# predicting_ed_outcomes

Kevin McGowan

2024-11-30

## Abstract

This Predicting_ed_outcomes project report, is part of the HarvardX: PH125.9x Data Science: Capstone course, and builds two machine learning algorithm to predict 3-levels of college outcomes (dropout, enrolled, graduate). These models include decision trees and gradient boosted model with cross validation. Both models have the potential to direct early intervention decisions by highlighting who would benefit from such programs and factors are most predictive of student outcomes. This report and its associated R script are designed to run using local directories. To ensure proper functionality, the entire GitHub repository should be downloaded as a zip file. If any issues are encountered, the README file provides additional guidance. The analysis adopts a step-by-step methodology to build and validate two machine learning models, presented through the following outline:

- **Introduction**
- **Methodology**
  - Exploratory Data Analysis
  - Feature Development
  - Modeling
  - Visualization
- **Results**
  - Testing on Independent Final Holdout Set
- **Discussion**

## Introduction

This report processes the "Predict Students' Dropout and Academic Success" dataset, developed by Martins et al. (2021). It includes data cleaning, exploratory data analysis, feature engineering, and utilizes machine learning models to predict student outcomes: dropout, enrolled, and graduated.

College is not only an expensive endeavor but as US Department of Health and Human Services reports, one that is also predictive of long-term health outcomes and income (2023). As such, being able to predict student dropout before it happens could enable early intervention programs that support students toward success. With universities often having thousands of students and in some cases only a few counselors, a predictive model like the one developed here could be immensely helpful in giving counselors a prioritized starting point for their efforts.

This report aims to support counselors, administrators, and teachers by identifying which students might need the most help. Ultimately, the goal is to ensure better outcomes for students by leveraging predictive insights.

### The Dataset

This dataset, created by Martins et al. (2021), originates from a single Portuguese university. It captures three main categories of information available at the beginning of a student's academic journey:

1. **Academic Performance**: Information about students' grades and course credits
2. **Demographics**: Age, gender, and other personal characteristics.
3. **Socio-Economic Factors**: Data related to students' economic and social background.

The dataset includes students from various degrees of study, including agronomy, design, education, nursing, journalism, management, social service, and technologies. These diverse academic paths provide a rich context for analyzing predictors of student success.

The dataset consists of: **- 4424 students - 37 features**, representing factors potentially associated with academic outcomes.

# Method/Analysis

Before training the education outcome models out of the Martins and colleagues (2021) dataset, the data needs to be prepared for analysis. The steps in doing so are as follows:

- Load Data: download, unzip the data
- Clean Data: clean the column names and types of 'variable_table' and 'data'
- Explore Data: variables, dimensions, and insights
- Feature Development: leveraging descriptive statistics of students and their descriptors
- Model Development: Decision Tree & Gradiant Boosted Tree w/ CV
- Prediction: Making predictions on an independent validation set.

### Load Libraries

```r
# List of required packages
packages <- c("readr", "gbm", "tinytex", "rpart.plot", "dplyr", "caret", "rpart", "stringr", "tidyverse

# Check and install missing packages
for (pkg in packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    install.packages(pkg)
  }
}

# Load the libraries
invisible(lapply(packages, library, character.only = TRUE))
```

### Load Data & Variable Table from Repo Zip File

This report, and it's associated R script, are set up to run with local directories. As such, the zip file should be downloaded as described in the README file. If done so, the following will run.

```r
# Define the GitHub repository URL and destination path
repo_url <- "https://github.com/KevinWMcGowan/predicting_ed_outcomes/archive/refs/heads/main.zip"
zip_file <- here("data", "predicting_ed_outcomes-main.zip")
unzip_dir <- here("data", "predicting_ed_outcomes-main")

# Check if the repository is already downloaded
if (!file.exists(zip_file)) {
  cat("Downloading the repository...\n")
  download.file(repo_url, zip_file, mode = "wb")
}

# Check if the repository is already unzipped
```

```r
if (!dir.exists(unzip_dir)) {
  cat("Unzipping the repository...\n")
  unzip(zip_file, exdir = here("data"))
}
```

With the repository downloaded and unzipped, the following code loads the two data tables into the environment: the numerically coded data.csv and the definitions in the variable_table:

```r
# Define the paths to the data folder within the unzipped repository
data_folder <- file.path(unzip_dir, "data")  # Path to the "data" folder
data_path <- file.path(data_folder, "data.csv")
variable_table_path <- file.path(data_folder, "variable_table.csv")

# Check if the "data" folder exists
if (!dir.exists(data_folder)) {
  stop("The 'data' folder is not found in the unzipped repository.
       Please check the repository structure.")
}

# Check if data.csv exist in the "data" folder
if (!file.exists(data_path)) {
  stop("The file 'data.csv' is not found in the 'data' folder.
       Please ensure it exists in the repository structure.")
}

# Check if variable_table.csv exists in teh "data" folder
if (!file.exists(variable_table_path)) {
  stop("The file 'variable_table.csv' is not found in the 'data' folder.
       Please ensure it exists in the repository structure.")
}

# Load the datasets
data <- read.csv(data_path, header = TRUE, sep = ";")
variable_table <- read_csv(variable_table_path)
```

```
## Rows: 37 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): Variable Name, Role, Type, Demographic, Description, Missing Values
## lgl (1): Units
##
## 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
cat("Data and variable table loaded successfully from the 'data' folder.\n")
```

```
## Data and variable table loaded successfully from the 'data' folder.
```

### Clean the datasets

With the data now downloaded and loaded into the R Studio environment, data cleaning can commence. As seen below, the column names in data table and the variable and the variable_name column include spaces and parenthesis. All of which will cause issues with analysis later on.

```r
cat("Column names in `data` before cleaning:\n")
colnames(data)
```

```
cat("\nColumn names in `variable_table` before cleaning:\n")
colnames(variable_table)
cat("\nVariable names in `variable_table` before cleaning:\n")
print(variable_table$`Variable Name`)
```

# remove print column codes below if results = hide hides the output but not the code

The following regex code removes these problematic characters.

```
clean_names <- function(names) {
  names %>%
    tolower() %>%
    gsub("[ .\\(\\)/'\\-]", "_", .) %>%
    gsub("_+", "_", .) %>%
    gsub("^_|_$", "", .) %>%
    trimws()
}

# Apply the cleaning function to `data` and `variable_table` names
colnames(data) <- clean_names(colnames(data))
colnames(variable_table) <- clean_names(colnames(variable_table))

# Print cleaned column names
cat("\nColumn names in `data` after cleaning:\n")
print(colnames(data))
cat("\nColumn names in `variable_table` after cleaning:\n")
print(colnames(variable_table))
variable_table$variable_name <- clean_names(variable_table$variable_name)
```

Now that the problematic characters have been removed, ensure all variable names are consistent:

```
matching_variables <- intersect(colnames(data), variable_table$variable_name)
non_matching_in_data <- setdiff(colnames(data), variable_table$variable_name)
non_matching_in_variable_table <- setdiff(variable_table$variable_name, colnames(data))

# Display the matching and non-matching variables
cat("\nMatching Variables:\n")
```

```
##
## Matching Variables:
```

```
print(matching_variables)
```

```
##  [1] "marital_status"
##  [2] "application_mode"
##  [3] "application_order"
##  [4] "course"
##  [5] "daytime_evening_attendance"
##  [6] "previous_qualification"
##  [7] "previous_qualification_grade"
##  [8] "nacionality"
##  [9] "mother_s_qualification"
## [10] "father_s_qualification"
## [11] "mother_s_occupation"
## [12] "father_s_occupation"
## [13] "admission_grade"
## [14] "displaced"
## [15] "educational_special_needs"
## [16] "debtor"
## [17] "tuition_fees_up_to_date"
## [18] "gender"
## [19] "scholarship_holder"
## [20] "age_at_enrollment"
## [21] "international"
## [22] "curricular_units_1st_sem_credited"
## [23] "curricular_units_1st_sem_enrolled"
## [24] "curricular_units_1st_sem_evaluations"
## [25] "curricular_units_1st_sem_approved"
## [26] "curricular_units_1st_sem_grade"
## [27] "curricular_units_1st_sem_without_evaluations"
## [28] "curricular_units_2nd_sem_credited"
## [29] "curricular_units_2nd_sem_enrolled"
## [30] "curricular_units_2nd_sem_evaluations"
## [31] "curricular_units_2nd_sem_approved"
## [32] "curricular_units_2nd_sem_grade"
## [33] "curricular_units_2nd_sem_without_evaluations"
## [34] "unemployment_rate"
## [35] "inflation_rate"
## [36] "gdp"
## [37] "target"
```

The code above shows that all 37 variable names havre printed, and therefore are matching.

### Encode Target as Numeric

The last cleaning step is to encode Target numerically:

```
# Encode 'target' variable with a numeric representation
data$target <- ifelse(data$target == "Dropout", 1,
                      ifelse(data$target == "Enrolled", 2,
                             ifelse(data$target == "Graduate", 3, NA)))
sort(unique(data$target))
```

```
## [1] 1 2 3
```

The target variable is now numeric: - “Dropout” -> 1 - “Enrolled” -> 2 - “Graduate” -> 3

Next, the dataset is split to avoid over training and other validation challenges with model training.

## Split the dataset & Justify

Now before any analysis or exploration can be done, the best practice is to split the dataset in order to avoid over training. A quick inspection of the dataset shows a relatively small dataset for model training (4424 students) and a strong class imbalance (Dropout = 1421, Enrolled = 794, Graduated = 2209). This imbalance and small sample size will inform the splitting of the data for training and testing, and the modeling techniques used in the method section of this report.

```
nrow(data)
```

```
## [1] 4424
```

```
table(data$target)
```

```
##
##    1    2    3
## 1421  794 2209
```

A 20% holdout set is reserved before final testing, ensuring the dataset's class distribution is maintained, sufficient representation for all three outcome classes are retained, and 80% of the data can be used for training and validation.

```
set.seed(123)
trainindex <- createDataPartition(data$target, p = .8,
                                  list = FALSE,
                                  times = 1)
traindata <- data[trainindex,]
final_holdout_set  <- data[-trainindex,]
```

From here on, this report will only work with the traindata set. This will prevent any patterns and observations about the holdout set from informing the training of the learning models. In this way, the final holdout set will be an independent test of model effectiveness.

## Exploratory Data Analysis

Now that the data has been cleaned, some exploration can be done to better understand the students in the traindata by inspecting the variables that describe them. Any findings can contribute to the modeling done later.

```
# See number of students in dataset
nrow(traindata)
```

```
## [1] 3540
```

```
ncol(traindata)
```

```
## [1] 37
```

```
unique(traindata$target)
```

```
## [1] 1 3 2
```

```
table(traindata$target)
```

```
##
##    1    2    3
## 1125  647 1768
```

```
head(traindata)
```

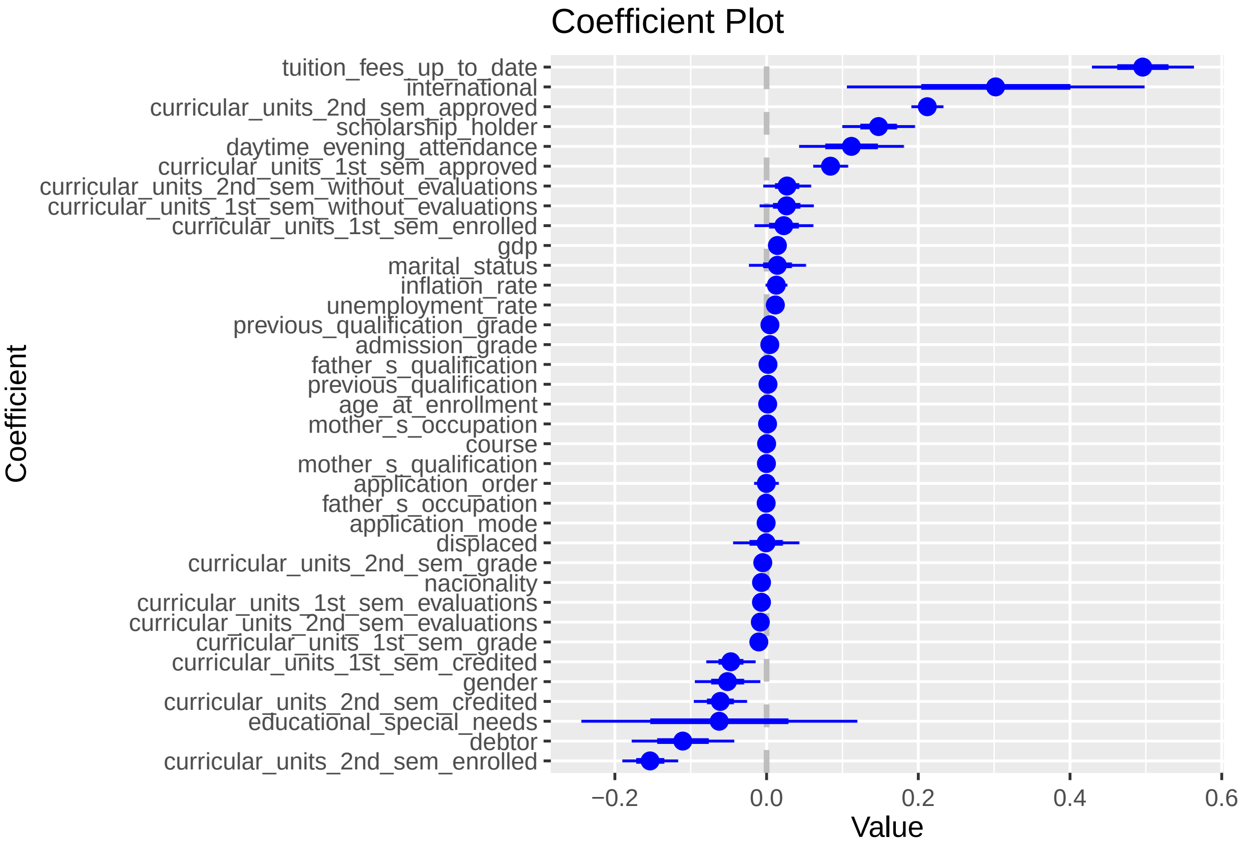
With the data now split and only using `traindata`, the code above shows the data now contains 3540 students, 36 variables to help predict target (37th), #3 levels of outcomes to predict, 1137 have dropped out, 636 are currently enrolled, and 1768 have graduated. The relatively small number of currently enrolled students will likely lead to difficulty in modeling due to class imbalance. Lastly, all values are coded with numbers.

In order to have a functional and targeted exploration, the following section performs a linear regression to get an idea of what variables are predictive of dropout, enrolled, and graduated. Afterwards, these predictive variables will be explored more thoroughly.

### Linear Regression

To better understand the predictive variables in the dataset, this regression will visualize how certain variables might be predictive of target as a whole.

```r
# Extract variable names for the formula
variables_train <- colnames(traindata)

# Remove the 'target' variable from the list of predictors
variables_train <- variables_train[variables_train != "target"]

# Create the formula as a string
formula_string_train <- paste("target ~", paste(variables_train, collapse = " + "), - 1)

# Convert the string to a formula object
formula_train <- as.formula(formula_string_train)

# Perform linear regression
value1 <- lm(formula_train, data = traindata)

# Visualize the coefficients
regression_plot<-coefplot(value1, sort = 'magnitude', conf.int = TRUE)
regression_plot
```



The coefficient plot above provides insights into the variables' relationships with the target outcomes and their relative strengths. It highlights variables with significant predictive power, as well as those with wide error margins crossing zero, suggesting limited or no effect.

**Key Regression Findings:**

- **Strong Predictors (Positive Association):** Due to the positive coefficients, the plot suggests focusing on variables such as: `tuition_fees_up_to_date`, `international`, `curricular_units_2nd_sem_approved`, `scholarship_holder`, `daytime_evening_attendance`, and `curricular_units_1st_sem_approved` These range approximately from 0.1 to 0.5, indicating a positive contribution toward favorable outcomes like graduation or enrollment.

Conversely, certain factors show negative correlations with the target, as indicated by their negative coefficients. These include:`gender`, `curricular_units_1st_sem_credited`, `curricular_units_2nd_sem_credited`, `educational_special_needs`, `debtor`, and `curricular_units_2nd_sem_enrolled`. These insights suggest these variables are just as important, because they may be associated with lower values of target, such as dropout and enrolled.

Together, these findings provide a foundation for deeper exploration into how these variables influence the target outcomes and how they can be utilized effectively in predictive modeling.

## Investigate Predictive Variables

Based on the regression above, the following exploration targets variables identified above as predictive.

```
binary_vars <- c("tuition_fees_up_to_date", "gender", "scholarship_holder",
                 "debtor", "international", "educational_special_needs")
categorical_vars <- c("marital_status", "application_mode", "daytime_evening_attendance", "nacionality"
continuous_vars <- c("admission_grade", "curricular_units_1st_sem_grade", "gdp")
```

## Binary variables

Since all variables are encoded numerically, the following code will tell us what the binary values mean.

```
variable_table %>%
  slice(c(14:19, 21)) %>%  # Select rows 14 to 19 and 21
  select(variable_name, description) %>%  # Select specific columns
  print()
```

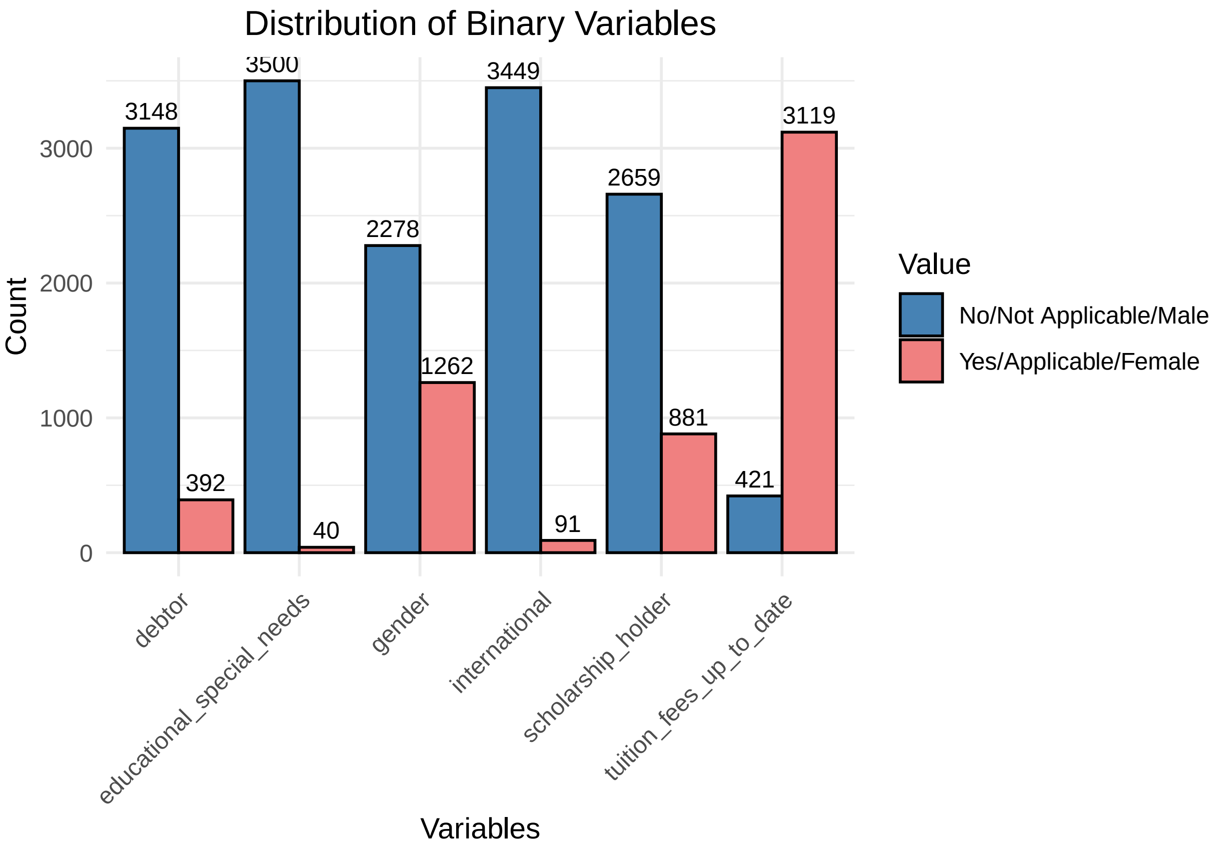
```
## # A tibble: 7 x 2
##   variable_name             description
##   <chr>                     <chr>
## 1 displaced                 1 - yes 0 - no
## 2 educational_special_needs 1 - yes 0 - no
## 3 debtor                    1 - yes 0 - no
## 4 tuition_fees_up_to_date   1 - yes 0 - no
## 5 gender                    1 - male 0 - female
## 6 scholarship_holder        1 - yes 0 - no
## 7 international             1 - yes 0 - no
```

The above shows 1 = yes/female & 0 = no/male for binary values.

Now plot the students across these variables.

```
# Plot distribution of binary data
binary_data <- traindata %>%
  select(all_of(binary_vars)) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value") %>%
  count(Variable, Value)
# Map human-readable names for binary variable values
value_labels <- c("0" = "No/Not Applicable/Male", "1" = "Yes/Applicable/Female")
ggplot(binary_data, aes(x = Variable, y = n, fill = as.factor(Value))) +
```

```r
geom_bar(stat = "identity", position = "dodge", color = "black") +
scale_fill_manual(
  values = c("0" = "steelblue", "1" = "lightcoral"),
  labels = value_labels
) +
geom_text(
  aes(label = n),
  position = position_dodge(width = 0.9),
  vjust = -0.5,
  size = 3
) +
labs(
  title = "Distribution of Binary Variables",
  x = "Variables",
  y = "Count",
  fill = "Value"
) +
theme_minimal() +
theme(
  axis.text.x = element_text(angle = 45, hjust = 1),
  plot.title = element_text(hjust = 0.5)
)
```



The chart above shows that the vast majority of students are not debtor (don't owe money to school), nor special needs. Most are male, aren't international, or scholarship holders. These findings warrant more digging, since the regression found all but gender to be predictive of the target variable.

Below, the chart compares the same variables against the 3 levels of target ( 0 = dropout, 1 = enrolled, 2 =

graduated)

```
binary_target_table <- traindata %>%
  select(all_of(binary_vars), target) %>%
  pivot_longer(cols = -target, names_to = "Variable", values_to = "Value") %>%
  count(Variable, Value, target) %>%
  pivot_wider(
    names_from = target,
    values_from = n,
    values_fill = 0, # Fill missing combinations with 0
    names_prefix = "Target_"
  )
print(binary_target_table)
```

```
## # A tibble: 12 x 5
##    Variable                 Value Target_1 Target_2 Target_3
##    <chr>                    <int>    <int>    <int>    <int>
##  1 debtor                       0      887      578     1683
##  2 debtor                       1      238       69       85
##  3 educational_special_needs    0     1111      639     1750
##  4 educational_special_needs    1       14        8       18
##  5 gender                       0      556      397     1325
##  6 gender                       1      569      250      443
##  7 international                0     1098      624     1727
##  8 international                1       27       23       41
##  9 scholarship_holder           0     1017      538     1104
## 10 scholarship_holder           1      108      109      664
## 11 tuition_fees_up_to_date      0      364       35       22
## 12 tuition_fees_up_to_date      1      761      612     1746
```

The chart above reflects the regression above that these variables do have a high number of graduated students, which is typical with the National Center for Education Statistics reporting average 6 year graduation of college at 64% in 2020.

A few powerful takeaways include:

- A significant number of students who dropped out (312) were debtors compared to those who graduated (101) or are still enrolled (90).
- A similar number of students without scholarships dropped out (1287) as those who graduated (1374), suggesting scholarships are not as predictive as debtors.

### Scholarship VS Dropout

Below, one can see the rate of scholarship holders dropping out (12%) is much lower than that of non_scholarship holders (38%).

```
# Compare scholarship holder drop out vs non_scholarship holder dropout
scholarship_holder_rate <- binary_target_table %>%
  filter(Variable == "scholarship_holder", Value == 1) %>%
  summarize(
    total_scholarship_dropouts = sum(Target_1),
    total_scholarship_students = sum(Target_1 + Target_2 + Target_3),
    scholarship_dropout_rate = total_scholarship_dropouts / total_scholarship_students
  )
scholarship_holder_rate
```

```r
# same as above for non_scholarship holders
non_scholarship_holder_rate <- binary_target_table %>%
  filter(Variable == "scholarship_holder", Value == 0) %>%
  summarize(
    total_non_scholarship_dropouts = sum(Target_1),
    total_non_scholarship_students = sum(Target_1 + Target_2 + Target_3),
    non_scholarship_dropout_rate = total_non_scholarship_dropouts / total_non_scholarship_students
  )
non_scholarship_holder_rate
```

This finding suggests financial support is a strong incentive to not dropout.

### Gender Vs Dropout

Although there are significantly more men (2278) than women (1262), in the dataset, they have roughly the same number of drop outs (men = 556 & women = 569). As a result, the rate of female_drop out is very high 45% for women vs 24% for men. This could be sampling error and possibly unique to the dataset which is taken from the following diverse degree programs: "agronomy, design, education, nursing, journalism, management, social service, and technologies" (Martins et al., 2021).

```r
gender_male_rate <- binary_target_table %>%
  filter(Variable == "gender", Value == 0) %>%
  summarize(
    total_male_dropouts = sum(Target_1),
    total_male_students = sum(Target_1 + Target_2 + Target_3),
    male_dropout_rate = total_male_dropouts / total_male_students
  )
gender_male_rate
```

```r
gender_female_rate <- binary_target_table %>%
  filter(Variable == "gender", Value == 1) %>%
  summarize(
    total_female_dropouts = sum(Target_1),
    total_female_students = sum(Target_1 + Target_2 + Target_3),
    female_dropout_rate = total_female_dropouts / total_female_students
  )
gender_female_rate
```

### Owing Fees VS Dropout

Almost all of students who owe tuition fees (87%) dropout. Shedding light again on the importance of financial support and reflecting the finding in the regression chart as the most predictive with a correlation coefficient of .5.

```r
owe_fees_drop_rate <- binary_target_table %>%
  filter(Variable == "tuition_fees_up_to_date", Value == 0) %>%
  summarize(
    total_owe_fee_dropouts = sum(Target_1),
    total_owe_fee_students = sum(Target_1 + Target_2 + Target_3),
    owe_fee_dropout_rate = total_owe_fee_dropouts / total_owe_fee_students
  )
owe_fees_drop_rate
```

### International Students & Dropout

International students have a drop out rate of 30%, which is similar to the total dropout rate of the data set ~32%. This is interesting when international status was the second most predictive variable after tuition and fees up to date.

```
international_rate <- binary_target_table %>%
  filter(Variable == "international", Value == 1) %>%
  summarize(
    total_international_dropouts = sum(Target_1),
    total_international_students = sum(Target_1 + Target_2 + Target_3),
    international_dropout_rate = total_international_dropouts / total_international_students
  )
international_rate
```

```
# Calculate overall dropout rate
overall_dropout_rate <- traindata %>%
  summarize(
    total_dropouts = sum(target == 1),
    total_students = n(),
    dropout_rate = total_dropouts / total_students
  )
overall_dropout_rate
```

### Students with Special Needs & Dropout

Although there were not many students with special needs in the dataset, the perseverence of these students, and possibly their support is clear when the dropout rate of 35% is not much higher than that of the dataset as a whole (~32%)

```
educational_special_needs_rate <- binary_target_table %>%
  filter(Variable == "educational_special_needs", Value == 1) %>%
  summarize(
    total_special_needs_dropouts = sum(Target_1),
    total_special_needs_students = sum(Target_1 + Target_2 + Target_3),
    special_needs_dropout_rate = total_special_needs_dropouts / total_special_needs_students
  )
educational_special_needs_rate
```

## Exploring Continuous Variables (Academic Performance Variables)

This section focuses on exploring the continuous variables like academic performance.

```
semester_variables <- variable_table %>%
  slice(22:33) %>%
  mutate(
    Semester = if_else(str_detect(variable_name, "1st_sem"), "1st Semester", "2nd Semester")
  )
print(semester_variables %>% select(variable_name, Semester))
```

```
## # A tibble: 12 x 2
##    variable_name                                Semester
##    <chr>                                        <chr>
##  1 curricular_units_1st_sem_credited            1st Semester
##  2 curricular_units_1st_sem_enrolled            1st Semester
##  3 curricular_units_1st_sem_evaluations         1st Semester
##  4 curricular_units_1st_sem_approved            1st Semester
##  5 curricular_units_1st_sem_grade               1st Semester
##  6 curricular_units_1st_sem_without_evaluations 1st Semester
##  7 curricular_units_2nd_sem_credited            2nd Semester
##  8 curricular_units_2nd_sem_enrolled            2nd Semester
##  9 curricular_units_2nd_sem_evaluations         2nd Semester
## 10 curricular_units_2nd_sem_approved            2nd Semester
## 11 curricular_units_2nd_sem_grade               2nd Semester
## 12 curricular_units_2nd_sem_without_evaluations 2nd Semester
```

As seen above the curricular unit variables give us 6 looks at how the students stand each semester:

- The number of credits earned
- The number of units enrolled in
- The number of evaluations/ tests taken
- The number of units they were approved to take
- Their Grade avg
- The number of units that didn't have tests/evaluations (conceivably easier courses.)

### Identify trends

The following code groups the continuous variables by semester and summarize their statistics

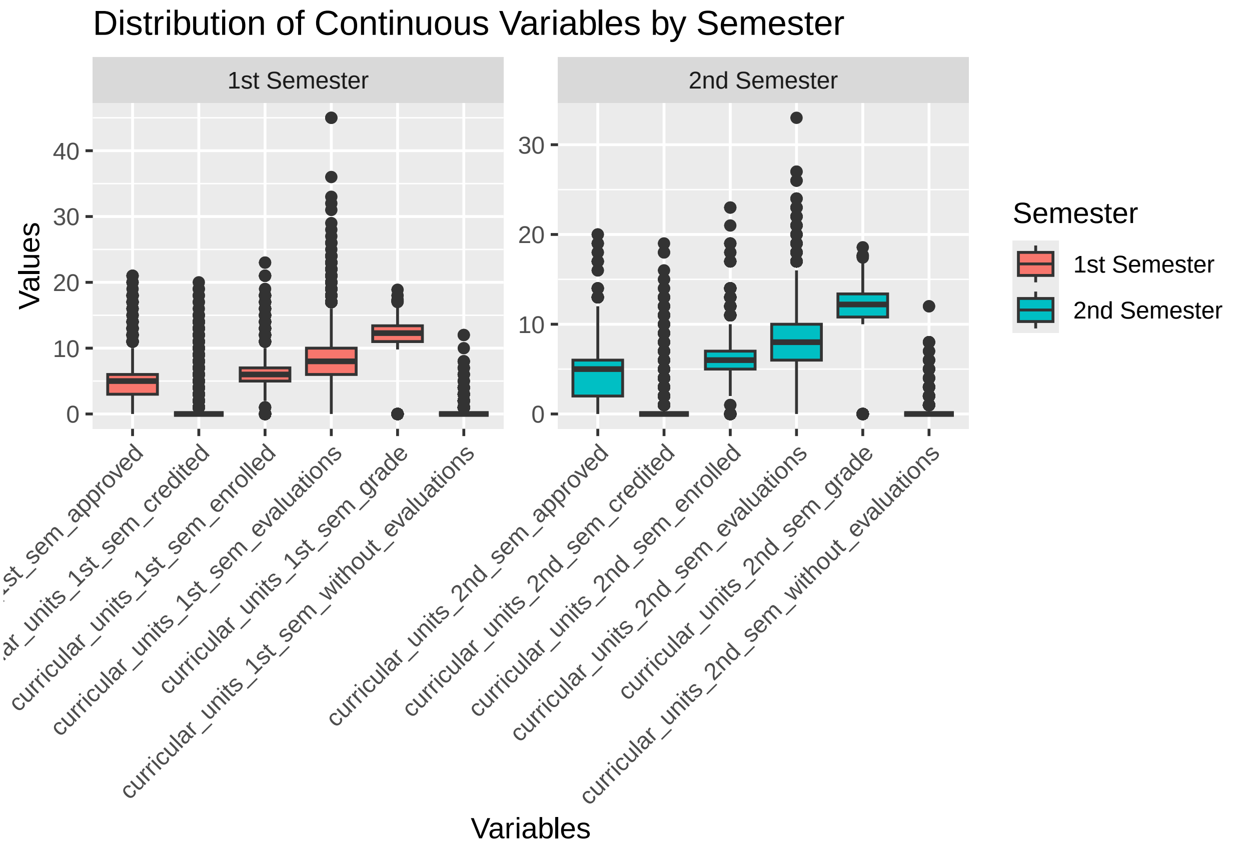
```
semester_summary <- traindata %>%
  select(all_of(semester_variables$variable_name)) %>%
  pivot_longer(everything(), names_to = "variable_name", values_to = "value") %>%
  left_join(semester_variables, by = "variable_name") %>%
  group_by(Semester, variable_name) %>%
  summarize(
    Mean = mean(value, na.rm = TRUE),
    Median = median(value, na.rm = TRUE),
    SD = sd(value, na.rm = TRUE),
    Min = min(value, na.rm = TRUE),
    Max = max(value, na.rm = TRUE),
    .groups = "drop"
  )
print(semester_summary)
```

```
## # A tibble: 12 x 7
##    Semester     variable_name                      Mean Median    SD   Min   Max
##    <chr>        <chr>                             <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1 1st Semester curricular_units_1st_sem_approv~  4.70     5   3.05      0  21
##  2 1st Semester curricular_units_1st_sem_credit~  0.673    0   2.27      0  20
##  3 1st Semester curricular_units_1st_sem_enroll~  6.25     6   2.41      0  23
##  4 1st Semester curricular_units_1st_sem_evalua~  8.30     8   4.15      0  45
##  5 1st Semester curricular_units_1st_sem_grade   10.7     12.3 4.83      0  18.9
##  6 1st Semester curricular_units_1st_sem_withou~  0.143    0   0.718     0  12
##  7 2nd Semester curricular_units_2nd_sem_approv~  4.43     5   2.99      0  20
##  8 2nd Semester curricular_units_2nd_sem_credit~  0.514    0   1.85      0  19
##  9 2nd Semester curricular_units_2nd_sem_enroll~  6.22     6   2.15      0  23
## 10 2nd Semester curricular_units_2nd_sem_evalua~  8.07     8   3.91      0  33
## 11 2nd Semester curricular_units_2nd_sem_grade   10.3     12.2 5.19      0  18.6
## 12 2nd Semester curricular_units_2nd_sem_withou~  0.153    0   0.778     0  12
```

The summary above shows on average students are approved for 4.7 to 4.4 units each semester, but enroll in 6.25 to 6.22, and are only credited .6 to .5 in the first and second semester, respectively. This finding suggests many students are not succeeding in passing their courses. Which is further exemplified in an average grade of 10.3/20 in sem 1 and sem2. which is only .3 on average above the minimum passing grade. Most course offer at least a few evaluations (evident by .14 -.15 avg units without an eval each semesters)

The following code visualizes the distribution of each of these variables.

```r
# Boxplot of variables grouped by semester
units_distributed<- traindata %>%
  select(all_of(semester_variables$variable_name)) %>%
  pivot_longer(everything(), names_to = "variable_name", values_to = "value") %>%
  left_join(semester_variables, by = c("variable_name" = "variable_name")) %>%
  ggplot(aes(x = variable_name, y = value, fill = Semester)) + # Match exact case of "Semester"
  geom_boxplot() +
  facet_wrap(~ Semester, scales = "free") + # Ensure "Semester" column exists and matches case
  labs(
    title = "Distribution of Continuous Variables by Semester",
    x = "Variables",
    y = "Values",
    fill = "Semester"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
units_distributed
```



Most clearly, students on average are earning 0 credits, which may be more descriptive of graduates who are no longer earning credits, but also telling of students who might have dropped out, more than students enrolled (which is the smallest group). Without more context into what units credited means, it's hard to draw conclusions.

Again, average grades are only just above 10/20 in semester 1, with some more variablility in semester 2, but a similar average.

## Explore Categorical variables,

This section looks at nationality, Previous qualification (education), previous qualification (education) grade, degree programs, and courses.

### Nationality

Below shows over 97% of the dataset is Portuguese. An appropriate finding for a Portugal based university.

```r
# Summarize nationality distribution in the dataset
nationality_summary <- traindata %>%
  group_by(nacionality) %>%
  summarize(
    count = n(),
    percentage = (n() / nrow(traindata)) * 100
  ) %>%
  arrange(desc(count))
print(nationality_summary)
```

```
## # A tibble: 21 x 3
##    nacionality count percentage
##          <int> <int>      <dbl>
##  1           1  3449    97.4
##  2          41    30     0.847
##  3          26    13     0.367
##  4           6    11     0.311
##  5          22     9     0.254
##  6          24     4     0.113
##  7         100     3     0.0847
##  8         103     3     0.0847
##  9           2     2     0.0565
## 10          11     2     0.0565
## # i 11 more rows
```

```r
# Pull description for nationality (row 8) in the variable table
variable_description <- variable_table %>%
  slice(8) %>%
  select(variable_name, description) %>%
  mutate(description = str_wrap(description, width = 80))
description_text <- variable_description$description
cat(description_text)
```

```
## 1 - Portuguese - 2 - German 6 - Spanish 11 - Italian 13 - Dutch 14 - English 17
## - Lithuanian 21 - Angolan 22 - Cape Verdean 24 - Guinean 25 - Mozambican 26 -
## Santomean 32 - Turkish 41 - Brazilian 62 - Romanian 100 - Moldova (Republic of)
## 101 - Mexican 103 - Ukrainian 105 - Russian 108 - Cuban 109 - Colombian
```

### Previous Educational Achievement

Result below show that 84% of students in the sample have recently come from secondary education (1) AKA High Schools suggesting the sample describes largely Freshman and Sophomore students who had not had other more recent achievements, like the 5% with Technological specialization course (39). Notably, ~4%, the third largest group, is coming with 9,10,11th grade completed, but not a secondary level degree. Suggesting these students are taking college credits while in High School. This group may perform better or worse than others and could be a useful feature.

```
previous_education_summary <- traindata %>%
  group_by(previous_qualification) %>%
  summarize(
    Count = n(),
    Percentage = (n() / nrow(traindata)) * 100
  ) %>%
  arrange(desc(Count))
print(previous_education_summary)
```

```
## # A tibble: 16 x 3
##    previous_qualification Count Percentage
##                     <int> <int>      <dbl>
##  1                      1  2971    83.9
##  2                     39   182     5.14
##  3                     19   129     3.64
##  4                      3    94     2.66
##  5                     12    38     1.07
##  6                     40    32     0.904
##  7                     42    29     0.819
##  8                      2    20     0.565
##  9                      6    11     0.311
## 10                      9     9     0.254
## 11                      4     7     0.198
## 12                     38     6     0.169
## 13                     43     5     0.141
## 14                     10     4     0.113
## 15                     15     2     0.0565
## 16                      5     1     0.0282
```

```
# Pull description for educational achievement (row 6) in the variable table
variable_description <- variable_table %>%
  slice(6) %>%
  select(variable_name, description) %>%
  mutate(description = str_wrap(description, width = 80))
description_text <- variable_description$description
cat(description_text)
```

```
## 1 - Secondary education 2 - Higher education - bachelor's degree 3 - Higher
## education - degree 4 - Higher education - master's 5 - Higher education -
## doctorate 6 - Frequency of higher education 9 - 12th year of schooling - not
## completed 10 - 11th year of schooling - not completed 12 - Other - 11th year
## of schooling 14 - 10th year of schooling 15 - 10th year of schooling - not
## completed 19 - Basic education 3rd cycle (9th/10th/11th year) or equiv. 38
## - Basic education 2nd cycle (6th/7th/8th year) or equiv. 39 - Technological
## specialization course 40 - Higher education - degree (1st cycle) 42 -
## Professional higher technical course 43 - Higher education - master (2nd cycle)
```

Below confirms the age of the sample is aligns with first and second year univeristy students. About half the sample is 18 or 19, with 80% falling between 18 and 27.

```
# Summarize age_at_enrollment
age_summary <- traindata %>%
  group_by(age_at_enrollment) %>%
  summarize(
    Count = n(),
    Percentage = (n() / nrow(traindata)) * 100
  ) %>%
  arrange(desc(Percentage))
print(age_summary)
```

```
## # A tibble: 46 x 3
##    age_at_enrollment Count Percentage
##                <int> <int>      <dbl>
##  1                18   825      23.3
##  2                19   746      21.1
##  3                20   483      13.6
##  4                21   255       7.20
##  5                22   144       4.07
##  6                24   101       2.85
##  7                23    83       2.34
##  8                25    72       2.03
##  9                26    71       2.01
## 10                27    68       1.92
## # i 36 more rows
```

### Previous Education Grade

The average grade coming into university was about 133/200, with a standard deviation of 13.

```
# Summarize previous_qualification_grade (index 7)
grade_summary <- traindata %>%
  summarize(
    Min_Grade = min(previous_qualification_grade, na.rm = TRUE),
    Max_Grade = max(previous_qualification_grade, na.rm = TRUE),
    Mean_Grade = mean(previous_qualification_grade, na.rm = TRUE),
    Median_Grade = median(previous_qualification_grade, na.rm = TRUE),
    SD_Grade = sd(previous_qualification_grade, na.rm = TRUE)
  )
print(grade_summary)
```

```
##   Min_Grade Max_Grade Mean_Grade Median_Grade SD_Grade
## 1        95       190   132.6108        133.1 13.30456
```

```
# Pull description for previous education grade (row 7) in the variable table
variable_description <- variable_table %>%
  slice(7) %>%
  select(variable_name, description) %>%
  mutate(description = str_wrap(description, width = 80))
description_text <- variable_description$description
cat(description_text)
```

```
## Grade of previous qualification (between 0 and 200)
```

This finding shows that on average students got grades of 65% which is 15% above passing on the Portuguese scale. In comparison to the average grades for first and second semester at university being at 50%, the data suggest that the students are perofrming more poorly than they did in their previous education..

### Courses

Investigating the distribution of courses [row 4] taken in the sample will contribute to understanding the study and where possible student support is need. Below shows that out of 17 courses and 3540 students, the majority (17%) are in the nursing program, 8.5% in Management, 8% in social services, and ~8% in both veterinary nursing and Journalism and communication. The rest of the course have 6% - .2% each. These results suggest a good distribution across fields of study.

```
course_summary <- traindata %>%
  group_by(course) %>%
  summarize(
    Count = n(),
    Percentage = (n() / nrow(traindata)) * 100
  ) %>%
  arrange(desc(Percentage))
print(course_summary)
```

```
## # A tibble: 17 x 3
##    course Count Percentage
##     <int> <int>      <dbl>
##  1   9500   615     17.4
##  2   9147   299      8.45
##  3   9238   283      7.99
##  4   9085   273      7.71
##  5   9773   267      7.54
##  6   9991   222      6.27
##  7   9670   213      6.02
##  8   9254   203      5.73
##  9   9070   177      5
## 10   8014   175      4.94
## 11   9003   164      4.63
## 12    171   161      4.55
## 13   9853   151      4.27
## 14   9119   141      3.98
## 15   9130   121      3.42
## 16   9556    67      1.89
## 17     33     8      0.226
```

```
# Pull description for course (row 4) in the variable table
variable_description <- variable_table %>%
  slice(4) %>%
  select(variable_name, description) %>%
  mutate(description = str_wrap(description, width = 80))
# Print Description
description_text <- variable_description$description
cat(description_text)
```

```
## 33 - Biofuel Production Technologies 171 - Animation and Multimedia Design
## 8014 - Social Service (evening attendance) 9003 - Agronomy 9070 - Communication
## Design 9085 - Veterinary Nursing 9119 - Informatics Engineering 9130 -
## Equinculture 9147 - Management 9238 - Social Service 9254 - Tourism 9500 -
## Nursing 9556 - Oral Hygiene 9670 - Advertising and Marketing Management 9773 -
## Journalism and Communication 9853 - Basic Education 9991 - Management (evening
## attendance)
```

**Exploratory Data Analysis Summary:**

- Examined binary variables (e.g., debtor status, scholarship holder, gender) and their distributions.
- Identified that scholarship holders have a significantly lower dropout rate (12%) compared to non-scholarship holders (38%).
- Noted that international students have a dropout rate similar to the overall rate (~30%), despite being a predictive variable.
- Analyzed semester-wide academic performance, observing that dropouts have lower average grades compared to graduates.
- Visualized data using bar charts and boxplots to highlight key differences across groups.

Based on the insights from the exploratory analysis, the next section engineers new features. These features will capture important factors such as financial status, academic performance, and course difficulty, enhancing the predictive power of the models to be built.

## Feature Development

Feature development is the practice of getting the most out of a dataset by introducing new variables. The feature engineering function below creates 26 new features in 10 categories. The description of each category of features is found at the end of each chunk of code in the following function

```r
feature_engineering <- function(data) {

  # 1. Grading Scale Mapping (Numeric Categories)
  map_grades_to_category <- function(grade) {
    case_when(
      grade >= 18 & grade <= 20 ~ 5,  # Very Good with Distinction
      grade >= 16 & grade < 18 ~ 4,   # Very Good
      grade >= 14 & grade < 16 ~ 3,   # Good
      grade >= 10 & grade < 14 ~ 2,   # Sufficient
      grade >= 7 & grade < 10 ~ 1,    # Poor
      grade < 7 ~ 0,                  # Very Poor
      TRUE ~ NA_real_                 # Handle missing values
    )
  }
  data <- data %>%
    mutate(
      grade_category_1st_sem = map_grades_to_category(curricular_units_1st_sem_grade),
      grade_category_2nd_sem = map_grades_to_category(curricular_units_2nd_sem_grade)
    ) # The grade_categories features use the Portuguese grading standards to define 6 ordinal academic

  # 2. Owe Money Feature
  data <- data %>%
    mutate(
      owe_money = ifelse(debtor == 1 & tuition_fees_up_to_date == 0, 1, 0)
    ) # The owe_money feature attempts to create meaning out of students who are financially strained v

  # 3. Failing Features
  data <- data %>%
    mutate(
      failing_first_sem = ifelse(grade_category_1st_sem <= 1, 1, 0),
      failing_both_semesters = ifelse(grade_category_1st_sem <= 1 & grade_category_2nd_sem <= 1, 1, 0)
    ) # These failing semester features define class failure vs. pass by using the grade_category in fe

  # 4. Under-Enrolled
```

```
data <- data %>%
  mutate(
    under_enrolled_1st_sem = ifelse(curricular_units_1st_sem_enrolled < 10, 1, 0),
    under_enrolled_2nd_sem = ifelse(curricular_units_2nd_sem_enrolled < 10, 1, 0),
    under_enrolled_both_sem = ifelse(under_enrolled_1st_sem == 1 & under_enrolled_2nd_sem == 1, 1, 0)
  ) # The under-enrolled feature identifies students who are off track for graduating on time by
# using a threshold of 50% of the required pace. To graduate in 4 years with 180 credits,
# students need 22.5 credits per semester. This feature flags students enrolled in fewer than
# half that amount (10 credits in the 1st semester and 2nd semester).

# 5. Course Difficulty
data <- data %>%
  group_by(course) %>%
  mutate(
    avg_grade_1st_sem = mean(curricular_units_1st_sem_grade, na.rm = TRUE),
    std_grade_1st_sem = sd(curricular_units_1st_sem_grade, na.rm = TRUE),
    avg_grade_2nd_sem = mean(curricular_units_2nd_sem_grade, na.rm = TRUE),
    std_grade_2nd_sem = sd(curricular_units_2nd_sem_grade, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  mutate(
    overall_avg_grade = (avg_grade_1st_sem + avg_grade_2nd_sem) / 2,
    overall_std_grade = (std_grade_1st_sem + std_grade_2nd_sem) / 2
  ) # The course difficulty feature quantifies how challenging a course is based on all student grade

# 6. Hard Courses
data <- data %>%
  mutate(
    hard_courses = ifelse(curricular_units_1st_sem_evaluations > 10, 1, 0)
  ) # Hard courses assume that many evaluations/tests make a course harder.

# 7. Discouraging Courses
data <- data %>%
  group_by(course) %>%
  mutate(
    avg_course_grade = mean(curricular_units_1st_sem_grade, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  mutate(
    discouraging_courses = ifelse(avg_course_grade < 10, 1, 0)
  )# Courses with low average grades (below the passing threshold of 10) are flagged as discouraging.

# 8. Historical Success
data <- data %>%
  mutate(
    normalized_previous_grade = previous_qualification_grade / 200,
    strong_historical_success = ifelse(normalized_previous_grade >= 0.663, 1, 0),
    weak_historical_success = ifelse(normalized_previous_grade < 0.5966, 1, 0)
  ) # Uses normalized thresholds to categorize students with strong or weak prior success before comi

# 9. Grade-Based Summaries
data <- data %>%
  mutate(
      average_grade = (curricular_units_1st_sem_grade + curricular_units_2nd_sem_grade) / 2,
      semester_grade_gap = curricular_units_2nd_sem_grade - curricular_units_1st_sem_grade
    ) # Summarizes overall grade performance and tracks grade changes between semesters.

  # 10. Prior Education Group Distributions
  data <- data %>%
    group_by(previous_qualification) %>%
    mutate(
      prior_avg_grade_1st_sem = mean(curricular_units_1st_sem_grade, na.rm = TRUE),
      prior_std_grade_1st_sem = sd(curricular_units_1st_sem_grade, na.rm = TRUE),
      prior_avg_grade_2nd_sem = mean(curricular_units_2nd_sem_grade, na.rm = TRUE),
      prior_std_grade_2nd_sem = sd(curricular_units_2nd_sem_grade, na.rm = TRUE)
    ) %>%
    ungroup()
  # Tracks grade patterns for students by prior education level.

  return(data)
}


# Apply to training data
traindata <- feature_engineering(traindata)
```

## Validate Feature Development on Training Data

This section validates that the feature engineering function above was effective and didn't introduce NA's into the dataset.

The following code prints the new size of the datasets (now 63 Features):

```
ncol(traindata)
```

```
## [1] 63
```

```
colnames(traindata)
```

```
##  [1] "marital_status"
##  [2] "application_mode"
##  [3] "application_order"
##  [4] "course"
##  [5] "daytime_evening_attendance"
##  [6] "previous_qualification"
##  [7] "previous_qualification_grade"
##  [8] "nacionality"
##  [9] "mother_s_qualification"
## [10] "father_s_qualification"
## [11] "mother_s_occupation"
## [12] "father_s_occupation"
## [13] "admission_grade"
## [14] "displaced"
## [15] "educational_special_needs"
## [16] "debtor"
## [17] "tuition_fees_up_to_date"
## [18] "gender"
## [19] "scholarship_holder"
## [20] "age_at_enrollment"
## [21] "international"
## [22] "curricular_units_1st_sem_credited"
## [23] "curricular_units_1st_sem_enrolled"
## [24] "curricular_units_1st_sem_evaluations"
## [25] "curricular_units_1st_sem_approved"
## [26] "curricular_units_1st_sem_grade"
## [27] "curricular_units_1st_sem_without_evaluations"
## [28] "curricular_units_2nd_sem_credited"
## [29] "curricular_units_2nd_sem_enrolled"
## [30] "curricular_units_2nd_sem_evaluations"
## [31] "curricular_units_2nd_sem_approved"
## [32] "curricular_units_2nd_sem_grade"
## [33] "curricular_units_2nd_sem_without_evaluations"
## [34] "unemployment_rate"
## [35] "inflation_rate"
## [36] "gdp"
## [37] "target"
## [38] "grade_category_1st_sem"
## [39] "grade_category_2nd_sem"
## [40] "owe_money"
## [41] "failing_first_sem"
## [42] "failing_both_semesters"
## [43] "under_enrolled_1st_sem"
## [44] "under_enrolled_2nd_sem"
## [45] "under_enrolled_both_sem"
## [46] "avg_grade_1st_sem"
## [47] "std_grade_1st_sem"
## [48] "avg_grade_2nd_sem"
## [49] "std_grade_2nd_sem"
## [50] "overall_avg_grade"
## [51] "overall_std_grade"
## [52] "hard_courses"
## [53] "avg_course_grade"
## [54] "discouraging_courses"
## [55] "normalized_previous_grade"
## [56] "strong_historical_success"
## [57] "weak_historical_success"
## [58] "average_grade"
## [59] "semester_grade_gap"
## [60] "prior_avg_grade_1st_sem"
## [61] "prior_std_grade_1st_sem"
## [62] "prior_avg_grade_2nd_sem"
## [63] "prior_std_grade_2nd_sem"
```

The following function does 4 key things to evaluate the newly engineered features:

- 1. Count total NAs in the dataset
- 2. Count total NAs in each feature
- 3. Count if any features are 100% 0 or 1 to indicate a lack of predictive power
- 4. Plots 8 of a few of the 26 new features

```r
validate_features <- function(data) {
  # 1. Count Total NAs in the Dataset
  total_na <- sum(is.na(data))
  cat("Total NA Count in Dataset:", total_na, "\n")
  # 2. Count NAs per Feature
  na_counts <- data %>%
    summarise(across(everything(), ~ sum(is.na(.)))) %>%
    pivot_longer(everything(), names_to = "Feature", values_to = "NA_Count") %>%
    arrange(desc(NA_Count))

  print("NA Counts for Each Feature:")
  print(na_counts)

  # 3. Identify Features with 100% 0s or 1s
  constant_features <- data %>%
    summarise(across(everything(), ~ all(. == 0, na.rm = TRUE) | all(. == 1, na.rm = TRUE))) %>%
    pivot_longer(everything(), names_to = "Feature", values_to = "Is_Constant") %>%
    filter(Is_Constant == TRUE)

  if (nrow(constant_features) > 0) {
    cat("Features with 100% 0s or 1s:\n")
    print(constant_features$Feature)
  } else {
    cat("No features have 100% 0s or 1s.\n")
  }

  # 4. Visualize Key Features
  key_features <- c(
    "owe_money",
    "failing_first_sem", "under_enrolled_both_sem",
    "discouraging_courses", "weak_historical_success",
    "average_grade", "semester_grade_gap"
  )

  feature_plots <- list()
  for (feature in key_features) {
    if (feature %in% names(data)) {
      p <- ggplot(data, aes_string(x = feature)) +
        geom_bar(fill = "steelblue", color = "black") +
        labs(
          title = paste("Distribution of", feature),
          x = feature,
          y = "Count"
        ) +
        theme_minimal()
      feature_plots[[feature]] <- p
    }
  }

  # Return validation results
  return(list(
    na_counts = na_counts,
    total_na = total_na,
    constant_features = constant_features,
    feature_plots = feature_plots
  ))
}
```

```r
# Apply the updated validation function
validation_train_results <- validate_features(traindata)
```

```
## Total NA Count in Dataset: 2
## [1] "NA Counts for Each Feature:"
## # A tibble: 63 x 2
##    Feature                      NA_Count
##    <chr>                           <int>
##  1 prior_std_grade_1st_sem             1
##  2 prior_std_grade_2nd_sem             1
##  3 marital_status                      0
##  4 application_mode                    0
##  5 application_order                   0
##  6 course                              0
##  7 daytime_evening_attendance          0
##  8 previous_qualification              0
##  9 previous_qualification_grade        0
## 10 nacionality                         0
## # i 53 more rows
## No features have 100% 0s or 1s.
```

With the function applied, the following inspects the traindata for errors.

```r
# Total NAs in the dataset
cat("Total NA Count:", validation_train_results$total_na, "\n")
```

```
## Total NA Count: 2
```

```r
# NA counts per feature
print(validation_train_results$na_counts)
```

```
## # A tibble: 63 x 2
##    Feature                      NA_Count
##    <chr>                           <int>
##  1 prior_std_grade_1st_sem             1
##  2 prior_std_grade_2nd_sem             1
##  3 marital_status                      0
##  4 application_mode                    0
##  5 application_order                   0
##  6 course                              0
##  7 daytime_evening_attendance          0
##  8 previous_qualification              0
##  9 previous_qualification_grade        0
## 10 nacionality                         0
## # i 53 more rows
```

The dataset only has 2 NAs. These two instances can be removed for simplicity. Additionally, no variables appear to have no predictive power by having only 1s or 0s.

```r
# Count features with 100% 0s or 1s
if (nrow(validation_train_results$constant_features) > 0) {
  cat("Features with constant values (100% 0s or 1s):\n")
  print(validation_train_results$constant_features$Feature)
} else {
  cat("No features have 100% 0s or 1s.\n")
}
```

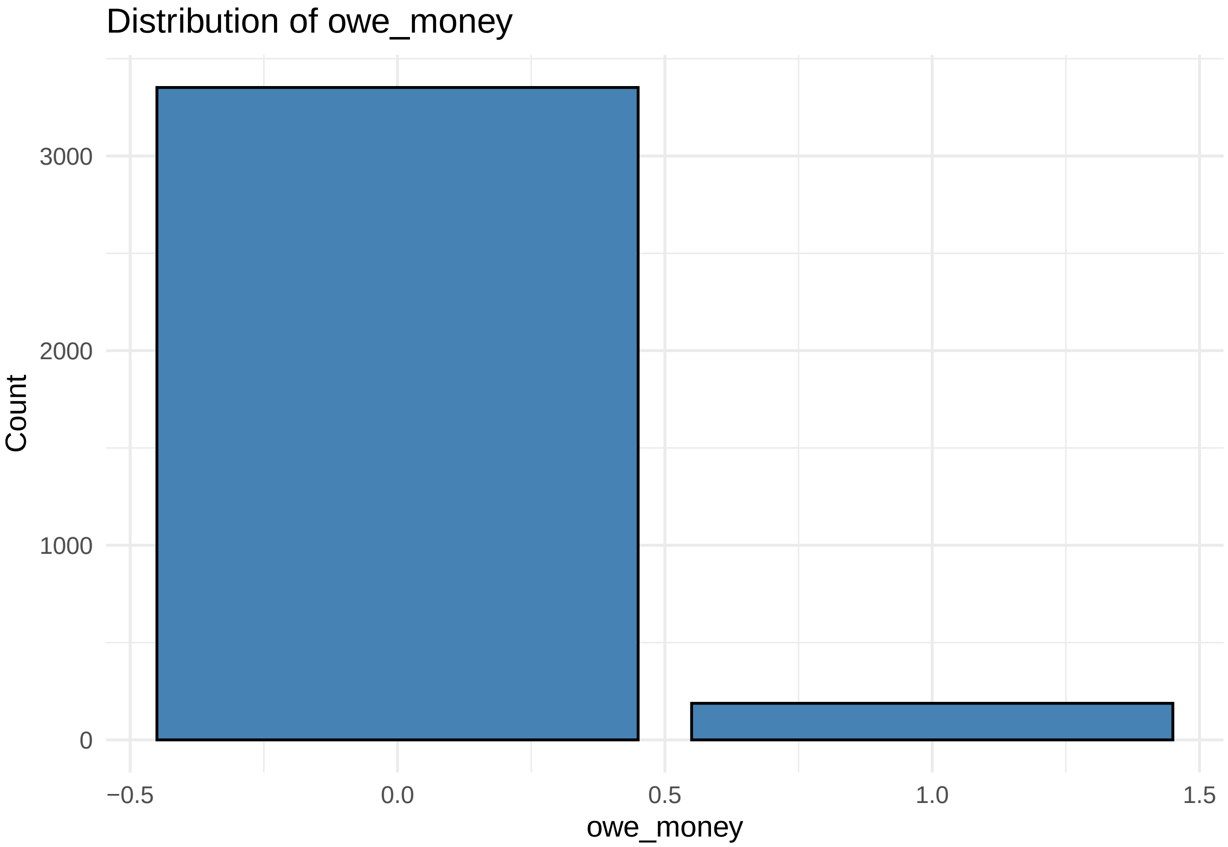
```
## No features have 100% 0s or 1s.
```

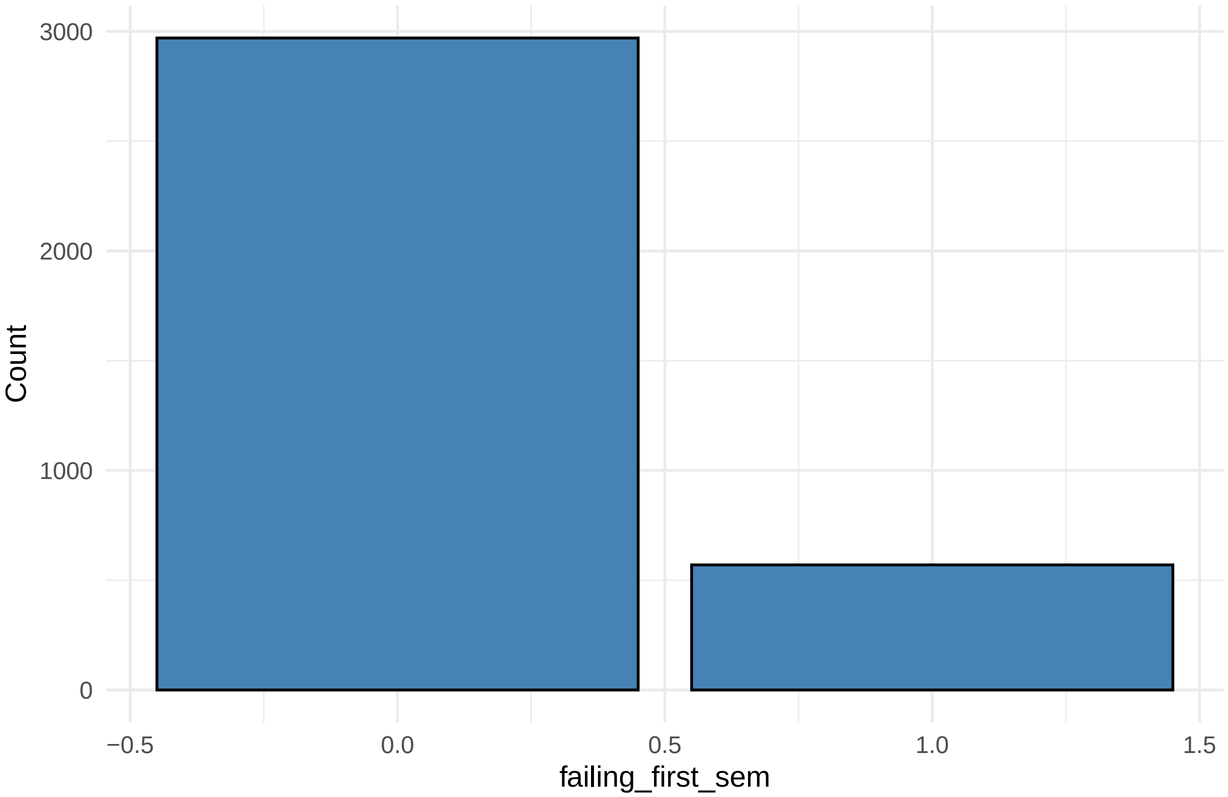
```
traindata <- traindata %>% drop_na()
# Validate the dataset no longer contain NAs
cat("Total NA Count in Training Dataset after removal:", sum(is.na(traindata)), "\n")
```

```
## Total NA Count in Training Dataset after removal: 0
```

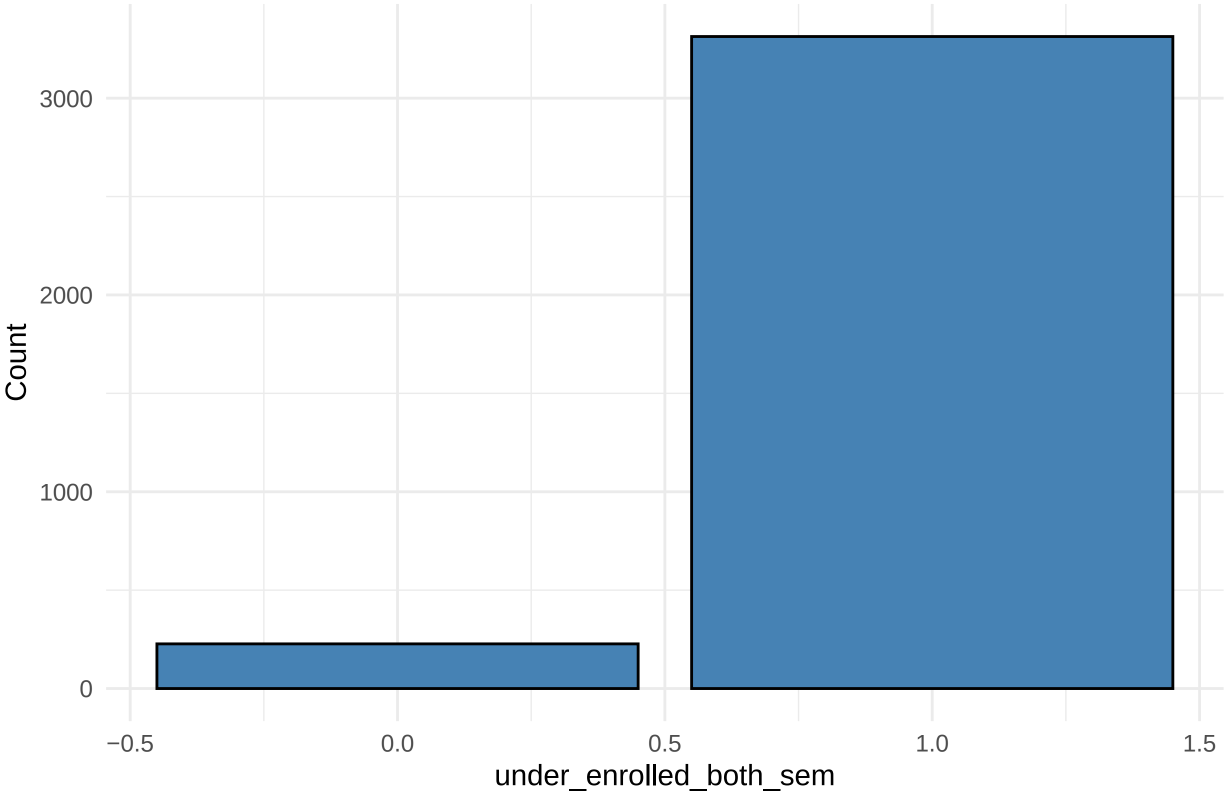
Below visualizes several of the new features for added insight. Their descriptions are below.

```
# View the plots for key features
for (feature_name in names(validation_train_results$feature_plots)) {
  print(validation_train_results$feature_plots[[feature_name]])
}
```

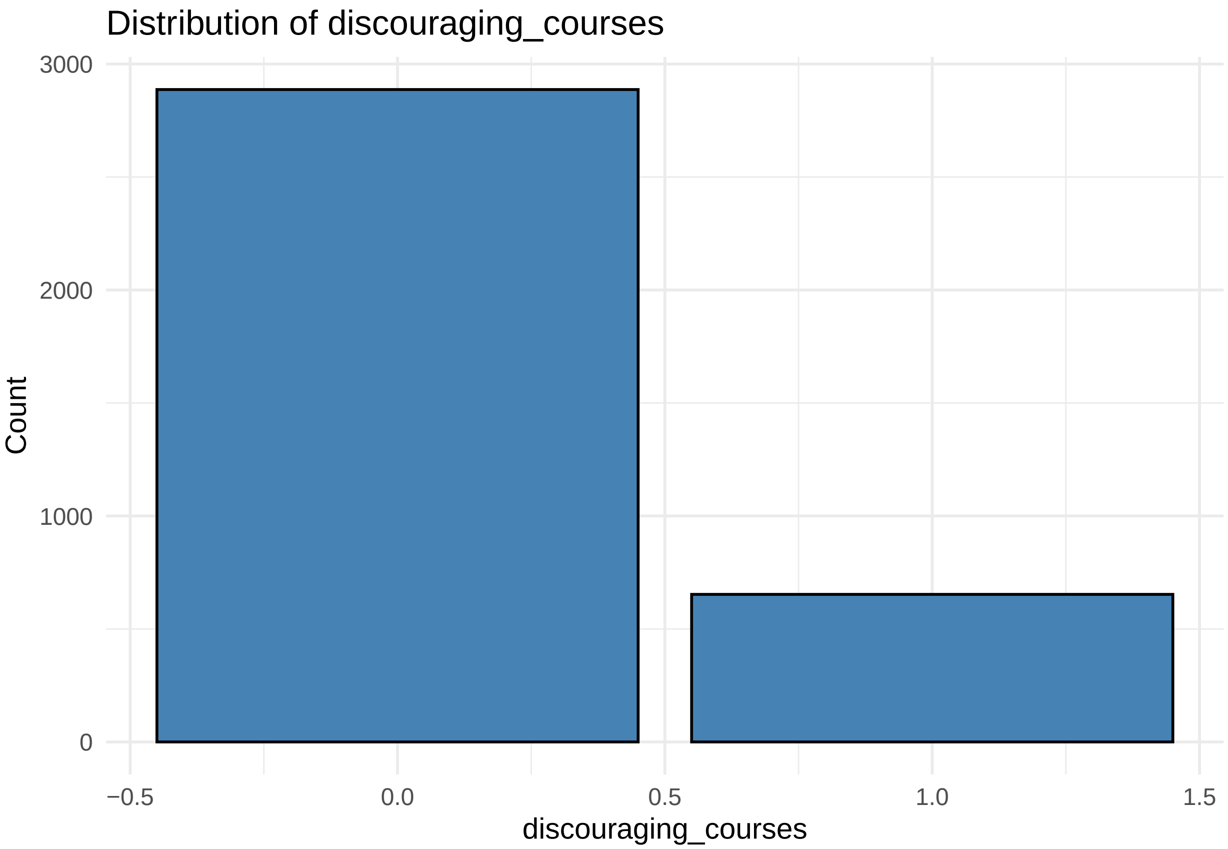


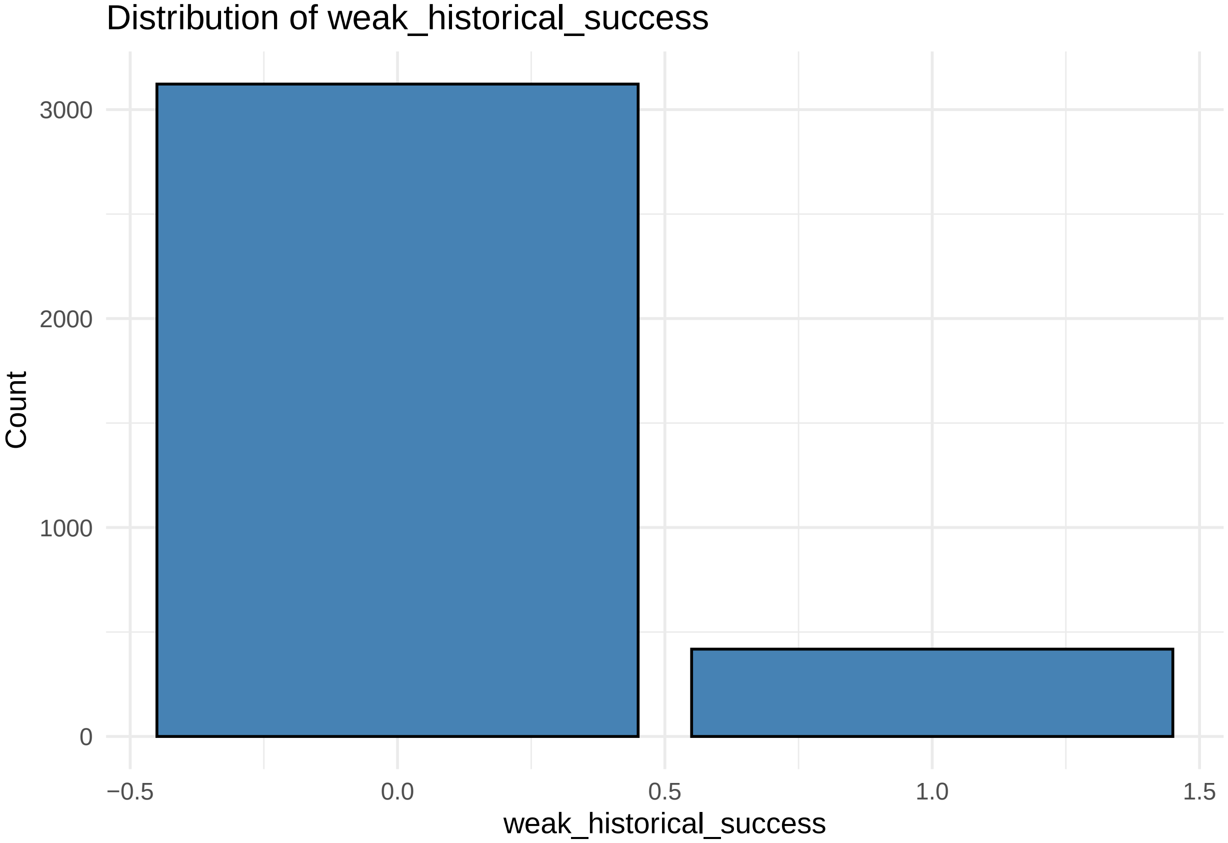


Distribution of failing_first_sem

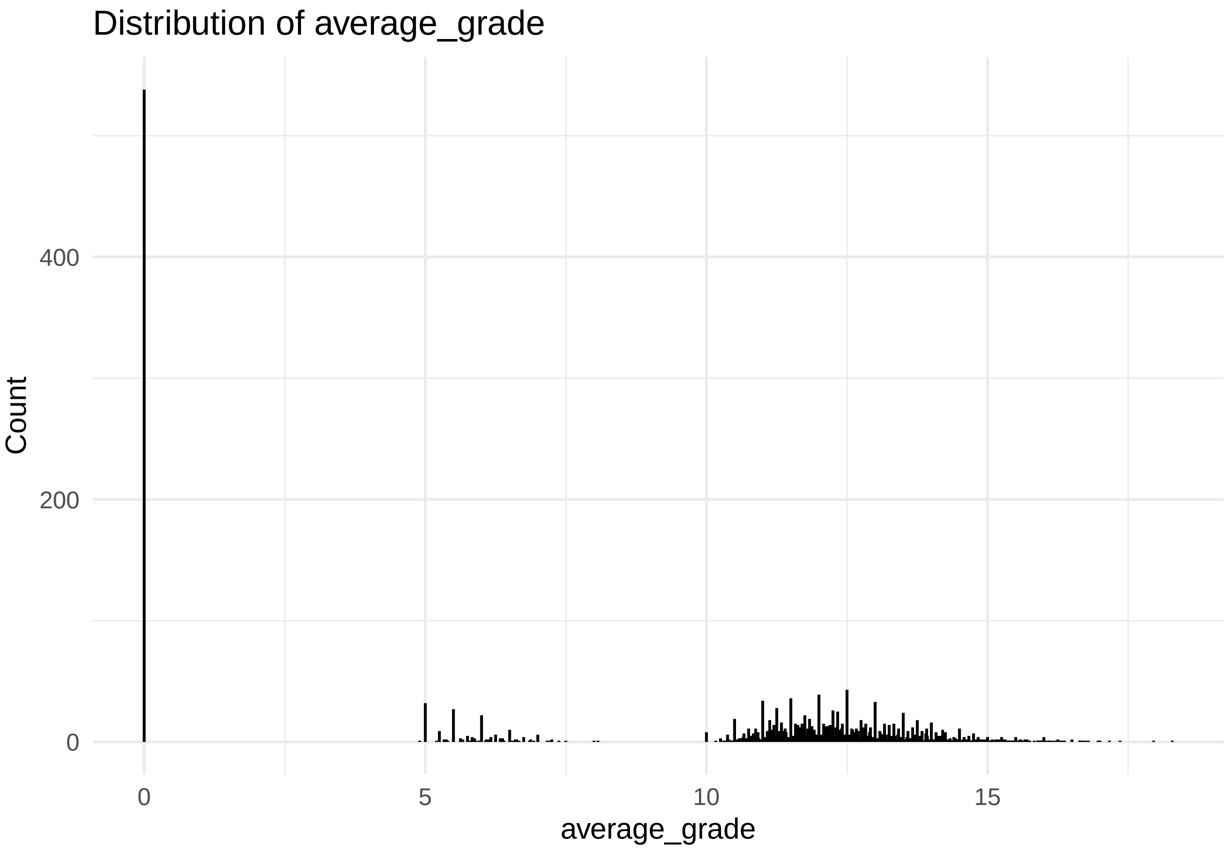


Distribution of under_enrolled_both_sem

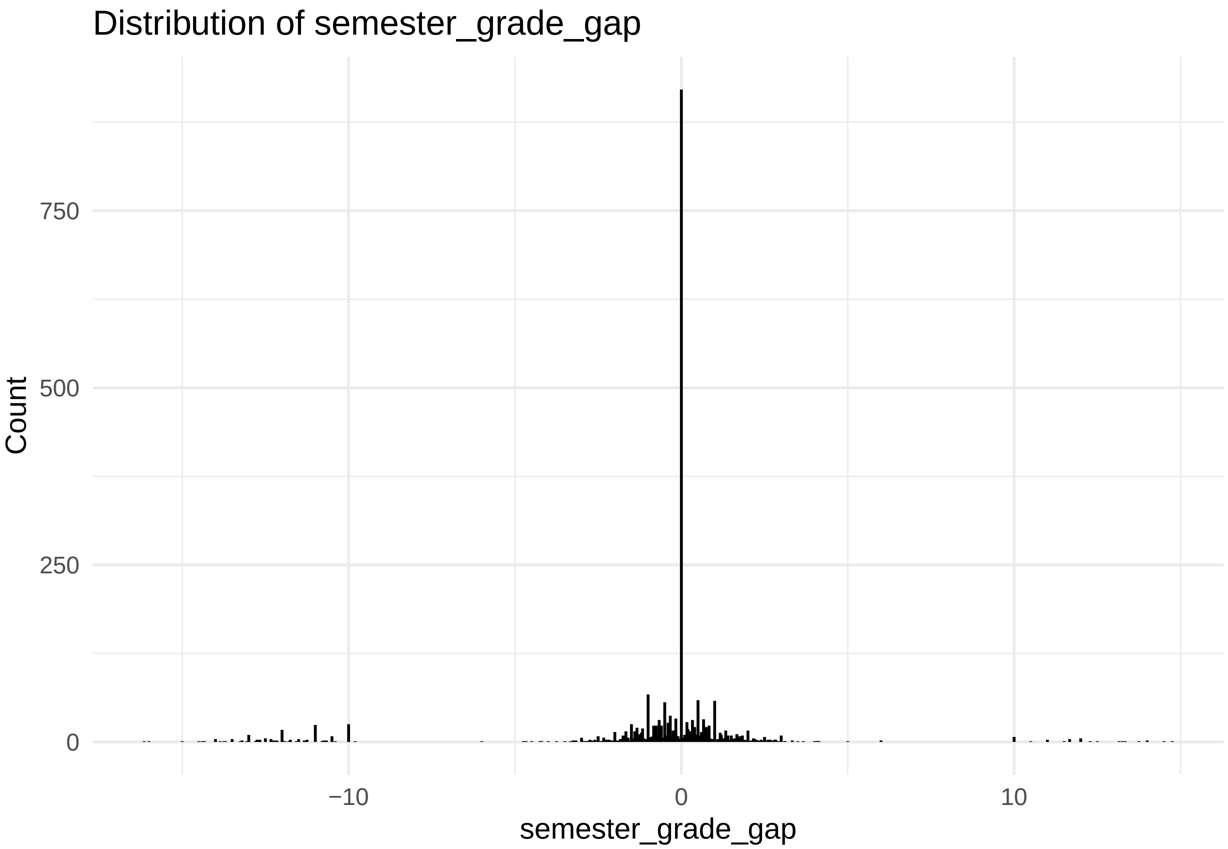




```
## Warning: `position_stack()` requires non-overlapping x intervals.
```



```
## Warning: `position_stack()` requires non-overlapping x intervals.
```



- **Owe money:** distribution is not capturing many students, but since tuition fees up to date was most predictive of target in the regression before, it will be retained.
- **Distribution of failing first semester:** shows about 500 students are getting grades below 10/20.
- **Distribution of Under Enrolled in Both Semesters:** strangely, most students fall into the category of being under enrolled in both semesters (less than 10 credits). This may be due to missing domain knowledge on part of the author.
- **Discouraging courses:** show about 100 students have taken classes that led to them receiving a failing grade (below 10). Strangely, most students fall into the category of being under enrolled in both semesters (less than 10 credits). This may be due to missing domain knowledge on part of the author.
- **Distribution of weak historical success:** shows majority of students performed well in past, but about 500 did not.
- **Distribution of Average grade:** shows largest cluster between 10/20 and 15/20, with a very large concentration at 0, and a smaller concentration between 5 and 10 suggesting some students are in trouble of failing or simply not taking courses.
- **Distribution of Semester gap:** shows most students have no little difference in grades with a normal distribution around 0 with some outliars at -10 and + 10, which may point to dropouts or students who return from a break.

This feature validation section shows that the number of features has been nearly doubled. This could increase predictive power, but could also increase the likelihood of over-fitting. To avoid over fitting steps will be taken in modeling, like cross validation.

## Choosing Classification

This section justifies classification as the task for predicting student outcomes.

Now that the dataset has been loaded, cleaned, and split into training and test sets, followed by exploratory analysis and the development of features to maximize the information within the dataset, it's time to start modeling an algorithm to predict college outcomes.

Since the college outcomes in this dataset are categorized into three distinct groups—dropout, enrolled, and graduated—a classification approach is the most suitable modeling strategy. Classification enables the algorithm to learn from the dataset and classify each student into one of these three outcome groups based on the provided features. This approach aligns well with the structure of the target variable and ensures predictions are interpretable and actionable.

Start by splitting the dataset as before, this time for training the traindata:

```
# Partition traindata
set.seed(456)
train_model_index <- createDataPartition(traindata$target, p = 0.8, list = FALSE)
training_set <- traindata[train_model_index, ]
validation_set <- traindata[-train_model_index, ]

# Convert target to a factor in the training and evaluation sets
training_set$target <- as.factor(training_set$target)
validation_set$target <- as.factor(validation_set$target)
```

## Model #1 Decision Tree Classification

With the dataset split decision trees will be the first model employed. Decision trees will allow for logical separation of variables that predict the three outcomes.

### Train the Decision Tree (DT) Model

```
#Training
decision_tree_model <- rpart(
  target ~ .,
  data = training_set,
  method = "class"
)
```

With the model trained, now use it to predict target on the validation set.

```
## Predict on the validation set
dt_predictions <- predict(decision_tree_model, validation_set, type = "class")

## Evaluate the model
dt_conf_matrix <- confusionMatrix(dt_predictions, validation_set$target)
print(dt_conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   1   2   3
##          1 186  44  17
##          2   0   0   0
##          3  43  81 336
##
## Overall Statistics
##
##                Accuracy : 0.7383
##                  95% CI : (0.7043, 0.7704)
##     No Information Rate : 0.4993
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5344
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8122   0.0000   0.9518
## Specificity            0.8724   1.0000   0.6497
## Pos Pred Value         0.7530      NaN   0.7304
## Neg Pred Value         0.9065   0.8232   0.9312
## Prevalence             0.3239   0.1768   0.4993
## Detection Rate         0.2631   0.0000   0.4752
## Detection Prevalence   0.3494   0.0000   0.6506
## Balanced Accuracy      0.8423   0.5000   0.8008
```

The model performs well in predicting students who will drop out and graduate:

- **81% of actual dropouts were correctly identified as dropouts (Sensitivity = 0.8122).**
- **73% of predicted graduates were correct (Positive Predictive Value = 0.7304).**

However, there are notable challenges:

- The model entirely fails to predict the "enrolled" class (Sensitivity = 0, Precision = NaN). This suggests a lack of information or patterns in the data for this group or is the impact of class imbalance (as seen below):

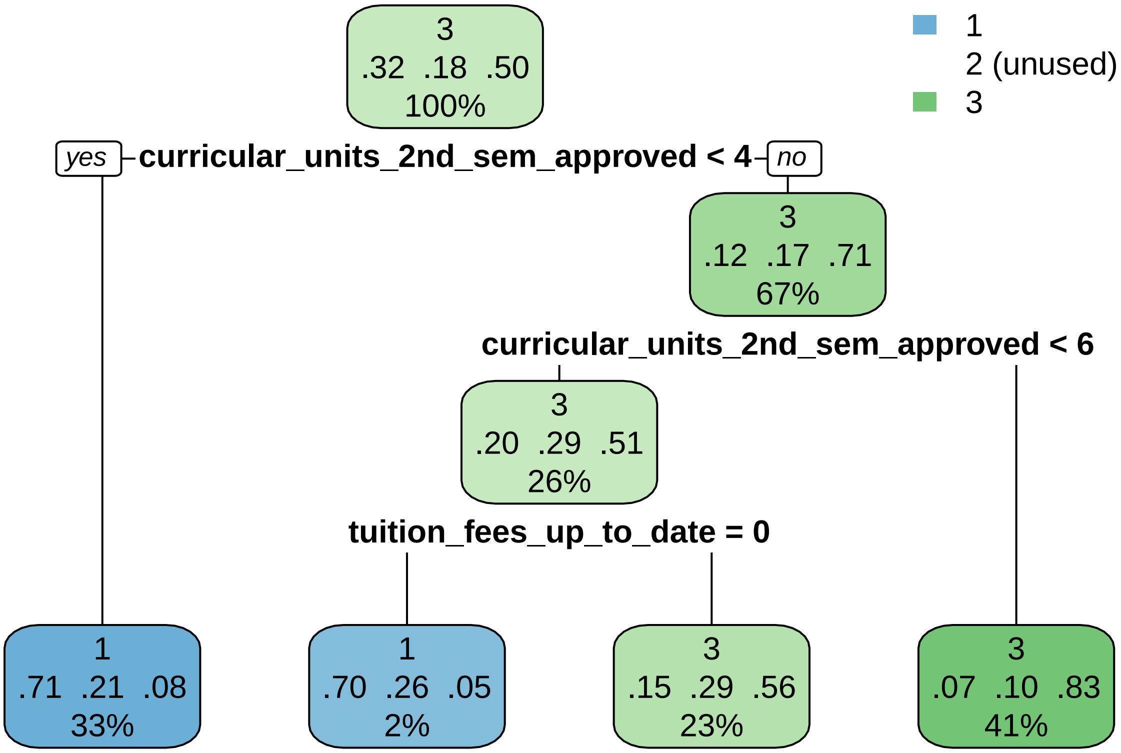
```
table(training_set$target)
```

```
##
##    1    2    3
##  895  522 1415
```

- A concerning mis-classification occurs where **43 students who dropped out were classified as graduates.
- This is problematic because mis-classifying a dropout as a graduate could lead to overlooking at-risk students needing intervention.

## Visualize the Decision Tree

```
rpart.plot(decision_tree_model, type = 2, extra = 104)
```



As seen in the decision tree, the model used two key variables to make it's classifications:

- curricular_units_2nd_sem_approved
- tuition_fees_up_to_date. These splits effectively separated dropouts (Class 1) and graduates (Class 3) but struggled to distinctly classify enrolled (Class 2) students, as evidenced by low proportions of Class 2 in terminal nodes. Unfortunately, this means the model decided the rest of the features were not helpful. This is called overfitting when the model assumes the other variables are just noise and not worth listening to.

Practically, this finding tells counselors to focus on students who are not approved for at least 4 credits in their second semester, and who do not have their tuition fees up to date as these students are most likely to be dropout.

## Model #2: Gradient Boosted Trees with Cross-Validation

In order to account for the class imbalance, gradient boosted trees with crass-validation will be employed below. Boosted trees increase the number of iterations and samplings for enrolled AKA Target =2. Additionally, to account for the many features, this model uses cross validation as the method.

```r
# Define training control for cross-validation
train_control_gbt <- trainControl(
  method = "cv",           # k-fold cross-validation
  number = 5,              # folds
  verboseIter = TRUE
)

# Define the grid of hyperparameters to tune
gbt_grid <- expand.grid(
  n.trees = seq(100, 700, by = 100),  # Number of boosting iterations
  interaction.depth = c(1, 2, 3),     # Max depth of each tree
  shrinkage = c(0.05, 0.075, 0.1),    # Learning rate
  n.minobsinnode = c(5, 10)           # Minimum number of observations in a node
)
```

With the parameters and controls set for the Gradient Boosted Trees (GBT) Model, the following food trains the model by searching for the best parameters within the ranges set above.

```
# Train the Gradient Boosted Trees model (This will take a few minutes)
set.seed(123)
gbt_model <- train(
  target ~ .,
  data = training_set,
  method = "gbm",
  trControl = train_control_gbt,
  tuneGrid = gbt_grid,
  verbose = FALSE
)
```

Thanks to cross validations 5 folds, the model is trained on 4 folds and validated on the 5th. This ensures that all data is used for both training and validation. Boosting re-weights categories that are mis-classified within the folds to handle class imbalance (e.g., Target = 2)

The following code prints what the trained model above found to be optimal

```
# Print the best model parameters from cross-validation
cat("Best Model Parameters from Cross-Validation:\n")
```

```
## Best Model Parameters from Cross-Validation:
```

```
print(gbt_model$bestTune)
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 66     300                 2     0.075             10
```

With the best parameters identified, the predict function below from the caret package will automatically take these parameters and apply them to the dataset we choose. In this case, below the parameters are applied to the validation set that was created from the traindata.

```
## Predict on the validation set
gbt_predictions <- predict(gbt_model, validation_set)

## Evaluate the model
gbt_conf_matrix <- confusionMatrix(gbt_predictions, validation_set$target)
print(gbt_conf_matrix)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2   3
##          1 173  23  14
##          2  30  52  14
##          3  26  50 325
## 
## Overall Statistics
## 
##                Accuracy : 0.7779
##                  95% CI : (0.7455, 0.8081)
##     No Information Rate : 0.4993
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                    Kappa : 0.6278
##
##  Mcnemar's Test P-Value : 1.721e-05
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.7555  0.41600   0.9207
## Specificity            0.9226  0.92440   0.7853
## Pos Pred Value         0.8238  0.54167   0.8105
## Neg Pred Value         0.8873  0.88052   0.9085
## Prevalence             0.3239  0.17680   0.4993
## Detection Rate         0.2447  0.07355   0.4597
## Detection Prevalence   0.2970  0.13579   0.5672
## Balanced Accuracy      0.8390  0.67020   0.8530
```

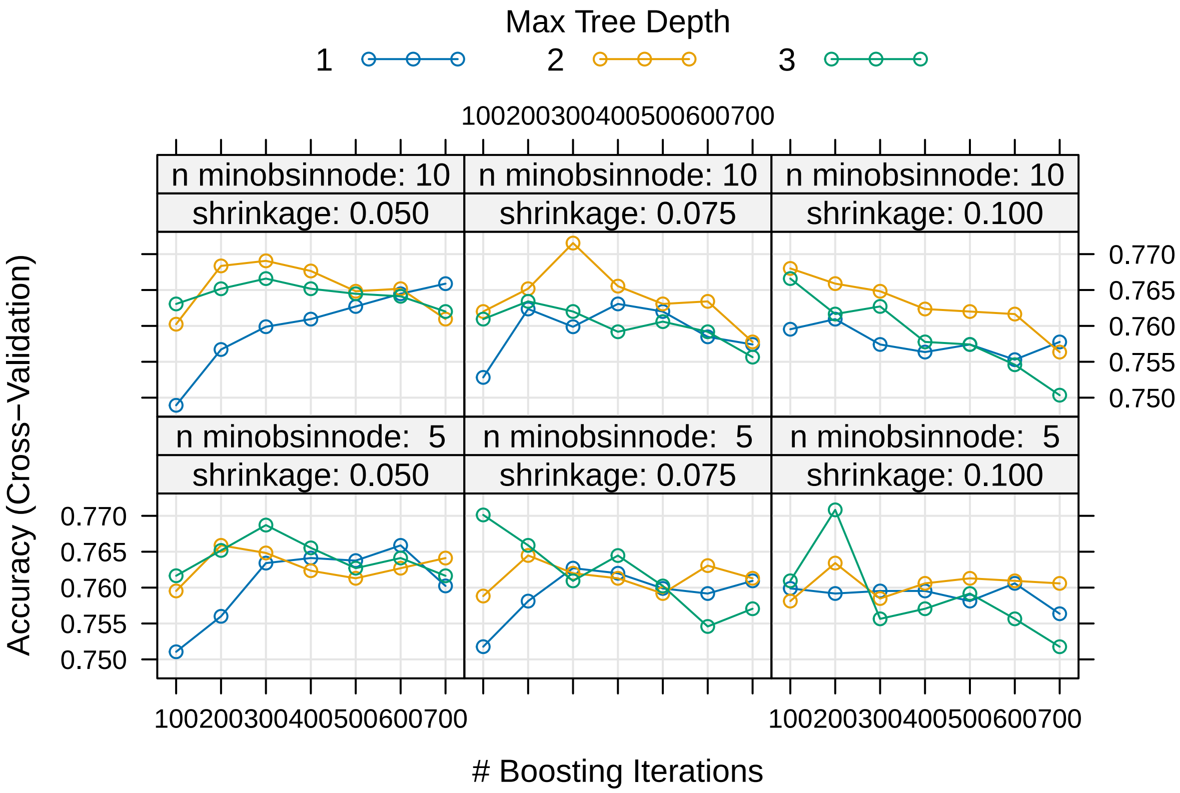
While the GBT model shows improved performance in predicting the "enrolled" class (Class 2) compared to the Decision Tree, it still struggled, as evidenced by its sensitivity for Class 2 being only 38.4%. This suggests that the GBT model use of additional trees handles some aspects of class imbalance through its iterative learning process, although not with 100% success.

The GBT model maintained strong sensitivity for Class 1 (dropout) at 75.98% and improved precision for this class (Positive Predictive Value: 84.88%) compared to the Decision Tree. As a result, the GBT model achieves a good balance of sensitivity and precision for the critical drop out class. GBT implementation already demonstrates reasonable performance for sensitivity.

For Class 2 (Enrolled), the GPT model achieving a sensitivity of 38.40%. While this is still relatively low, it is a significant improvement over the Decision Tree model, which failed to predict any enrolled students. The specificity was 92.44%, and the positive predictive value was 52.17%.

For Class 3 (Graduate), the sensitivity was 92.07%, with a specificity of 75.99% and a positive predictive value of 79.27%. These results are comparable to those of the Decision Tree model, with a slight decrease in sensitivity but improved specificity and precision.

```r
plot(gbt_model)
```



The plot shows that boosting iterations (x-axis) did greatly impact accuracy (as seen on y axis) for all three outcomes (dropout=1, enrolled =2, graduated =3). Cross validation automatically picked the best tune that optimizes for the best outcomes across all 2. The chart shows that the three values align best around 200 trees, a depth of 3, shrinkage of .05 and 10 observations per node.

Overall, the GBT model outperformed the Decision Tree model, particularly in its ability to predict the enrolled class. The improved sensitivity and positive predictive value for enrolled students suggest that the GBT model is better at capturing the nuances in the data that distinguish this class. This enhancement may be attributed to the GBT model's ability to model complex interactions and its robustness against overfitting due to cross-validation.

## Results

With two models developed and tested on the trainingsets' partitioned evaluation set, this section now tests each model on the final_holdout_set. Until this point, this dataset has not been inspected or in any way used to inform the developed of the GBT or Decision tree models.

Starting by applying the feature engineering function and NA check to the final holdout set:

```
# Step 1: Feature Engineering
#apply feature engineering function to final_holdout_set
final_holdout_set <- feature_engineering(final_holdout_set)
# Apply the feature engineering validation function
validation_holdout_results <- validate_features(final_holdout_set)
```

```
## Total NA Count in Dataset: 8
## [1] "NA Counts for Each Feature:"
## # A tibble: 63 x 2
##    Feature                              NA_Count
##    <chr>                                   <int>
##  1 prior_std_grade_1st_sem             4
##  2 prior_std_grade_2nd_sem             4
##  3 marital_status                      0
##  4 application_mode                    0
##  5 application_order                   0
##  6 course                              0
##  7 daytime_evening_attendance          0
##  8 previous_qualification              0
##  9 previous_qualification_grade        0
## 10 nacionality                         0
## # i 53 more rows
## No features have 100% 0s or 1s.
```

```
# There are 8 NAs. A negligable amount. So they are dropped
cat("Total NA Count in final_holdout_set  before removal:", sum(is.na(final_holdout_set)), "\n")
```

```
## Total NA Count in final_holdout_set  before removal: 8
```

```
final_holdout_set <- final_holdout_set %>% drop_na()
cat("Total NA Count in final_holdout_set after removal:", sum(is.na(final_holdout_set)), "\n")
```

```
## Total NA Count in final_holdout_set after removal: 0
```

```
# Step 2: Set Target as factor
# Ensure the target variable in the final_holdout_set is a factor
final_holdout_set$target <- as.factor(final_holdout_set$target)
```

With the final holdout set now having the same treatment as the traindata set from which the models were developed, predictions can be made.

```
# Predict with Decision Tree Model
dt_holdout_predictions <- predict(decision_tree_model, final_holdout_set, type = "class")
dt_holdout_conf_matrix <- confusionMatrix(dt_holdout_predictions, final_holdout_set$target)
cat("Decision Tree Results on Final Holdout Set:\n")
```

```
## Decision Tree Results on Final Holdout Set:
```

```
print(dt_holdout_conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 226  62  23
##          2   0   0   0
##          3  67  85 417
##
## Overall Statistics
##
##                Accuracy : 0.7307
##                  95% CI : (0.7001, 0.7597)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5182
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.7713    0.000   0.9477
## Specificity            0.8552    1.000   0.6545
## Pos Pred Value         0.7267      NaN   0.7329
## Neg Pred Value         0.8822    0.833   0.9260
## Prevalence             0.3330    0.167   0.5000
## Detection Rate         0.2568    0.000   0.4739
## Detection Prevalence   0.3534    0.000   0.6466
## Balanced Accuracy      0.8133    0.500   0.8011
```

```
# Predict with Gradient Boosted Trees Model
gbt_holdout_predictions <- predict(gbt_model, final_holdout_set)
gbt_holdout_conf_matrix <- confusionMatrix(gbt_holdout_predictions, final_holdout_set$target)
cat("Gradient Boosted Trees Results on Final Holdout Set:\n")
```

```
## Gradient Boosted Trees Results on Final Holdout Set:
```

```
print(gbt_holdout_conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 217  38  27
##          2  39  56  17
##          3  37  53 396
##
## Overall Statistics
##
##                Accuracy : 0.7602
##                  95% CI : (0.7306, 0.7881)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5976
##
##  Mcnemar's Test P-Value : 0.0001626
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.7406  0.38095   0.9000
## Specificity            0.8893  0.92360   0.7955
## Pos Pred Value         0.7695  0.50000   0.8148
## Neg Pred Value         0.8729  0.88151   0.8883
## Prevalence             0.3330  0.16705   0.5000
## Detection Rate         0.2466  0.06364   0.4500
## Detection Prevalence   0.3205  0.12727   0.5523
## Balanced Accuracy      0.8149  0.65228   0.8477
```

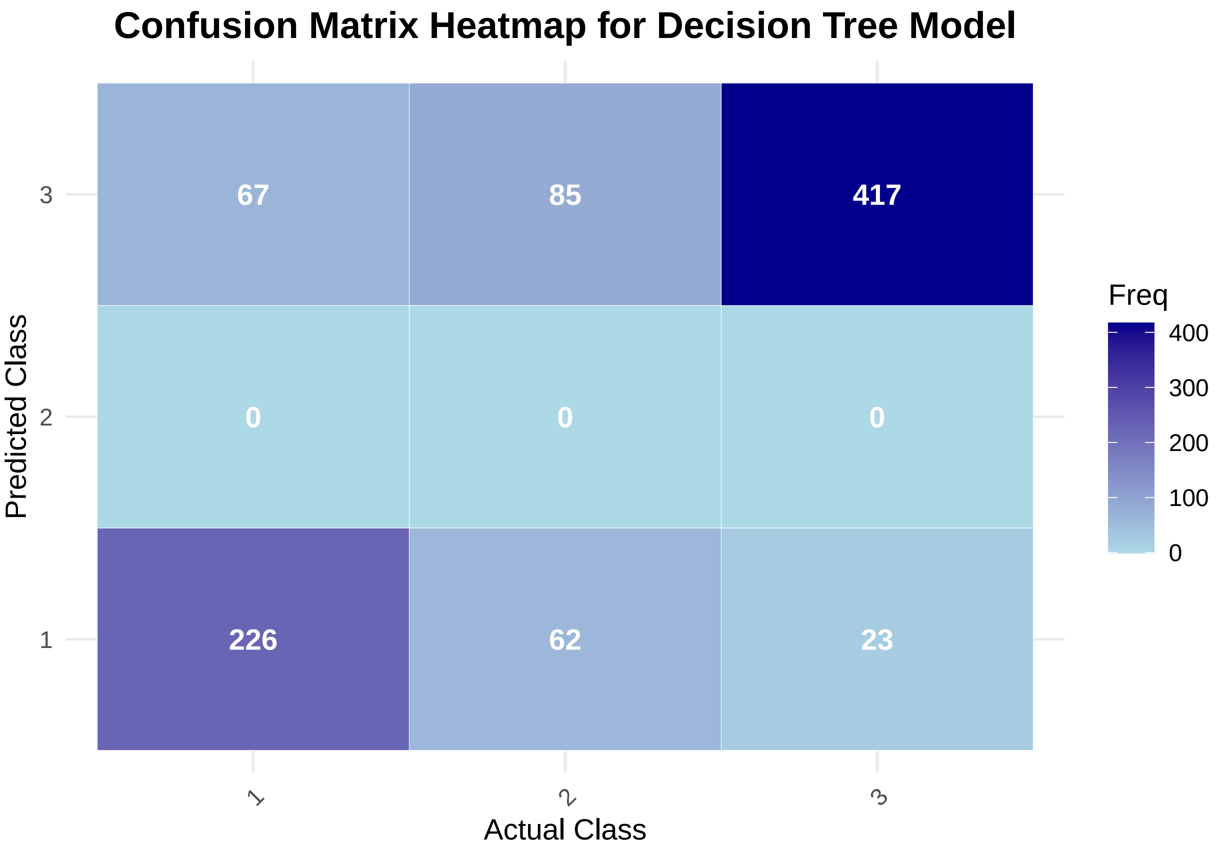
The results above show, in terms of balanced accuracy, the boosted tree model was able to maintain similar performance to the DT model in predicting dropout (GBT = .81 vs DT = .81) and enrolled (GBT = .85 vs DT = .80) while making up 15% in enrolled (GBT = .65 vs DT = .50). This difference is certainly attributed to cross validations and the boosted resampling.

Below, the GBT & Decision Tree models Performance on Final Holdout are visualized.

```r
#Function to plot confusion matrix heatmap
plot_confusion_matrix <- function(conf_matrix, model_name) {
  conf_matrix_df <- as.data.frame(conf_matrix$table)
  ggplot(conf_matrix_df, aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile(color = "white") +
    geom_text(aes(label = Freq), color = "white", fontface = "bold") +
    scale_fill_gradient(low = "lightblue", high = "darkblue") +
    labs(
      title = paste("Confusion Matrix Heatmap for", model_name, "Model"),
      x = "Actual Class",
      y = "Predicted Class"
    ) +
    theme_minimal() +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1),
      plot.title = element_text(size = 14, face = "bold", hjust = 0.5)
    )
}
```

Now visualize the DT model performance on final holdout:

```r
# Visualize Decision Tree Model
dt_conf_matrix_plot <- plot_confusion_matrix(dt_holdout_conf_matrix, "Decision Tree")
print(dt_conf_matrix_plot)
```



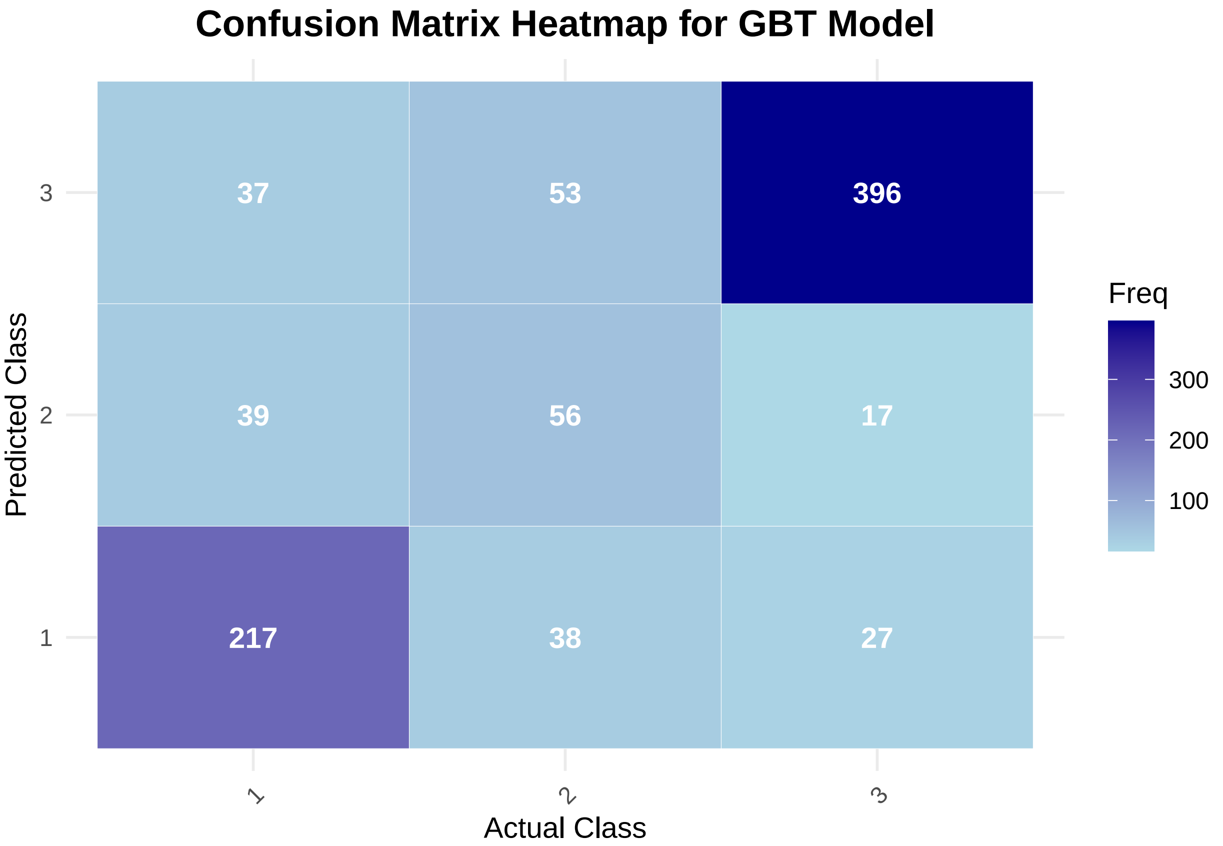
Represented by the darker colors in the top right and bottom left, the model greatly favored enrollment and dropout, and completely missed enrolled. This performance was expected and negligably different from the

DT performance on the validation set:

- validation set results = balanced accuracy of 0.8423 0.5000 0.8008 in Dropout, enrolled, and graduate, respectively. Final_holdout_set results = 0.8133 0.500 0.8011

Now visualize the GBT Model performance:

```
# Visualize Gradient Boosted Trees Model
gbt_conf_matrix_plot <- plot_confusion_matrix(gbt_holdout_conf_matrix, "GBT")
print(gbt_conf_matrix_plot)
```



The GBT heat map shows slightly lower correct predictions for enrolled and dropout compared to DT, but actually made predictions for enrolled, making it the stronger of the two models.

# Conclusion

In conclusion, this report has cleaned the dataset, explored the variables, built new features, and developed two machine learning models to predict student outcomes at university (dropout, enrolled, graduate). The two models developed were decision trees & gradiaent boosted trees. The performance of these two models are mediocre at best. Due to class imbalance, precision/pos predictive value is one of the best measures of success. As shown above, GBT performs much better than decision trees in this regaard (DT = 0.7267 NaN 0.7329) vs (GPT = 0.7695 0.50000 0.8148). This difference is largely due to cross validation and the use of boosted sampling for mis-classified target in it's training. However, with GBT only identified 38% of enrolled students (sensitivity), as a result, the model is left wanting.

On the positive side, enrolled is the least important group for prediction since it is the most easily identified in the present. Notably, accurately predicting 74% of dropouts and 90% (sensitivity) of graduates are strong outcomes and suggests that with further refinement as discussed in the following section greater outcomes can be achieved.

The potential impact of these findings include informing college counselors and advisers to focus early intervention efforts that combat dropout with students who do not have tuition fes up to date and/or are not approved for less than 4 credits. Additionally, pulling all students who were classified as dropout would and exploring their college expereince could yeild qualitative insights into who to support currently enrolled students.

## Limitations

The dataset itself has several limitations that constrain the analysis and predictive modeling. One key limitation is the lack of longitudinal data, which prevents analyzing year-over-year trends in enrollment, graduation, and dropout rates. Incorporating temporal data could reveal how outcomes evolve annually, offering valuable insights for intervention planning.

Another limitation is the absence of information on financial support. Financial aid is a known predictor of student outcomes, but the dataset does not include details about the university's financial aid policies. This restricts the ability to assess how financial support impacts dropout and graduation rates. Including such data in future analyses could provide a more comprehensive understanding.

The dataset also does not clearly map courses to specific degree programs. This missing connection makes it challenging to identify the unique support needs of students pursuing different academic paths. Establishing these relationships could improve the ability to tailor interventions to individual degree programs.

Class imbalance and limited data further challenge model performance. The dataset contains 62 features but only 3539 students, with a significant imbalance in class distribution. For example, only 647 students are currently enrolled (class 2), which complicates the models' ability to learn effectively and increases susceptibility to overfitting.

```r
ncol(traindata)
```

```
## [1] 63
```

```r
nrow(traindata)
```

```
## [1] 3539
```

```r
table(traindata$target)
```

```
##
##    1    2    3
## 1124  647 1768
```

The complexity of the features relative to the dataset size also introduces noise and reduces the generalizability of the models. Dimensionality reduction or feature selection could help mitigate this issue and improve predictive accuracy. Lastly, the high number of features and limited sample size have resulted in signs of overfitting in the decision tree and gradient-boosted tree models. This indicates the need for simpler models, additional data, or regularization techniques to enhance performance on unseen data.

## Future Directions

Many variables in the dataset were not explored further because their predictability in their current form was insufficient to warrant immediate feature development. Variables with coefficients near zero in the regression plot, for instance, were largely left examined. This creates an opportunity to develop new features, particularly demographic ones, such as more detailed descriptions of parent occupation and educational qualifications, which could provide better insights into factors that increase or decrease the likelihood of success in university.

Class imbalance remains a key challenge in the analysis. While some strategies were employed, such as cross-validation in gradient-boosted trees, additional methods like SMOTE (Synthetic Minority Oversampling Technique) or weighted loss functions could further enhance performance for minority classes. Neither the decision tree nor gradient-boosted tree models explicitly utilized techniques such as oversampling, undersampling, or class-weighted learning, leaving room for improvement in handling imbalanced data.

Future research could also explore elastic net or lasso regression to strategically select the most relevant features for predictive modeling. Another potential avenue is reframing the problem as a two-class classification task, focusing solely on distinguishing between graduates and dropouts. This simplified approach could yield more actionable insights for supporting at-risk students.

Additionally, a probabilistic approach, such as multinomial logistic regression, could provide counselors and stakeholders with actionable probabilities of students falling into dropout, enrollment, or graduation categories. Such a model would equip decision-makers with the nuanced information needed to prioritize early interventions and support strategies effectively.

Lastly, since the current data only identifies students who have already dropped out, a next step would include using this model to identify similar students who are currently enrolled and performing early intervention activities to hopefully ensure their outcomes are more positive.

# Citations

- Realinho, V., Vieira Martins, M., Machado, J., & Baptista, L. (2021). Predict Students' Dropout and Academic Success [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5MC89.
- U.S. Department of Education, National Center for Education Statistics, Integrated Postsecondary Education Data System (IPEDS), Winter 2020–21, Graduation Rates component. See Digest of Education Statistics 2021, table 326.20. Retreieved on 11/25/24 from https://nces.ed.gov/fastfacts/display.asp?id=40
- US Department of Health and Human Services. (2023). Enrollment in higher education. Enrollment in Higher Education - Healthy People 2030. https://odphp.health.gov/healthypeople/priority-areas/social-determinants-health/literature-summaries/enrollment-higher-education