Project Report

Saniya Bekova

2024-12-04

```r
library(tidymodels)
library(tidyverse)
library(bonsai)
library(themis)
library(readxl)
library(haven) #for loading other datafiles (SAS, STATA, SPSS, etc.)
library(stringr)
library(lubridate)
library(ggplot2)
library(dplyr)
library(ggrepel)
library(scales)
```

1. Introduction and Data

Data was collected from following resources:

UNESCO Institute for Statistics (UIS): Educational indicators data collected from https://sdg4-data.uis.unesco.org/. This data provides comprehensive educational metrics for various countries across multiple years and was last updated in September 2024. Data from 2013 to 2024

TidyTuesday GitHub Repository: Data related to the International Mathematical Olympiad (IMO) collected from https://github.com/rfordatascience/tidytuesday/blob/master/data/2024/2024-09-24/readme.md. The IMO data tracks country-level performance, including scores, medals, and rankings, and was also updated in September 2024. Data from 1959 to 2024

DataBank: https://databank.worldbank.org/source/education-statistics-%5e-all-indicators # other educational series was collected from this site.

Units of Analysis

- **Countries**: Each country represents a unit of analysis in this dataset, with attributes related to educational performance (such as completion rates, expenditure) and their success in the IMO (average score, team size, medals won).
- **Time**: The dataset spans multiple years, allowing for the analysis of trends over time in both education indicators and IMO performance.

Topic Description

This project aims to explore how a country's education system impacts its performance in the **International Mathematical Olympiad (IMO)**. I am particularly interested in studying the relationship between **government spending on education, primary and secondary education completion rates, literacy rates** and a country's success in the IMO.

Why This Topic?

This topic interests me because I want to understand how a country's investment in education and the quality of its education system influence its ability to succeed in international academic competitions like the IMO. By exploring these relationships, I hope to identify the factors that most strongly contribute to winning medals or achieving high scores in the IMO.

Expectations

I expect to find a **positive relationship** between a country's investment in education and its success in the IMO.

```r
#| message: false
#| warning: false
#load data
education_data <- read_csv('data/education_data.csv')  # Dataset 1: Educational indicators
```

```
Rows: 1999 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Country
dbl (7): Year, Value_primary_edu_completion_rate, Value_lower_sec_edu_comple...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
imo_data <- read_csv('data/imo_data.csv')  # Dataset 2: IMO competition results
```

```
Rows: 3780 Columns: 18
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): country, leader, deputy_leader
dbl (14): year, team_size_all, team_size_male, team_size_female, p1, p2, p3,...
lgl  (1): p7

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Main Outcome/Target (Y Variable):

The main outcome or target variable in this analysis is the **average score per contestant for each country in a given year**. This is calculated by summing the scores from problems 1 to 7 for each country's team and dividing the total by the number of participants (team_size_all). This variable represents how well the entire team from each country performed in the International Mathematical Olympiad (IMO).

This outcome is a good fit for the study because it provides a clear measure of how well a country's education system prepares students for international competitions. By using the average score, the analysis captures the performance of the whole team, not just the top individual performers. This is important for understanding the impact of educational investments, such as government spending on education, literacy rates, and school completion rates, on a country's success in the IMO.

The average score per contestant gives a more detailed and fair comparison between countries. It helps to evaluate the overall strength of the team, making it a useful measure for examining how education systems contribute to performance in international competitions.

By focusing on the average score, this analysis can effectively explore the connection between educational investments and a country's overall performance in the IMO, making it a suitable target for this project.

2.Exploratory Data Analysis

2.1 Data Cleaning

```r
literacy_rate_by_country_and_region <- read_csv("data/literacy_data.csv")

country_and_region_data <- read_xlsx("data/P_Data_Extract_From_Education_Statistics_-_All_Inc

youth_literacy_rate <- literacy_rate_by_country_and_region |>
  filter(Series == "Youth literacy rate, population 15-24 years, both sexes (%)")

youth_literacy_rate <- pivot_longer(youth_literacy_rate,
                                    cols = c('2009 [YR2009]', '2010 [YR2010]', '2011 [YR2011]
                                    names_to = "Year",
                                    values_to = "Literacy_Rate") |>
  mutate(Year = str_replace(Year, " \\[YR[0-9]+\\]", "")) |>
  select("Country Code", "Country Name", Year, Literacy_Rate)

youth_literacy_rate <- youth_literacy_rate |>
  mutate(Year = as.double(Year),
         Literacy_Rate = as.double(Literacy_Rate))


education_data_joined <- education_data |>
  left_join(youth_literacy_rate,
            by = c("Country" = "Country Name", "Year" = "Year"),
            suffix = c("", "_new"))

education_data_updated <- education_data_joined |>
  mutate(Value_literacy_rate = coalesce(Value_literacy_rate,
                                        Literacy_Rate)) |>
  select(-Literacy_Rate)
```

```r
education_with_region <- education_data_updated |>
  left_join(country_and_region_data, by = c("Country" = "Long Name"))

education_full <- education_with_region |>
  left_join(youth_literacy_rate,
            by = c("Region" = "Country Name", "Year" = "Year"),
            suffix = c("", "_region")) |>
  left_join(youth_literacy_rate,
            by = c("Country" = "Country Name", "Year" = "Year"),
            suffix = c("", "_country")) |>
  left_join(youth_literacy_rate,
            by = c("Income Group" = "Country Name", "Year" = "Year"),
            suffix = c("", "_income"))
```

```r
education_data_updated <- education_full |>
  mutate(Value_literacy_rate = coalesce(Value_literacy_rate,
                                        Literacy_Rate_country,
                                        Literacy_Rate)) |>
  select(-Literacy_Rate_country, -Literacy_Rate)

education_data_updated <- education_data_updated |>
  mutate(Value_literacy_rate = coalesce(Value_literacy_rate,
                                        Literacy_Rate_income)) |>
  select(Country,
         Year,
         `Country Code`,
         Value_gross_enr_ratio_for_tertiary_edu,
         Value_gov_expen_as_perc_of_GPP,
         Value_literacy_rate,
         Region,
         `Income Group`)
```

2.2 Merging Educational Data:

The educational indicators from the UNESCO Institute for Statistics were split across multiple variables (e.g., primary and secondary education completion rates, government expenditure on education). These were merged into a single dataset, ensuring all relevant indicators were available for each country and year. The merging process involved handling mismatched country names between the datasets. For example, differences such as "Kyrgyz Republic" vs. "Kyrgyzstan" were corrected manually to ensure proper alignment of the data.

2.3 Combining IMO Data with Educational Data:

The educational data (which now included literacy rates, completion rates, and government expenditure) was merged with the IMO performance data (e.g., team scores, medals won) to create a comprehensive dataset. This allowed for the analysis of the relationship between a country's educational indicators and its performance in the IMO.

2.4 Creating New Variables :

`Medal_Efficiency` This variable was created by dividing the total number of medals (gold, silver, and bronze) won by a country by its team size (team_size_all). It measures how efficiently a country converts its team into medals, providing insights into performance relative to team size.

`Gov_Investment_Per_Medal` This variable measures the amount of government expenditure on education required to produce one IMO medal. It was created by dividing the government expenditure as a percentage of GDP by the total number of medals won.

`Lit_Performance_Ratio` This variable measures the ratio between a country's youth literacy rate and its average IMO score or total number of medals won, helping to explore the link between literacy and performance.

These variables were created before the training-test split to avoid any issues related to leakage between the datasets.

```
# Calculate total score by summing problem scores p1 to p7
imo_data <- imo_data |>
  rowwise() |>
  mutate(total_score = sum(c_across(p1:p7), na.rm = TRUE)) |>
  ungroup()

# Calculate average score per contestant by dividing total score by team size
imo_data <- imo_data |>
  mutate(average_score_per_contestant = total_score / team_size_all)

imo_data <- imo_data |>
  mutate(medal_Efficiency = ifelse(team_size_all > 0,
                                   (awards_gold + awards_silver + awards_bronze) / team_size_
                                   NA))

# Merging 'imo_data' with 'education_data_updated'
combined_data <- imo_data |>
  left_join(education_data_updated, by = c("country" = "Country", "year" = "Year"))

combined_data <- combined_data |>
  mutate(Gov_Investment_Per_Medal = ifelse((awards_gold + awards_silver + awards_bronze) > 0
                                           Value_gov_expen_as_perc_of_GPP / (awards_gold + aw
                                           NA),
         Lit_Performance_Ratio = ifelse(average_score_per_contestant > 0,
                                        Value_literacy_rate / average_score_per_contestant,
                                        NA))

combined_data <- combined_data |>
  filter(year > 2008 & year < 2020)
```

```r
summary(combined_data)
```

```
      year         country           team_size_all   team_size_male
 Min.   :2009   Length:1142        Min.   :1.000   Min.   :1.000
 1st Qu.:2011   Class :character   1st Qu.:6.000   1st Qu.:5.000
 Median :2014   Mode  :character   Median :6.000   Median :5.000
 Mean   :2014                      Mean   :5.511   Mean   :5.001
 3rd Qu.:2017                      3rd Qu.:6.000   3rd Qu.:6.000
 Max.   :2019                      Max.   :6.000   Max.   :6.000
                                                   NA's   :11
 team_size_female       p1              p2              p3
 Min.   :1.000    Min.   : 0.00   Min.   : 0.00   Min.   : 0.000
 1st Qu.:1.000    1st Qu.:17.00   1st Qu.: 2.00   1st Qu.: 0.000
 Median :1.000    Median :34.00   Median : 9.00   Median : 0.000
 Mean   :1.331    Mean   :28.25   Mean   :13.01   Mean   : 2.986
 3rd Qu.:2.000    3rd Qu.:41.00   3rd Qu.:22.00   3rd Qu.: 3.000
 Max.   :6.000    Max.   :42.00   Max.   :42.00   Max.   :42.000
 NA's   :668      NA's   :1       NA's   :1       NA's   :1
       p4              p5              p6              p7
 Min.   : 0.00   Min.   : 0.00   Min.   : 0.000   Mode:logical
 1st Qu.:11.00   1st Qu.: 2.00   1st Qu.: 0.000   NA's:1142
 Median :26.00   Median : 7.00   Median : 0.000
 Mean   :24.47   Mean   :11.54   Mean   : 2.186
 3rd Qu.:39.00   3rd Qu.:18.00   3rd Qu.: 2.000
 Max.   :42.00   Max.   :42.00   Max.   :36.000
 NA's   :1       NA's   :1       NA's   :1
  awards_gold     awards_silver     awards_bronze   awards_honorable_mentions
 Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   Median :0.0000   Median :1.000   Median :1.000
 Mean   :0.461   Mean   :0.9351   Mean   :1.339   Mean   :1.409
 3rd Qu.:0.000   3rd Qu.:2.0000   3rd Qu.:2.000   3rd Qu.:2.000
 Max.   :6.000   Max.   :6.0000   Max.   :6.000   Max.   :6.000
 NA's   :1       NA's   :1        NA's   :1       NA's   :1
    leader          deputy_leader       total_score
 Length:1142        Length:1142        Min.   :  0.00
 Class :character   Class :character   1st Qu.: 39.25
 Mode  :character   Mode  :character   Median : 79.00
                                       Mean   : 82.37
                                       3rd Qu.:119.75
                                       Max.   :227.00
```

```
 average_score_per_contestant medal_Efficiency Country Code
 Min.   : 0.000               Min.   :0.0000   Length:1142
 1st Qu.: 8.167               1st Qu.:0.0000   Class :character
 Median :13.667               Median :0.4000   Mode  :character
 Mean   :14.350               Mean   :0.4689
 3rd Qu.:20.000               3rd Qu.:0.8333
 Max.   :37.833               Max.   :1.0000
                              NA's   :1
 Value_gross_enr_ratio_for_tertiary_edu Value_gov_expen_as_perc_of_GPP
 Min.   :  4.02                         Min.   : 0.390
 1st Qu.: 37.49                         1st Qu.: 3.540
 Median : 59.18                         Median : 4.520
 Mean   : 56.98                         Mean   : 4.564
 3rd Qu.: 77.23                         3rd Qu.: 5.450
 Max.   :143.96                         Max.   :10.670
 NA's   :573                            NA's   :573
 Value_literacy_rate    Region          Income Group
 Min.   : 50.00      Length:1142        Length:1142
 1st Qu.: 98.25      Class :character   Class :character
 Median : 99.57      Mode  :character   Mode  :character
 Mean   : 96.77
 3rd Qu.: 99.66
 Max.   :100.00
 NA's   :571
 Gov_Investment_Per_Medal Lit_Performance_Ratio
 Min.   :0.160            Min.   :  2.620
 1st Qu.:0.780            1st Qu.:  5.054
 Median :1.126            Median :  7.470
 Mean   :1.969            Mean   : 15.681
 3rd Qu.:2.632            3rd Qu.: 12.208
 Max.   :8.960            Max.   :446.425
 NA's   :718              NA's   :575
```

```
write_csv(combined_data, "data/combined_data.csv")
```

2.5 Excluded Observations

Observations from years prior to 2009 and after 2019 were excluded due to insufficient data availability.

Additionally, certain features were excluded due to a significant number of missing values (approximately 1,500 NAs out of 1,999 total observations). These features included:

1. Completion rate, primary education, both sexes (%)
2. Completion rate, lower secondary education, both sexes (%)
3. Completion rate, upper secondary education, both sexes (%)

Since no relevant data was available to fill the missing values, these features were omitted from the analysis.

2.6 Handling missing data

The summary showed that the literacy_rate feature had about 1,500 missing values, indicating that we lacked sufficient data. To address this, I sourced an additional dataset for literacy rates from the World Bank (https://databank.worldbank.org/source/education-statistics-%5e-all-indicators). When missing data was'nt found for a specific country, the missing NA values were replaced with regional data. After all of these if we have NA's it will be replaced with mean value Other NA's from numeric features will be imputed with mean value

Also added the following features to improve performance:

Government expenditure on education, constant US$ (millions)

Government expenditure on education, US$ (millions)

GDP per capita (current US$)

Expenditure on education as % of total government expenditure (%),

Current expenditure as % of total expenditure in public institutions (%)

Annual statutory teacher salaries in public institutions in USD. Primary. 10 years of experience

Labor force with advanced education (% of total labor force)

Internet users (per 100 people)

Percentage of graduates from tertiary education graduating from Natural Sciences, Mathematics and Statistics programmes, both sexes (%)'

```
education_data_new <- read_csv('data/New_Education_Data.csv')

education_data_new_1 <- education_data_new |>
  filter(Series %in% c('Government expenditure on education, constant US$ (millions)',
                       'Government expenditure on education, US$ (millions)',
                       'GDP per capita (current US$)',
                       'Expenditure on education as % of total government expenditure (%)',
                       'Current expenditure as % of total expenditure in public institutions
                       'Annual statutory teacher salaries in public institutions in USD. Pri
                       'Labor force with advanced education (% of total labor force)',
                       'Internet users (per 100 people)',
                       'Percentage of graduates from tertiary education graduating from Natu


head(education_data_new_1)
```

```
# A tibble: 6 x 15
  `Country Name` `Country Code` Series             `Series Code` `2019 [YR2019]`
  <chr>          <chr>          <chr>              <chr>         <chr>
1 Afghanistan    AFG            Percentage of gra~ SE.TER.GRAD.~ ..
2 Afghanistan    AFG            Internet users (p~ IT.NET.USER.~ ..
3 Afghanistan    AFG            Labor force with ~ SL.TLF.ADVN.~ ..
4 Afghanistan    AFG            Government expend~ UIS.X.US.FSG~ ..
5 Afghanistan    AFG            Government expend~ UIS.X.USCONS~ ..
6 Afghanistan    AFG            GDP per capita (c~ NY.GDP.PCAP.~ ..
# i 10 more variables: `2018 [YR2018]` <chr>, `2017 [YR2017]` <chr>,
#   `2016 [YR2016]` <chr>, `2015 [YR2015]` <chr>, `2014 [YR2014]` <chr>,
#   `2013 [YR2013]` <chr>, `2012 [YR2012]` <chr>, `2011 [YR2011]` <chr>,
#   `2010 [YR2010]` <chr>, `2009 [YR2009]` <chr>
```

```
educational_data_2 <-  pivot_longer(education_data_new_1,
                                    cols = c('2009 [YR2009]', '2010 [YR2010]', '2011 [YR2011]
                                    names_to = "Year",
                                    values_to = "Value") |>
  mutate(Year = str_replace(Year, " \\[YR[0-9]+\\]", ""))

educational_data_2 |>
  pivot_wider(names_from = `Series Code`,
              values_from = Value) |>
  select(`Country Name`, `Country Code`, Year, SE.TER.GRAD.SC.ZS,
         IT.NET.USER.P2, SL.TLF.ADVN.ZS, UIS.X.US.FSGOV, UIS.X.USCONST.FSGOV, NY.GDP.PCAP.CD
```

```
# A tibble: 26,928 x 12
   `Country Name` `Country Code` Year  SE.TER.GRAD.SC.ZS IT.NET.USER.P2
   <chr>          <chr>          <chr> <chr>             <chr>
 1 Afghanistan    AFG            2009  ..                <NA>
 2 Afghanistan    AFG            2010  ..                <NA>
 3 Afghanistan    AFG            2011  ..                <NA>
 4 Afghanistan    AFG            2012  ..                <NA>
 5 Afghanistan    AFG            2013  ..                <NA>
 6 Afghanistan    AFG            2014  ..                <NA>
 7 Afghanistan    AFG            2015  ..                <NA>
 8 Afghanistan    AFG            2016  ..                <NA>
 9 Afghanistan    AFG            2017  ..                <NA>
10 Afghanistan    AFG            2018  ..                <NA>
# i 26,918 more rows
# i 7 more variables: SL.TLF.ADVN.ZS <chr>, UIS.X.US.FSGOV <chr>,
#   UIS.X.USCONST.FSGOV <chr>, NY.GDP.PCAP.CD <chr>, SE.XPD.TOTL.GB.ZS <chr>,
#   SE.XPD.CUR.TOTL.ZS <chr>, OECD.TSAL.1.E10 <chr>
```

```
SE.TER.GRAD.SC.ZS <- educational_data_2 |>
  filter(`Series Code` == "SE.TER.GRAD.SC.ZS") |>
  mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)


summary(SE.TER.GRAD.SC.ZS)
```

```
      Year      Country Name       Country Code           Value
 Min.   :2009   Length:2992        Length:2992        Min.   : 0.000
 1st Qu.:2011   Class :character   Class :character   1st Qu.: 2.572
 Median :2014   Mode  :character   Mode  :character   Median : 4.370
 Mean   :2014                                         Mean   : 4.679
 3rd Qu.:2017                                         3rd Qu.: 6.075
 Max.   :2019                                         Max.   :23.572
                                                      NA's   :2260
```

```
IT.NET.USER.P2 <- educational_data_2 |>
  filter(`Series Code` == "IT.NET.USER.P2") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)
```

```r
SL.TLF.ADVN.ZS <- educational_data_2 |>
  filter(`Series Code` == "SL.TLF.ADVN.ZS") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

UIS.X.US.FSGOV <- educational_data_2 |>
  filter(`Series Code` == "UIS.X.US.FSGOV") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

UIS.X.USCONST.FSGOV <- educational_data_2 |>
  filter(`Series Code` == "UIS.X.USCONST.FSGOV") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

NY.GDP.PCAP.CD <- educational_data_2 |>
  filter(`Series Code` == "NY.GDP.PCAP.CD") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

SE.XPD.TOTL.GB.ZS <- educational_data_2 |>
  filter(`Series Code` == "SE.XPD.TOTL.GB.ZS") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

SE.XPD.CUR.TOTL.ZS <- educational_data_2 |>
  filter(`Series Code` == "SE.XPD.CUR.TOTL.ZS") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)

OECD.TSAL.1.E10 <- educational_data_2 |>
  filter(`Series Code` == "OECD.TSAL.1.E10") |>
    mutate(Year =  as.numeric(Year),
         Value = as.numeric(Value)) |>
  select(Year, `Country Name`, `Country Code`, Value)
```

```r
combined_educ_data <- read_csv("data/combined_data.csv")

combined_educ_data <- combined_educ_data |>
  left_join(SE.TER.GRAD.SC.ZS,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(SE.TER.GRAD.SC.ZS = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(IT.NET.USER.P2,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(IT.NET.USER.P2 = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(SL.TLF.ADVN.ZS,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(SL.TLF.ADVN.ZS = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(UIS.X.US.FSGOV,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(UIS.X.US.FSGOV = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(UIS.X.USCONST.FSGOV,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(UIS.X.USCONST.FSGOV = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(NY.GDP.PCAP.CD,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(NY.GDP.PCAP.CD = as.numeric(Value)) |>
```

```r
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(SE.XPD.TOTL.GB.ZS,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(SE.XPD.TOTL.GB.ZS = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(SE.XPD.CUR.TOTL.ZS,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(SE.XPD.CUR.TOTL.ZS = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)

combined_educ_data <- combined_educ_data |>
  left_join(OECD.TSAL.1.E10,
            by = c("country" = "Country Name", "year" = "Year"),
            suffix = c("", "_new")) |>
  mutate(OECD.TSAL.1.E10 = as.numeric(Value)) |>
  select(-`Country Code_new`, -Value)


summary(combined_educ_data)
```

```
      year        country          team_size_all   team_size_male
 Min.   :2009   Length:1142        Min.   :1.000   Min.   :1.000
 1st Qu.:2011   Class :character   1st Qu.:6.000   1st Qu.:5.000
 Median :2014   Mode  :character   Median :6.000   Median :5.000
 Mean   :2014                      Mean   :5.511   Mean   :5.001
 3rd Qu.:2017                      3rd Qu.:6.000   3rd Qu.:6.000
 Max.   :2019                      Max.   :6.000   Max.   :6.000
                                                   NA's   :11
 team_size_female       p1              p2              p3
 Min.   :1.000    Min.   : 0.00   Min.   : 0.00   Min.   : 0.000
 1st Qu.:1.000    1st Qu.:17.00   1st Qu.: 2.00   1st Qu.: 0.000
 Median :1.000    Median :34.00   Median : 9.00   Median : 0.000
 Mean   :1.331    Mean   :28.25   Mean   :13.01   Mean   : 2.986
 3rd Qu.:2.000    3rd Qu.:41.00   3rd Qu.:22.00   3rd Qu.: 3.000
 Max.   :6.000    Max.   :42.00   Max.   :42.00   Max.   :42.000
 NA's   :668      NA's   :1       NA's   :1       NA's   :1
```

```
      p4              p5              p6             p7
 Min.   : 0.00   Min.   : 0.00   Min.   : 0.000   Mode:logical
 1st Qu.:11.00   1st Qu.: 2.00   1st Qu.: 0.000   NA's:1142
 Median :26.00   Median : 7.00   Median : 0.000
 Mean   :24.47   Mean   :11.54   Mean   : 2.186
 3rd Qu.:39.00   3rd Qu.:18.00   3rd Qu.: 2.000
 Max.   :42.00   Max.   :42.00   Max.   :36.000
 NA's   :1       NA's   :1       NA's   :1
  awards_gold    awards_silver    awards_bronze   awards_honorable_mentions
 Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   Median :0.0000   Median :1.000   Median :1.000
 Mean   :0.461   Mean   :0.9351   Mean   :1.339   Mean   :1.409
 3rd Qu.:0.000   3rd Qu.:2.0000   3rd Qu.:2.000   3rd Qu.:2.000
 Max.   :6.000   Max.   :6.0000   Max.   :6.000   Max.   :6.000
 NA's   :1       NA's   :1        NA's   :1       NA's   :1
    leader          deputy_leader       total_score
 Length:1142        Length:1142        Min.   :  0.00
 Class :character   Class :character   1st Qu.: 39.25
 Mode  :character   Mode  :character   Median : 79.00
                                       Mean   : 82.37
                                       3rd Qu.:119.75
                                       Max.   :227.00

 average_score_per_contestant medal_Efficiency Country Code
 Min.   : 0.000               Min.   :0.0000   Length:1142
 1st Qu.: 8.167               1st Qu.:0.0000   Class :character
 Median :13.667               Median :0.4000   Mode  :character
 Mean   :14.350               Mean   :0.4689
 3rd Qu.:20.000               3rd Qu.:0.8333
 Max.   :37.833               Max.   :1.0000
                              NA's   :1
 Value_gross_enr_ratio_for_tertiary_edu Value_gov_expen_as_perc_of_GPP
 Min.   :  4.02                         Min.   : 0.390
 1st Qu.: 37.49                         1st Qu.: 3.540
 Median : 59.18                         Median : 4.520
 Mean   : 56.98                         Mean   : 4.564
 3rd Qu.: 77.23                         3rd Qu.: 5.450
 Max.   :143.96                         Max.   :10.670
 NA's   :573                            NA's   :573
 Value_literacy_rate    Region          Income Group
 Min.   : 50.00      Length:1142        Length:1142
 1st Qu.: 98.25      Class :character   Class :character
```

```
 Median : 99.57      Mode  :character   Mode  :character
 Mean   : 96.77
 3rd Qu.: 99.66
 Max.   :100.00
 NA's   :571
 Gov_Investment_Per_Medal Lit_Performance_Ratio SE.TER.GRAD.SC.ZS
 Min.   :0.160            Min.   :  2.620       Min.   : 0.000
 1st Qu.:0.780            1st Qu.:  5.054       1st Qu.: 2.989
 Median :1.126            Median :  7.470       Median : 4.442
 Mean   :1.969            Mean   : 15.681       Mean   : 4.912
 3rd Qu.:2.632            3rd Qu.: 12.208       3rd Qu.: 6.277
 Max.   :8.960            Max.   :446.425       Max.   :23.572
 NA's   :718              NA's   :575           NA's   :667
 IT.NET.USER.P2  SL.TLF.ADVN.ZS  UIS.X.US.FSGOV     UIS.X.USCONST.FSGOV
 Min.   : 0.53   Min.   :57.08   Min.   :   146.7   Min.   :   170.7
 1st Qu.:37.31   1st Qu.:75.16   1st Qu.:  1848.0   1st Qu.:  1802.9
 Median :60.31   Median :80.02   Median :  6618.9   Median :  6878.5
 Mean   :56.68   Mean   :78.87   Mean   : 24732.0   Mean   : 23150.8
 3rd Qu.:78.89   3rd Qu.:83.07   3rd Qu.: 28688.3   3rd Qu.: 27215.4
 Max.   :99.01   Max.   :94.33   Max.   :227371.3   Max.   :179812.0
 NA's   :325     NA's   :508     NA's   :596        NA's   :598
 NY.GDP.PCAP.CD   SE.XPD.TOTL.GB.ZS SE.XPD.CUR.TOTL.ZS OECD.TSAL.1.E10
 Min.   :   476   Min.   : 5.644    Min.   : 63.95     Min.   : 1855
 1st Qu.:  4114   1st Qu.:11.142    1st Qu.: 89.60     1st Qu.:28262
 Median :  9934   Median :13.411    Median : 92.84     Median :37609
 Mean   : 21210   Mean   :14.134    Mean   : 91.72     Mean   :39316
 3rd Qu.: 32483   3rd Qu.:16.221    3rd Qu.: 95.29     3rd Qu.:48419
 Max.   :178846   Max.   :31.372    Max.   :100.00     Max.   :96224
 NA's   :280      NA's   :603       NA's   :697        NA's   :872
```

```r
write_csv(combined_educ_data, "data/combined_educ_data.csv")
```

2.7 Split data

```r
#-country, -p7, -leader, -deputy_leader, -`Country Code`,-Region, -`Income Group`, -medal_Ef


#year, team_size_all, team_size_male, Value_gross_enr_ratio_for_tertiary_edu, Value_gov_expe
combined_educ_data <-read_csv("data/combined_educ_data.csv")
```

```r
education_numeric_data <- combined_educ_data |>
  select(-country, -p7, -leader, -deputy_leader, -`Country Code`,-Region, -`Income Group`, -r

str(education_numeric_data)
```

```
tibble [1,142 x 19] (S3: tbl_df/tbl/data.frame)
 $ year                                 : num [1:1142] 2019 2019 2019 2019 2019 ...
 $ team_size_all                        : num [1:1142] 6 6 6 6 6 6 6 6 6 6 ...
 $ team_size_male                       : num [1:1142] 6 6 6 6 6 6 6 6 5 5 ...
 $ team_size_female                     : num [1:1142] NA NA NA NA NA NA NA NA 1 1 ...
 $ average_score_per_contestant         : num [1:1142] 37.8 37.8 37.7 31.2 30.8 ...
 $ Value_gross_enr_ratio_for_tertiary_edu: num [1:1142] NA 87.9 94 NA 45.4 ...
 $ Value_gov_expen_as_perc_of_GPP       : num [1:1142] NA 4.96 4.68 NA 3.02 3.7 NA 2.73 3.6
 $ Value_literacy_rate                  : num [1:1142] NA NA 98.7 NA 98.5 ...
 $ Gov_Investment_Per_Medal             : num [1:1142] NA 0.827 0.78 NA 0.503 ...
 $ Lit_Performance_Ratio                : num [1:1142] NA NA 2.62 NA 3.2 ...
 $ SE.TER.GRAD.SC.ZS                    : num [1:1142] NA NA NA NA NA ...
 $ IT.NET.USER.P2                       : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ SL.TLF.ADVN.ZS                       : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ UIS.X.US.FSGOV                       : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ UIS.X.USCONST.FSGOV                  : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ NY.GDP.PCAP.CD                       : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ SE.XPD.TOTL.GB.ZS                    : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ SE.XPD.CUR.TOTL.ZS                   : num [1:1142] NA NA NA NA NA NA NA NA NA NA ...
 $ OECD.TSAL.1.E10                      : num [1:1142] NA NA NA NA NA ...
```

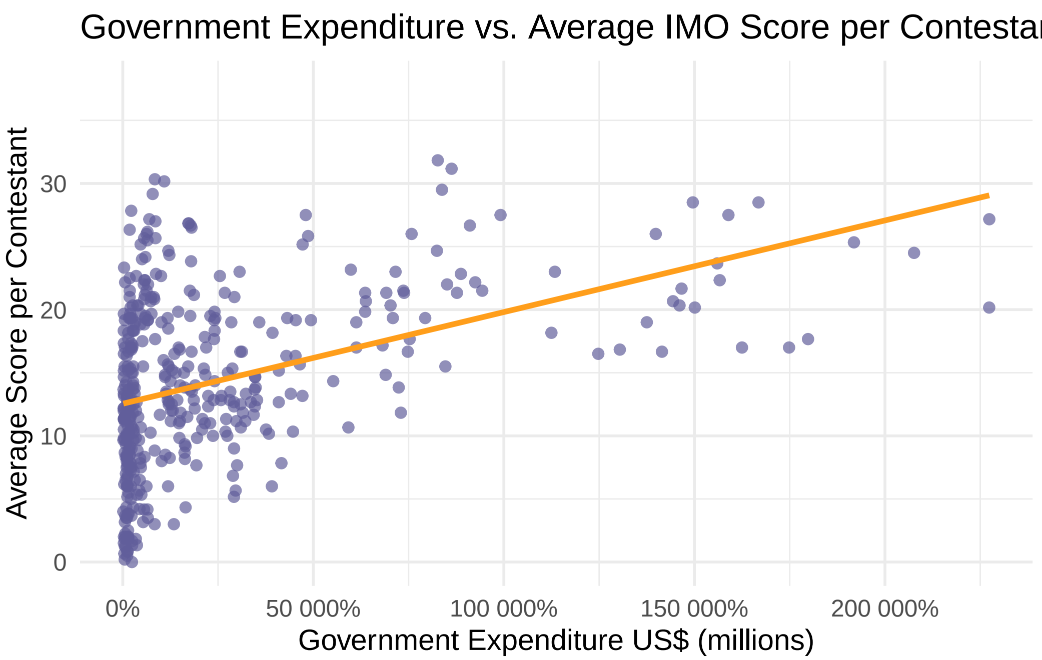
```r
set.seed(1234)
educ_data_split <- initial_split(education_numeric_data, prop = 3/4, strata = Value_gov_exper
train_data <- training(educ_data_split)
test_data <- testing(educ_data_split)

edu_recipe <- recipe(average_score_per_contestant ~ ., data = train_data) |>
  step_nzv(all_predictors()) |>   # Remove near-zero variance predictors
  step_impute_mean(all_numeric(), -all_outcomes()) |> # Impute missing values for numeric pre
 step_impute_mode(all_nominal()) |>
  step_unknown(all_nominal(), -all_outcomes()) |>
 step_normalize(all_numeric_predictors())            # Normalize numeric predictors
```

2.8 Data Visualization

2.8.1.Scatter Plot: Government Expenditure vs. Average Score Per Contestant

This plot shows the relationship between government spending on education (US$ (millions)) and the average score achieved by a country's team in the IMO.

```r
# Scatter plot of government expenditure vs. average score per contestant
ggplot(train_data, aes(x = UIS.X.US.FSGOV, y = average_score_per_contestant)) +
  geom_point(alpha = 0.7, color = "#615e9b") +
  geom_smooth(method = "lm", color = "#ff9e1b", se = FALSE) +
  ggtitle("Government Expenditure vs. Average IMO Score per Contestant") +
  xlab("Government Expenditure US$ (millions)") +
  ylab("Average Score per Contestant") +
  scale_x_continuous(labels = label_percent(suffix = "%", scale = 1)) +
  theme_minimal()
```



Interpretation:There is a positive correlation between government expenditure and the average IMO score per contestant. As government expenditure increases, the average IMO score tends to increase, as indicated by the orange regression line

2.8.2.Line Plot: Medal Counts of the Top 3 Countries in 2019 Over the Period 2009–2019

This line plot displays the total number of medals won by the top 3 countries from 2009 to 2019, selected based on their medal counts in 2019. Each line represents a country and tracks its medal achievements over time. The colors of the lines correspond to different countries, and the labels for each country are positioned next to the last point (2019) for easy identification.

This visualization allows us to observe the trend and consistency of each country's performance in terms of medal counts over the 10-year period.

```
library(ggrepel)
#| message: false
#| warning: false

# 1) Selecting the top 3 countries by the number of medals in 2019
top_countries_2019 <- combined_data |>
  filter(year == 2019) |>
  group_by(country) |>
  summarize(total_medals_2019 = sum(awards_gold + awards_silver + awards_bronze, na.rm = TRUE
  arrange(desc(total_medals_2019)) |>
  slice_head(n = 3) |>
  pull(country)

# 2) Filtering data for the selected countries from 2009 to 2019
medal_data <- combined_data |>
  filter(country %in% top_countries_2019, year >= 2009, year <= 2019) |>
  group_by(year, country) |>
  summarize(total_medals = sum(awards_gold + awards_silver + awards_bronze, na.rm = TRUE)) |>
  ungroup()
```

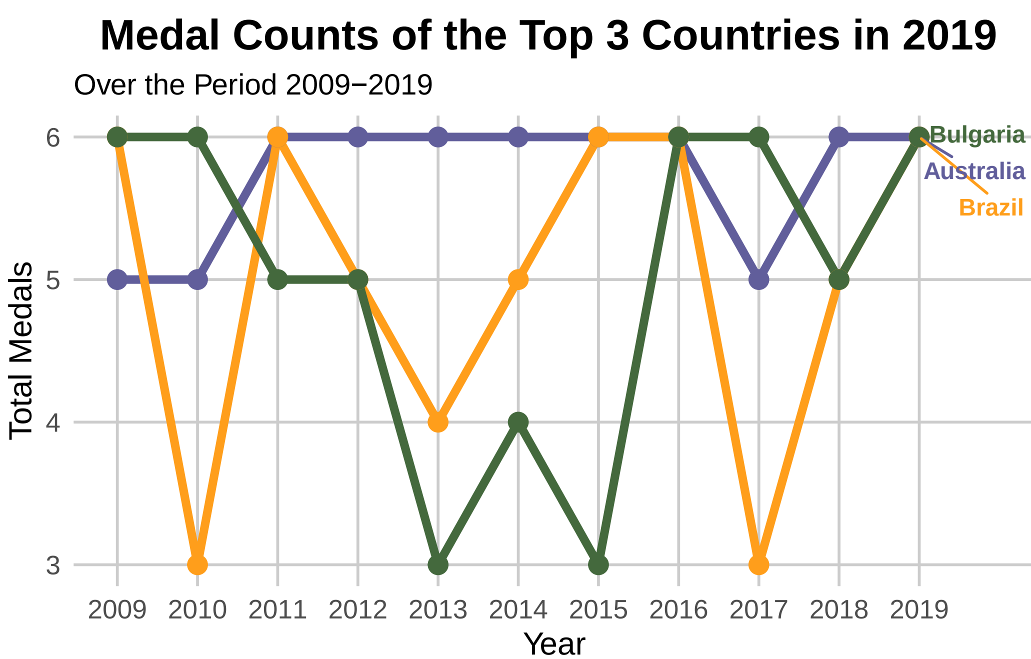
```
`summarise()` has grouped output by 'year'. You can override using the
`.groups` argument.
```

```
# Set colors for each country
country_colors <- setNames(c("#615e9b", "#ff9e1b", "#44693d"), top_countries_2019)

# Plot the graph
ggplot(medal_data, aes(x = year, y = total_medals, color = country, group = country)) +
  geom_line(size = 1.5) + # Line for each country
  geom_point(size = 3) + # Points on the lines
  scale_color_manual(values = country_colors) +
  scale_x_continuous(breaks = seq(2009, 2019, by = 1), labels = as.character(seq(2009, 2019,
  labs(
    title = "Medal Counts of the Top 3 Countries in 2019 ",
    subtitle = "Over the Period 2009-2019",
    x = "Year",
    y = "Total Medals",
    color = "Country"
  ) +
```

```r
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    panel.grid.major = element_line(color = "gray80", size = 0.5),
    panel.grid.minor = element_blank(),
    legend.position = "none" # Remove legend for cleaner design
  ) +
# Add annotations for each country next to the latest values
  geom_text_repel(data = medal_data %>% filter(year == 2019),
                  aes(label = country),
                  nudge_x = 0.9, # Slightly shift text to the right
                  direction = "y", # Repel text in the y direction
                  size = 3, fontface = "bold", color = country_colors)
```

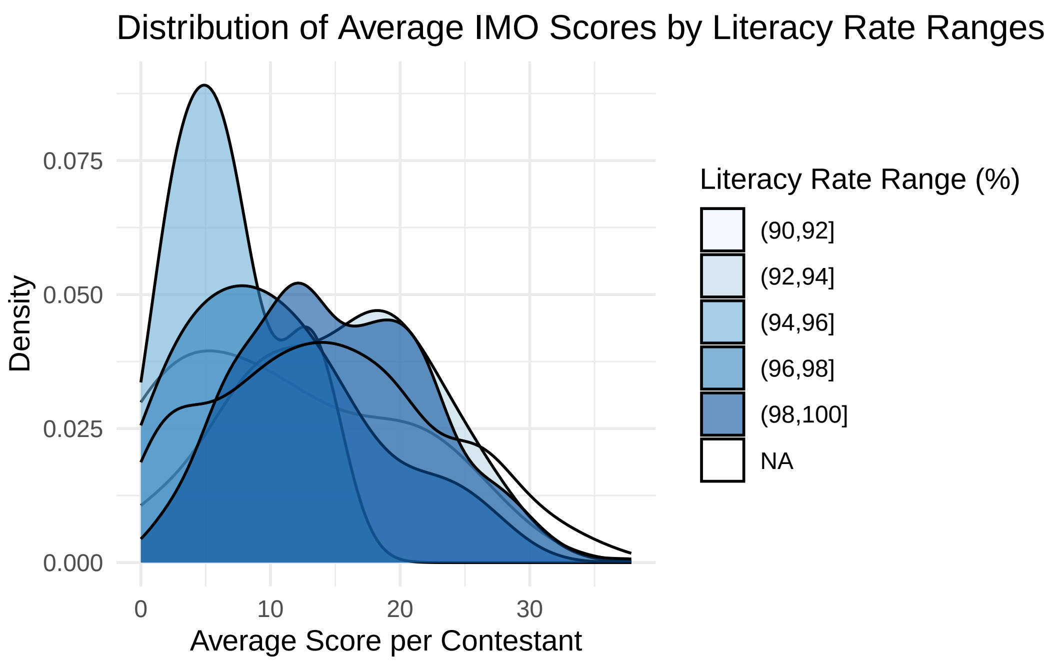
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
```



2.8.3. Density plot: Distribution of Average IMO Scores by Literacy Rate Ranges

```r
# Density plot showing the distribution of average scores by literacy rate
ggplot(train_data, aes(x = average_score_per_contestant, fill = cut(Value_literacy_rate, brea
  geom_density(alpha = 0.6) +
  scale_fill_brewer(palette = "Blues", name = "Literacy Rate Range (%)") +
  labs(
    title = "Distribution of Average IMO Scores by Literacy Rate Ranges",
    x = "Average Score per Contestant",
    y = "Density"
  ) +
  theme_minimal()
```

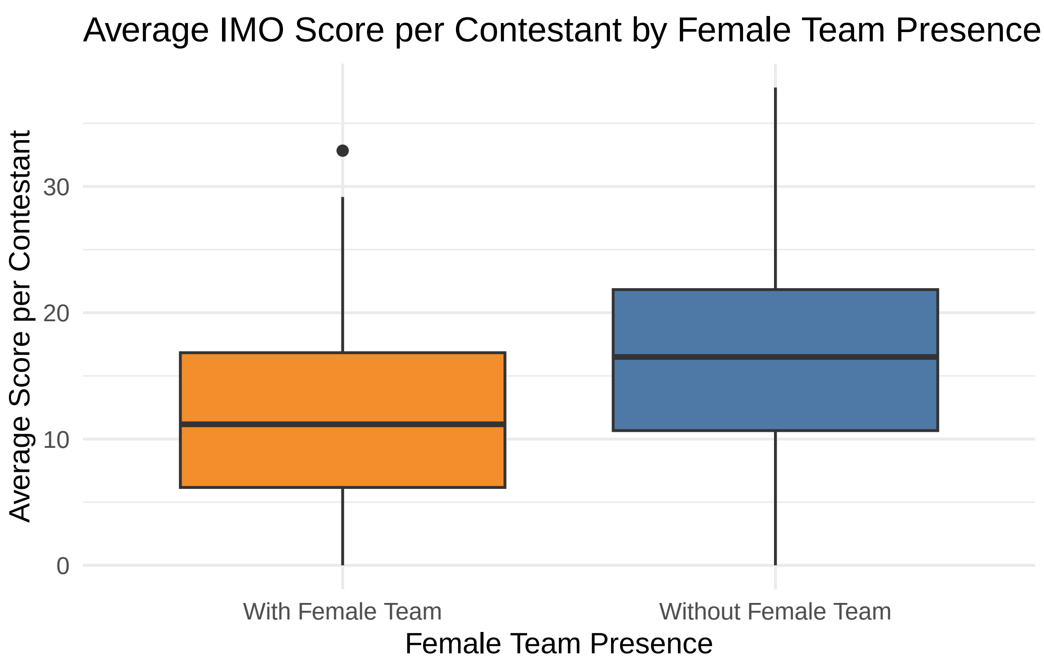


Interpretation:

The plot suggests that literacy rate does not strongly impact the distribution of average IMO scores per contestant. Countries with both lower and higher literacy rates show similar distributions of average scores, implying that literacy rate alone does not significantly influence IMO performance.

2.8.4. Boxplot: Compares the average IMO scores between countries with and without female team members:

```
train_data_fem <- train_data |>
  mutate(
    team_size_female = ifelse(is.na(team_size_female), 0, team_size_female), # Treat NA as 0
    has_female_team = ifelse(team_size_female > 0, "With Female Team", "Without Female Team")
  )

ggplot(train_data_fem, aes(x = has_female_team, y = average_score_per_contestant, fill = has_
  geom_boxplot() +
  labs(
    title = "Average IMO Score per Contestant by Female Team Presence",
    x = "Female Team Presence",
    y = "Average Score per Contestant"
  ) +
  scale_fill_manual(values = c("With Female Team" = "#F28E2B", "Without Female Team" = "#4E79
  theme_minimal() +
  theme(legend.position = "none")
```



The plot suggests a slight association between the absence of female team members and higher average IMO scores, although the difference is not very large.

3. Evaluation Metric

RMSE and R-squared were used as evaluation metrics due to the regression nature of the task.

4.Fit Models

4.1 Data Preprocessing

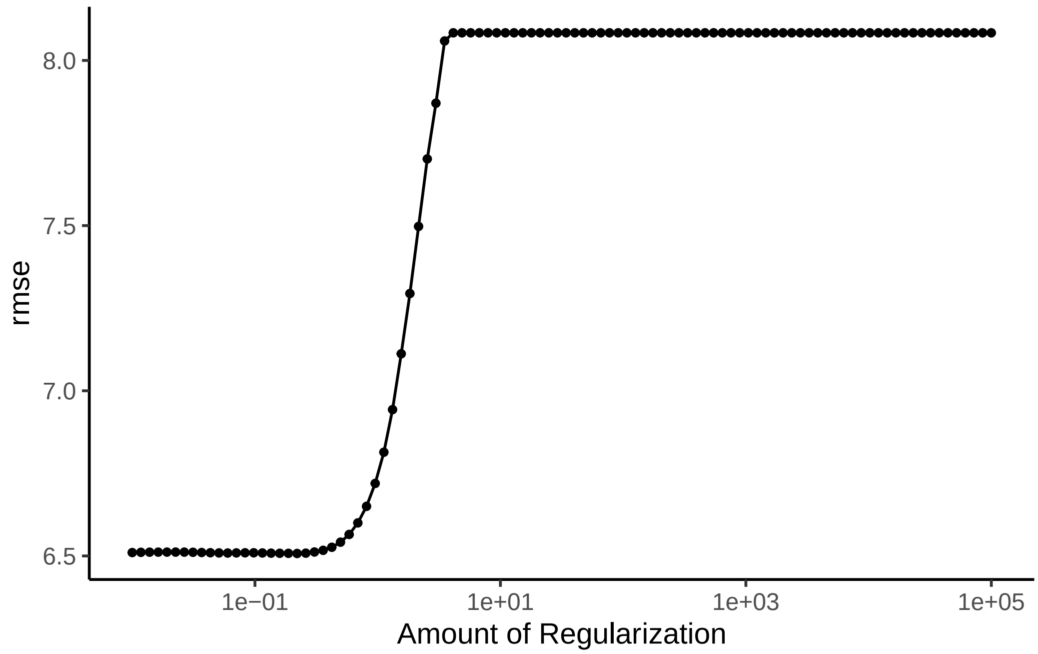
step_nzv(all_predictors()) to remove near-zero variance predictors step_impute_mean(all_numeric(), -all_outcomes()) to impute missing values for numeric predictors step_impute_mode(all_nominal()) to impute missing categorical values step_normalize(all_numeric_predictors()) to ormalize numeric predictors

```
library(tidymodels)
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
```

```
library(dplyr)


lasso_spec_tune <- linear_reg() |>
  set_engine("glmnet") |>
  set_args(mixture = 1, penalty = tune()) |>
  set_mode("regression")


lasso_recipe <- recipe(average_score_per_contestant ~ ., data = train_data) |>
  step_nzv(all_predictors()) |>
  step_impute_mean(all_numeric(), -all_outcomes()) |>
```

```
  step_normalize(all_numeric_predictors())


lasso_wf <- workflow() |>
  add_recipe(lasso_recipe) |>
  add_model(lasso_spec_tune)


penalty_grid <- grid_regular(
  penalty(range = c(-2, 5)),
  levels = 100
)


data_cv5 <- vfold_cv(train_data, v = 5)


tune_output <- tune_grid(
  lasso_wf,
  resamples = data_cv5,
  metrics = metric_set(yardstick::rmse),
  grid = penalty_grid
)


autoplot(tune_output) + theme_classic()
```




```r
collect_metrics(tune_output) |>
  filter(.metric == "rmse") |>
  select(penalty, mean_rmse = mean)
```

```
# A tibble: 100 x 2
   penalty mean_rmse
     <dbl>     <dbl>
 1  0.01        6.51
 2  0.0118      6.51
 3  0.0138      6.51
 4  0.0163      6.51
 5  0.0192      6.51
 6  0.0226      6.51
 7  0.0266      6.51
 8  0.0313      6.51
 9  0.0368      6.51
10  0.0433      6.51
# i 90 more rows
```

```r
best_pen_lasso <- select_best(tune_output, metric = "rmse")


lasso_final_fit <- lasso_wf |>
```

```r
  finalize_workflow(best_pen_lasso) |>
  fit(data = train_data)


lasso_coefs <- coef(
  lasso_final_fit |>
    extract_fit_engine(),
  s = best_pen_lasso$penalty
)


tibble(
  Predictor = rownames(lasso_coefs),
  Coefficient = as.vector(lasso_coefs)
)
```

```
# A tibble: 19 x 2
   Predictor                                    Coefficient
   <chr>                                              <dbl>
 1 (Intercept)                                      14.5
 2 year                                             -0.0938
 3 team_size_all                                     0
 4 team_size_male                                    2.61
 5 team_size_female                                 -0.190
 6 Value_gross_enr_ratio_for_tertiary_edu            0.847
 7 Value_gov_expen_as_perc_of_GPP                    0
 8 Value_literacy_rate                               0.374
 9 Gov_Investment_Per_Medal                         -2.00
10 Lit_Performance_Ratio                            -1.32
11 SE.TER.GRAD.SC.ZS                                 0
12 IT.NET.USER.P2                                    0
13 SL.TLF.ADVN.ZS                                    0
14 UIS.X.US.FSGOV                                    0.990
15 UIS.X.USCONST.FSGOV                               0
16 NY.GDP.PCAP.CD                                    0.111
17 SE.XPD.TOTL.GB.ZS                                 0
18 SE.XPD.CUR.TOTL.ZS                                0.0113
19 OECD.TSAL.1.E10                                   0
```

4.2 Choose Hyperparameters; Fit and Test Models

4.2.1 Linear Regression

```r
library(tidymodels)
library(Metrics)
library(dplyr)

# Create a linear regression model specification
lm_spec <- linear_reg() |>
  set_engine("lm")

# Create a workflow to combine preprocessing and modeling
lm_workflow <- workflow() |>
  add_recipe(edu_recipe) |>
  add_model(lm_spec)

# Fit the model to the training data
lm_fit <- fit(lm_workflow, data = train_data)

# View model summary using the new function
summary_model <- extract_fit_parsnip(lm_fit) |>
  tidy()
print(summary_model)
```

```
# A tibble: 19 x 5
   term                               estimate std.error statistic  p.value
   <chr>                                 <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                          14.5       0.221    65.8   0
 2 year                                 -0.292     0.244    -1.19  2.33e- 1
 3 team_size_all                         0.250     0.385     0.650 5.16e- 1
 4 team_size_male                        2.62      0.389     6.73  3.06e-11
 5 team_size_female                     -0.389     0.233    -1.67  9.56e- 2
 6 Value_gross_enr_ratio_for_tertirary_edu  0.992     0.268     3.70  2.27e- 4
 7 Value_gov_expen_as_perc_of_GPP       -0.266     0.274    -0.973 3.31e- 1
 8 Value_literacy_rate                   0.485     0.279     1.74  8.29e- 2
 9 Gov_Investment_Per_Medal             -2.15      0.257    -8.38  2.29e-16
10 Lit_Performance_Ratio                -1.44      0.244    -5.91  5.07e- 9
11 SE.TER.GRAD.SC.ZS                    -0.176     0.233    -0.757 4.50e- 1
12 IT.NET.USER.P2                       -0.103     0.361    -0.286 7.75e- 1
13 SL.TLF.ADVN.ZS                       -0.290     0.234    -1.24  2.14e- 1
14 UIS.X.US.FSGOV                        3.28      1.56      2.10  3.58e- 2
```

```
15 UIS.X.USCONST.FSGOV                         -2.14      1.57     -1.37  1.72e- 1
16 NY.GDP.PCAP.CD                               0.511     0.355     1.44  1.51e- 1
17 SE.XPD.TOTL.GB.ZS                            0.368     0.243     1.52  1.30e- 1
18 SE.XPD.CUR.TOTL.ZS                           0.310     0.228     1.36  1.75e- 1
19 OECD.TSAL.1.E10                             -0.0988    0.250    -0.395 6.93e- 1
```

```r
# Preprocess and predict on the test data
y_pred <- predict(lm_fit, new_data = test_data) |>
  pull(.pred)

# Replace negative predictions with a small positive value to avoid NaNs in log
y_pred <- ifelse(y_pred < 0, 1e-6, y_pred)

# Evaluate performance metrics
mse_train <- mean((train_data$average_score_per_contestant - predict(lm_fit, new_data = trai
                     pull(.pred))^2)
r2_train <- caret::R2(predict(lm_fit, new_data = train_data) |> pull(.pred), train_data$avera

mse_test <- mean((test_data$average_score_per_contestant - y_pred)^2)
r2_test <- caret::R2(y_pred, test_data$average_score_per_contestant)

# Calculate additional error metrics
msle_test <- msle(test_data$average_score_per_contestant, y_pred)
rmsle_test <- sqrt(msle_test)

# Print results
cat("Training MSE:", mse_train, "\n")
```

```
Training MSE: 40.69752
```

```r
cat("Training R-squared:", r2_train, "\n")
```

```
Training R-squared: 0.3768742
```

```r
cat("Test MSE:", mse_test, "\n")
```

```
Test MSE: 41.71663
```

```r
cat("Test R-squared:", r2_test, "\n")
```

```
Test R-squared: 0.3984548
```

```r
cat("Mean Squared Log Error (MSLE):", msle_test, "\n")
```

```
Mean Squared Log Error (MSLE): 0.399063
```

```r
cat("Root Mean Squared Log Error (RMSLE):", rmsle_test, "\n")
```

```
Root Mean Squared Log Error (RMSLE): 0.6317143
```

4.2.2 Gradient Boosting

`trees (500 to 3000, step 500)`: This range was chosen to balance computational efficiency with predictive accuracy. Smaller numbers of trees (e.g., 500) allow for faster training and provide a baseline for performance, while larger numbers (up to 3000) enable the model to capture more complex patterns in the data, covering a broad range to explore optimal tree count.

`tree_depth (1 to 5)`: Tree depth controls the complexity of each decision tree. A shallow depth (e.g., 1) promotes simpler and faster models, reducing the risk of overfitting, while deeper trees (up to 5) allow for capturing more intricate patterns in the data, providing a balanced exploration of model complexity.

`learn_rate (0.01, 0.05, 0.1)`: The learning rate determines how quickly the model adjusts during training. A smaller rate (e.g., 0.01) ensures careful and incremental adjustments, minimizing the risk of overshooting optimal solutions, while a larger rate (e.g., 0.1) speeds up training, with values chosen to balance accuracy and convergence speed.

`iter = 100`: The number of iterations (100) ensures a comprehensive exploration of the parameter space, allowing the model to evaluate a wide range of potential combinations and converge on the most effective hyperparameters.

```r
boost_edu <- boost_tree(mode = "regression",
                          engine = "lightgbm",
                          # B
                          trees = tune(),
                          # d
                          tree_depth = tune(),
                          # lambda
```

```r
                        learn_rate = tune())


boost_wf <- workflow() |>
  add_recipe(edu_recipe) |>
  add_model(boost_edu)
```

````
```{r cv-bayes-r}
#| eval: false

folds <- vfold_cv(train_data,
               v = 6)

boost_grid <- crossing(
 trees =  seq(500, 3000, by = 500),
 tree_depth = 1:5,
 learn_rate = c(0.01, 0.05, 0.1)
)

boost_cv_edu <- tune_grid(boost_wf,
                      resamples = folds,
                      grid = boost_grid,
                      metrics = metric_set(yardstick::rmse)
                      )

boost_params <- extract_parameter_set_dials(boost_wf)

boost_params <- boost_params |>
  update(trees = trees(range = c(1000, 3000)))

set.seed(756)
boost_cv_bayes_edu <- boost_wf |>
  tune_bayes(
    resamples = folds,
    param_info = boost_params,
    initial = boost_cv_edu,
    iter = 50,
    metrics = metric_set(yardstick::rmse),
    control = control_bayes(no_improve = 15)
  )

save(boost_cv_bayes_edu, file = "data/boost_cv_bayes_edu.RData")
````
````

```
```

```
load(file = "data/boost_cv_bayes_edu.RData")
```

```
collect_metrics(boost_cv_bayes_edu) |>
  arrange(desc(mean))
```

```
# A tibble: 131 x 10
   trees tree_depth learn_rate .metric .estimator  mean     n std_err .config
   <dbl>      <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1   500          1       0.01 rmse    standard    5.82     6   0.127 Preproces~
 2  1000          1       0.01 rmse    standard    5.58     6   0.130 Preproces~
 3  1500          1       0.01 rmse    standard    5.49     6   0.129 Preproces~
 4  2000          1       0.01 rmse    standard    5.45     6   0.127 Preproces~
 5  2500          1       0.01 rmse    standard    5.43     6   0.126 Preproces~
 6   500          1       0.05 rmse    standard    5.43     6   0.125 Preproces~
 7  3000          1       0.01 rmse    standard    5.40     6   0.123 Preproces~
 8  1000          1       0.05 rmse    standard    5.35     6   0.116 Preproces~
 9   500          1       0.1  rmse    standard    5.35     6   0.117 Preproces~
10  1500          1       0.05 rmse    standard    5.34     6   0.110 Preproces~
# i 121 more rows
# i 1 more variable: .iter <int>
```

```
```{r eval-bayes-r}
#| eval: true

boost_wf_best_bayes <- boost_wf |>
  finalize_workflow(select_best(boost_cv_bayes_edu,
                                metric = "rmse")) |>
  fit(train_data)

edu_aug_bayes <- boost_wf_best_bayes |>
  augment(new_data = test_data)

# Calculate RMSE and R^2
boost_metrics <- metrics(
  edu_aug_bayes,
  truth = average_score_per_contestant,
  estimate = .pred
)
```
```

````
print(boost_metrics)  # Displays RMSE and R^2
```
````

```
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       4.71
2 rsq     standard       0.684
3 mae     standard       3.11
```

4.2.3 GAM

Smoothing Basis (`bs = "cr"`): Cubic regression splines were chosen because they efficiently model nonlinear patterns while reducing the risk of overfitting.

Maximum Degrees of Freedom (`k = 10`): This setting controls the complexity of the spline, ensuring the model stays simple and easy to interpret while capturing enough flexibility to fit the data well.

```
library(mgcv)

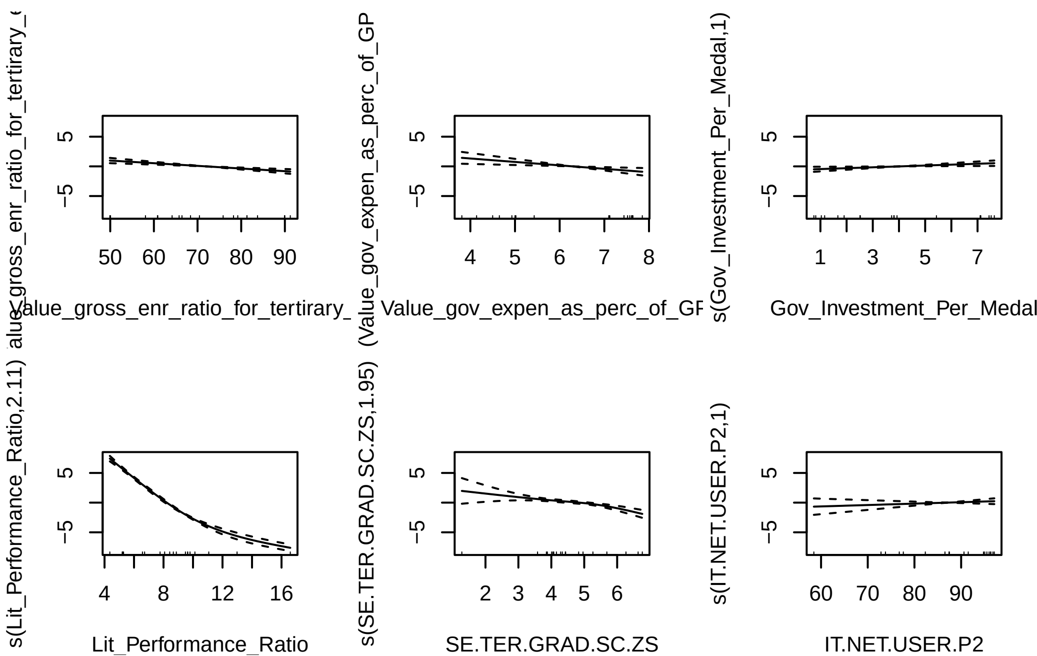
# Define GAM model formula
gam_formula <- average_score_per_contestant ~
  team_size_all+  team_size_male+ team_size_female+
  s(Value_gross_enr_ratio_for_tertiary_edu, k = 10, bs = "cr") +
  s(Value_gov_expen_as_perc_of_GPP, k = 10, bs = "cr") +
  Value_literacy_rate +
  s(Gov_Investment_Per_Medal, k = 10, bs = "cr") +
  s(Lit_Performance_Ratio, k = 10, bs = "cr") +
  s(SE.TER.GRAD.SC.ZS, k = 10, bs = "cr") +
  s(IT.NET.USER.P2, k = 10, bs = "cr") +
  s(SL.TLF.ADVN.ZS, k = 10, bs = "cr") +
  s(UIS.X.US.FSGOV, k = 12, bs = "cr") +
  s(UIS.X.USCONST.FSGOV , k = 10, bs = "cr") +
  s(NY.GDP.PCAP.CD, k = 10, bs = "cr") +
  s(SE.XPD.TOTL.GB.ZS, k = 10, bs = "cr") +
  s(SE.XPD.CUR.TOTL.ZS, k = 10, bs = "cr") +
  s(OECD.TSAL.1.E10, k = 10, bs = "cr")

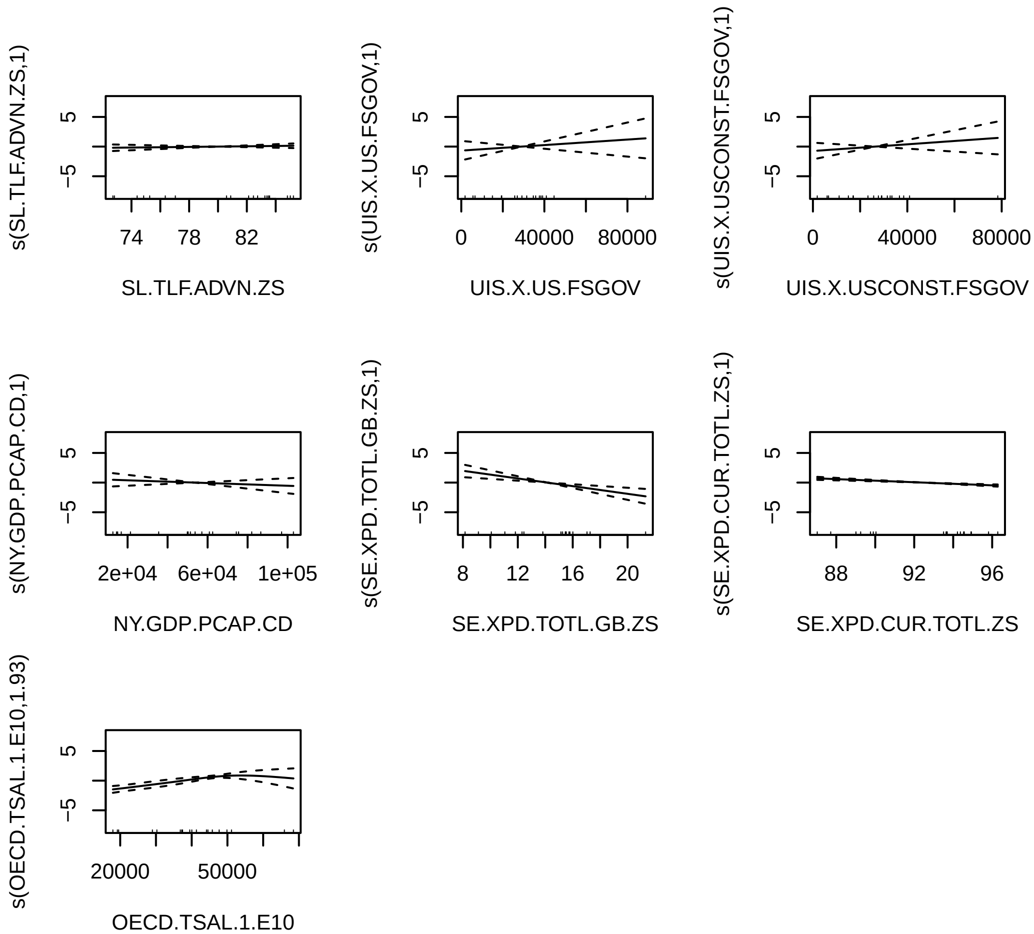
preproc_form <- average_score_per_contestant ~ .
```
```

```r
gam_mod <- gen_additive_mod()  |>
  set_engine("mgcv") |>
  set_mode("regression")

gam_pre <- recipe(preproc_form, data = train_data)

gam_wf <- workflow() |>
  add_recipe(gam_pre) |>
  add_model(gam_mod, formula = gam_formula)

gam_fit <- gam_wf |>
  fit(train_data)
```

Partial Dependency Plots

```r
par(mfrow = c(2, 3))
gam_fit_pd <- gam_fit |>
  extract_fit_engine() |>
  plot()
```






```
library(yardstick)
#| warning: false
#| message: false
mls_test_gam <- gam_fit |>
  augment(new_data = test_data)
actual <- mls_test_gam$average_score_per_contestant
predicted <- mls_test_gam$.pred
valid_data <- mls_test_gam[!is.na(actual) & !is.na(predicted), ]

rmsle_gam <- rmsle(actual = valid_data$average_score_per_contestant, predicted = valid_data$

rmsle_gam
```

```
[1] 0.1296193
```

```r
# Create a tibble with actual and predicted values
results <- tibble(
  truth = valid_data$average_score_per_contestant,
  estimate = valid_data$.pred
)
# Calculate R-squared
r2_gam_yardstick <- rsq(results, truth = truth, estimate = estimate)
# Print R-squared
cat("R-squared (R2) using yardstick:", r2_gam_yardstick$.estimate, "\n")
```

```
R-squared (R2) using yardstick: 0.7717786
```

5.Comparing Models

Overfitting vs. Underfitting

Linear Regression: Tends to underfit since it assumes simple linear relationships. The R-squared values (~0.37 and ~0.39) show it doesn't capture the complexity of the data well.

Gradient Boosting: Strikes a good balance by adjusting hyperparameters to avoid both overfitting and underfitting. It performs better, with a test R-squared of 0.684.

GAM: Handles nonlinear relationships best, giving the highest test R-squared (0.772). However, it risks overfitting if the smoothing parameters aren't tuned properly.

Bias vs. Variance

Linear Regression: Has high bias (makes simple assumptions) but low variance (predictions don't change much between datasets). This makes it consistent but not very accurate.

Gradient Boosting: Reduces bias by iteratively improving predictions and balances variance well with proper tuning.

GAM: Reduces bias by modeling complex patterns but can have higher variance depending on the smoothing settings.

Flexibility vs. Interpretability

Linear Regression: Very simple and easy to interpret, but lacks flexibility for capturing complex relationships.

Gradient Boosting: Flexible enough to model complex data but harder to interpret without extra tools like feature importance analysis.

GAM: Combines flexibility with decent interpretability, especially through partial dependency plots.

Key Takeaways

Gradient Boosting is a strong choice for balancing flexibility and accuracy, making it practical for predictions.

GAM is the most accurate and excels at modeling complex relationships, but it needs careful tuning to avoid overfitting.

Linear Regression is a good starting point for understanding basic relationships but doesn't handle complex data well.

Each model has its strengths depending on the need: simplicity (Linear Regression), flexibility and reliability (Gradient Boosting), or detailed nonlinear modeling (GAM).

6. Ethical Implications

The model could show bias if the data favors certain countries, like those with more government spending or higher literacy rates. This might lead to unfair decisions, like giving more resources to already advantaged countries.