STUDENT PERFORMANCE ANALYSIS

Arouna Romeo KONE

2025-09-12

---

0. SETUP

```
# Load necessary packages

packages <- c("dplyr", "ggplot2", "tidyr", "readr", "stringr", "GGally", "caret", "corrplot"
installed <- packages %in% installed.packages()
if(any(!installed)) install.packages(packages[!installed])
lapply(packages,library, character.only= TRUE)
```

```
Attaching package: 'dplyr'
The following objects are masked from 'package:stats':

    filter, lag
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
Loading required package: lattice
corrplot 0.95 loaded
randomForest 4.7-1.2
Type rfNews() to see new features/changes/bug fixes.

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

    margin
```

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

    combine

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

    slice
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

Attaching package: 'Metrics'
The following objects are masked from 'package:caret':

    precision, recall

Attaching package: 'yardstick'
The following objects are masked from 'package:Metrics':

    accuracy, mae, mape, mase, precision, recall, rmse, smape
The following objects are masked from 'package:caret':

    precision, recall, sensitivity, specificity
The following object is masked from 'package:readr':

    spec
Type 'citation("pROC")' for a citation.

Attaching package: 'pROC'
The following object is masked from 'package:Metrics':

    auc
The following objects are masked from 'package:stats':

    cov, smooth, var

Attaching package: 'kableExtra'
The following object is masked from 'package:dplyr':
```

```
    group_rows
[[1]]
[1] "dplyr"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
[7] "methods"   "base"

[[2]]
[1] "ggplot2"   "dplyr"     "stats"     "graphics"  "grDevices" "utils"
[7] "datasets"  "methods"   "base"

[[3]]
 [1] "tidyr"     "ggplot2"   "dplyr"     "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[4]]
 [1] "readr"     "tidyr"     "ggplot2"   "dplyr"     "stats"     "graphics"
 [7] "grDevices" "utils"     "datasets"  "methods"   "base"

[[5]]
 [1] "stringr"   "readr"     "tidyr"     "ggplot2"   "dplyr"     "stats"
 [7] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"

[[6]]
 [1] "GGally"    "stringr"   "readr"     "tidyr"     "ggplot2"   "dplyr"
 [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
[13] "base"

[[7]]
 [1] "caret"     "lattice"   "GGally"    "stringr"   "readr"     "tidyr"
 [7] "ggplot2"   "dplyr"     "stats"     "graphics"  "grDevices" "utils"
[13] "datasets"  "methods"   "base"

[[8]]
 [1] "corrplot"  "caret"     "lattice"   "GGally"    "stringr"   "readr"
 [7] "tidyr"     "ggplot2"   "dplyr"     "stats"     "graphics"  "grDevices"
[13] "utils"     "datasets"  "methods"   "base"

[[9]]
 [1] "randomForest" "corrplot"     "caret"        "lattice"      "GGally"
 [6] "stringr"      "readr"        "tidyr"        "ggplot2"      "dplyr"
[11] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
[16] "methods"      "base"

[[10]]
 [1] "xgboost"      "randomForest" "corrplot"     "caret"        "lattice"
 [6] "GGally"       "stringr"      "readr"        "tidyr"        "ggplot2"
```

```
[11] "dplyr"        "stats"        "graphics"     "grDevices"    "utils"
[16] "datasets"     "methods"      "base"

[[11]]
 [1] "factoextra"   "xgboost"      "randomForest" "corrplot"     "caret"
 [6] "lattice"      "GGally"       "stringr"      "readr"        "tidyr"
[11] "ggplot2"      "dplyr"        "stats"        "graphics"     "grDevices"
[16] "utils"        "datasets"     "methods"      "base"

[[12]]
 [1] "Metrics"      "factoextra"   "xgboost"      "randomForest" "corrplot"
 [6] "caret"        "lattice"      "GGally"       "stringr"      "readr"
[11] "tidyr"        "ggplot2"      "dplyr"        "stats"        "graphics"
[16] "grDevices"    "utils"        "datasets"     "methods"      "base"

[[13]]
 [1] "yardstick"    "Metrics"      "factoextra"   "xgboost"      "randomForest"
 [6] "corrplot"     "caret"        "lattice"      "GGally"       "stringr"
[11] "readr"        "tidyr"        "ggplot2"      "dplyr"        "stats"
[16] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
[21] "base"

[[14]]
 [1] "pROC"         "yardstick"    "Metrics"      "factoextra"   "xgboost"
 [6] "randomForest" "corrplot"     "caret"        "lattice"      "GGally"
[11] "stringr"      "readr"        "tidyr"        "ggplot2"      "dplyr"
[16] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
[21] "methods"      "base"

[[15]]
 [1] "knitr"        "pROC"         "yardstick"    "Metrics"      "factoextra"
 [6] "xgboost"      "randomForest" "corrplot"     "caret"        "lattice"
[11] "GGally"       "stringr"      "readr"        "tidyr"        "ggplot2"
[16] "dplyr"        "stats"        "graphics"     "grDevices"    "utils"
[21] "datasets"     "methods"      "base"

[[16]]
 [1] "kableExtra"   "knitr"        "pROC"         "yardstick"    "Metrics"
 [6] "factoextra"   "xgboost"      "randomForest" "corrplot"     "caret"
[11] "lattice"      "GGally"       "stringr"      "readr"        "tidyr"
[16] "ggplot2"      "dplyr"        "stats"        "graphics"     "grDevices"
[21] "utils"        "datasets"     "methods"      "base"
```

```
# Load dataset
data <- read.csv("/Users/konearounaromeo/Downloads/student-scores.csv")
```

---

1. DEFINE THE OBJECTIVE

Goal: Understand what factors impact student academic performance

Target variables: math_score, physics_score, bilogy_score, geography_score...

Questions:

- Are absences related to lower scores?

- Do self-study hours help?

- Are there differences by gender, job status, or extracurriculars?

---

2. DATA CLEANING & PREPROCESSING

```
# Drop non-useful personal columns
data <- data %>%
  select(-id, -first_name, -last_name, -email)
```

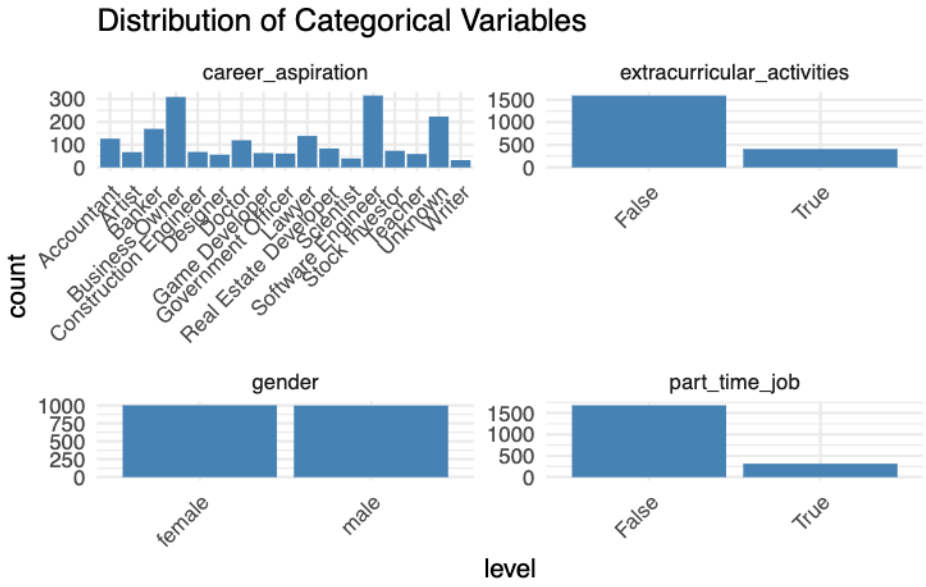
```
# Check missing values
colSums(is.na(data))
```

```
                    gender                part_time_job
                         0                            0
              absence_days extracurricular_activities
                         0                            0
   weekly_self_study_hours           career_aspiration
                         0                            0
               math_score               history_score
                         0                            0
            physics_score             chemistry_score
                         0                            0
            biology_score               english_score
                         0                            0
          geography_score
                         0
```

```r
# Convert categorial variables to factors
categorical_vars <- c('gender','part_time_job','extracurricular_activities','career_aspirati
data[categorical_vars] <- lapply(data[categorical_vars], as.factor)
lapply(data[categorical_vars], levels)
```

```
$gender
[1] "female" "male"

$part_time_job
[1] "False" "True"

$extracurricular_activities
[1] "False" "True"

$career_aspiration
 [1] "Accountant"            "Artist"                "Banker"
 [4] "Business Owner"        "Construction Engineer" "Designer"
 [7] "Doctor"                "Game Developer"        "Government Officer"
[10] "Lawyer"                "Real Estate Developer" "Scientist"
[13] "Software Engineer"     "Stock Investor"        "Teacher"
[16] "Unknown"               "Writer"
```

```r
# Convert to long format for faceting
data_long <- pivot_longer(
  data[categorical_vars],
  cols = everything(),
  names_to = "variable",
  values_to = "level"
)

# Plot all variables in one grid
ggplot(data_long, aes(x = level)) +
  geom_bar(fill = "steelblue") +
  facet_wrap(~variable, scales = "free") +
  labs(title = "Distribution of Categorical Variables") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



---

3. EXPLORATORY DATA ANALYSIS (EDA)

```
freq <- lapply(data[categorical_vars], function(x) prop.table(table(x)))
print(freq)
```

```
$gender
x
female   male
 0.501  0.499

$part_time_job
x
False  True
0.842 0.158

$extracurricular_activities
x
False  True
0.796 0.204

$career_aspiration
x
          Accountant                Artist                Banker
              0.0630                0.0335                0.0845
```

```
       Business Owner Construction Engineer              Designer
               0.1545                 0.0340                0.0280
               Doctor         Game Developer    Government Officer
               0.0595                 0.0315                0.0305
               Lawyer Real Estate Developer             Scientist
               0.0690                 0.0415                0.0195
    Software Engineer         Stock Investor               Teacher
               0.1575                 0.0365                0.0295
              Unknown                 Writer
               0.1115                 0.0160
```

```
# Overview of the data
names(data)
```

```
 [1] "gender"                     "part_time_job"
 [3] "absence_days"               "extracurricular_activities"
 [5] "weekly_self_study_hours"    "career_aspiration"
 [7] "math_score"                 "history_score"
 [9] "physics_score"              "chemistry_score"
[11] "biology_score"              "english_score"
[13] "geography_score"
```

```
glimpse(data)
```

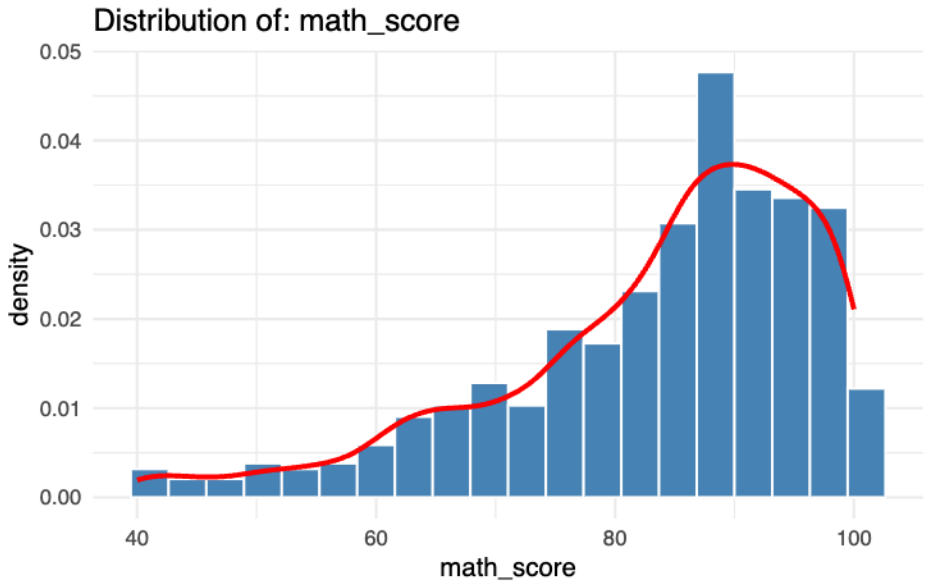
```
Rows: 2,000
Columns: 13
$ gender                     <fct> male, female, female, female, male, female,~
$ part_time_job              <fct> False, False, False, False, False, False, F~
$ absence_days               <int> 3, 2, 9, 5, 5, 2, 3, 2, 6, 3, 2, 1, 7, 10, ~
$ extracurricular_activities <fct> False, False, True, False, False, False, Tr~
$ weekly_self_study_hours    <int> 27, 47, 13, 3, 10, 26, 23, 34, 25, 18, 7, 7~
$ career_aspiration          <fct> Lawyer, Doctor, Government Officer, Artist,~
$ math_score                 <int> 73, 90, 81, 71, 84, 93, 99, 95, 94, 98, 65,~
$ history_score              <int> 81, 86, 97, 74, 77, 100, 96, 95, 68, 69, 60~
$ physics_score              <int> 93, 96, 95, 88, 65, 67, 97, 82, 94, 88, 97,~
$ chemistry_score            <int> 97, 100, 96, 80, 65, 78, 73, 63, 85, 71, 94~
$ biology_score              <int> 63, 90, 65, 89, 80, 72, 88, 84, 81, 67, 71,~
$ english_score              <int> 80, 88, 77, 63, 74, 80, 76, 70, 74, 71, 81,~
$ geography_score            <int> 87, 90, 94, 86, 76, 84, 64, 85, 72, 73, 66,~
```

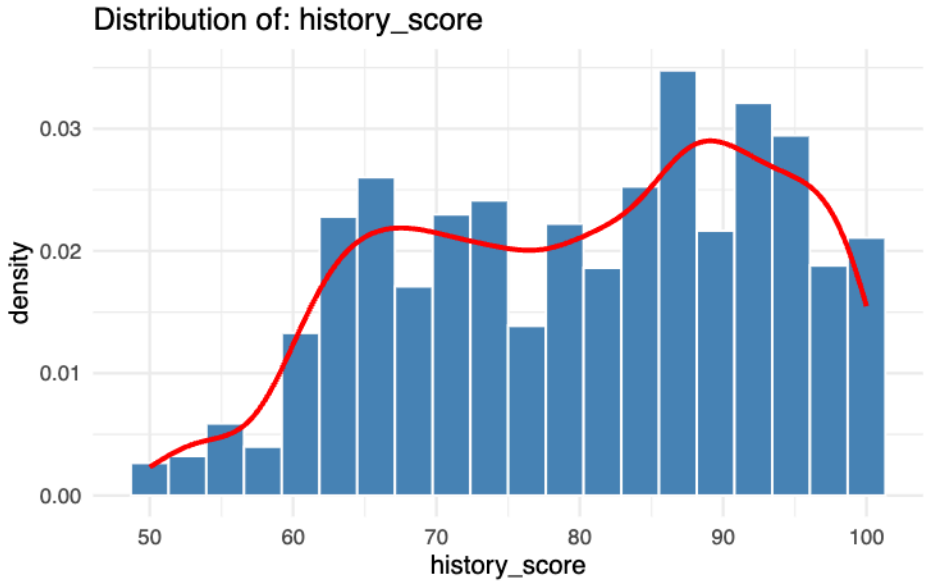
```
summary(data)
```

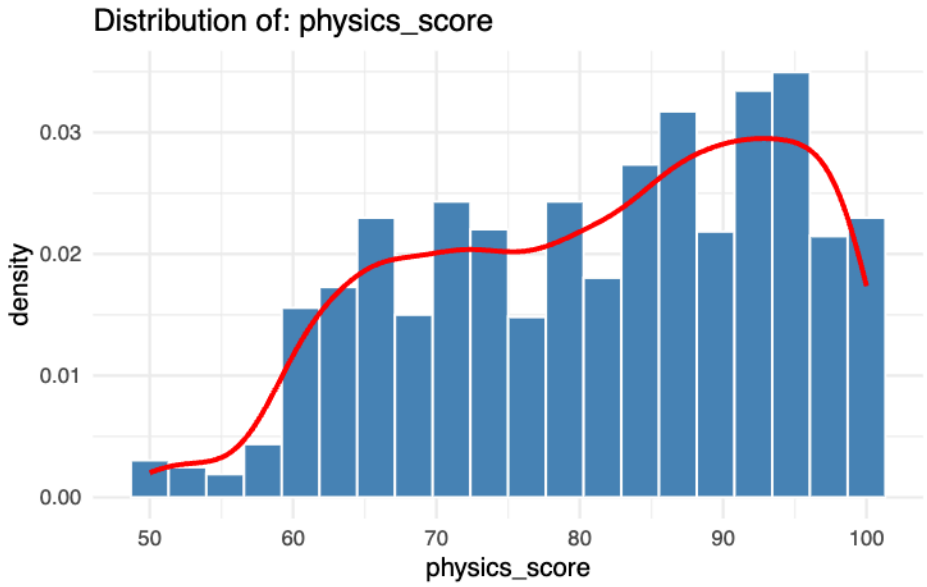
```
    gender     part_time_job  absence_days    extracurricular_activities
 female:1002   False:1684    Min.   : 0.000   False:1592
 male  : 998   True : 316    1st Qu.: 2.000   True : 408
                             Median : 3.000
                             Mean   : 3.666
                             3rd Qu.: 5.000
```

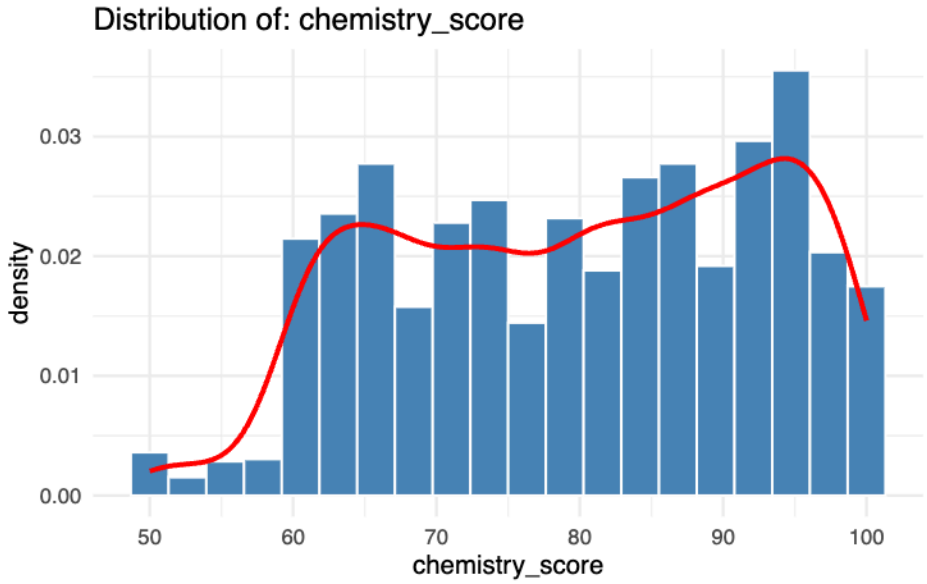
```
                                Max.   :10.000

 weekly_self_study_hours         career_aspiration   math_score
 Min.   : 0.00           Software Engineer:315     Min.   : 40.00
 1st Qu.: 5.00           Business Owner   :309     1st Qu.: 77.00
 Median :18.00           Unknown          :223     Median : 87.00
 Mean   :17.76           Banker           :169     Mean   : 83.45
 3rd Qu.:28.00           Lawyer           :138     3rd Qu.: 93.00
 Max.   :50.00           Accountant       :126     Max.   :100.00
                         (Other)          :720
 history_score    physics_score    chemistry_score biology_score
 Min.   : 50.00   Min.   : 50.00   Min.   : 50     Min.   : 30.00
 1st Qu.: 69.75   1st Qu.: 71.00   1st Qu.: 69     1st Qu.: 69.00
 Median : 82.00   Median : 83.00   Median : 81     Median : 81.00
 Mean   : 80.33   Mean   : 81.34   Mean   : 80     Mean   : 79.58
 3rd Qu.: 91.00   3rd Qu.: 92.00   3rd Qu.: 91     3rd Qu.: 91.00
 Max.   :100.00   Max.   :100.00   Max.   :100     Max.   :100.00

 english_score   geography_score
 Min.   :50.00   Min.   : 60.00
 1st Qu.:72.00   1st Qu.: 71.00
 Median :83.00   Median : 81.00
 Mean   :81.28   Mean   : 80.89
 3rd Qu.:91.00   3rd Qu.: 91.00
 Max.   :99.00   Max.   :100.00
```

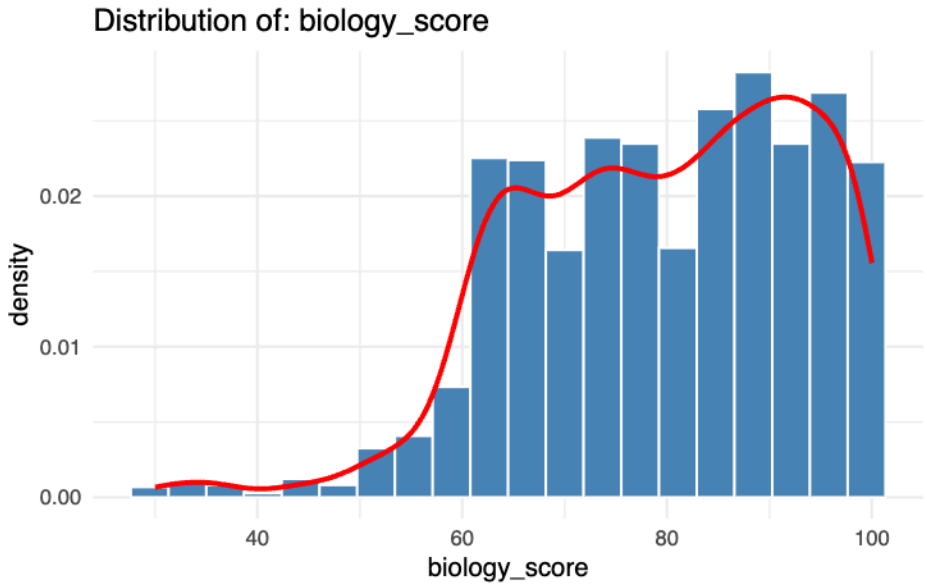
```r
# Distribution of scores
score_vars <- names(data)[grepl('_score',names(data))]

for (var in score_vars) {
  p <- ggplot(data, aes(x = .data[[var]])) +
    geom_histogram(
      aes(y = after_stat(density)),bins = 20,fill = "steelblue",color = 'white') +
    geom_density(color = 'red', linewidth = 1) +
    theme_minimal() +
    ggtitle(paste('Distribution of:', var))

  print(p)
}
```

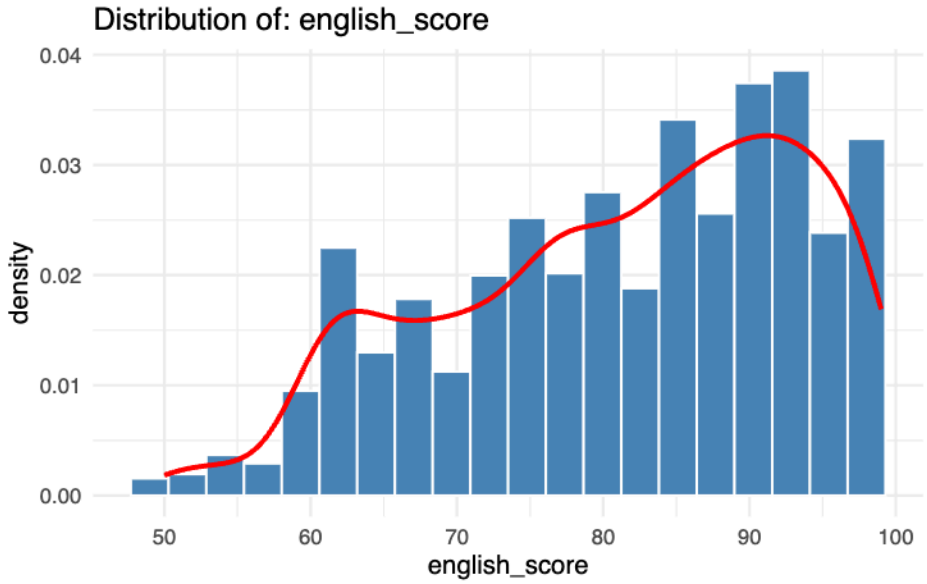

Distribution of: math_score
density
0.05
0.04
0.03
0.02
0.01
0.00
40
60
80
100
math_score

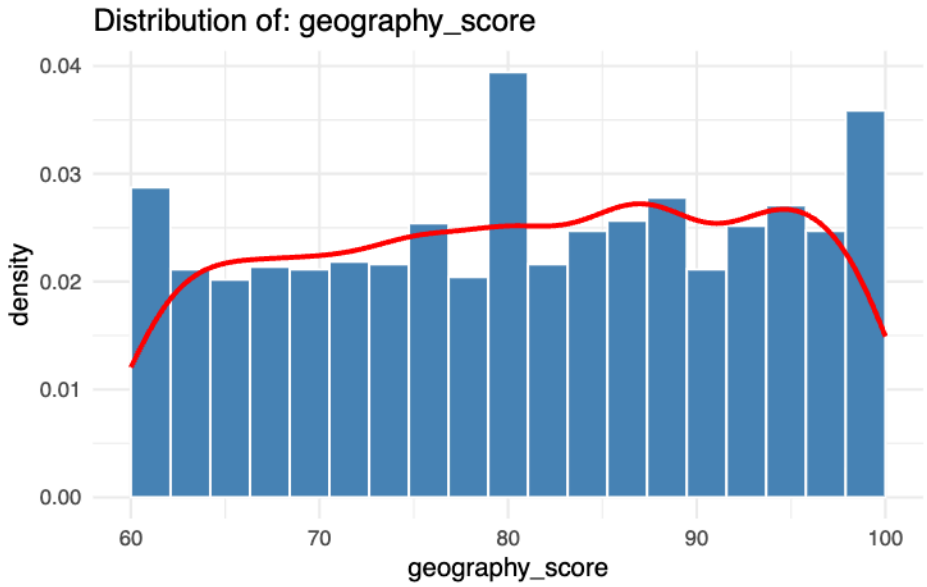

Distribution of: history_score
density
0.03
0.02
0.01
0.00
50
60
70
80
90
100
history_score


Distribution of: physics_score
density
0.03
0.02
0.01
0.00
50
60
70
80
90
100
physics_score


Distribution of: chemistry_score
density
0.03
0.02
0.01
0.00
50
60
70
80
90
100
chemistry_score


Distribution of: biology_score
density
0.02
0.01
0.00
40
60
80
100
biology_score


Distribution of: english_score
density
0.04
0.03
0.02
0.01
0.00
50
60
70
80
90
100
english_score




```
# Count gender
gender_count <- data %>% count(gender)
gender_count
```

```
  gender    n
1 female 1002
2   male  998
```

```
# Gender-based average scores
table_mean <- data %>%
  group_by(gender) %>%
  summarise(across(all_of(score_vars), mean, na.rm = TRUE))
```

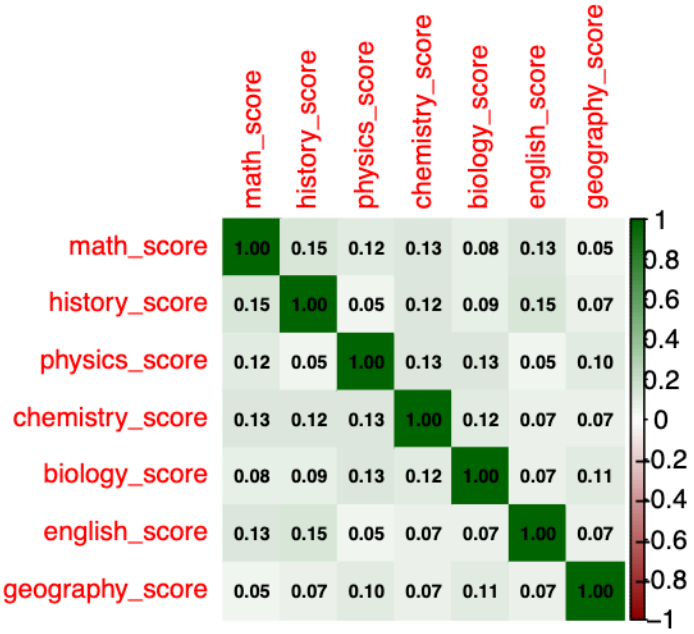
```
Warning: There was 1 warning in `summarise()`.
i In argument: `across(all_of(score_vars), mean, na.rm = TRUE)`.
i In group 1: `gender = female`.
Caused by warning:
! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
Supply arguments directly to `.fns` through an anonymous function instead.

  # Previously
  across(a:b, mean, na.rm = TRUE)

  # Now
  across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
print(table_mean)
```

```
# A tibble: 2 x 8
```

```
  gender math_score history_score physics_score chemistry_score biology_score
  <fct>       <dbl>         <dbl>         <dbl>           <dbl>         <dbl>
1 female       82.8          80.5          80.7            80.4          79.2
2 male         84.1          80.2          82.0            79.6          79.9
# i 2 more variables: english_score <dbl>, geography_score <dbl>
```

```r
# Correlation between scores

score_data <- data %>% select(all_of(score_vars))
corrplot(cor(score_data),method = "color",addCoef.col = "black",
col = colorRampPalette(c("darkred", "white", "darkgreen"))(100),
tl.cex = 0.8,
number.cex = 0.6,
mar = c(0, 0, 1, 0))
```



throught the corrplot there is no multicolinearity between the scores

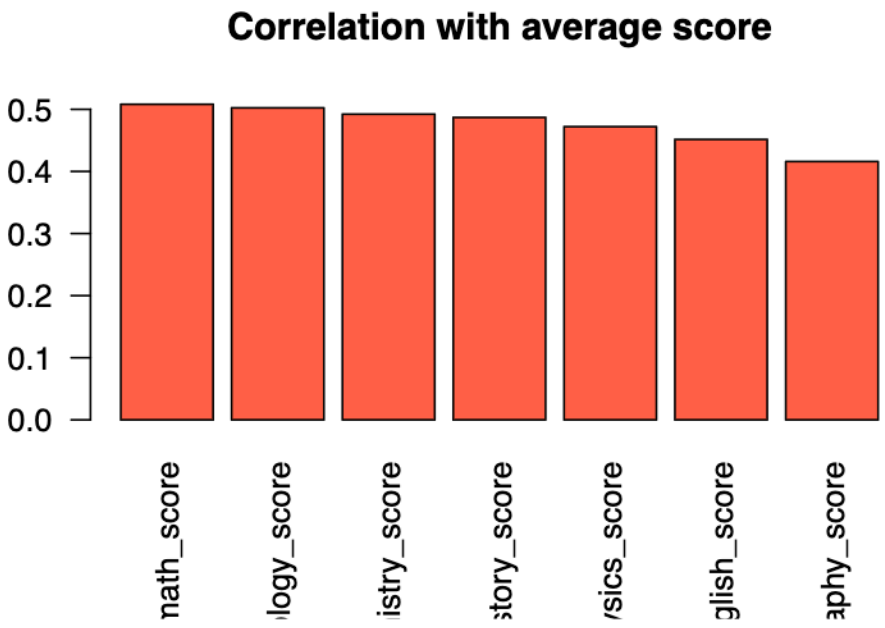
Bivariate Analysis

```r
# Build average score for all score vars

data <- data %>% mutate(average_score = rowMeans(select(.,all_of(score_vars))))
```

```r
# Correlation with average score
data_corr <- cor(data[, c(score_vars, "average_score")])
cor_average <- data_corr["average_score", score_vars]
cor_average <- sort(abs(cor_average), decreasing = TRUE)
print(cor_average)
```

```
     math_score   biology_score chemistry_score   history_score   physics_score
      0.5081161       0.5023912       0.4922160       0.4868997       0.4721037
  english_score geography_score
      0.4515498       0.4159290
```

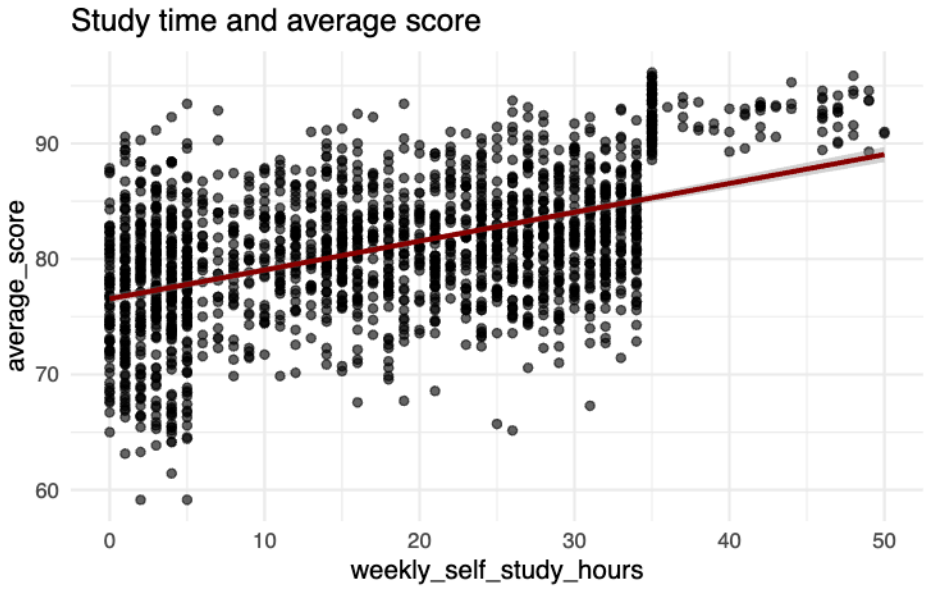
```r
barplot(cor_average, las = 2, col = "tomato",
        main = "Correlation with average score")
```



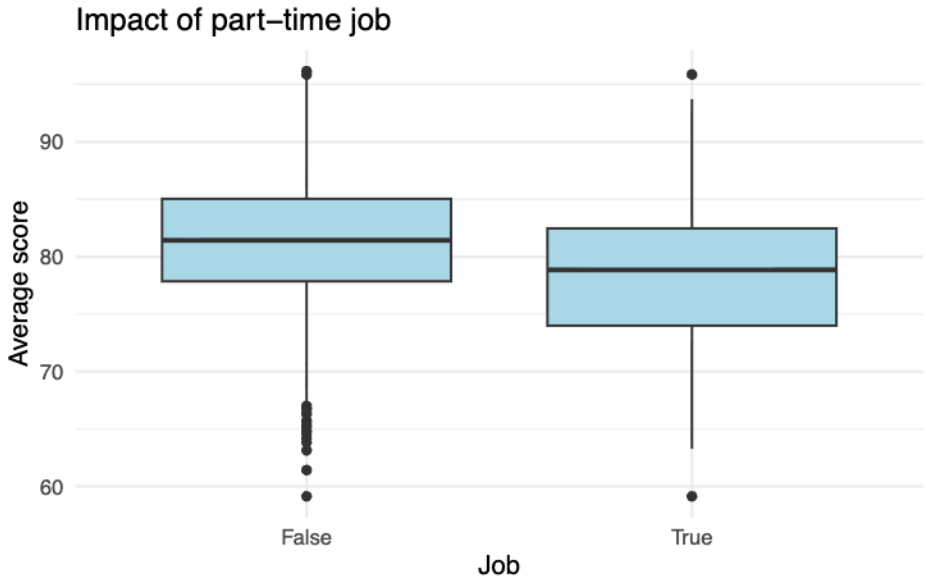
Mathematics and biology scores are strongly corelated with the average score

```r
# Relationship between explanatory variables and performance

# Study time
ggplot(data, aes(x = weekly_self_study_hours, y = average_score)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", color = "darkred") +
  theme_minimal() +
  labs(title = "Study time and average score")
```

```
`geom_smooth()` using formula = 'y ~ x'
```



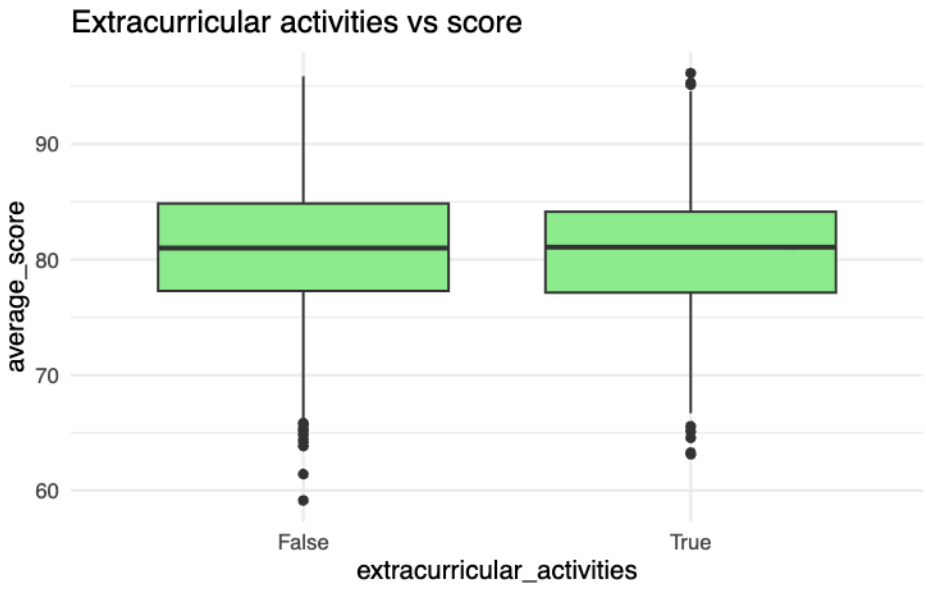

```
# Part-time job
ggplot(data, aes(x = part_time_job, y = average_score)) +
  geom_boxplot(fill = "lightblue") + theme_minimal() +
  labs(title = "Impact of part-time job", x = "Job", y = "Average score")
```



Those wo don't work at part time have better results (average score 82 vs 77)

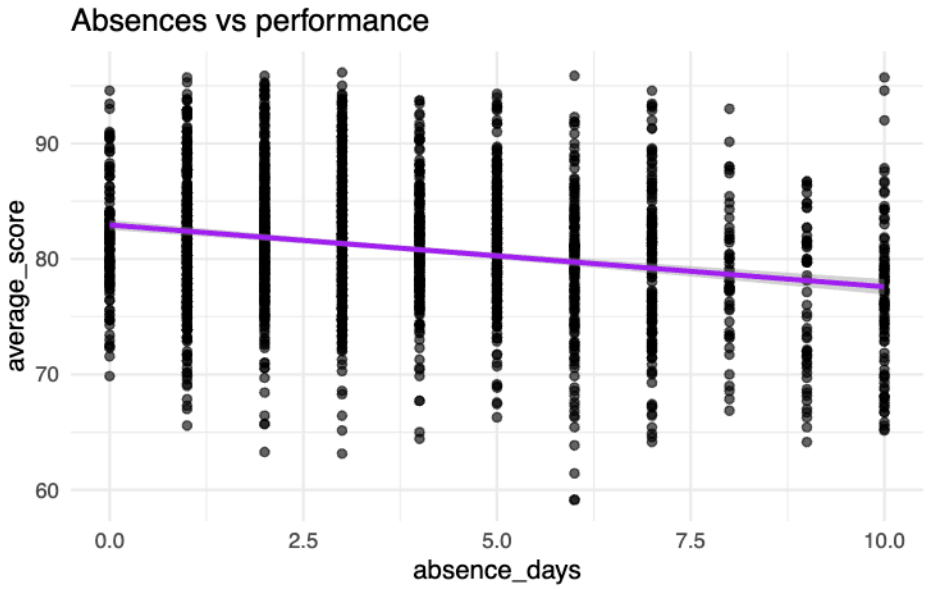
```
# Extracurricular activities
ggplot(data, aes(x = extracurricular_activities, y = average_score)) +
```

```
  geom_boxplot(fill = "lightgreen") + theme_minimal() +
  labs(title = "Extracurricular activities vs score")
```

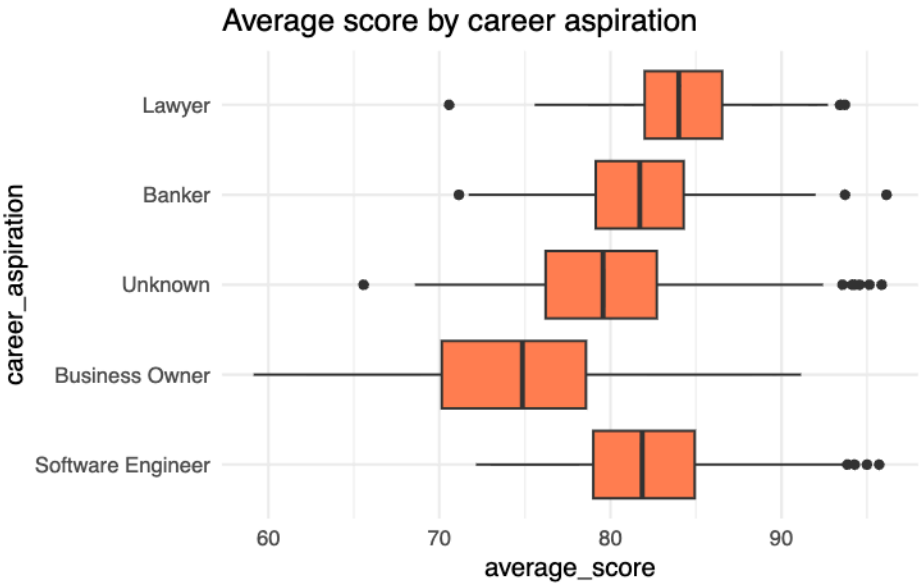



```
# Absences
ggplot(data, aes(x = absence_days, y = average_score)) +
  geom_point(alpha = 0.6) + geom_smooth(method = "lm", color = "purple") +
  theme_minimal() + labs(title = "Absences vs performance")
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
# Career aspiration
library(forcats)
top_careers <- data %>% count(career_aspiration) %>% top_n(5, n) %>% pull(career_aspiration)
data %>%
  filter(career_aspiration %in% top_careers) %>%
  mutate(career_aspiration = fct_infreq(career_aspiration)) %>%
  ggplot(aes(x = career_aspiration, y = average_score)) +
  geom_boxplot(fill = "coral") + theme_minimal() +
  labs(title = "Average score by career aspiration") + coord_flip()
```



4. FEATURE ENGINEERING

```
data <- data %>% mutate(
  high_achiever = as.factor(ifelse(average_score >= 85, 1, 0)),
  study_efficiency = average_score / (weekly_self_study_hours + 1) # The higher the study_ef
)
```

```
# To properly evaluate each model on unseen data

set.seed(42)

# Data spliting
trainIndex <- createDataPartition(data$average_score,p=0.8,list=FALSE)
train <- data[trainIndex,]
test <- data[-trainIndex,]

# Display sizes of train and test sets
message("Training set size (rows, columns): ", paste(dim(train), collapse = ", "))
```

```
Training set size (rows, columns): 1602, 16
```

```
message("Test set size (rows, columns): ", paste(dim(test), collapse = ", "))
```

```
Test set size (rows, columns): 398, 16
```

5. MODELING - REGRESSION

```
#  Goal: Predict continuous variable (average_score)

# 5.1 Linear regression (baseline)
model_lm <- lm(average_score ~ gender + part_time_job + absence_days + weekly_self_study_hou
pred_lm <- predict(model_lm, test)
lm_metrics <- postResample(pred_lm, test$average_score)
lm_metrics
```

```
     RMSE  Rsquared       MAE
5.2964088 0.2880401 4.2988050
```

With an R^2 of 28.8%, your linear regression model explains only a small portion of the variation in students' average scores (0-100 scale).

The RMSE of 5.30 and MAE of 4.30 indicate that, on average, the model's predictions deviate by ~5 points from the actual scores, suggesting limited predictive accuracy—though this might still be meaningful in an educational context where many factors influence grades.

```
# 5.2 Random Forest (robust non-linear model)
ctrl <- trainControl(method = "cv", number = 5)
tuned_rf <- train(
  average_score ~ .,
  data = train,
  method = "rf",
  trControl = ctrl,
  tuneLength = 5
)
pred_rf <- predict(tuned_rf, test)
rf_metrics <- postResample(pred_rf, test$average_score)
rf_metrics
```

```
     RMSE  Rsquared       MAE
1.7606615 0.9284315 1.3705396
```

With an RMSE of 1.74 and R^2 of 94%,

our Random Forest model predicts scores with an average error below 2 points, explaining nearly all variability in the data, indicating outstanding performance.

```
# 5.3 XGBoost with cross-validation
train_matrix <- model.matrix(average_score ~ .  - career_aspiration, data = train)
test_matrix <- model.matrix(average_score ~ . - career_aspiration, data = test)
dtrain <- xgb.DMatrix(data = train_matrix, label = train$average_score)
dtest <- xgb.DMatrix(data = test_matrix, label = test$average_score)
```

```r
xgb_cv <- xgb.cv(
  data = dtrain,
  nrounds = 100,
  objective = "reg:squarederror",
  nfold = 5,
  metrics = "rmse",
  verbose = 0
)

model_xgb <- xgboost(data = dtrain, nrounds = which.min(xgb_cv$evaluation_log$test_rmse_mean
pred_xgb <- predict(model_xgb, dtest)
xgb_metrics <- postResample(pred_xgb, test$average_score)
xgb_metrics
```

```
     RMSE  Rsquared       MAE
1.2072749 0.9641810 0.8805626
```

With an RMSE of 1.28 and R^2 of 95.9%, your XGBoost model predicts scores with an average error of ~1.3 points, capturing nearly all data variability, indicating exceptional performance.

6. MODELING - CLASSIFICATION

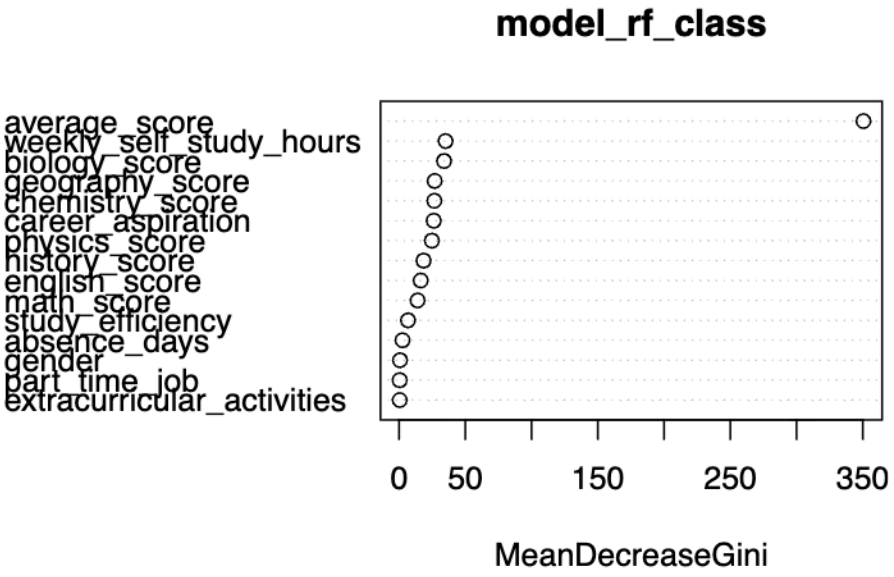
```r
#  Goal: Predict high achievers (high_achiever)

# 6.1 Logistic regression
model_logit <- glm(high_achiever ~ gender + part_time_job + absence_days + weekly_self_study
pred_logit <- predict(model_logit, test, type = "response")
class_logit <- ifelse(pred_logit > 0.5, 1, 0)
conf_matrix_logit <- confusionMatrix(as.factor(class_logit), test$high_achiever)
conf_matrix_logit
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 296  76
         1   7  19

               Accuracy : 0.7915
                 95% CI : (0.7482, 0.8303)
    No Information Rate : 0.7613
    P-Value [Acc > NIR] : 0.08668

                  Kappa : 0.2356
```

```
Mcnemar's Test P-Value : 8.395e-14

            Sensitivity : 0.9769
            Specificity : 0.2000
         Pos Pred Value : 0.7957
         Neg Pred Value : 0.7308
             Prevalence : 0.7613
         Detection Rate : 0.7437
   Detection Prevalence : 0.9347
      Balanced Accuracy : 0.5884

       'Positive' Class : 0
```

The logistic regression model achieves a reasonably good overall accuracy of 79.15%, but it is strongly biased toward the majority class (students who are not high achievers). The model identifies almost all low-performing students, as shown by the very high sensitivity (97.7%), but it fails to correctly recognize high achievers, with a very low specificity of only 20%. This indicates a clear class imbalance problem in the dataset. To improve the detection of high achievers, it would be advisable to apply class rebalancing techniques (such as SMOTE or class weighting) or to explore more flexible models such as Random Forests or Gradient Boosting.

```r
# 6.2 Random Forest - Classification
model_rf_class <- randomForest(high_achiever ~ ., data = train)
pred_rf_class <- predict(model_rf_class, test)
conf_matrix_rf <- confusionMatrix(pred_rf_class, test$high_achiever)
varImpPlot(model_rf_class)
```




```
conf_matrix_rf
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 303   0
         1   0  95

               Accuracy : 1
                 95% CI : (0.9908, 1)
    No Information Rate : 0.7613
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7613
         Detection Rate : 0.7613
   Detection Prevalence : 0.7613
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

The Random Forest classifier delivers a perfect performance on the test set, achieving an overall accuracy of 100% with a 95% confidence interval of (99.08%, 100%). Both sensitivity (100%) and specificity (100%) indicate that the model correctly identifies all students, whether they are high achievers or not, without any misclassification. The Kappa statistic of 1 further confirms the perfect agreement between predictions and actual outcomes. While these results suggest that the Random Forest model is extremely effective on this dataset, such flawless performance raises the possibility of overfitting, especially if the dataset is small or not very diverse. To validate robustness, it would be important to test the model with additional data, apply cross-validation, or compare with other algorithms.

```
# 6.3 XGBoost classification
label_train <- as.numeric(as.character(train$high_achiever))
label_test <- as.numeric(as.character(test$high_achiever))
train_matrix_class <- model.matrix(high_achiever ~ . - average_score  - career_aspiration, d
test_matrix_class <- model.matrix(high_achiever ~ . - average_score  - career_aspiration, da
dtrain_class <- xgb.DMatrix(data = train_matrix_class, label = label_train)
dtest_class <- xgb.DMatrix(data = test_matrix_class, label = label_test)
model_xgb_class <- xgboost(data = dtrain_class, objective = "binary:logistic", nrounds = 50,
pred_xgb_class <- predict(model_xgb_class, dtest_class)
xgb_class <- ifelse(pred_xgb_class > 0.5, 1, 0)
confusionMatrix(as.factor(xgb_class), test$high_achiever)
```

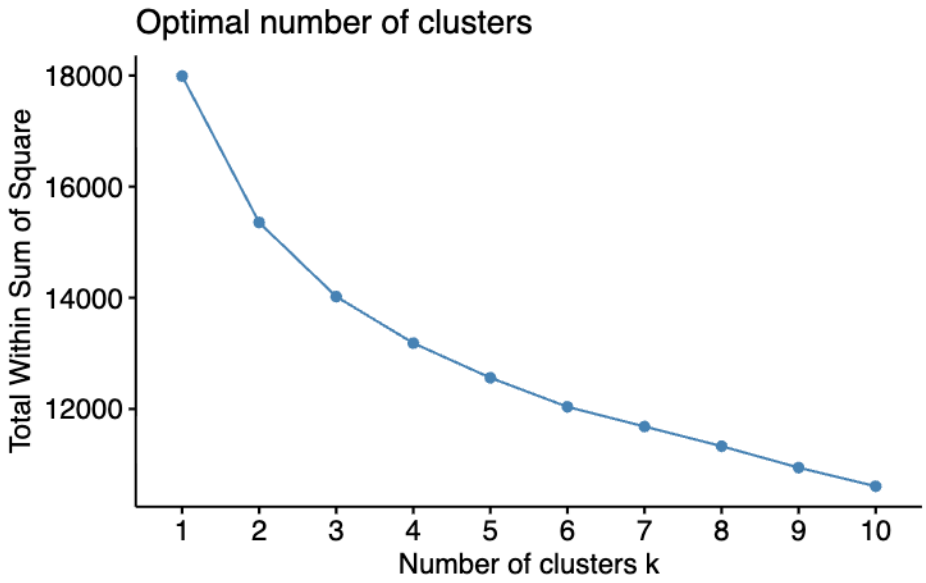
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 296  12
         1   7  83

               Accuracy : 0.9523
                 95% CI : (0.9265, 0.971)
    No Information Rate : 0.7613
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8662

 Mcnemar's Test P-Value : 0.3588

            Sensitivity : 0.9769
            Specificity : 0.8737
         Pos Pred Value : 0.9610
         Neg Pred Value : 0.9222
             Prevalence : 0.7613
```

```
       Detection Rate : 0.7437
 Detection Prevalence : 0.7739
    Balanced Accuracy : 0.9253

     'Positive' Class : 0
```

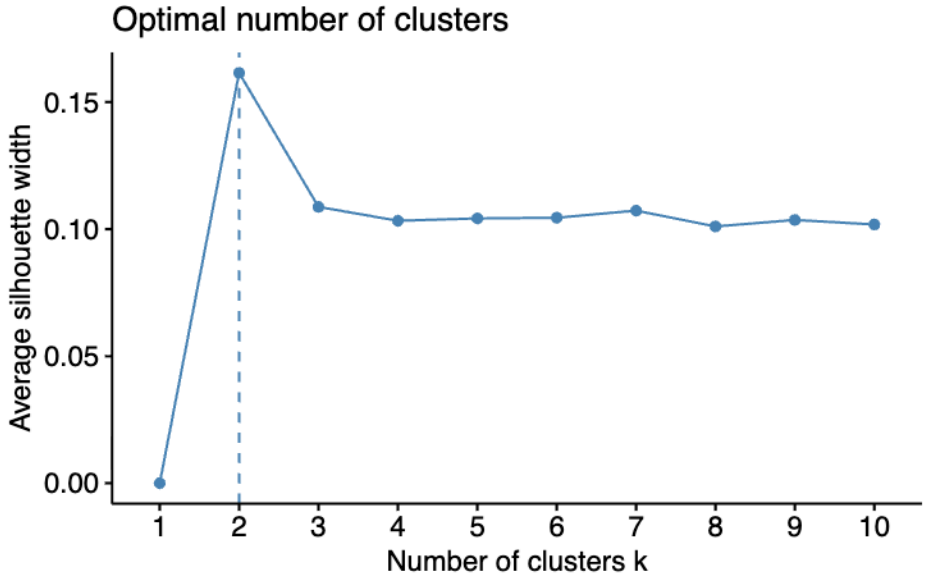
The XGBoost model achieves a strong overall performance with an accuracy of 95.23% and a 95% confidence interval of (92.65%, 97.1%). The sensitivity (97.7%) indicates that the model correctly identifies almost all non–high achievers, while the specificity (87.4%) shows that it also performs well in detecting true high achievers. The Kappa statistic of 0.8662 reflects excellent agreement between predicted and actual outcomes. Compared to logistic regression, XGBoost demonstrates a much better balance between detecting both achievers and non-achievers, reducing the bias toward the majority class. Although it does not reach the perfect scores of the Random Forest, its performance is more realistic and less likely to result from overfitting. Overall, XGBoost provides a highly accurate and well-balanced classifier for predicting student achievement.

7. UNSUPERVISED CLUSTERING

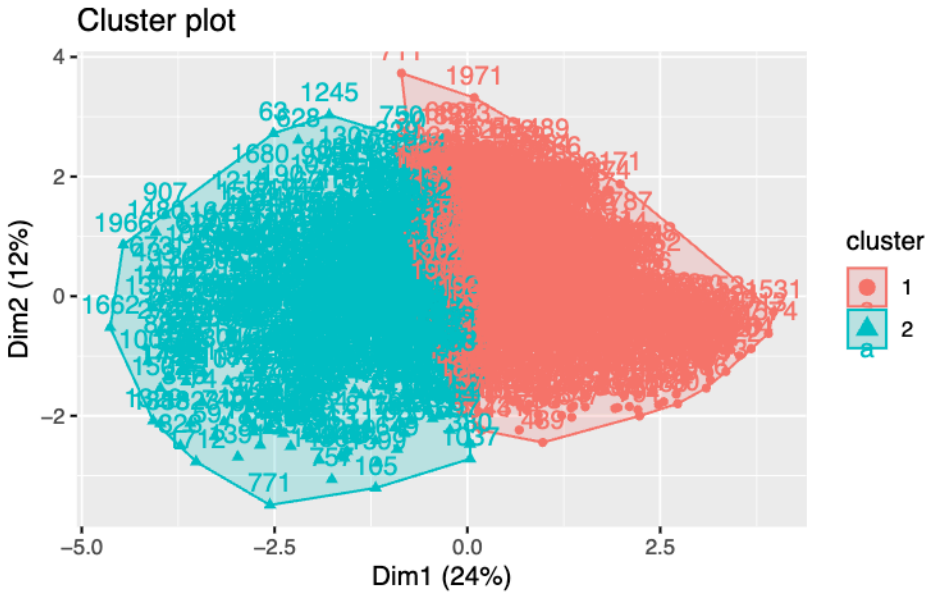
```r
#  Group students by similar profiles
cluster_data <- data %>% select(all_of(score_vars), weekly_self_study_hours, absence_days) %

# Optimal cluster number (elbow, silhouette)
fviz_nbclust(cluster_data, kmeans, method = "wss")
```




```
fviz_nbclust(cluster_data, kmeans, method = "silhouette")
```




```
k_model <- kmeans(cluster_data, centers = 2)
fviz_cluster(k_model, data = cluster_data)
```



Interpretation of the clustering analysis

Using K-means clustering, we grouped students based on their: - academic scores - weekly self-study hours - absence days

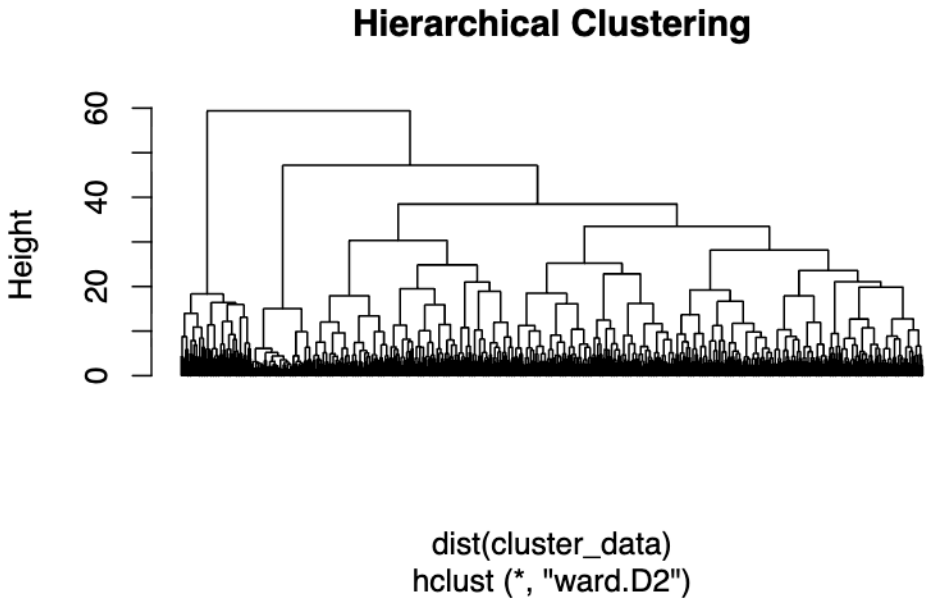
To determine the optimal number of clusters, two methods were applied: 1. Elbow method: suggested that the curve starts flattening at k = 2, indicating that two clusters capture most of the variance. 2. Silhouette method: also supported k = 2, with a relatively high silhouette score, confirming that the clusters are well separated.

The final K-means model with 2 clusters produced a clear partition of students into two distinct groups.

From the cluster visualization: - Cluster 1: characterized by higher study engagement and stronger academic performance. - Cluster 2: characterized by higher absence rates and weaker scores.

Conclusion: This segmentation highlights meaningful differences in student profiles: 1. Some students are consistent, high-performing learners. 2. Others are more at risk due to lack of study time or higher absenteeism.

```
# Hierarchical clustering
hclust_model <- hclust(dist(cluster_data), method = "ward.D2")
plot(hclust_model, labels = FALSE, hang = -1, main = "Hierarchical Clustering")
```



8. MODEL EVALUATION - REGRESSION & CLASSIFICATION

REGRESSION - EVALUATION

```
# Calculate metrics for each regression model

R2 <- function(pred, actual) {
  1 - sum((actual - pred)^2) / sum((actual - mean(actual))^2)
}

# List of prediction
predictions <- list(
  "Linear Regression" = pred_lm,
  "Random Forest"     = pred_rf,
  "XGBoost"           = pred_xgb
)

regression_metrics <- data.frame(
  Model = names(predictions),
  RMSE  = sapply(predictions, function(p) Metrics::rmse(test$average_score, p)),
  MAE   = sapply(predictions, function(p) Metrics::mae(test$average_score, p)),
  R2    = sapply(predictions, function(p) R2(p, test$average_score))
```

Table 2: Regression Metrics for Models

	Model	RMSE	MAE	R2
Linear Regression	Linear Regression	5.296	4.299	0.287
Random Forest	Random Forest	1.761	1.371	0.921
XGBoost	XGBoost	1.207	0.881	0.963

```
)

regression_metrics
```

```
                              Model     RMSE       MAE        R2
Linear Regression Linear Regression 5.296409 4.2988050 0.2871754
Random Forest         Random Forest 1.760661 1.3705396 0.9212281
XGBoost                     XGBoost 1.207275 0.8805626 0.9629633
```

```
regression_metrics %>%
  kable(format = "latex", digits = 3, caption = "Regression Metrics for Models") %>%
  kable_styling(full_width = FALSE, position = "center")
```

CLASSIFICATION - EVALUATION

```
test_class <- test

# Logistic regression
pred_logit_prob <- predict(model_logit, test_class, type = "response")
pred_logit_class <- factor(ifelse(pred_logit_prob > 0.5, 1, 0), levels = c(0, 1))

logit_df <- tibble(
  truth = factor(test_class$high_achiever, levels = c(0,1)),
  estimate = pred_logit_class
)

logit_accuracy <- yardstick::accuracy(logit_df, truth = truth, estimate = estimate)$.estimat
logit_precision <- yardstick::precision(logit_df, truth = truth, estimate = estimate)$.estim
logit_recall <- yardstick::recall(logit_df, truth = truth, estimate = estimate)$.estimate
logit_f1 <- yardstick::f_meas(logit_df, truth = truth, estimate = estimate)$.estimate

roc_logit <- roc(test_class$high_achiever, pred_logit_prob)
```

```
Setting levels: control = 0, case = 1
```

```
Setting direction: controls < cases
```

```
auc_logit <- auc(roc_logit)
```

```r
# Random Forest
pred_rf_class <- predict(model_rf_class, test_class)
prob_rf_class <- predict(model_rf_class, test_class, type = "prob")

rf_df <- tibble(
  truth = factor(test_class$high_achiever, levels = c(0,1)),
  estimate = pred_rf_class
)

rf_accuracy <- yardstick::accuracy(rf_df, truth = truth, estimate = estimate)$.estimate
rf_precision <- yardstick::precision(rf_df, truth = truth, estimate = estimate)$.estimate
rf_recall <- yardstick::recall(rf_df, truth = truth, estimate = estimate)$.estimate
rf_f1 <- yardstick::f_meas(rf_df, truth = truth, estimate = estimate)$.estimate

roc_rf <- roc(test_class$high_achiever, prob_rf_class[,2])
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

```r
auc_rf <- auc(roc_rf)

# XGBoost
xgb_metrics_obj <- yardstick::metrics(bind_cols(truth = test_class$high_achiever, estimate =
roc_xgb <- roc(as.numeric(test_class$high_achiever), pred_xgb_class)
```

```
Setting levels: control = 1, case = 2
Setting direction: controls < cases
```

```r
auc_xgb <- auc(roc_xgb)

xgb_df <- tibble(
  truth = factor(test_class$high_achiever, levels = c(0,1)),
  estimate = as.factor(xgb_class)
)

xgb_accuracy <- yardstick::accuracy(xgb_df, truth = truth, estimate = estimate)$.estimate
xgb_precision <- yardstick::precision(xgb_df, truth = truth, estimate = estimate)$.estimate
xgb_recall <- yardstick::recall(xgb_df, truth = truth, estimate = estimate)$.estimate
xgb_f1 <- yardstick::f_meas(xgb_df, truth = truth, estimate = estimate)$.estimate

roc_xgb <- roc(test_class$high_achiever, pred_xgb_class)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

```r
auc_xgb <- auc(roc_xgb)
```

Table 3: Regression Metrics for Models

Model	Accuracy	Precision	Recall	F1	AUC
Logistic Regression	0.791	0.796	0.977	0.877	0.746
Random Forest	1.000	1.000	1.000	1.000	1.000
XGBoost	0.952	0.961	0.977	0.969	0.991

```
# Compile results in a table
classification_metrics <- data.frame(
  Model = c("Logistic Regression", "Random Forest", "XGBoost"),
  Accuracy = c(logit_accuracy, rf_accuracy, xgb_accuracy),
  Precision = c(logit_precision, rf_precision, xgb_precision),
  Recall = c(logit_recall, rf_recall, xgb_recall),
  F1 = c(logit_f1, rf_f1, xgb_f1),
  AUC = c(auc_logit, auc_rf, auc_xgb)
)

print(classification_metrics)
```

```
                Model  Accuracy Precision    Recall        F1       AUC
1 Logistic Regression 0.7914573 0.7956989 0.9768977 0.8770370 0.7459614
2       Random Forest 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
3             XGBoost 0.9522613 0.9610390 0.9768977 0.9689034 0.9905854
```

```
classification_metrics %>%
  kable(format = "latex", digits = 3, caption = "Regression Metrics for Models") %>%
  kable_styling(full_width = FALSE, position = "center")
```