Elearning data analysis

2025-02-14

Load packages

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.1
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6     ✓ purrr   1.0.2
## ✓ tibble  3.1.8     ✓ dplyr   1.0.9
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.2.1
```

```
## Warning: package 'tibble' was built under R version 4.2.1
```

```
## Warning: package 'tidyr' was built under R version 4.2.1
```

```
## Warning: package 'readr' was built under R version 4.2.1
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.1
```

```
## Warning: package 'forcats' was built under R version 4.2.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
library(caret) #for classification and regression training
```

```
## Warning: package 'caret' was built under R version 4.2.1
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.2.1
```

```
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ranger) #for random forests
```

```
## Warning: package 'ranger' was built under R version 4.2.3
```

```r
library(e1071) #for statistics functions
```

```
## Warning: package 'e1071' was built under R version 4.2.1
```

```r
library(tidylog)
```

```
##
## Attaching package: 'tidylog'
##
## The following objects are masked from 'package:dplyr':
##
##     add_count, add_tally, anti_join, count, distinct, distinct_all,
##     distinct_at, distinct_if, filter, filter_all, filter_at, filter_if,
##     full_join, group_by, group_by_all, group_by_at, group_by_if,
##     inner_join, left_join, mutate, mutate_all, mutate_at, mutate_if,
##     relocate, rename, rename_all, rename_at, rename_if, rename_with,
##     right_join, sample_frac, sample_n, select, select_all, select_at,
##     select_if, semi_join, slice, slice_head, slice_max, slice_min,
##     slice_sample, slice_tail, summarise, summarise_all, summarise_at,
##     summarise_if, summarize, summarize_all, summarize_at, summarize_if,
##     tally, top_frac, top_n, transmute, transmute_all, transmute_at,
##     transmute_if, ungroup
##
## The following objects are masked from 'package:tidyr':
##
##     drop_na, fill, gather, pivot_longer, pivot_wider, replace_na,
##     spread, uncount
##
## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(dataedu) #package for educational data
```

Import and view data

```
setwd("D:/Data-analytics-project/Elearning-data-analysis")
df <- dataedu::sci_mo_with_text
glimpse(df)
```

```
## Rows: 606
## Columns: 74
## $ student_id            <dbl> 43146, 44638, 47448, 47979, 48797, 51943, 52326,…
## $ course_id             <chr> "FrScA-S216-02", "OcnA-S116-01", "FrScA-S216-01"…
## $ total_points_possible <dbl> 3280, 3531, 2870, 4562, 2207, 4208, 4325, 2086, …
## $ total_points_earned   <dbl> 2220, 2672, 1897, 3090, 1910, 3596, 2255, 1719, …
## $ percentage_earned     <dbl> 0.6768293, 0.7567261, 0.6609756, 0.6773345, 0.86…
## $ subject               <chr> "FrScA", "OcnA", "FrScA", "OcnA", "PhysA", "FrSc…
## $ semester              <chr> "S216", "S116", "S216", "S216", "S116", "S216", …
## $ section               <chr> "02", "01", "01", "01", "01", "03", "01", "01", …
## $ Gradebook_Item        <chr> "POINTS EARNED & TOTAL COURSE POINTS", "ATTEMPTE…
## $ Grade_Category        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …
## $ final_grade           <dbl> 93.45372, 81.70184, 88.48758, 81.85260, 84.00000…
## $ Points_Possible       <dbl> 5, 10, 10, 5, 438, 5, 10, 10, 443, 5, 12, 10, 5,…
## $ Points_Earned         <dbl> NA, 10.00, NA, 4.00, 399.00, NA, NA, 10.00, 425.…
## $ Gender                <chr> "M", "F", "M", "M", "F", "F", "M", "F", "F", "M"…
## $ q1                    <dbl> 5, 4, 5, 5, 4, NA, 5, 3, 4, NA, NA, 4, 3, 5, NA,…
## $ q2                    <dbl> 4, 4, 4, 5, 3, NA, 5, 3, 3, NA, NA, 5, 3, 3, NA,…
## $ q3                    <dbl> 4, 3, 4, 3, 3, NA, 3, 3, 3, NA, NA, 3, 3, 5, NA,…
## $ q4                    <dbl> 5, 4, 5, 5, 4, NA, 5, 3, 4, NA, NA, 5, 3, 5, NA,…
## $ q5                    <dbl> 5, 4, 5, 5, 4, NA, 5, 3, 4, NA, NA, 5, 4, 5, NA,…
## $ q6                    <dbl> 5, 4, 4, 5, 4, NA, 5, 4, 3, NA, NA, 5, 3, 5, NA,…
## $ q7                    <dbl> 5, 4, 4, 4, 4, NA, 4, 3, 3, NA, NA, 5, 3, 5, NA,…
## $ q8                    <dbl> 5, 5, 5, 5, 4, NA, 5, 3, 4, NA, NA, 4, 3, 5, NA,…
## $ q9                    <dbl> 4, 4, 3, 5, NA, NA, 5, 3, 2, NA, NA, 5, 2, 2, NA…
## $ q10                   <dbl> 5, 4, 5, 5, 3, NA, 5, 3, 5, NA, NA, 4, 4, 5, NA,…
## $ time_spent            <dbl> 1555.1667, 1382.7001, 860.4335, 1598.6166, 1481.…
## $ TimeSpent_hours       <dbl> 25.91944500, 23.04500167, 14.34055833, 26.643610…
## $ TimeSpent_std         <dbl> -0.18051496, -0.30780313, -0.69325954, -0.148446…
## $ int                   <dbl> 5.0, 4.2, 5.0, 5.0, 3.8, 4.6, 5.0, 3.0, 4.2, NA,…
## $ pc                    <dbl> 4.50, 3.50, 4.00, 3.50, 3.50, 4.00, 3.50, 3.00, …
## $ uv                    <dbl> 4.333333, 4.000000, 3.666667, 5.000000, 3.500000…
## $ enrollment_status     <chr> "Approved/Enrolled", "Approved/Enrolled", "Appro…
## $ enrollment_reason     <chr> "Course Unavailable at Local School", "Course Un…
## $ cogproc               <dbl> 15.069737, 7.106667, 15.165854, 14.508000, 16.69…
## $ male                  <dbl> 0.51210526, 0.00000000, 0.11121951, 0.00000000, …
## $ female                <dbl> 0.16657895, 0.00000000, 0.15219512, 0.00000000, …
## $ friend                <dbl> 0.00000000, 0.00000000, 0.01268293, 0.00000000, …
## $ family                <dbl> 0.006052632, 0.000000000, 0.084878049, 0.0000000…
## $ social                <dbl> 6.200526, 6.140000, 5.052927, 6.133000, 7.534000…
## $ sad                   <dbl> 0.18078947, 0.00000000, 0.09097561, 0.00000000, …
## $ anger                 <dbl> 0.41868421, 0.00000000, 0.14097561, 0.10800000, …
## $ anx                   <dbl> 0.080000000, 0.000000000, 0.275365854, 0.7880000…
## $ negemo                <dbl> 1.1363158, 0.0000000, 1.4187805, 1.1520000, 1.28…
## $ posemo                <dbl> 3.555526, 19.010000, 2.906098, 5.591000, 3.79400…
## $ affect                <dbl> 4.756053, 19.010000, 4.330732, 6.743000, 5.07500…
## $ quant                 <dbl> 2.046842, 2.743333, 3.245366, 3.214000, 2.551000…
## $ number                <dbl> 0.9131579, 3.4733333, 2.3065854, 0.2570000, 0.21…
## $ interrog              <dbl> 1.2857895, 0.4433333, 1.7868293, 1.1030000, 1.71…
## $ compare               <dbl> 2.4213158, 4.1466667, 3.9021951, 2.6990000, 3.94…
## $ adj                   <dbl> 5.106842, 5.480000, 5.614390, 5.213000, 4.618000…
## $ verb                  <dbl> 18.11368, 11.02333, 16.34366, 16.31100, 17.11700…
```

```
## $ negate             <dbl> 1.2060526, 0.0000000, 1.6809756, 1.1300000, 0.74…
## $ conj               <dbl> 5.565526, 6.660000, 5.370244, 6.203000, 7.244000…
## $ adverb             <dbl> 6.243421, 6.660000, 5.824878, 5.314000, 6.492000…
## $ auxverb            <dbl> 11.298421, 9.246667, 10.226341, 8.890000, 9.4940…
## $ prep               <dbl> 12.301579, 11.850000, 12.132927, 13.626000, 12.8…
## $ article            <dbl> 7.828947, 2.223333, 6.767805, 9.119000, 9.830000…
## $ ipron              <dbl> 6.936316, 2.743333, 5.145122, 4.335000, 7.841000…
## $ they               <dbl> 1.01026316, 0.00000000, 0.84341463, 1.86300000, …
## $ shehe              <dbl> 0.54342105, 0.00000000, 0.16951220, 0.00000000, …
## $ you                <dbl> 1.7442105, 3.4733333, 1.1487805, 2.0490000, 2.62…
## $ we                 <dbl> 0.06578947, 0.00000000, 0.03317073, 0.30200000, …
## $ i                  <dbl> 3.646579, 7.993333, 4.689268, 3.449000, 3.142000…
## $ ppron              <dbl> 7.010000, 11.470000, 6.882927, 7.662000, 6.77900…
## $ pronoun            <dbl> 13.98868, 14.20667, 12.02756, 12.21900, 14.61900…
## $ `function`         <dbl> 55.15447, 44.63000, 49.40293, 53.12700, 57.50900…
## $ Dic                <dbl> 86.27895, 86.31000, 80.72220, 86.49700, 90.48700…
## $ Sixltr             <dbl> 20.89316, 22.20333, 20.80780, 21.80200, 15.30600…
## $ WPS                <dbl> 17.413947, 9.833333, 17.922439, 18.824000, 15.66…
## $ Tone               <dbl> 56.62395, 96.38000, 49.41610, 78.36900, 55.38400…
## $ Authentic          <dbl> 44.13079, 70.25333, 41.22366, 49.03800, 42.25000…
## $ Clout              <dbl> 49.52079, 53.58333, 40.11024, 53.08800, 54.08500…
## $ Analytic           <dbl> 55.70316, 56.04000, 58.98098, 69.95700, 55.82000…
## $ WC                 <dbl> 88.31579, 34.66667, 69.34146, 61.20000, 47.10000…
## $ n                  <dbl> 38, 3, 41, 10, 10, 2, 21, 18, 31, 37, 37, 18, 12…
```

Data processing

Select important variables

```
df <-
  df %>%
  select(
    int, #student think this course is interesting
    uv, #utility value: what I'm learning in this course is relevant to my life
    pc, #perceived competence: this topic is one of my best subjects
    time_spent, #time spent in the course
    final_grade,
    subject,
    enrollment_reason,
    semester,
    enrollment_status,
    cogproc, #student's cognitive processing
    social, #social-related discourse in discussion board posts
    posemo, #positive emotions in discussion board posts
    negemo, #negative emotions in discussion board posts
    n #number of discussion board posts in the course in the semester
  )
```

```
## select: dropped 60 variables (student_id, course_id, total_points_possible,
## total_points_earned, percentage_earned, …)
```

Analysis

Analyze data

```
# Check number of rows in dataset
nrow(df)
```

```
## [1] 606
```

```
# Drop rows with missing data (N/A)
df<-na.omit(df)
# Check number of rows after dropping N/A
nrow(df)
```

```
## [1] 464
```

```
glimpse(df)
```

```
## Rows: 464
## Columns: 14
## $ int               <dbl> 5.0, 4.2, 5.0, 5.0, 3.8, 5.0, 3.0, 4.2, 4.4, 3.4, 4.…
## $ uv                <dbl> 4.333333, 4.000000, 3.666667, 5.000000, 3.500000, 5.…
## $ pc                <dbl> 4.50, 3.50, 4.00, 3.50, 3.50, 3.50, 3.00, 3.00, 4.00…
## $ time_spent        <dbl> 1555.1667, 1382.7001, 860.4335, 1598.6166, 1481.8000…
## $ final_grade       <dbl> 93.45372, 81.70184, 88.48758, 81.85260, 84.00000, 83…
## $ subject           <chr> "FrScA", "OcnA", "FrScA", "OcnA", "PhysA", "AnPhA", …
## $ enrollment_reason <chr> "Course Unavailable at Local School", "Course Unavai…
## $ semester          <chr> "S216", "S116", "S216", "S216", "S116", "S216", "S11…
## $ enrollment_status <chr> "Approved/Enrolled", "Approved/Enrolled", "Approved/…
## $ cogproc           <dbl> 15.069737, 7.106667, 15.165854, 14.508000, 16.692000…
## $ social            <dbl> 6.200526, 6.140000, 5.052927, 6.133000, 7.534000, 7.…
## $ posemo            <dbl> 3.555526, 19.010000, 2.906098, 5.591000, 3.794000, 5…
## $ negemo            <dbl> 1.1363158, 0.0000000, 1.4187805, 1.1520000, 1.282000…
## $ n                 <dbl> 38, 3, 41, 10, 10, 21, 18, 31, 18, 12, 3, 16, 7, 42,…
```

```
# Determine if there are variables with no variability
nearZeroVar(df, saveMetrics=TRUE)
```

```
##                   freqRatio percentUnique zeroVar   nzv
## int                1.314815     9.0517241   FALSE FALSE
## uv                 1.533333     6.4655172   FALSE FALSE
## pc                 1.488372     3.8793103   FALSE FALSE
## time_spent         1.000000   100.0000000   FALSE FALSE
## final_grade        1.333333    93.1034483   FALSE FALSE
## subject            1.648649     1.0775862   FALSE FALSE
## enrollment_reason  3.154762     1.0775862   FALSE FALSE
## semester           1.226601     0.6465517   FALSE FALSE
## enrollment_status  0.000000     0.2155172    TRUE  TRUE
## cogproc            1.000000    96.9827586   FALSE FALSE
## social             1.500000    96.1206897   FALSE FALSE
## posemo             1.000000    96.7672414   FALSE FALSE
## negemo            13.000000    90.7327586   FALSE FALSE
## n                  1.333333    10.1293103   FALSE FALSE
```

zeroVar column of enrollment_status is True, so we will remove it. Variables with no variability may cause problems in some models.

```
df <-
  df %>%
  select(-enrollment_status)
```

```
## select: dropped one variable (enrollment_status)
```

```
# Convert string categorical variables to factors
df <-
  df %>%
  mutate_if(is.character, as.factor)
```

```
## mutate_if: converted 'subject' from character to factor (0 new NA)
##            converted 'enrollment_reason' from character to factor (0 new NA)
##            converted 'semester' from character to factor (0 new NA)
```

Prepare train and test sets

```
# Set seed
set.seed(2025)
# Train 70%, Test 30%
# Create train set
trainIdx <- createDataPartition(df$final_grade,
                                p=.7,
                                list=FALSE,
                                times=1)

# Add new variable to dataset temporarily
# Select rows according to their row number
df <-
  df %>%
  mutate(temp_id = 1:464)
```

```
## mutate: new variable 'temp_id' (integer) with 464 unique values and 0% NA
```

```
# Filter dataset to get only rows indicated trainIdx vector
df_train <-
  df %>%
  filter(temp_id %in% trainIdx)
```

```
## filter: removed 136 rows (29%), 328 rows remaining
```

```
# Filter out to get test set
df_test <-
  df %>%
  filter(!temp_id %in% trainIdx)
```

```
## filter: removed 328 rows (71%), 136 rows remaining
```

```
# Delete temp_id from the original data
df <-
  df %>%
  select(-temp_id)
```

```
## select: dropped one variable (temp_id)
```

```
df_train <-
  df_train %>%
  select(-temp_id)
```

```
## select: dropped one variable (temp_id)
```

```
df_test <-
  df_test %>%
  select(-temp_id)
```

```
## select: dropped one variable (temp_id)
```

Estimate the model

Random forests - bootstrap resampling

```
# Set seed
set.seed(2025)
# Run random forest model
# Final grade is y, other variables are x
# Syntax: final_grade ~. => y ~ all other variables except y
# Resampling method: Bootstrap resampling
rf_fit <- train(final_grade ~.,
                data = df_train,
                method = "ranger")
rf_fit
```

```
## Random Forest
##
## 328 samples
##  12 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 328, 328, 328, 328, 328, 328, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   RMSE      Rsquared   MAE
##    2    variance    16.07891  0.4814206  12.00116
##    2    extratrees  17.37091  0.4558964  12.58493
##   10    variance    14.81596  0.5150913  10.98432
##   10    extratrees  14.37030  0.5813905  10.70215
##   19    variance    14.88024  0.5093763  10.89464
##   19    extratrees  13.78887  0.5957179  10.30089
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 19, splitrule = extratrees
##  and min.node.size = 5.
```

```
# Results: Best RMSE: 13.79, Best R^2: 0.6
```

Random forests - Cross validation

```
#Set seed
set.seed(2025)
# Use cross validation
train_control <-
  trainControl(method="repeatedcv",
               number = 10, #10 folds
               repeats = 10) #repeat 10 times

rf_fit1 <-
  train(final_grade ~ .,
        data=df_train,
        method="ranger",
        trControl=train_control)

rf_fit1
```

```
## Random Forest
##
## 328 samples
##  12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 294, 296, 294, 296, 295, 294, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   RMSE      Rsquared   MAE
##    2    variance    15.48641  0.5306322  11.72206
##    2    extratrees  17.03196  0.5068189  12.51838
##   10    variance    13.90578  0.5720739  10.43246
##   10    extratrees  13.79550  0.6149065  10.45750
##   19    variance    13.80944  0.5732509  10.24774
##   19    extratrees  13.20623  0.6280336  10.01536
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 19, splitrule = extratrees
##  and min.node.size = 5.
```

```
# Result: Best RMSE: 13.2, Best R^2: 0.63
```

Tuning random forest model

Previously: min.node.size is fixed to 5 Now: change min.node.size and mtry

```r
set.seed(2025)

# Create a grid of different values of mtry, split rules and min node sizes to test
tune_grid <-
  expand.grid(
    mtry = c(2, 3, 7, 10, 19),
    splitrule = c("variance", "extratrees"),
    min.node.size=c(1, 5, 10, 15, 20)
  )

# Fit a new model using tuning grid
rf_fit2 <-
  train(final_grade~.,
        data=df_train,
        method="ranger",
        tuneGrid=tune_grid)

rf_fit2
```

```
## Random Forest
##
## 328 samples
##  12 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 328, 328, 328, 328, 328, 328, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   min.node.size  RMSE      Rsquared   MAE
##    2    variance     1             15.99371  0.4861212  11.92869
##    2    variance     5             16.09348  0.4811132  12.01389
##    2    variance    10             16.18863  0.4746928  12.10158
##    2    variance    15             16.35733  0.4644302  12.23053
##    2    variance    20             16.46616  0.4581469  12.32386
##    2    extratrees   1             17.25220  0.4577243  12.47415
##    2    extratrees   5             17.37693  0.4552814  12.58400
##    2    extratrees  10             17.60935  0.4414375  12.76655
##    2    extratrees  15             17.78587  0.4314077  12.91368
##    2    extratrees  20             17.98658  0.4163722  13.07164
##    3    variance     1             15.53336  0.4947565  11.61643
##    3    variance     5             15.57180  0.4947766  11.64949
##    3    variance    10             15.69477  0.4856603  11.78690
##    3    variance    15             15.81486  0.4802026  11.89772
##    3    variance    20             15.99589  0.4696509  12.04563
##    3    extratrees   1             16.18111  0.5086335  11.73106
##    3    extratrees   5             16.35841  0.5011881  11.88576
##    3    extratrees  10             16.64786  0.4857649  12.12168
##    3    extratrees  15             16.90480  0.4740047  12.29219
##    3    extratrees  20             17.08276  0.4681437  12.43796
##    7    variance     1             14.91422  0.5138677  11.12181
##    7    variance     5             14.97616  0.5101048  11.16496
##    7    variance    10             15.02833  0.5065927  11.23460
##    7    variance    15             15.11661  0.5022217  11.34332
##    7    variance    20             15.20459  0.4968687  11.43802
##    7    extratrees   1             14.82990  0.5616703  10.94610
##    7    extratrees   5             14.87008  0.5631145  11.01215
##    7    extratrees  10             15.08067  0.5566656  11.17367
##    7    extratrees  15             15.33269  0.5447313  11.34670
##    7    extratrees  20             15.54238  0.5378536  11.49999
##   10    variance     1             14.77988  0.5181068  10.96699
##   10    variance     5             14.80325  0.5158682  10.99447
##   10    variance    10             14.81722  0.5147974  11.04178
##   10    variance    15             14.86410  0.5126755  11.09651
##   10    variance    20             14.93172  0.5092059  11.19363
##   10    extratrees   1             14.36338  0.5781267  10.67644
##   10    extratrees   5             14.39487  0.5797341  10.72813
##   10    extratrees  10             14.57807  0.5735542  10.86363
##   10    extratrees  15             14.78028  0.5663489  11.01265
##   10    extratrees  20             14.94316  0.5593999  11.14028
##   19    variance     1             14.88535  0.5095078  10.89231
```

```
##   19    variance     5            14.87368  0.5100980  10.89955
##   19    variance    10            14.88364  0.5086409  10.93401
##   19    variance    15            14.83982  0.5107337  10.91608
##   19    variance    20            14.79437  0.5138788  10.92149
##   19    extratrees   1            13.81893  0.5930555  10.32942
##   19    extratrees   5            13.82810  0.5940922  10.32892
##   19    extratrees  10            13.90730  0.5912578  10.40617
##   19    extratrees  15            13.99420  0.5891199  10.49275
##   19    extratrees  20            14.12019  0.5860253  10.60280
## 
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 19, splitrule = extratrees
##  and min.node.size = 1.
```

```
# Result: Best MRSE: 13.82, Best R^2: 0.59
```

```
# See details of final model output of rf_fit2
rf_fit2$finalModel
```

```
## Ranger result
## 
## Call:
##  ranger::ranger(dependent.variable.name = ".outcome", data = x,      mtry = min(param$mtry, n
col(x)), min.node.size = param$min.node.size,      splitrule = as.character(param$splitrule), wr
ite.forest = TRUE,      probability = classProbs, ...) 
## 
## Type:                             Regression 
## Number of trees:                  500 
## Sample size:                      328 
## Number of independent variables:  19 
## Mtry:                             19 
## Target node size:                 1 
## Variable importance mode:         none 
## Splitrule:                        extratrees 
## Number of random splits:          1 
## OOB prediction error (MSE):       174.9095 
## R squared (OOB):                  0.6188466
```

Examine predictive accuracy on test

```
set.seed(2025)
# Create new testing data including predicted values
df_test_augmented <-
  df_test %>%
  mutate(pred=predict(rf_fit2, df_test),
         obs=final_grade)
```

```
## mutate: new variable 'pred' (double) with 136 unique values and 0% NA
##         new variable 'obs' (double) with 133 unique values and 0% NA
```

```
# Transform object to data frame
defaultSummary(as.data.frame(df_test_augmented))
```

```
##       RMSE   Rsquared        MAE
## 12.0703944  0.6792062  9.0596247
```

- RMSE on test set = 12.07 => better than RMSE 13.82 of train set.
- R^2 on test set = 0.68 => better than R^2 0.59 of train set
- Therefore, the model performs well on unseen data (Test data)

Results

Variable Importance

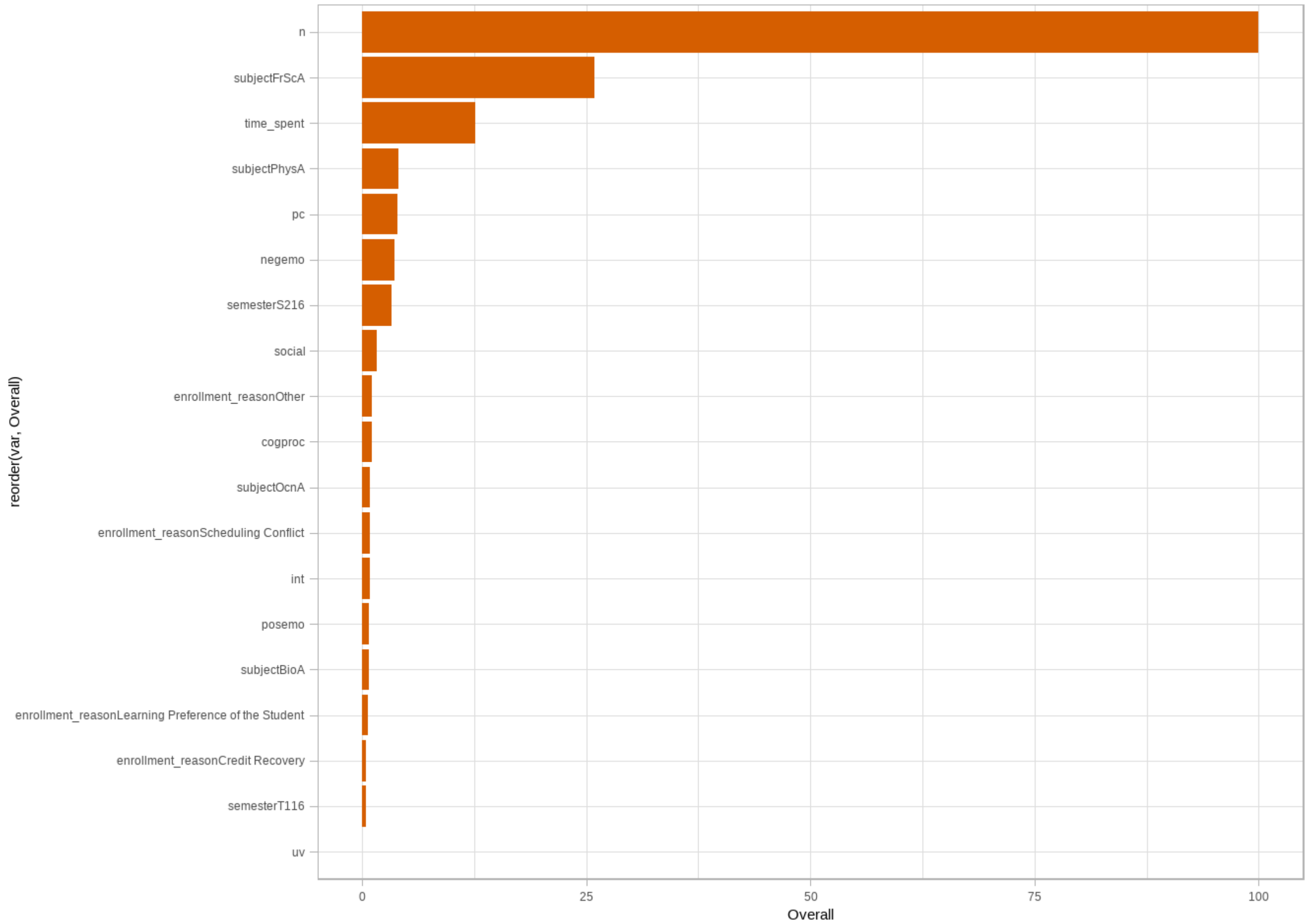
```
set.seed(2025)
# Learn which variables contribute most strongly to the model prediction
rf_fit2_imp <-
  train(
    final_grade ~.,
    data=df_train,
    method="ranger",
    tuneGrid=tune_grid,
    importance="permutation"
  )

#Extract variable importance from the new model
varImp(rf_fit2_imp)
```

```
## ranger variable importance
##
##                                                     Overall
## n                                                  100.0000
## subjectFrScA                                        25.9573
## time_spent                                          12.6614
## subjectPhysA                                         4.0058
## pc                                                   4.0008
## negemo                                               3.6652
## semesterS216                                         3.2900
## social                                               1.6859
## enrollment_reasonOther                               1.1195
## cogproc                                              1.0923
## subjectOcnA                                          0.9272
## enrollment_reasonScheduling Conflict                 0.9037
## int                                                  0.8653
## posemo                                               0.7464
## subjectBioA                                          0.7180
## enrollment_reasonLearning Preference of the Student  0.5989
## enrollment_reasonCredit Recovery                     0.4383
## semesterT116                                         0.3946
## uv                                                   0.0000
```

Visualize variable importance

```
varImp(rf_fit2_imp) %>%
    pluck(1) %>%
    rownames_to_column("var") %>%
    ggplot(aes(x = reorder(var, Overall), y = Overall)) +
    geom_col(fill = "#D55E00") +
    coord_flip() +
    theme_light()
```



Insights * Most important: n: number of student’s discussion posts * 2nd most: subject Forensic Science: enrolled in Forensic Science course has great impact on student’s * 3rd most: timespent: time student spent in the course

Compare random forest to regression

```
# Convert character variables to factors
df_train_lm <-
  df_train %>%
  mutate_if(is.character, as.factor)
```

```
## mutate_if: no changes
```

```
# Create a linear regression model
lm_fit <-
  train(final_grade~.,
    data=df_train_lm,
    method="lm")
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
# Append predicted values to train set for linear model
df_train_lm <-
  df_train %>%
  mutate(obs=final_grade,
         pred=predict(lm_fit, df_train_lm))
```

```
## mutate: new variable 'obs' (double) with 312 unique values and 0% NA
##         new variable 'pred' (double) with 328 unique values and 0% NA
```

```r
#Append predicted values to train set for random forest
df_train_rf <-
  df_train %>%
  mutate(pred=predict(rf_fit2, df_train),
         obs=final_grade)
```

```
## mutate: new variable 'pred' (double) with 328 unique values and 0% NA
##         new variable 'obs' (double) with 312 unique values and 0% NA
```

```r
# Summarize linear model
defaultSummary(as.data.frame(df_train_lm))
```

```
##       RMSE   Rsquared        MAE
## 15.0208404  0.5068252 11.2760167
```

```r
# Summarize random forest
defaultSummary(as.data.frame(df_train_rf))
```

```
##      RMSE  Rsquared       MAE
## 4.8600844 0.9663748 3.6447460
```

Conclusions

- Random forest has higher R squared (0.97 compared to 0.51 for the regression model)
- Random forest has lower RMSE, meaning random forest fits the data better than linear model.