Project Report

Saniya Bekova

2024-12-04

```r
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.2.0 --

v broom        1.0.6     v recipes      1.1.0
v dials        1.3.0     v rsample      1.2.1
v dplyr        1.1.4     v tibble       3.2.1
v ggplot2      3.5.1     v tidyr        1.3.1
v infer        1.0.7     v tune         1.2.1
v modeldata    1.4.0     v workflows    1.1.4
v parsnip      1.2.1     v workflowsets 1.1.0
v purrr        1.0.2     v yardstick    1.3.1

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x purrr::discard() masks scales::discard()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
x recipes::step()  masks stats::step()
* Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.5
v lubridate 1.9.3     v stringr   1.5.1
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x readr::col_factor() masks scales::col_factor()
x purrr::discard()    masks scales::discard()
x dplyr::filter()     masks stats::filter()
x stringr::fixed()    masks recipes::fixed()
x dplyr::lag()        masks stats::lag()
x readr::spec()       masks yardstick::spec()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becon
```

```r
library(bonsai)
library(themis)
```

```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")
```

```
Rows: 1142 Columns: 38
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (6): country, leader, deputy_leader, Country Code, Region, Income Group
dbl (31): 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.
```

```r
str(combined_educ_data)
```

```
spc_tbl_ [1,142 x 38] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ year                                    : num [1:1142] 2019 2019 2019 2019 2019 ...
 $ country                                 : chr [1:1142] "People's Republic of China" "Unitec
 $ 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 ...
 $ p1                                      : num [1:1142] 40 42 42 41 42 39 39 42 42 42 ...
 $ p2                                      : num [1:1142] 41 40 39 41 35 40 31 30 26 28 ...
 $ p3                                      : num [1:1142] 27 26 31 17 5 17 10 13 19 9 ...
 $ p4                                      : num [1:1142] 41 42 42 42 42 35 40 40 31 40 ...
 $ p5                                      : num [1:1142] 42 42 42 42 40 42 42 42 37 42 ...
```

```
$ p6                                    : num [1:1142] 36 35 30 4 21 6 15 7 16 7 ...
$ p7                                    : logi [1:1142] NA NA NA NA NA NA ...
$ awards_gold                           : num [1:1142] 6 6 6 3 3 2 2 2 3 1 ...
$ awards_silver                         : num [1:1142] 0 0 0 3 3 4 4 4 1 3 ...
$ awards_bronze                         : num [1:1142] 0 0 0 0 0 0 0 0 2 2 ...
$ awards_honorable_mentions             : num [1:1142] 0 0 0 0 0 0 0 0 0 0 ...
$ leader                                : chr [1:1142] "Bin Xiong" "Po-Shen Loh" "Yongjin S
$ deputy_leader                         : chr [1:1142] "Yijie He" "Yang Liu" "Suyoung Choi"
$ total_score                           : num [1:1142] 227 227 226 187 185 179 177 174 171
$ average_score_per_contestant          : num [1:1142] 37.8 37.8 37.7 31.2 30.8 ...
$ medal_Efficiency                      : num [1:1142] 1 1 1 1 1 1 1 1 1 1 ...
$ Country Code                          : chr [1:1142] NA NA NA NA ...
$ Value_gross_enr_ratio_for_tertirary_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 ...
$ Region                                : chr [1:1142] NA "North America" "East Asia & Paci
$ Income Group                          : chr [1:1142] NA "High income: OECD" "High income
$ 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 ...
- attr(*, "spec")=
 .. cols(
 ..   year = col_double(),
 ..   country = col_character(),
 ..   team_size_all = col_double(),
 ..   team_size_male = col_double(),
 ..   team_size_female = col_double(),
 ..   p1 = col_double(),
 ..   p2 = col_double(),
 ..   p3 = col_double(),
 ..   p4 = col_double(),
 ..   p5 = col_double(),
 ..   p6 = col_double(),
 ..   p7 = col_logical(),
 ..   awards_gold = col_double(),
```

```
  ..   awards_silver = col_double(),
  ..   awards_bronze = col_double(),
  ..   awards_honorable_mentions = col_double(),
  ..   leader = col_character(),
  ..   deputy_leader = col_character(),
  ..   total_score = col_double(),
  ..   average_score_per_contestant = col_double(),
  ..   medal_Efficiency = col_double(),
  ..   `Country Code` = col_character(),
  ..   Value_gross_enr_ratio_for_tertirary_edu = col_double(),
  ..   Value_gov_expen_as_perc_of_GPP = col_double(),
  ..   Value_literacy_rate = col_double(),
  ..   Region = col_character(),
  ..   `Income Group` = col_character(),
  ..   Gov_Investment_Per_Medal = col_double(),
  ..   Lit_Performance_Ratio = col_double(),
  ..   SE.TER.GRAD.SC.ZS = col_double(),
  ..   IT.NET.USER.P2 = col_double(),
  ..   SL.TLF.ADVN.ZS = col_double(),
  ..   UIS.X.US.FSGOV = col_double(),
  ..   UIS.X.USCONST.FSGOV = col_double(),
  ..   NY.GDP.PCAP.CD = col_double(),
  ..   SE.XPD.TOTL.GB.ZS = col_double(),
  ..   SE.XPD.CUR.TOTL.ZS = col_double(),
  ..   OECD.TSAL.1.E10 = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

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

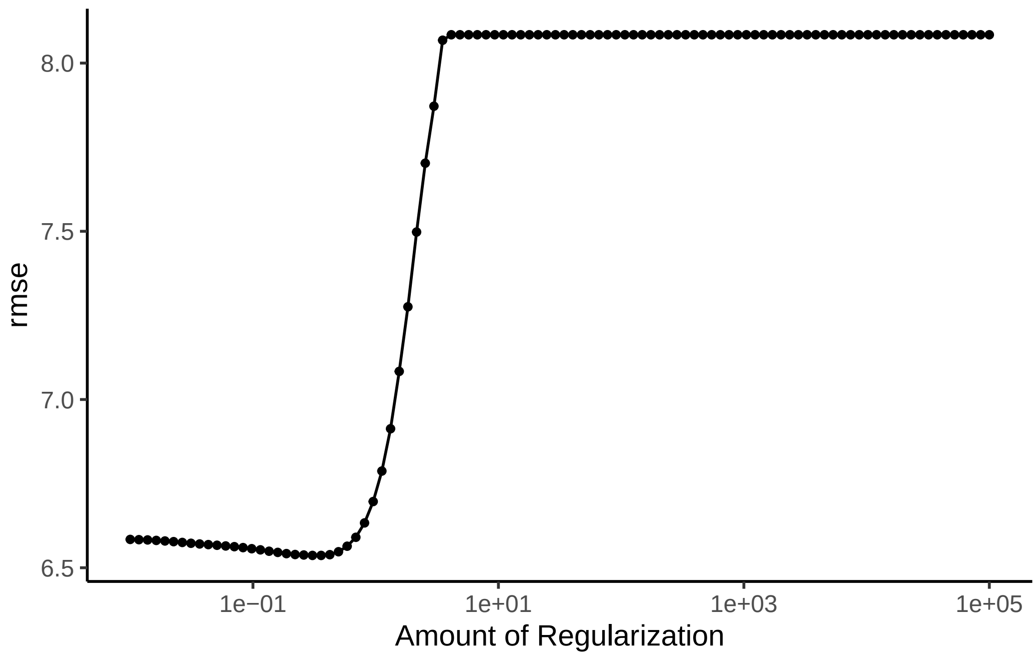
```r
set.seed(1234)
educ_data_split <- initial_split(education_numeric_data, prop = 3/4, strata = Value_gov_exper
train_data <- training(educ_data_split)
test_data <- testing(educ_data_split)

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
```

```r
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
```

```r
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
)
```

```r
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        6.58
 2  0.0118      6.58
```

```
 3  0.0138      6.58
 4  0.0163      6.58
 5  0.0192      6.58
 6  0.0226      6.58
 7  0.0266      6.58
 8  0.0313      6.57
 9  0.0368      6.57
10  0.0433      6.57
# i 90 more rows
```

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


lasso_final_fit <- lasso_wf |>
  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
 3 team_size_all                                 0
 4 team_size_male                                2.53
 5 team_size_female                             -0.0789
 6 Value_gross_enr_ratio_for_tertiary_edu        0.788
 7 Value_gov_expen_as_perc_of_GPP                0
 8 Value_literacy_rate                           0.334
 9 Gov_Investment_Per_Medal                     -1.87
```

```
10 Lit_Performance_Ratio                          -1.22
11 SE.TER.GRAD.SC.ZS                               0
12 IT.NET.USER.P2                                  0
13 SL.TLF.ADVN.ZS                                  0
14 UIS.X.US.FSGOV                                  0.919
15 UIS.X.USCONST.FSGOV                             0
16 NY.GDP.PCAP.CD                                  0.00193
17 SE.XPD.TOTL.GB.ZS                               0
18 SE.XPD.CUR.TOTL.ZS                              0
19 OECD.TSAL.1.E10                                 0
```

Linear Regression

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

```
Attaching package: 'Metrics'

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

    accuracy, mae, mape, mase, precision, recall, rmse, smape
```

```r
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_tertiary_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
```

```
# 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)
```

```r
# 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
```

Gradient Boosting

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

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

```r
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    5.81     6   0.203 Preproces~
 2  1000          1       0.01 rmse    standard    5.57     6   0.201 Preproces~
 3  1500          1       0.01 rmse    standard    5.47     6   0.193 Preproces~
 4  2000          1       0.01 rmse    standard    5.43     6   0.187 Preproces~
 5  2500          1       0.01 rmse    standard    5.40     6   0.182 Preproces~
 6   500          1       0.05 rmse    standard    5.40     6   0.181 Preproces~
 7  3000          1       0.01 rmse    standard    5.39     6   0.177 Preproces~
 8   500          1       0.1  rmse    standard    5.36     6   0.167 Preproces~
 9  1000          1       0.05 rmse    standard    5.36     6   0.166 Preproces~
10  1500          1       0.05 rmse    standard    5.34     6   0.161 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       4.72
2 rsq     standard       0.684
3 mae     standard       3.09
```
```