.3040239  0.32333543                   0.3184525         1.8169551
nursery     1.1640589  2.32866591                   2.6450890         2.0709782
higher      8.2603815 10.78585427                  13.3000307         6.0349992
internet   -0.9734204  1.23929495                   0.4373140         1.6069450
romantic    0.8178356  0.94714208                   1.4321200         2.5469623
famrel      5.5423390  3.66719179                   6.3073915         6.7670239
freetime    1.7281589 -3.28913038                  -2.2019527         6.2058309
goout       1.9509136  0.00652265                   1.1398874         7.1022976
Dalc       -1.1523620  5.05715203                   4.0445543         4.6644272
Walc        0.9359551  5.02044989                   4.9310812         5.9893684
health     -1.7409117 -0.50991774                  -1.3092199         5.1247424
absences   -0.6257365  2.25550033                   1.8006001         8.5779611
```

```
varImpPlot(RF6)
```



Failure seems to be very important in improving the accuracy, followed by school and absences.

```r
plot(RF6)
```



The above graph shows the overall error rate of the model (black line) and the false positives (green line) and false negatives (red line). After around 200 trees the error rate roughly stays the same, so we wouldn't need to do 500 iterations of trees if we wanted to save computing power.

We were alsp curious if changing the number of predictors considered at each branch to be the predictor amounts favored in previous models, like 21 and 9, would give better results:

```
set.seed(627)
randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=21, importance=TRUE, ntree =

randomForest(pass_fail ~ . -G3-G2-G1, data=portuguese.data, mtry=9, importance=TRUE)->RF9

RF21
```

```
Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 21
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 21

        OOB estimate of  error rate: 16.76%
Confusion matrix:
     Fail Pass class.error
Fail   25   59  0.70238095
Pass   28  407  0.06436782
```

```
RF9
```

```
Call:
 randomForest(formula = pass_fail ~ . - G3 - G2 - G1, data = portuguese.data,      mtry = 9,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 16.57%
Confusion matrix:
     Fail Pass class.error
Fail   14   70  0.83333333
Pass   16  419  0.03678161
```

Our error rate is 16.76%, but we have a relatively better prediction ability for failing students, 10% improvement. We will keep the model with the smaller OOB error rate, which is mtry=6.

Model 5: Support Vector Machine

```r
library(e1071)
```

```

Attaching package: 'e1071'

The following object is masked from 'package:ggplot2':

    element
```

```r
set.seed(627)
#Linear kernel
svm.lin<-svm(pass_fail ~ ., kernel="linear", data=portuguese.data_scaled, cross=10)
svm.lin$tot.accuracy
```

```
[1] 84.58574
```

```r
#Polynomial Kernel
svm.pol<-svm(pass_fail ~ ., kernel="polynomial", data=portuguese.data_scaled, cross=10)
svm.pol$tot.accuracy
```

```
[1] 83.81503
```

```r
#Radial kernel
svm.rad<-svm(pass_fail ~ ., kernel="radial", data=portuguese.data_scaled, cross=10)
svm.rad$tot.accuracy
```

```
[1] 83.81503
```

```r
#Sigmoid kernel
svm.sig<-svm(pass_fail ~ ., kernel="sigmoid", data=portuguese.data_scaled, cross=10)
svm.sig$tot.accuracy
```

```
[1] 84.00771
```

I used 10-fold cross validation to create 4 initial SVM models with each kernel and a default cost of 1. They all had similar predication accuracy with the best being a linear kernel at around 84.59% accuracy.

Now I willl use tune() to find the best kernel for each cost, and then pick the best model based on the smallest CV prediction error.

```
set.seed(627)
svm.tuning <- tune(svm, pass_fail ~ . , data = portuguese.data_scaled, ranges = list(cost = c
svm.tuning
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost kernel
  100 linear

- best performance: 0.1349925
```

```
svm.tuning$best.parameters
```

```
  cost kernel
5  100 linear
```

```
svm.tuning$best.model
```

```
Call:
best.tune(METHOD = svm, train.x = pass_fail ~ ., data = portuguese.data_scaled,
    ranges = list(cost = c(0.01, 0.1, 1, 10, 100), kernel = c("linear",
        "polynomial", "radial", "sigmoid")))


Parameters:
   SVM-Type:  C-classification
```

```
 SVM-Kernel:  linear 
       cost:  100 

Number of Support Vectors:  173
```

The best model has a cost of 100, uses the linear kernel, and has a CV prediction error of 13.5%.

Next I will look at the best model in all of the cost levels:

```
library(dplyr)
set.seed(627)
result.df<-svm.tuning$performances

try<-result.df |>
  arrange(cost) |>
  select(cost, kernel, error)
try
```

```
    cost     kernel     error
1  1e-02     linear 0.1696078
2  1e-02 polynomial 0.1619155
3  1e-02     radial 0.1619155
4  1e-02    sigmoid 0.1619155
5  1e-01     linear 0.1561463
6  1e-01 polynomial 0.1619155
7  1e-01     radial 0.1619155
8  1e-01    sigmoid 0.1619155
9  1e+00     linear 0.1407617
10 1e+00 polynomial 0.1619532
11 1e+00     radial 0.1580317
12 1e+00    sigmoid 0.1541855
13 1e+01     linear 0.1369155
14 1e+01 polynomial 0.1677225
15 1e+01     radial 0.1677602
16 1e+01    sigmoid 0.1599925
17 1e+02     linear 0.1349925
18 1e+02 polynomial 0.1696456
19 1e+02     radial 0.1639140
20 1e+02    sigmoid 0.1656863
```

The best kernel for cost 100 is linear with an error of 13.5%.

The best kernel for cost 10 is linear with an error of 13.69%.

The best kernel for cost 1 is linear with an error of 14.08%.

The best kernel for cost 0.1 is linear with an error of 15.61%.

The best kernel for cost 0.01 is not linear with an error of 16.19%.

```r
best.svm<-svm(pass_fail ~ ., data = portuguese.data_scaled, cost=100, kernel="linear")
best.svm
```

```


Call:
svm(formula = pass_fail ~ ., data = portuguese.data_scaled, cost = 100,
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  100

Number of Support Vectors:  173
```

Using Testing data on models

Logistic Regression Model with PCA - Binary Setup C

```r
# creating 0,1 for holdout
portuguese.data_01_holdout <- holdout.data |>
  mutate(pass_fail = ifelse(pass_fail == "Pass", 1, 0))

testing_pcr <- predict(portguese_pcr, newdata = portuguese.data_01_holdout,
                         ncomp = 19)

# Accuracy
predictions <- testing_pcr[,1, 1] # wouldnt work with just [,1] needed both
real_values <- portuguese.data_01_holdout$pass_fail

pred_class <- ifelse(predictions > 0.5, 1, 0) # funneling to 0 or 1
accuracy_rate <- mean(pred_class == real_values)
accuracy_rate
```

```
[1] 0.8923077
```

```r
table(Predicted = pred_class, Actual = real_values)
```

```
         Actual
Predicted   0   1
        0   3   1
        1  13 113
```

Our testing data did even better than our model on the training data, we received 89.23% of accurate predictions, or a misclassification rate of 10.77%.

Lasso Regression - Binary Setup C

```r
holdout.data_scaled_01 <- holdout.data_scaled |>
  mutate(pass_fail = ifelse(pass_fail == "Pass", 1, 0))

portuguese.glm.holdout <- glm(pass_fail ~ ., data = holdout.data_scaled_01)

X <- model.matrix(portuguese.glm.holdout) [,-1]
```

```r
set.seed(627)

predictions <- predict(portuguese_lasso, newx = X, s = "lambda.min", type = "response")
classification <- ifelse(predictions > 0.5, 1, 0)
mean(classification == holdout.data_scaled_01$pass_fail)
```

```
[1] 0.9076923
```

The model was better at predicting on the testing data, as the training data was 84.97% and the testing data was 90.77% prediction accuracy, or a misclassification error rate of 9.23%. This model also reflects predictive ability well.

Decision Tree Model - Binary Setup C

```r
pred_val<-predict(tree_initial, newdata=holdout.data, type="class")
table(pred_val, holdout.data$pass_fail)
```

```
pred_val Fail Pass
    Fail    9   10
    Pass    7  104
```

```r
mis.class<-mean(pred_val != holdout.data$pass_fail)
mis.class
```

```
[1] 0.1307692
```

```r
pred_val2<-predict(tuned_tree, newdata=holdout.data, type="class")
table(pred_val2, holdout.data$pass_fail)
```

```
pred_val2 Fail Pass
     Fail    8    7
     Pass    8  107
```

```r
mis.class2<-mean(pred_val2 != holdout.data$pass_fail)
mis.class2
```

```
[1] 0.1153846
```

Our initial tree model had a misclassification rate of around 13.08% on our testing data. Tuning the tree with cross validation brings our misclassification rate down to 11.54% which is better, but the model is more inflexible now.

Random Forest Model - Binary Setup C

Testing data on tuned model:

```r
rf.pred<-predict(RF6, newdata=holdout.data, type="class")
table(rf.pred, holdout.data$pass_fail)
```

```
rf.pred Fail Pass
   Fail    3    3
   Pass   13  111
```

```r
mis.class3<-mean(rf.pred != holdout.data$pass_fail)
mis.class3
```

```
[1] 0.1230769
```

On our tuned model we get a misclassification rate of 12.3%.

Testing data on a non-optimal RF model:

```r
rf.pred2<-predict(RF5, newdata=holdout.data, type="class")
table(rf.pred2, holdout.data$pass_fail)
```

```
rf.pred2 Fail Pass
    Fail    2    1
    Pass   14  113
```

```r
mis.class4<-mean(rf.pred2 != holdout.data$pass_fail)
mis.class4
```

```
[1] 0.1153846
```

The model where mtry=5 had a higher OOB error rate on the training data, but produces a slightly smaller msiclassification rate of 11.53% compared to our choosen RF model. These are similar misclassification rates and likely wouldn't create a big difference when tested on more data.

Support Vector Machine Model - Binary Setup C

```r
svm.pred <- predict(best.svm, newdata=holdout.data_scaled)
err.tab <- mean(holdout.data_scaled$pass_fail != svm.pred)
err.tab
```

```
[1] 0.1307692
```

Misclassification rate on new data: 13.08%.

```r
svm.pred2 <- predict(svm.sig, newdata=holdout.data_scaled)
err.tab2 <- mean(holdout.data_scaled$pass_fail != svm.pred2)
err.tab2
```

```
[1] 0.1076923
```

Misclassification rate for an untuned sigmoid model is lower than the tuned linear model, at 10.77%.

Conclusions

Below is a table of our final models misclassification error rate on our testing (holdout) data, alongside the papers misclassification error rate for the same models (if present in paper). Note: Our models are utilizing Portuguese data only with setup C (G3 included; G1 and G2 removed).

	Log w/ PCA	Lasso	DT	RF	SVM
Our Misclassification Error Rate	10.77%	9.23%	11.54%	12.3%	13.08%
Papers Misclassification Error Rate	**Not in initial paper**	**Not in initial paper**	15.6%	15%	15.2%
Our best Models Rank	2	1	3	4	5

Our models had a better misclassification error rate overall, which suggests differences in our calculation methods with the papers. Reading over the original paper, they provide detail on setup and validation methods. We made the decision to utilize more of the material learned in class, which included 80/20 split for most methods. We were able to use cross-validation (as used in the paper) for our Logistic Regression with PCA model, LASSO, Decision Tree and SVM. However, for time constraint, we did not utilize the 20 runs of the ten fold, and instead simplified to just one iteration of 10 fold. We did run one instance of 20 iterations of 10 fold on Logistic Regression with PCA, but that was due to the simplicity of the model and therefor its simple computation of these iterations. This would explain why our misclassification values are better than the original papers, for those that apply, as our 80/20 split is less conservative.

However, we replicated the study thoroughly otherwise, computing exactly what is asked of the paper with relevance to what we have learned thus far in the course and what we were able to further research. We believe this study was handled well with the researchers, and if we were to include any critiques it would be to remove the G1 and G2 values fully. While exploring this data, G1 and G2 inclusion affected the reliability of our models closer to a 95% range, indicating an overly predictive ability of G1 and G2.

Finally, it is important to note the difference in failures and passes in the data.

```r
table(portuguese.data$pass_fail)
```

```
Fail Pass
```

```
  84  435
```

```r
table(holdout.data$pass_fail)
```

```
Fail Pass
  16  114
```

Given the distribution of pass and fail, where fail is on a far lower end, a poor model could technically predict all values as "Pass" and still have an ~80-86% accuracy rate. While a small amount of failures does reflect the likely real-world distribution of pass and fail grades, it is worth mentioning that models with this type of setup may not be a reliable way to predict passing/failing at all. I believe a replication of this study where we have a much larger data set (>10000) may provide a more reliable basis for classifying pass and fail grades.

References

Cortez, P., & Silva, A. (2008). *Using Data Mining to Predict Secondary School Student Performance*. Proceedings of the 5th Annual Future Business Technology Conference.