Assignment3_FIT5145

Khoi

23/05/2025

The dataset that I use in this analysis is “Predict Students’ Dropout and Academic Success” was sourced from the UC Irvine Machine Learning Repository. It contains information on 4,424 students enrolled in various programs such as education, nursing. With 36 features, including demographic, academic, and institutional variables, the dataset supports classification tasks aimed at identifying students at risk of dropping out.

Data source link: https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success (https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success)

```
library(readr)
```

```
# Load the dataset
student_data <- read_delim("data.csv", delim = ";")
```

```
## Rows: 4424 Columns: 37
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ";"
## chr  (1): Target
## dbl (36): Marital status, Application mode, Application order, Course, Dayti...
##
## 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.
```

```
nrow(student_data) #Inspect number of rows of dataset
```

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

```
ncol(student_data) #Inspect number of columns of dataset
```

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

```
unique(sapply(student_data, class)) #View unique values of class
```

```
## [1] "numeric"   "character"
```

```
# View the first few rows
head(student_data)
```

```
## # A tibble: 6 × 37
##   `Marital status` `Application mode` `Application order` Course
##              <dbl>              <dbl>               <dbl>  <dbl>
## 1                1                 17                   5    171
## 2                1                 15                   1   9254
## 3                1                  1                   5   9070
## 4                1                 17                   2   9773
## 5                2                 39                   1   8014
## 6                2                 39                   1   9991
## # ℹ 33 more variables: `Daytime/evening attendance\t` <dbl>,
## #   `Previous qualification` <dbl>, `Previous qualification (grade)` <dbl>,
## #   Nacionality <dbl>, `Mother's qualification` <dbl>,
## #   `Father's qualification` <dbl>, `Mother's occupation` <dbl>,
## #   `Father's occupation` <dbl>, `Admission grade` <dbl>, Displaced <dbl>,
## #   `Educational special needs` <dbl>, Debtor <dbl>,
## #   `Tuition fees up to date` <dbl>, Gender <dbl>, …
```

Clean and Preprocess Data

```
# Count missing values
colSums(is.na(student_data))
```

```
##                                    Marital status
##                                                 0
##                                  Application mode
##                                                 0
##                                 Application order
##                                                 0
##                                            Course
##                                                 0
##                      Daytime/evening attendance\t
##                                                 0
##                            Previous qualification
##                                                 0
##                    Previous qualification (grade)
##                                                 0
##                                       Nacionality
##                                                 0
##                            Mother's qualification
##                                                 0
##                            Father's qualification
##                                                 0
##                               Mother's occupation
##                                                 0
##                               Father's occupation
##                                                 0
##                                   Admission grade
##                                                 0
##                                         Displaced
##                                                 0
##                         Educational special needs
##                                                 0
##                                            Debtor
##                                                 0
##                           Tuition fees up to date
##                                                 0
##                                            Gender
##                                                 0
##                                Scholarship holder
##                                                 0
##                                 Age at enrollment
##                                                 0
##                                     International
##                                                 0
##               Curricular units 1st sem (credited)
##                                                 0
##               Curricular units 1st sem (enrolled)
##                                                 0
##            Curricular units 1st sem (evaluations)
##                                                 0
##               Curricular units 1st sem (approved)
##                                                 0
##                  Curricular units 1st sem (grade)
##                                                 0
## Curricular units 1st sem (without evaluations)
##                                                 0
##               Curricular units 2nd sem (credited)
```

```
##                                           0
##            Curricular units 2nd sem (enrolled)
##                                           0
##         Curricular units 2nd sem (evaluations)
##                                           0
##            Curricular units 2nd sem (approved)
##                                           0
##               Curricular units 2nd sem (grade)
##                                           0
## Curricular units 2nd sem (without evaluations)
##                                           0
##                              Unemployment rate
##                                           0
##                                 Inflation rate
##                                           0
##                                            GDP
##                                           0
##                                         Target
##                                           0
```

```
# Find fully duplicated rows
duplicated_rows <- student_data[duplicated(student_data), ]

# View how many duplicated rows
nrow(duplicated_rows)
```

```
## [1] 0
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
student_data <- student_data %>%
  rename(Nationality = Nacionality) #Corrects the column name from the incorrectly sp
elled
```

Basic Exploratory Data Analysis (EDA)

```
# Class distribution
student_data %>%
  count(Target) %>% #Count number of records for each unique class in the 'Target' co
lumn
  mutate(percentage = n / sum(n) * 100) #Calculate percentage for each class
```

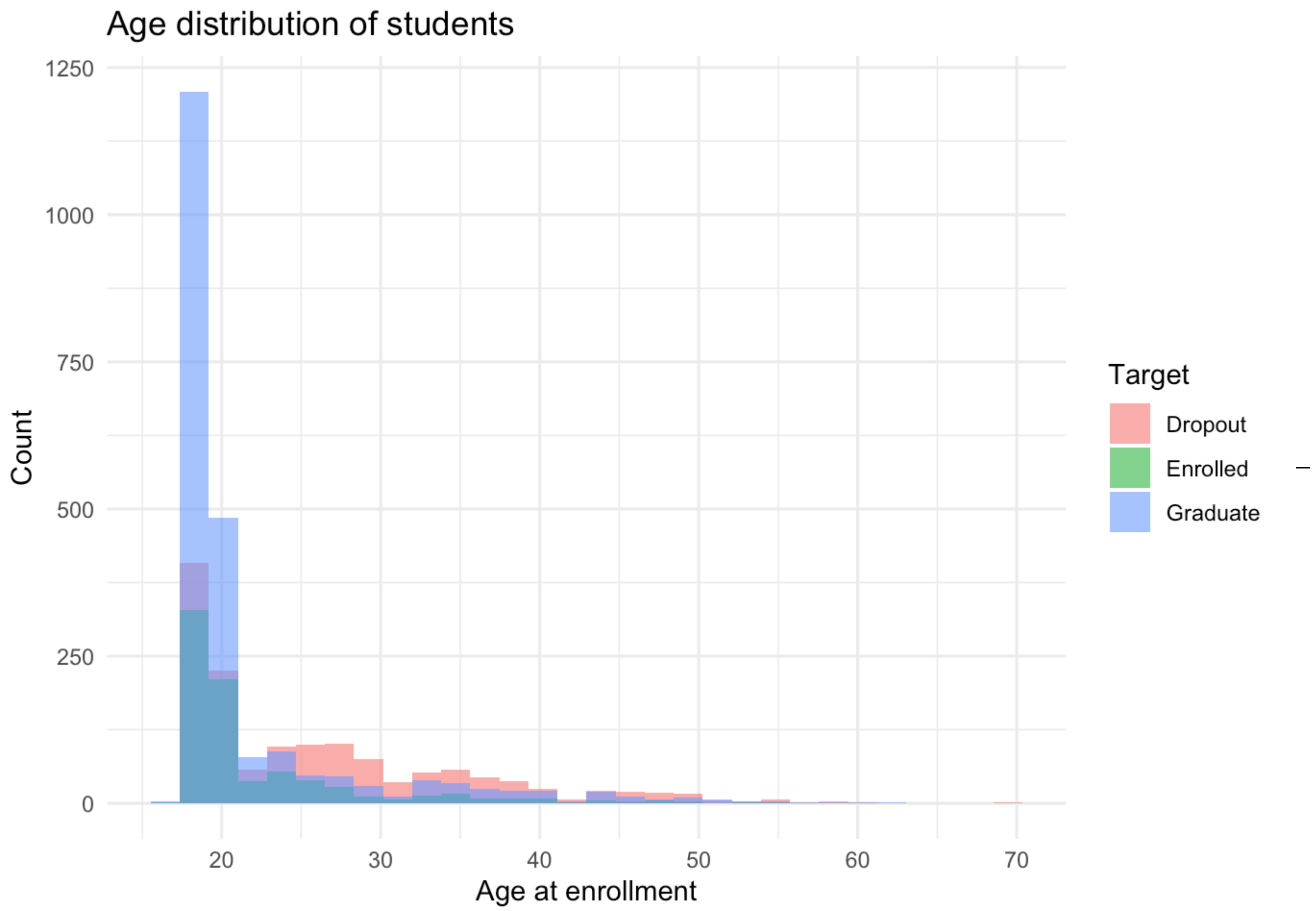
```
## # A tibble: 3 × 3
##   Target       n percentage
##   <chr>    <int>      <dbl>
## 1 Dropout   1421       32.1
## 2 Enrolled   794       17.9
## 3 Graduate  2209       49.9
```

```
summary(student_data) # Provides statistical summary of each column in the 'student_d
ata' dataset
```

```
##  Marital status  Application mode Application order     Course
##  Min.   :1.000   Min.   : 1.00    Min.   :0.000     Min.   :  33
##  1st Qu.:1.000   1st Qu.: 1.00    1st Qu.:1.000     1st Qu.:9085
##  Median :1.000   Median :17.00    Median :1.000     Median :9238
##  Mean   :1.179   Mean   :18.67    Mean   :1.728     Mean   :8857
##  3rd Qu.:1.000   3rd Qu.:39.00    3rd Qu.:2.000     3rd Qu.:9556
##  Max.   :6.000   Max.   :57.00    Max.   :9.000     Max.   :9991
##  Daytime/evening attendance\t Previous qualification
##  Min.   :0.0000              Min.   : 1.000
##  1st Qu.:1.0000              1st Qu.: 1.000
##  Median :1.0000              Median : 1.000
##  Mean   :0.8908              Mean   : 4.578
##  3rd Qu.:1.0000              3rd Qu.: 1.000
##  Max.   :1.0000              Max.   :43.000
##  Previous qualification (grade)  Nationality      Mother's qualification
##  Min.   : 95.0                  Min.   :  1.000   Min.   : 1.00
##  1st Qu.:125.0                  1st Qu.:  1.000   1st Qu.: 2.00
##  Median :133.1                  Median :  1.000   Median :19.00
##  Mean   :132.6                  Mean   :  1.873   Mean   :19.56
##  3rd Qu.:140.0                  3rd Qu.:  1.000   3rd Qu.:37.00
##  Max.   :190.0                  Max.   :109.000   Max.   :44.00
##  Father's qualification Mother's occupation Father's occupation Admission grade
##  Min.   : 1.00          Min.   :  0.00      Min.   :  0.00      Min.   : 95.0
##  1st Qu.: 3.00          1st Qu.:  4.00      1st Qu.:  4.00      1st Qu.:117.9
##  Median :19.00          Median :  5.00      Median :  7.00      Median :126.1
##  Mean   :22.28          Mean   : 10.96      Mean   : 11.03      Mean   :127.0
##  3rd Qu.:37.00          3rd Qu.:  9.00      3rd Qu.:  9.00      3rd Qu.:134.8
##  Max.   :44.00          Max.   :194.00      Max.   :195.00      Max.   :190.0
##    Displaced      Educational special needs     Debtor
##  Min.   :0.0000   Min.   :0.00000           Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000           1st Qu.:0.0000
##  Median :1.0000   Median :0.00000           Median :0.0000
##  Mean   :0.5484   Mean   :0.01153           Mean   :0.1137
##  3rd Qu.:1.0000   3rd Qu.:0.00000           3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.00000           Max.   :1.0000
##  Tuition fees up to date     Gender      Scholarship holder Age at enrollment
##  Min.   :0.0000          Min.   :0.0000   Min.   :0.0000     Min.   :17.00
##  1st Qu.:1.0000          1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:19.00
##  Median :1.0000          Median :0.0000   Median :0.0000     Median :20.00
##  Mean   :0.8807          Mean   :0.3517   Mean   :0.2484     Mean   :23.27
##  3rd Qu.:1.0000          3rd Qu.:1.0000   3rd Qu.:0.0000     3rd Qu.:25.00
##  Max.   :1.0000          Max.   :1.0000   Max.   :1.0000     Max.   :70.00
##  International     Curricular units 1st sem (credited)
##  Min.   :0.00000   Min.   : 0.00
##  1st Qu.:0.00000   1st Qu.: 0.00
##  Median :0.00000   Median : 0.00
##  Mean   :0.02486   Mean   : 0.71
##  3rd Qu.:0.00000   3rd Qu.: 0.00
##  Max.   :1.00000   Max.   :20.00
##  Curricular units 1st sem (enrolled) Curricular units 1st sem (evaluations)
##  Min.   : 0.000                      Min.   : 0.000
##  1st Qu.: 5.000                      1st Qu.: 6.000
##  Median : 6.000                      Median : 8.000
##  Mean   : 6.271                      Mean   : 8.299
##  3rd Qu.: 7.000                      3rd Qu.:10.000
```

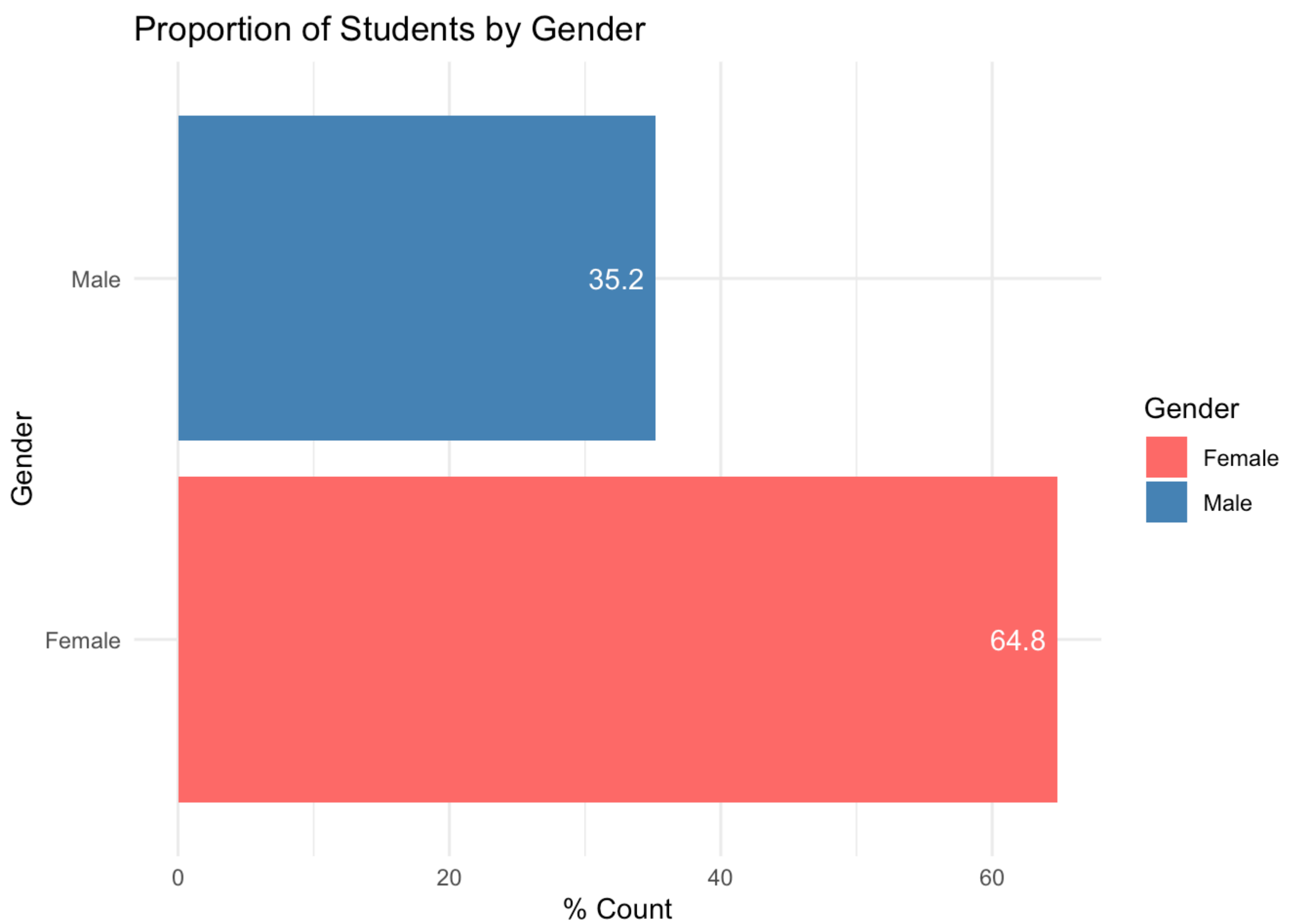
```
## Max.   :26.000                     Max.   :45.000
## Curricular units 1st sem (approved) Curricular units 1st sem (grade)
## Min.   : 0.000                      Min.   : 0.00
## 1st Qu.: 3.000                      1st Qu.:11.00
## Median : 5.000                      Median :12.29
## Mean   : 4.707                      Mean   :10.64
## 3rd Qu.: 6.000                      3rd Qu.:13.40
## Max.   :26.000                      Max.   :18.88
## Curricular units 1st sem (without evaluations)
## Min.   : 0.0000
## 1st Qu.: 0.0000
## Median : 0.0000
## Mean   : 0.1377
## 3rd Qu.: 0.0000
## Max.   :12.0000
## Curricular units 2nd sem (credited) Curricular units 2nd sem (enrolled)
## Min.   : 0.0000                     Min.   : 0.000
## 1st Qu.: 0.0000                     1st Qu.: 5.000
## Median : 0.0000                     Median : 6.000
## Mean   : 0.5418                     Mean   : 6.232
## 3rd Qu.: 0.0000                     3rd Qu.: 7.000
## Max.   :19.0000                     Max.   :23.000
## Curricular units 2nd sem (evaluations) Curricular units 2nd sem (approved)
## Min.   : 0.000                         Min.   : 0.000
## 1st Qu.: 6.000                         1st Qu.: 2.000
## Median : 8.000                         Median : 5.000
## Mean   : 8.063                         Mean   : 4.436
## 3rd Qu.:10.000                         3rd Qu.: 6.000
## Max.   :33.000                         Max.   :20.000
## Curricular units 2nd sem (grade)
## Min.   : 0.00
## 1st Qu.:10.75
## Median :12.20
## Mean   :10.23
## 3rd Qu.:13.33
## Max.   :18.57
## Curricular units 2nd sem (without evaluations) Unemployment rate
## Min.   : 0.0000                                Min.   : 7.60
## 1st Qu.: 0.0000                                1st Qu.: 9.40
## Median : 0.0000                                Median :11.10
## Mean   : 0.1503                                Mean   :11.57
## 3rd Qu.: 0.0000                                3rd Qu.:13.90
## Max.   :12.0000                                Max.   :16.20
## Inflation rate        GDP               Target
## Min.   :-0.800   Min.   :-4.060000   Length:4424
## 1st Qu.: 0.300   1st Qu.:-1.700000   Class :character
## Median : 1.400   Median : 0.320000   Mode  :character
## Mean   : 1.228   Mean   : 0.001969
## 3rd Qu.: 2.600   3rd Qu.: 1.790000
## Max.   : 3.700   Max.   : 3.510000
```

```
# Plot histogram of age at enrollment, colored by student status (Target)
library(ggplot2)
ggplot(student_data, aes(x = `Age at enrollment`, fill = Target)) +
  geom_histogram(position = "identity", alpha = 0.6, bins = 30) +
  labs(title = "Age distribution of students", x = "Age at enrollment", y = "Count")
+
  theme_minimal()
```



The distribution is right-skewed, indicating that the majority of students enrolled in higher education are in their late teens to early 20s, with a peak around age 18–20. _ As age increases beyond the early 20s, the proportion of dropouts increases noticeably, especially among students in their mid-20s to early 30s. _ Graduation is most common among younger students, suggesting that earlier enrollment may be associated with better academic outcomes.

```
### Bar chart of proportion of student distribution for gender
ggplot(
  student_data %>%
    count(Gender) %>%
    mutate(
      Gender = factor(Gender, levels = c(0, 1), labels = c("Female", "Male")), # Conv
ert to labelled factor
      Percent = round(n / sum(n) * 100, 1) # Calculate percentage and round to 1 deci
mal place
    ),
  aes(x = Percent, y = Gender, fill = Gender)
) +
  geom_col() +
  geom_text(aes(label = Percent), hjust = 1.2, color = "white") + # Add percentage la
bels inside bars
  scale_fill_manual(values = c("Female" = "indianred1", "Male" = "steelblue")) +
  labs(
    title = "Proportion of Students by Gender",
    x = "% Count",
    y = "Gender"
  ) +
  theme_minimal()
```

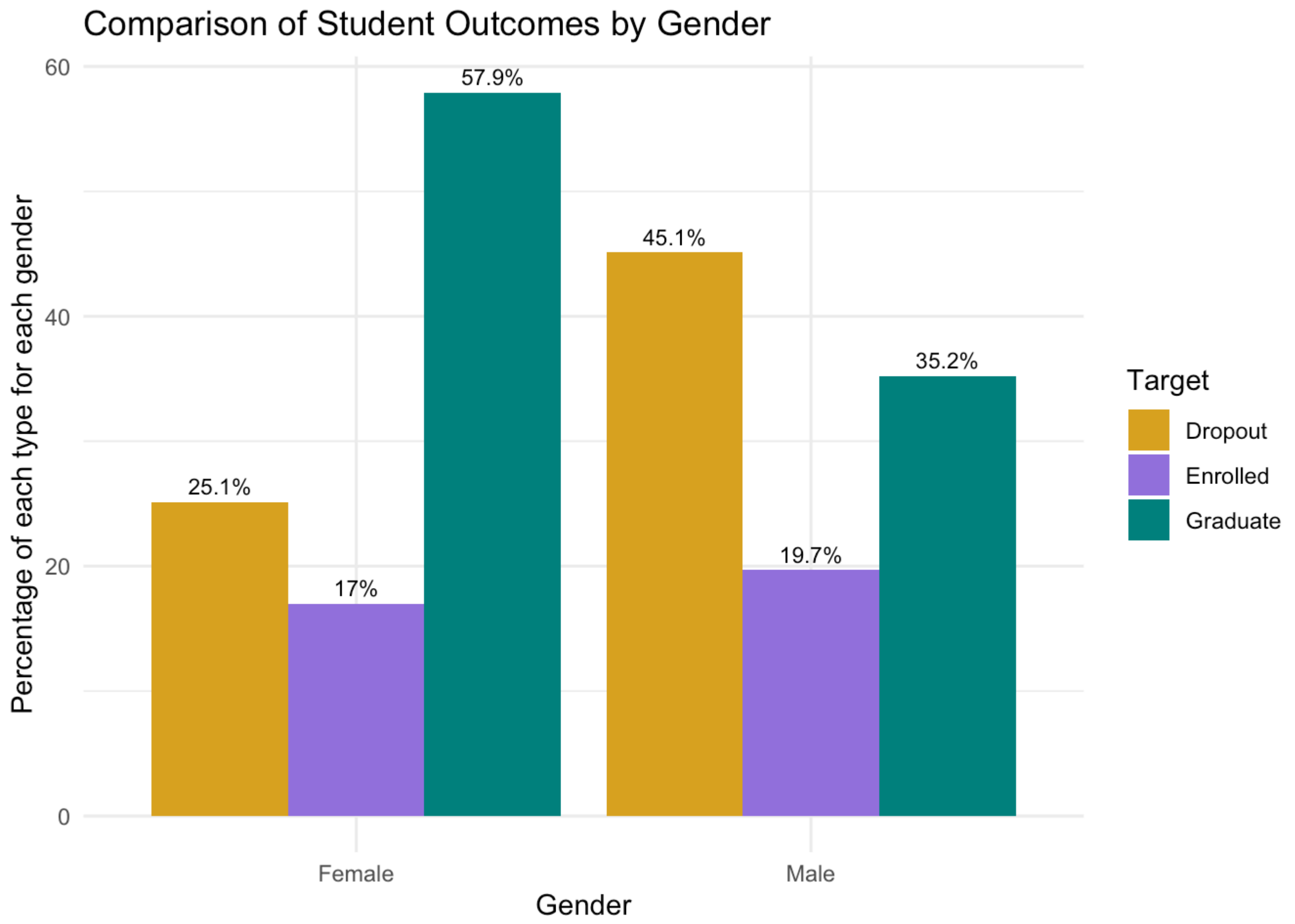


Females make up the majority of the dataset, accounting for approximately 64.8% of the students, while males comprise 35.2%

```
#cross-tabulation of gender and student outcomes (Target)
gender_target_summary <- student_data %>%
  mutate(Gender = factor(Gender, levels = c(0, 1), labels = c("Female", "Male"))) %>%
# Convert Gender to labeled factor
  group_by(Gender, Target) %>% # Group by Gender and Target class
  summarise(Count = n(), .groups = "drop") # Count observations per group and drop gr
ouping structure

print(gender_target_summary)
```

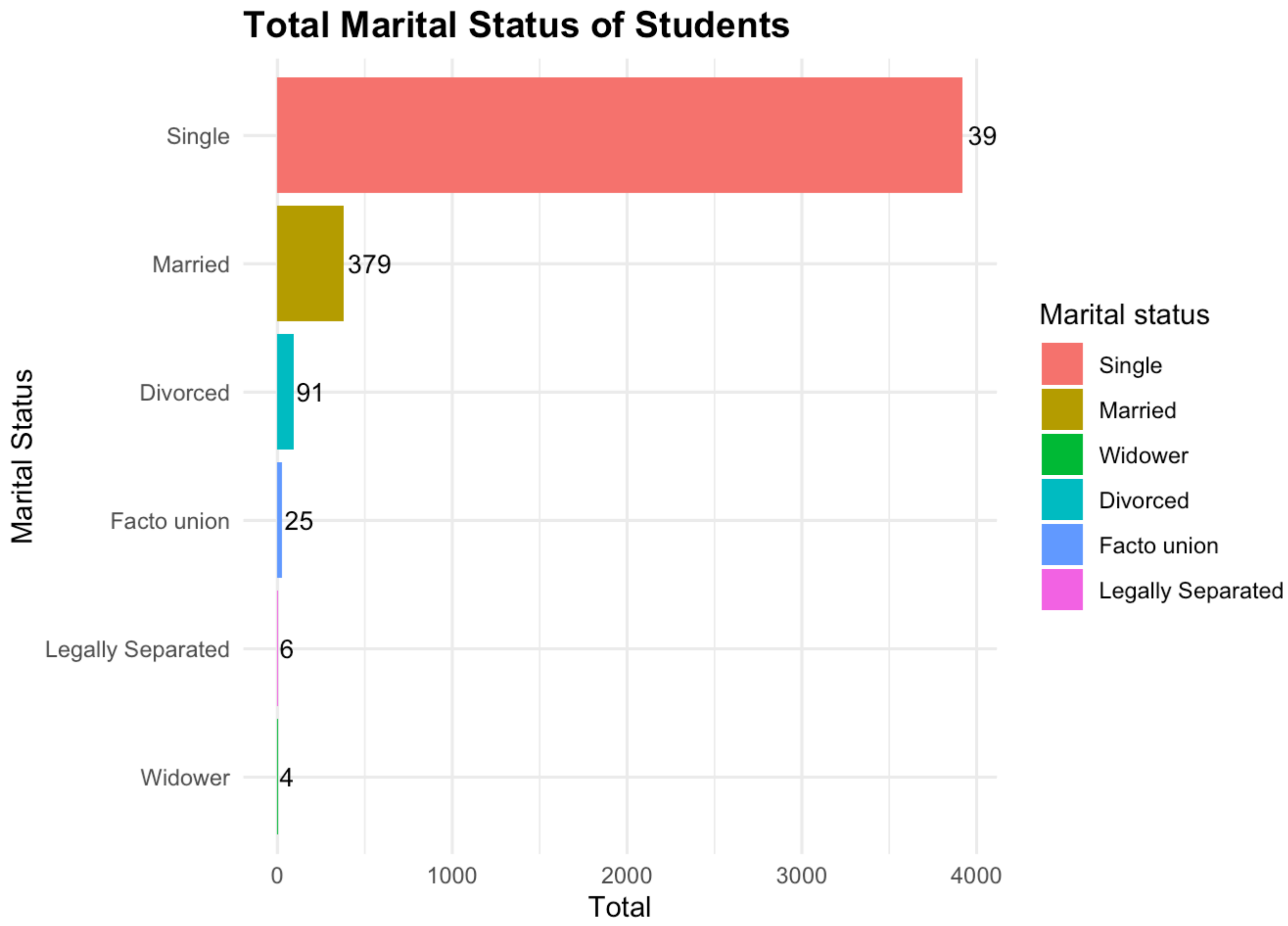
```
## # A tibble: 6 × 3
##   Gender Target   Count
##   <fct>  <chr>    <int>
## 1 Female Dropout    720
## 2 Female Enrolled   487
## 3 Female Graduate  1661
## 4 Male   Dropout    701
## 5 Male   Enrolled   307
## 6 Male   Graduate   548
```

```
# Summarise count of each target by gender in percentages
gender_target_percent <- gender_target_summary %>%
  group_by(Gender) %>%
  mutate(Percent = round(Count / sum(Count) * 100, 1))

# Plot grouped bar chart of percentages
ggplot(gender_target_percent, aes(x = Gender, y = Percent, fill = Target)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(Percent, "%")),
          position = position_dodge(width = 0.9),
          vjust = -0.5, size = 3) +
  labs(
    title = "Comparison of Student Outcomes by Gender",
    x = "Gender", y = "Percentage of each type for each gender"
  ) + #Custom color mapping for each Target class
  scale_fill_manual(values = c(
    "Dropout" = "goldenrod",
    "Enrolled" = "mediumpurple",
    "Graduate" = "#008080"
  )) +
  theme_minimal()
```



From the plot, we have: Around 57.9% of female students graduated, compared to only 35.2% of male students. This suggests female students are more likely to complete their studies. In addition, 45.1% of male students dropped out, significantly more than 25.1% of females. This indicates male students may be at greater risk of not completing their programs. There are 19.7% of males are still enrolled and 17% of females. This is a smaller difference but might suggest slightly slower progression for male students.

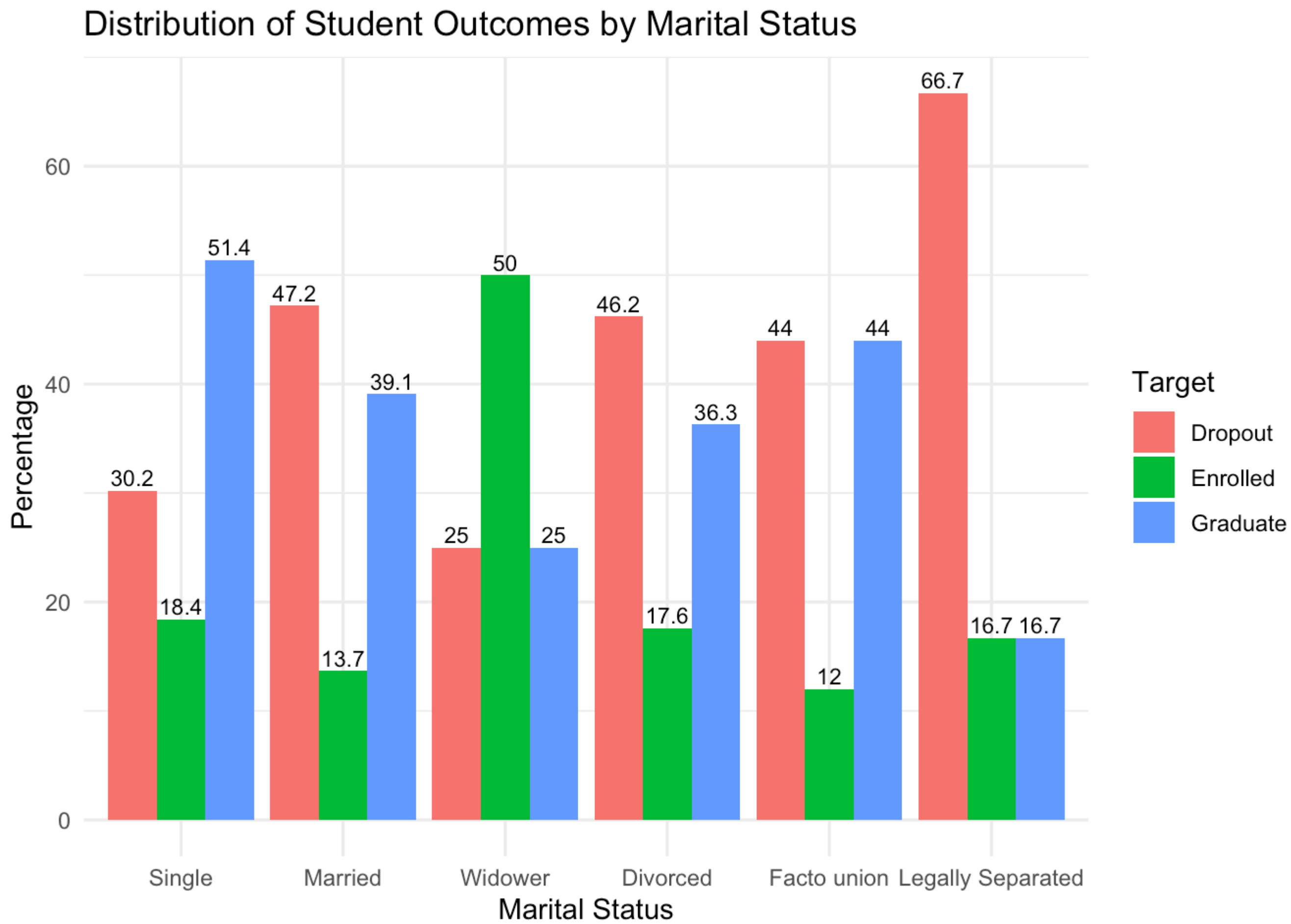
```
# Summarise marital status
marital_status_counts <- student_data %>%
  mutate(`Marital status` = factor(`Marital status`, levels = c(1, 2, 3, 4, 5, 6),
                                   labels = c("Single", "Married", "Widower", "Divorc
ed", "Facto union", "Legally Separated"))) %>%
  count(`Marital status`) # Count occurrences of each marital status

# Plot horizontal bar chart
ggplot(marital_status_counts, aes(x = n, y = reorder(`Marital status`, n), fill = `Ma
rital status`)) +
  geom_col() +
  geom_text(aes(label = n), hjust = -0.1, size = 3.5) +
  labs(
    title = "Total Marital Status of Students",
    x = "Total",
    y = "Marital Status"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```




```
# Calculate the percentage of each student outcome (Target) within each marital statu
s group
student_data_grouped <- student_data %>%
  mutate(`Marital status` = factor(`Marital status`,
                                   levels = c(1, 2, 3, 4, 5, 6),
                                   labels = c("Single", "Married", "Widower", "Divorc
ed", "Facto union", "Legally Separated"))) %>%
  group_by(`Marital status`, Target) %>%
  summarise(Count = n()) %>%
  mutate(Percent = round(Count / sum(Count) * 100, 1))  # percent within marital grou
p
```

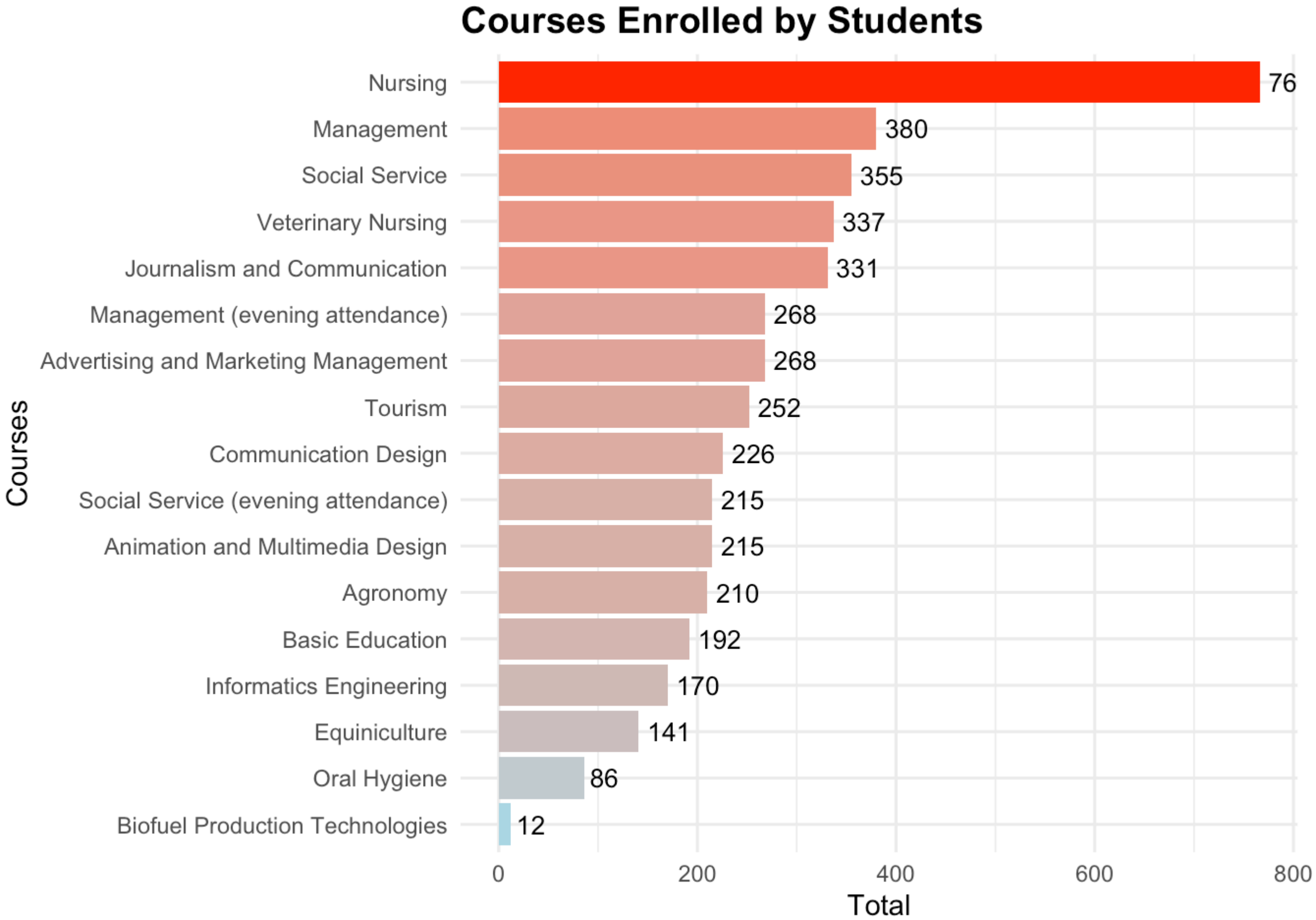
```
## `summarise()` has grouped output by 'Marital status'. You can override using
## the `.groups` argument.
```

```
# Plot grouped bar chart showing the distribution of student outcomes by marital stat
us
ggplot(student_data_grouped, aes(x = `Marital status`, y = Percent, fill = Target)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_text(aes(label = Percent),
            position = position_dodge(width = 0.9),
            vjust = -0.3, size = 3) +
  labs(title = "Distribution of Student Outcomes by Marital Status",
       x = "Marital Status", y = "Percentage") +
  theme_minimal()
```

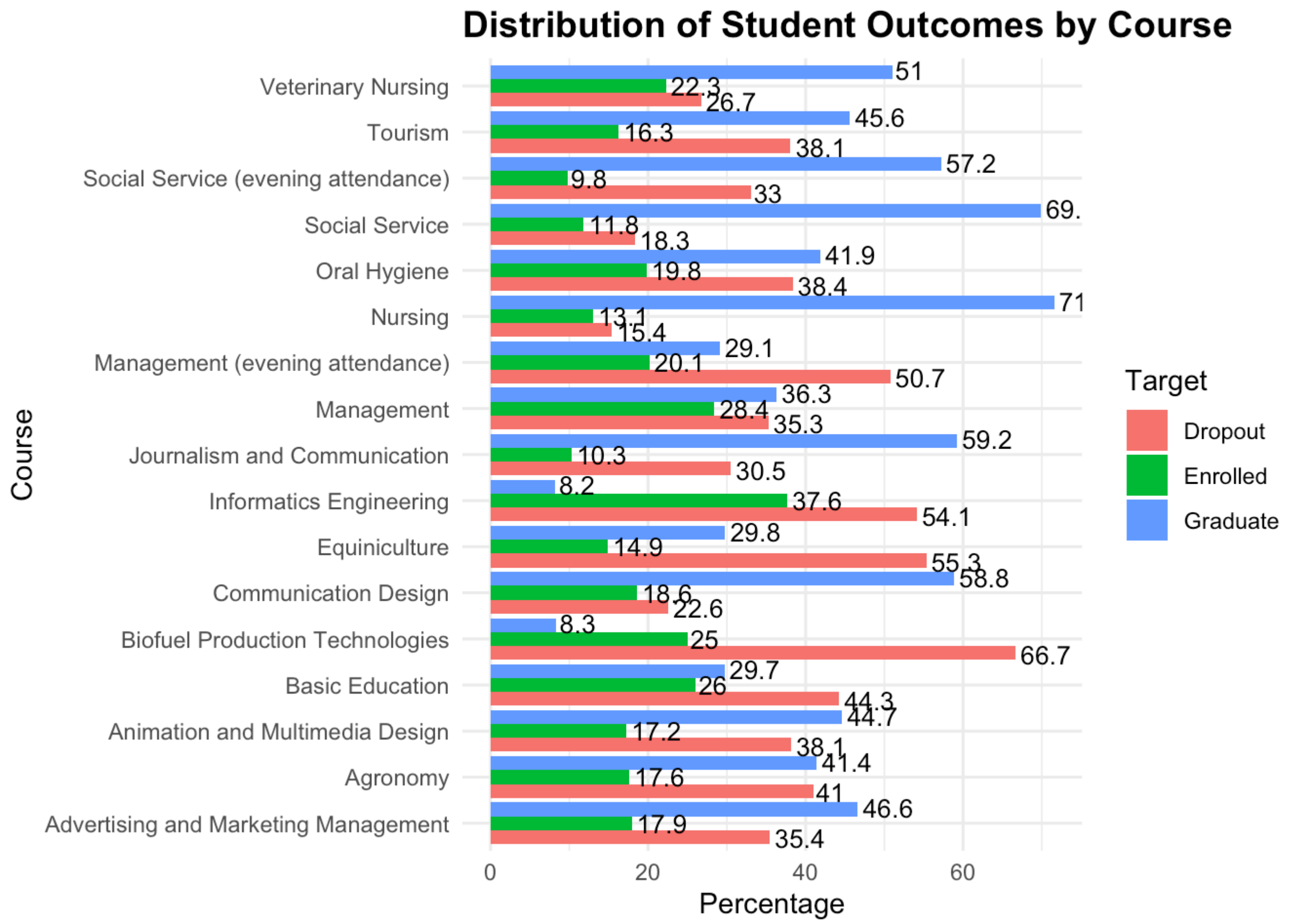


Based on the plot, we can conclude that: _ Single students have the highest graduation rate (51.4%), followed by those in a facto union (44%) and married students (39.1%). This suggests that students without complex family dynamics or with supportive relationships may be more likely to complete their studies. _ Legally separated students show the highest dropout rate (66.7%), indicating that recent or unresolved relationship changes may significantly disrupt academic progress. Divorced students also have a high dropout rate (46.2%). _ Widowed students show a unique pattern: 50% are still enrolled, while only 25% graduated and 25% dropped out. This may reflect ongoing study, small sample size, or unique life circumstances. _ Enrollment rates are notably high among widowers (50%) and single students (18.4%), suggesting these students may still be progressing through their studies.

```
# Define course name mapping based on actual course codes
course_map <- c(
  `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` = "Equiniculture",
  `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)"
)

# Convert course codes to course names in the student data
student_data_course <- student_data %>%
  mutate(Course = recode(as.character(Course), !!!course_map))

# Count students in each course
course_counts <- student_data_course %>%
  count(Course)
# Plot horizontal bar chart of course enrollments
ggplot(course_counts, aes(x = n, y = reorder(Course, n), fill = n)) +
  geom_col() +
  geom_text(aes(label = n), hjust = -0.2, size = 3.5) +
  labs(title = "Courses Enrolled by Students", x = "Total", y = "Courses") +
  scale_fill_gradient(low = "lightblue", high = "red") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14), legend.position = "non
e")
```



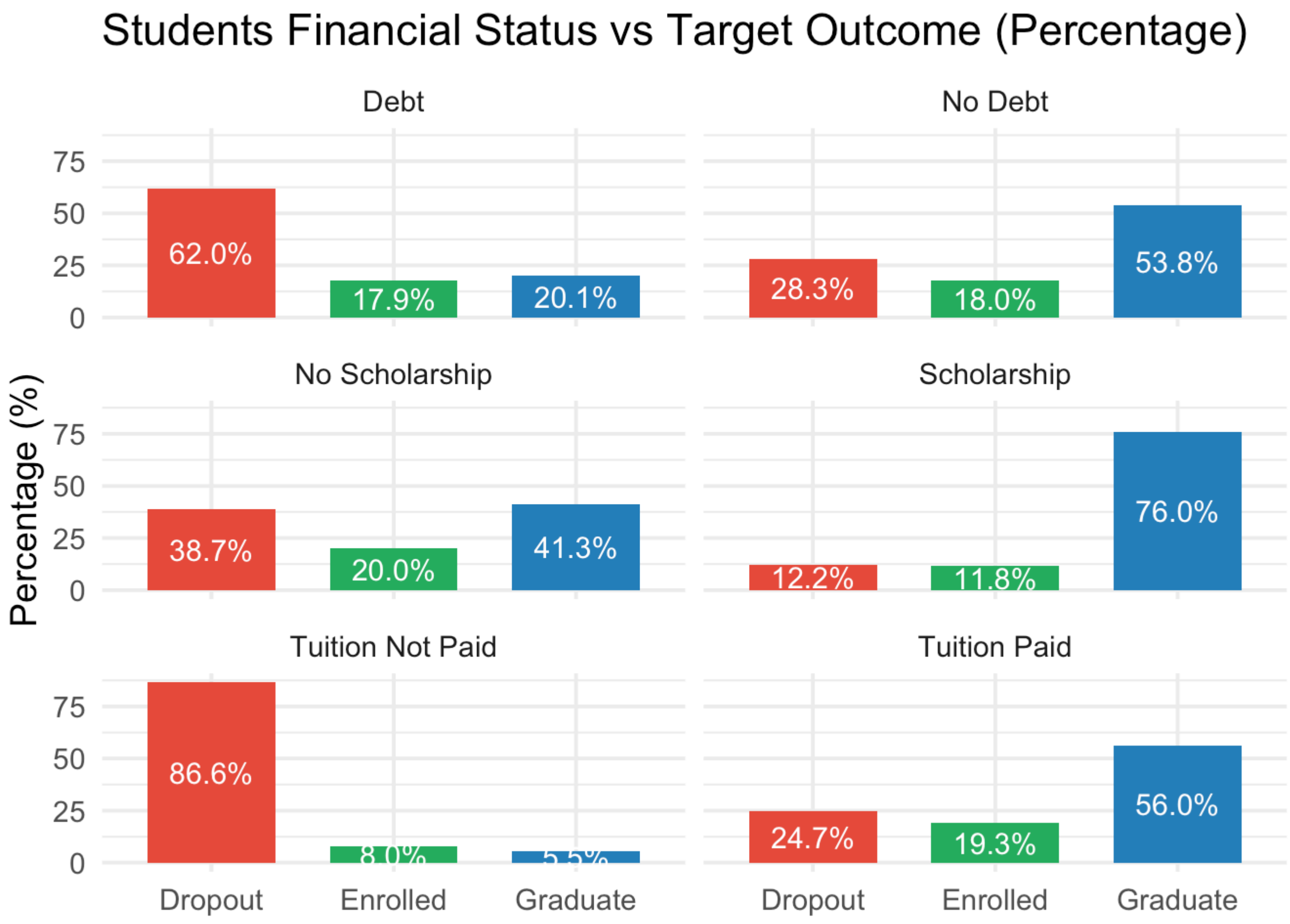

```
# Recode course numbers to names, then calculate % of each outcome (Target) within each course
course_plot_data <- student_data %>%
  mutate(
    Course_name = recode(as.character(Course), !!!course_map), # Replace codes with names
    Course_name = factor(Course_name)  # convert to factor for proper bar chart order
  ) %>%
  group_by(Course_name, Target) %>%
  summarise(n = n(), .groups = "drop") %>%
  group_by(Course_name) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup()

# Create grouped bar chart of student outcomes by course
ggplot(course_plot_data, aes(x = pct, y = Course_name, fill = Target)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = round(pct, 1)), position = position_dodge(width = 1), hjust = -0.1, size = 3.5) +
  labs(
    title = "Distribution of Student Outcomes by Course",
    x = "Percentage", y = "Course"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "right"
  )
```



_ The plot shows that Veterinary Nursing, Oral Hygiene, and Social Service show exceptionally high graduation rates (51% to 71.5%), suggesting strong academic progression and support structures in those programs. _ In contrast, Biofuel Production Technologies has the highest dropout rate (66.7%), which may indicate challenges in the curriculum or student fit. Equiniculture and Informatics Engineering also exhibit elevated dropout rates (55.3% and 54.1% respectively), possibly reflecting difficulties faced by part-time or working students. _ Courses such as Tourism, Journalism and Communication, and Animation and Multimedia Design have relatively balanced distributions, but with moderate dropout rates (30–40%).

```
library(dplyr)
library(ggplot2)

# Convert coded variables to readable labels and count combinations

# For Debt
debt_counts <- student_data %>%
  mutate(Debtor = ifelse(Debtor == 1, "Debt", "No Debt")) %>%
  group_by(Debtor, Target) %>%
  summarise(Count = n(), .groups = "drop") %>%
  rename(Category = Debtor)

# For Tuition status
tuition_counts <- student_data %>%
  mutate(`Tuition fees up to date` = ifelse(`Tuition fees up to date` == 1, "Tuition
Paid", "Tuition Not Paid")) %>%
  group_by(`Tuition fees up to date`, Target) %>%
  summarise(Count = n(), .groups = "drop") %>%
  rename(Category = `Tuition fees up to date`)

# For Scholarship
scholarship_counts <- student_data %>%
  mutate(`Scholarship holder` = ifelse(`Scholarship holder` == 1, "Scholarship", "No
Scholarship")) %>%
  group_by(`Scholarship holder`, Target) %>%
  summarise(Count = n(), .groups = "drop") %>%
  rename(Category = `Scholarship holder`)

# Combine all into one dataframe
financial_status <- bind_rows(debt_counts, tuition_counts, scholarship_counts)

# Compute percentages within each category
financial_status_pct <- financial_status %>%
  group_by(Category) %>%
  mutate(Percentage = Count / sum(Count) * 100)

#  Bar chart of Financial status categories and academic outcomes (Dropout, Enrolled,
Graduate)
ggplot(financial_status_pct, aes(x = Target, y = Percentage, fill = Target)) +
  geom_bar(stat = "identity", width = 0.7) +
  geom_text(aes(label = sprintf("%.1f%%", Percentage)),
            position = position_stack(vjust = 0.5),
            color = "white", size = 4) +
  scale_fill_manual(values = c(
    "Dropout" = "#e74c3c",
    "Enrolled" = "#27ae60",
    "Graduate" = "#2980b9"
  )) +
  facet_wrap(~ Category, ncol = 2) +
  labs(
    title = "Students Financial Status vs Target Outcome (Percentage)",
    x = NULL,
    y = "Percentage (%)"
  ) +
```

```
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```

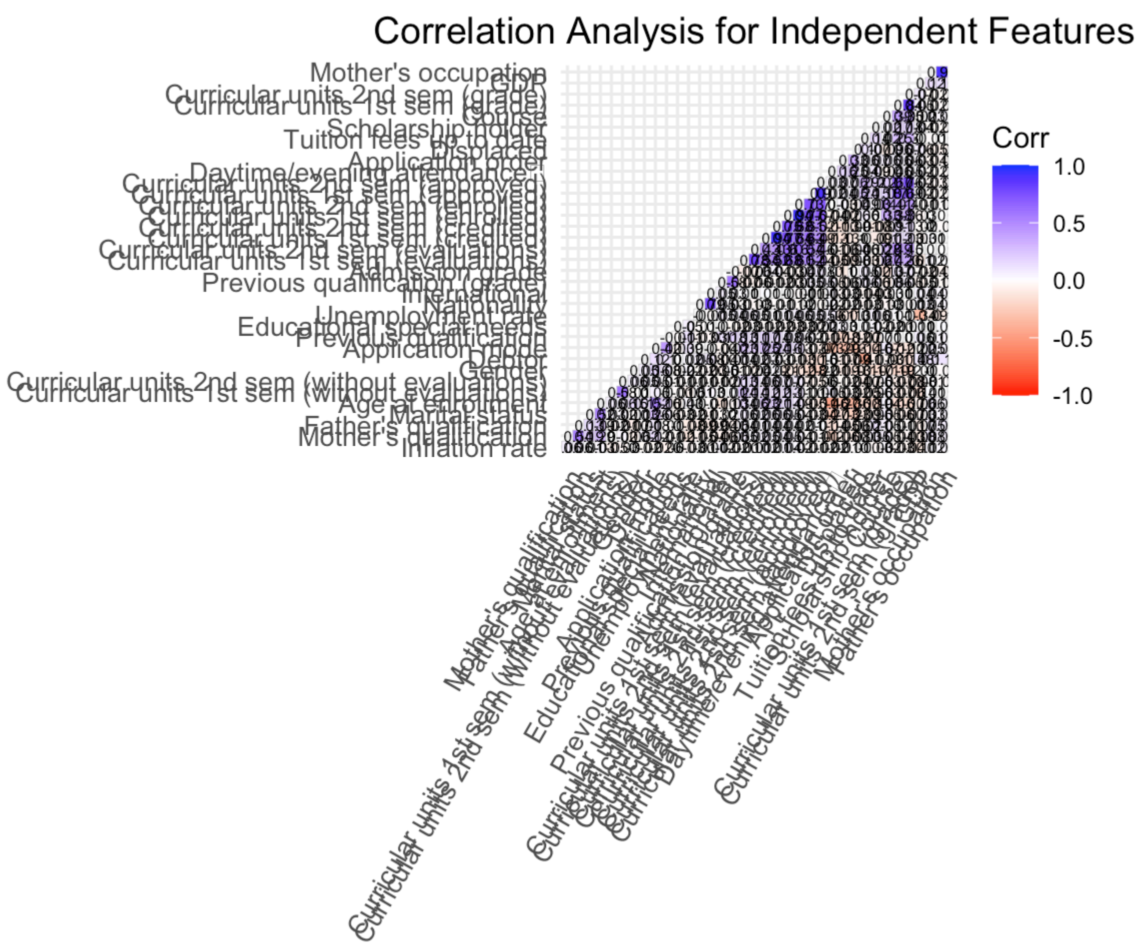


The chart illustrates: _ Students in financial debt have the highest dropout rate (62%), while those without debt show a much higher graduation rate (53.8%). This suggests that financial burdens may hinder academic success. _ Scholarship recipients display the strongest academic outcomes, with a graduation rate of 76% and a very low dropout rate (12.2%), compared to those without scholarships (dropout: 38.7%, graduate: 41.3%). _ Students who have not paid tuition show an alarming dropout rate of 86.6% and only 5.5% graduation. Meanwhile, students who have paid tuition have a much healthier outcome profile, with 56% graduating and only 24.7% dropping out.

```
# Load libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.4     ✔ tibble    3.2.1
## ✔ purrr     1.0.4     ✔ tidyr     1.3.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
library(ggcorrplot)

# Prepare data
numeric_data <- student_data %>%
  select(where(is.numeric)) %>%
  select(-matches("Target", ignore.case = TRUE))

# Compute and round correlation matrix
cor_matrix <- cor(numeric_data, use = "complete.obs")
cor_matrix_rounded <- round(cor_matrix, 2)

# Plot heatmap with larger size and better spacing
ggcorrplot(
  cor_matrix_rounded,
  lab = TRUE,
  lab_size = 2,
  type = "lower",
  colors = c("red", "white", "blue"),
  title = "Correlation Analysis for Independent Features",
  ggtheme = theme_minimal(),
  hc.order = TRUE
) +
theme(
  axis.text.x = element_text(angle = 60, hjust = 1, size = 10),  # steeper angle + sp
acing
  axis.text.y = element_text(size = 10),
  plot.title = element_text(hjust = 0.5, size = 14)
)
```



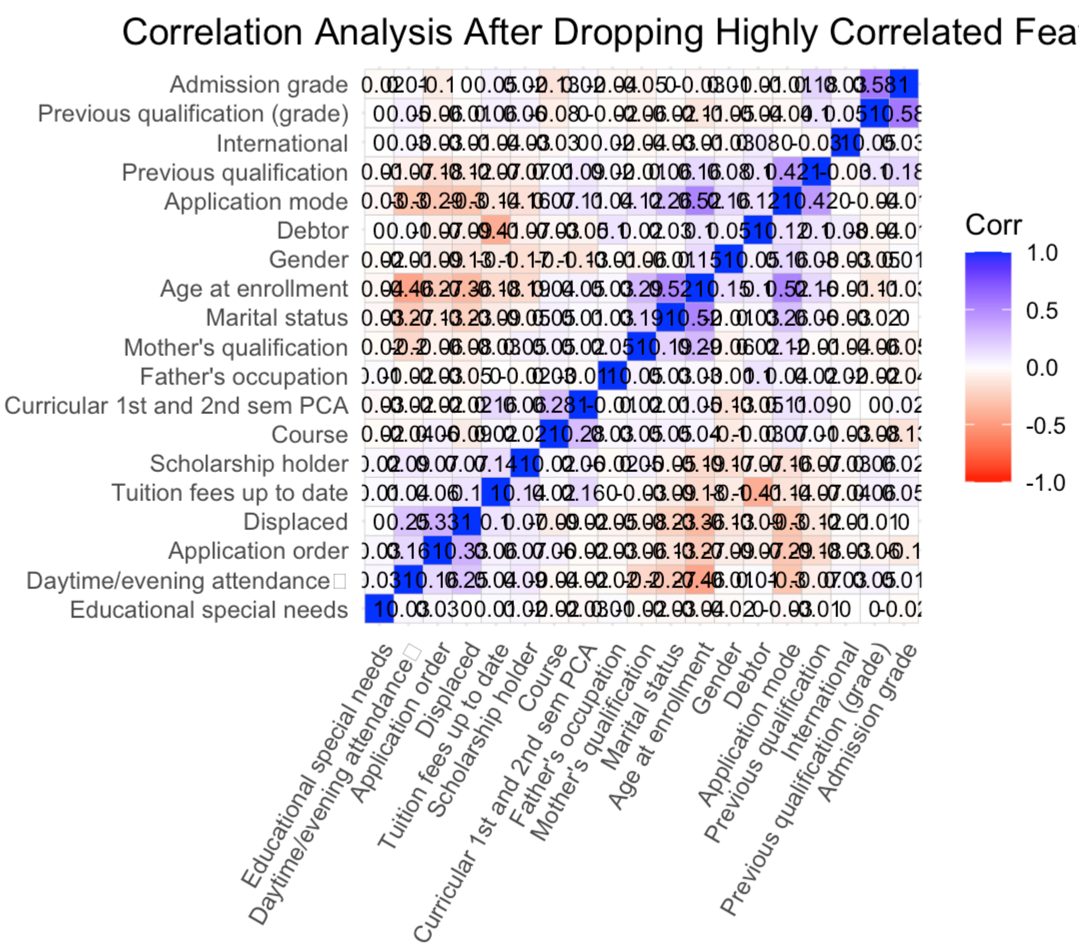
From the correlation matrix, we can see that several features are highly correlated with each other, which may introduce multicollinearity in predictive models: _ Mother's qualification and Father's qualification _ Mother's occupation and Father's occupation _ Nationality and International _ Variables such as GDP and Unemployment rate show very weak or near-zero correlations with other features in the dataset. Therefore, I decide to remove GDP and Unemployment rate from the feature set to streamline the model and reduce noise.

_ Moreover, the dataset includes a large number of variables related to students' academic performance in the 1st and 2nd semesters (e.g., grades, credits, approved units, evaluations). These features are highly correlated with one another, as shown in the correlation matrix. Rather than removing them which could lead to the loss of important information, we can apply Principal Component Analysis (PCA) to reduce dimensionality while preserving the overall academic signal.

```
# Select curriculum-related columns
data_for_pca <- student_data %>%
  select(contains("Curricular units"))

# Perform PCA
pca_result <- prcomp(data_for_pca, center = TRUE, scale. = TRUE)

# Get the first principal component
student_data$`Curricular 1st and 2nd sem PCA` <- pca_result$x[, 1]
```

```
# List of columns to drop due to high correlation
cols_to_drop <- c(
  "Nationality",
  "Mother's occupation",
  "Father's qualification",
  "Curricular units 1st sem (credited)",
  "Curricular units 1st sem (enrolled)",
  "Curricular units 1st sem (evaluations)",
  "Curricular units 1st sem (without evaluations)",
  "Curricular units 1st sem (approved)",
  "Curricular units 1st sem (grade)",
  "Curricular units 2nd sem (credited)",
  "Curricular units 2nd sem (enrolled)",
  "Curricular units 2nd sem (evaluations)",
  "Curricular units 2nd sem (without evaluations)",
  "Curricular units 2nd sem (approved)",
  "Curricular units 2nd sem (grade)",
  "Inflation rate",
  "GDP",
  "Unemployment rate"
)

# Drop them from data frame
student_data <- student_data %>% select(-all_of(cols_to_drop))
```

```r
# Select only numeric columns from the dataset (excluding target-related columns)
numeric_data <- student_data %>%
  select(where(is.numeric)) %>%
  select(-matches("Target", ignore.case = TRUE))

# Compute correlation matrix
cor_matrix <- cor(numeric_data, use = "complete.obs")
cor_matrix_rounded <- round(cor_matrix, 2) # Round the correlation values to 2 decima
l places

# Load correlation heatmap visualization
library(ggcorrplot)
ggcorrplot(
  cor_matrix_rounded,
  lab = TRUE,
  lab_size = 3,
  type = "full",
  colors = c("red", "white", "blue"),
  title = "Correlation Analysis After Dropping Highly Correlated Features",
  ggtheme = theme_minimal(),
  hc.order = TRUE
) +
theme(
  axis.text.x = element_text(angle = 60, hjust = 1, size = 9),
  axis.text.y = element_text(size = 9),
  plot.title = element_text(hjust = 0.5, size = 14)
)
```



Buiding model

```
student_data <- student_data %>%
  mutate(Target_bin = ifelse(Target == "Dropout", 1, 0)) # Create a binary target var
iable: 1 for "Dropout", 0 for all other outcomes (e.g., Enrolled, Graduate)
student_data$Target_bin <- as.factor(student_data$Target_bin) # Convert the binary va
riable to a factor, suitable for classification modeling
student_data <- student_data %>% select(-Target) # Remove the original multiclass Tar
get variable from the dataset
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

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

```
# Split full data into training (80%) and test (20%)
set.seed(42)
train_index <- createDataPartition(student_data$Target_bin, p = 0.8, list = FALSE)
train_data <- student_data[train_index, ]
test_data <- student_data[-train_index, ]
```

```
# Clean column names to ensure they are syntactically valid (remove spaces, special c
haracters)
colnames(train_data) <- make.names(colnames(train_data))
colnames(test_data) <- make.names(colnames(test_data))
```

```
# Create an empty list to store trained models
trained_models <- list()
```

Logistic regression

```
# Load libraries
library(caret)
set.seed(42)
# Train logistic regression model on the training set
logistic_model <- glm(Target_bin ~ ., data = train_data, family = "binomial")

# Generate predicted probabilities for training and test sets
pred_train <- predict(logistic_model, newdata = train_data, type = "response")
pred_test <- predict(logistic_model, newdata = test_data, type = "response")

# Convert predicted probabilities to binary class labels using threshold = 0.5
class_train <- ifelse(pred_train >= 0.5, 1, 0)
class_test <- ifelse(pred_test >= 0.5, 1, 0)

# Calculate accuracy for both training and test sets
acc_train_log <- mean(class_train == train_data$Target_bin)
acc_test_log <- mean(class_test == test_data$Target_bin)

# Output
cat("Accuracy of logistic regression model on the training set is", round(acc_train_l
og * 100, 2), "%\n")
```

```
## Accuracy of logistic regression model on the training set is 82.15 %
```

```
cat("Accuracy of logistic regression model on the test set is", round(acc_test_log *1
00, 2), "%\n")
```

```
## Accuracy of logistic regression model on the test set is 83.48 %
```

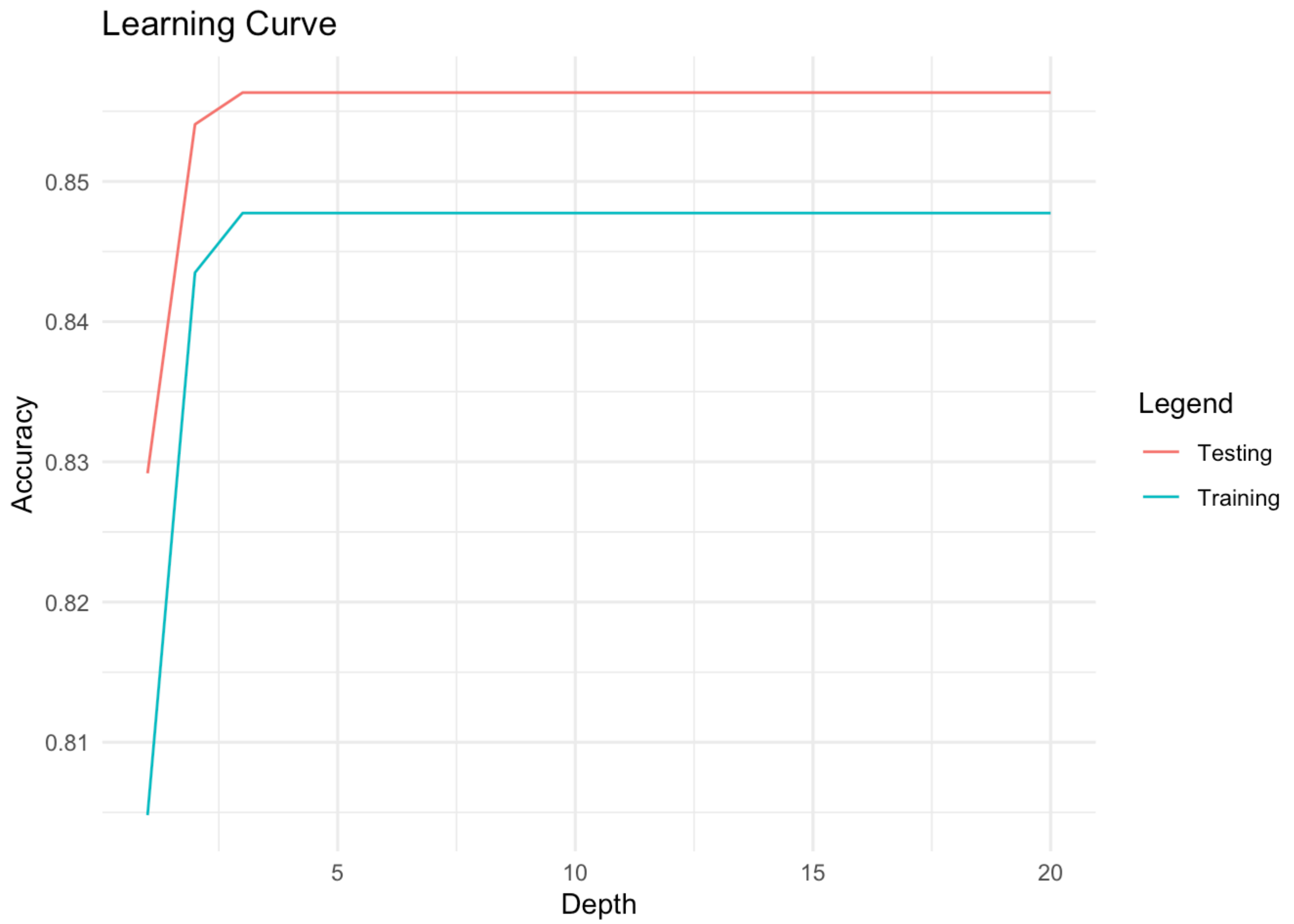
```
trained_models[["Logistic Regression"]] <- logistic_model # Store the trained model i
n a named list
```

Decision tree

```
# Install packages
# install.packages("rpart")
# install.packages("caret")

# Load required package
library(rpart)
library(caret)

# Create empty vectors to store accuracy
depth_values <- 1:20
train_acc <- c()
test_acc <- c()
```

```
set.seed(42)
# Loop through different tree depths to evaluate model performance
for (depth in depth_values) {
  model <- rpart(Target_bin ~ ., data = train_data,
                 method = "class",
                 control = rpart.control(maxdepth = depth))
  # Predict on training and testing sets
  pred_train <- predict(model, train_data, type = "class")
  pred_test <- predict(model, test_data, type = "class")

  # Calculate accuracy for each
  acc_train <- mean(pred_train == train_data$Target_bin)
  acc_test <- mean(pred_test == test_data$Target_bin)

  # Store accuracies
  train_acc <- c(train_acc, acc_train)
  test_acc <- c(test_acc, acc_test)
}
```

```
library(ggplot2)

# Combine depth and accuracy results into a data frame
learning_data <- data.frame(
  Depth = depth_values,
  Training_Accuracy = train_acc,
  Testing_Accuracy = test_acc
)

# Plot learning curve to visualize overfitting or underfitting trends
ggplot(learning_data, aes(x = Depth)) +
  geom_line(aes(y = Training_Accuracy, color = "Training")) +
  geom_line(aes(y = Testing_Accuracy, color = "Testing")) +
  labs(title = "Learning Curve", y = "Accuracy", color = "Legend") +
  theme_minimal()
```




```
# Identify the best tree depth based on highest test accuracy
best_depth <- depth_values[which.max(test_acc)]

# Retrain final model
final_tree <- rpart(Target_bin ~ .,
                    data = train_data,
                    method = "class",
                    control = rpart.control(maxdepth = best_depth))

# Predict on test set
pred_test <- predict(final_tree, test_data, type = "class")

# Final accuracy
final_acc_dec_tree <- mean(pred_test == test_data$Target_bin)
cat("Best depth:", best_depth, "\n")
```

```
## Best depth: 3
```

```
cat("Test accuracy with best depth:", round(final_acc_dec_tree * 100, 2), "%\n")
```

```
## Test accuracy with best depth: 85.63 %
```

```
trained_models[["Decision Tree"]] <- final_tree
```

Random Forest

```
# Install packages
# install.packages("randomForest")
# install.packages("caret")

# Load required package
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
library(caret)
# Set seed for reproducibility
set.seed(42)

# Train a Random Forest model using default parameters
rf_model <- randomForest(Target_bin ~ ., data = train_data)

# Predict on training data
pred_train <- predict(rf_model, train_data, type = "response")

# Calculate accuracy
acc_train <- round(mean(pred_train == train_data$Target_bin) * 100, 2)

# Print results
cat("With default parameters:\n")
```

```
## With default parameters:
```

```
cat("Accuracy of Random Forest model on training data is", acc_train, "%\n")
```

```
## Accuracy of Random Forest model on training data is 99.92 %
```

```
# Set up 5-fold cross-validation
control <- trainControl(method = "cv", number = 5)
# Custom train function to support ntree (which is not in grid by default)
rf_model <- train(
  Target_bin ~ .,
  data = train_data,
  method = "rf",
  metric = "Accuracy",
  trControl = control,
  tuneGrid = expand.grid(mtry = c(2, 3, 5)), # Explore different mtry values
  ntree = 100
)

# Display the best mtry value based on cross-validation accuracy
print(rf_model$bestTune)
```

```
##   mtry
## 3    5
```

```
# Retrain final Random Forest model with best mtry from grid search
final_rf_model <- randomForest(
  Target_bin ~ .,
  data = train_data,
  mtry = rf_model$bestTune$mtry,  # best mtry value from previous tuning
  ntree = 100                     # fixed number of trees
)

# Predict on train and test sets
pred_train <- predict(final_rf_model, train_data, type = "response")
pred_test <- predict(final_rf_model, test_data, type = "response")

# Calculate accuracy
acc_train_ran <- mean(pred_train == train_data$Target_bin)
acc_test_ran <- mean(pred_test == test_data$Target_bin)

# Print results
cat("With Grid search best estimator parameter:\n")
```

```
## With Grid search best estimator parameter:
```

```
cat("Accuracy of Random Forest model on training data is", round(acc_train_ran * 100,
2), "%\n")
```

```
## Accuracy of Random Forest model on training data is 99.94 %
```

```
cat("Accuracy of Random Forest model on test data is", round(acc_test_ran * 100, 2),
"%\n")
```

```
## Accuracy of Random Forest model on test data is 84.84 %
```

```
trained_models[["Random Forest"]] <- final_rf_model
```

K-Nearest Neighbor (KNN)

```
# Set cross-validation method
control <- trainControl(method = "cv", number = 5)

# Define grid of k values (number of neighbors) to tune
tune_grid <- expand.grid(k = 1:24)

set.seed(42)

# Train K-Nearest Neighbors (KNN) model with grid search to find best k
knn_model <- train(
  Target_bin ~ .,
  data = train_data,
  method = "knn",
  trControl = control,
  tuneGrid = tune_grid,
  metric = "Accuracy"
)

# View best k
print(knn_model$bestTune)
```

```
##     k
## 21 21
```

```
# Retrain KNN model with best k from tuning
final_knn_model <- knn3(
  Target_bin ~ .,
  data = train_data,
  k = knn_model$bestTune$k  # Use best k found earlier
)

# Predict on testing set
pred_train <- predict(final_knn_model, test_data, type = "class")
# Calculate test accuracy
acc_test_knn <- mean(pred_test == test_data$Target_bin)
cat("Accuracy of KNN model on test data is", round(acc_test_knn * 100, 2), "%\n")
```

```
## Accuracy of KNN model on test data is 84.84 %
```

```
trained_models[["KNN"]] <- final_knn_model
```

Support Vector Machine (SVM)

```
# install.packages("e1071")
# Load library
library(e1071)
set.seed(42)
# List of kernels to compare
kernels <- c("linear", "polynomial", "radial", "sigmoid")
svm_scores <- c()

# Loop over each kernel
for (kernel in kernels) {
  svm_model <- svm(Target_bin ~ .,
                   data = train_data,
                   kernel = kernel,
                   probability = FALSE)

  # Predict on training set
  pred_train <- predict(svm_model, train_data)

  # Compute accuracy
  acc <- mean(pred_train == train_data$Target_bin)

  # Store result
  svm_scores[kernel] <- acc
}

# Print the results
print(round(svm_scores * 100, 2))
```

```
##     linear polynomial     radial    sigmoid
##      82.66      83.64      86.02      73.25
```

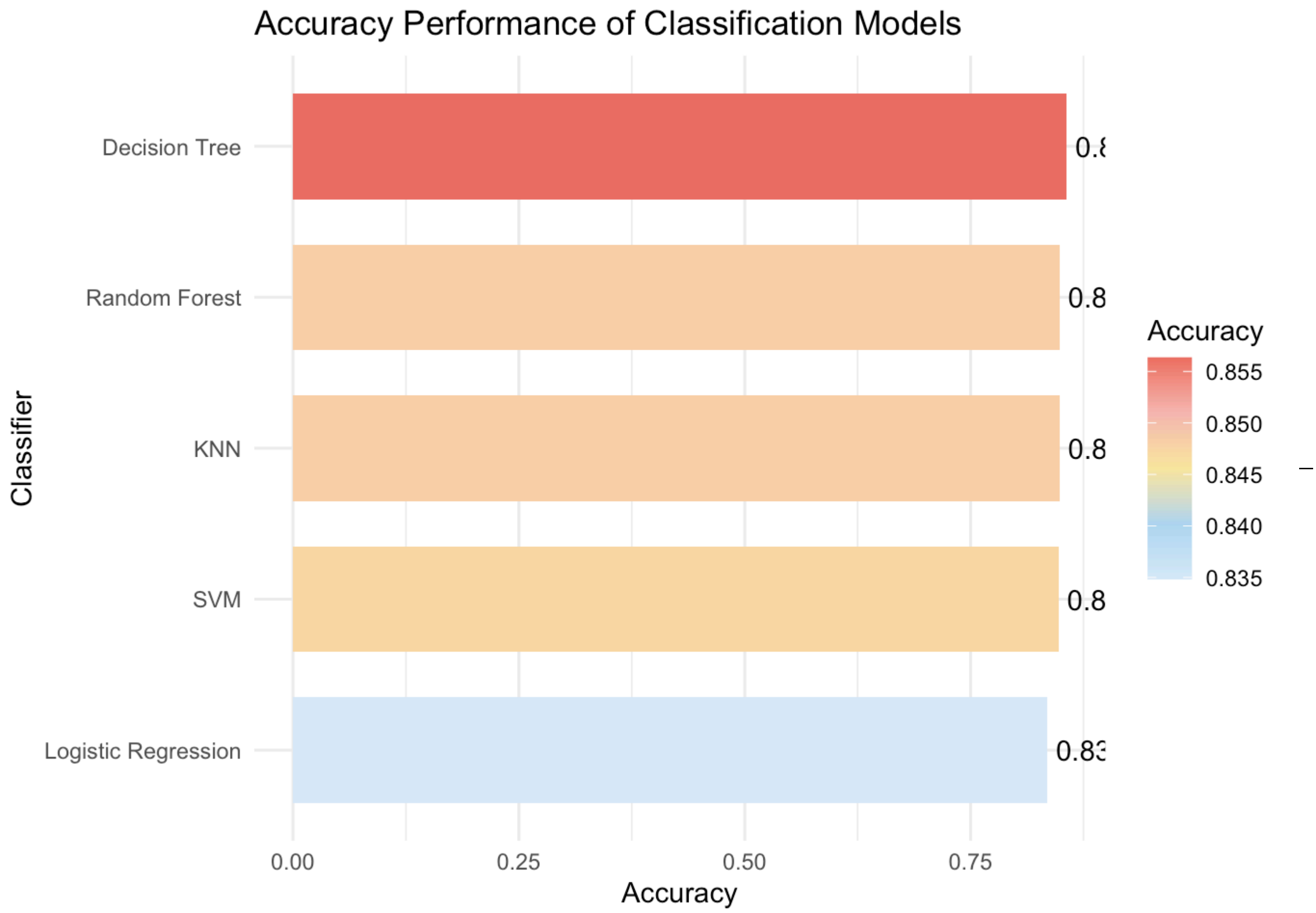
```
# Load required package
library(e1071)
best_kernel <- names(which.max(svm_scores)) # Identify best-performing kernel based o
n training accuracy
final_svm_model <- svm(Target_bin ~ ., data = train_data, kernel = best_kernel)  # Re
train final SVM model using the best kernel

# Predict on test data
pred_test <- predict(final_svm_model, test_data)

# Evaluate accuracy
acc_test_svm  <- mean(pred_test == test_data$Target_bin)

# Print accuracy
cat("Accuracy of SVM model on test data is", round(acc_test_svm * 100, 2), "%\n")
```

```
## Accuracy of SVM model on test data is 84.73 %
```

```
trained_models[["SVM"]] <- final_svm_model
```

Accuracy for each model

```
# Accuracy results
accuracy_scores <- c(
  "Decision Tree" = final_acc_dec_tree,
  "Logistic Regression" = acc_test_log,
  "Random Forest" = acc_test_ran,
  "SVM" = acc_test_svm,
  "KNN" = acc_test_knn
)

# Convert to dataframe for ggplot
accuracy_df <- data.frame(
  Classifier = names(accuracy_scores),
  Accuracy = as.numeric(accuracy_scores)
)

# Create horizontal bar plot to compare model accuracy
library(ggplot2)

ggplot(accuracy_df, aes(x = reorder(Classifier, Accuracy), y = Accuracy, fill = Accur
acy)) +
  geom_bar(stat = "identity", width = 0.7) +
  coord_flip() +
  geom_text(aes(label = round(Accuracy, 4)), hjust = -0.1) +
  scale_fill_gradientn(
    colours = c("#D6EAF8", "#AED6F1", "#F9E79F", "#F5B7B1", "#EC7063"),  # Light blue
to red gradient
    limits = c(min(accuracy_df$Accuracy), max(accuracy_df$Accuracy))
  ) +
  labs(
    title = "Accuracy Performance of Classification Models",
    x = "Classifier",
    y = "Accuracy"
  ) +
  theme_minimal()
```



The Decision Tree model achieved the highest accuracy (85.63%), slightly outperforming Random Forest and KNN (both at 84.84%). SVM followed closely with 84.73%, while Logistic Regression recorded the lowest accuracy (83.48%) but remains highly interpretable. _ This plot supports the conclusion that while all models perform similarly, the Decision Tree offers the best trade-off between accuracy and interpretability, making it the most suitable choice for this educational application.

```
# Identify the best-performing model based on highest test accuracy
best_model_name <- names(which.max(accuracy_scores))
best_model <- trained_models[[best_model_name]]
# Drop the target column before prediction
test_features <- test_data[, setdiff(names(test_data), "Target_bin")]

# Use probability thresholding for logistic regression, direct class prediction other
wise
if (best_model_name == "Logistic Regression") {
  probs <- predict(best_model, test_features, type = "response")
  best_pred <- ifelse(probs >= 0.5, 1, 0)
} else {
  best_pred <- predict(best_model, test_features, type = "class")
}


# Create confusion matrix to evaluate classification performance
conf_matrix <- table(
  Actual = test_data$Target_bin,
  Predicted = best_pred)
# Output the best model name and its confusion matrix
print(best_model_name)
```

```
## [1] "Decision Tree"
```

```
print(conf_matrix)
```

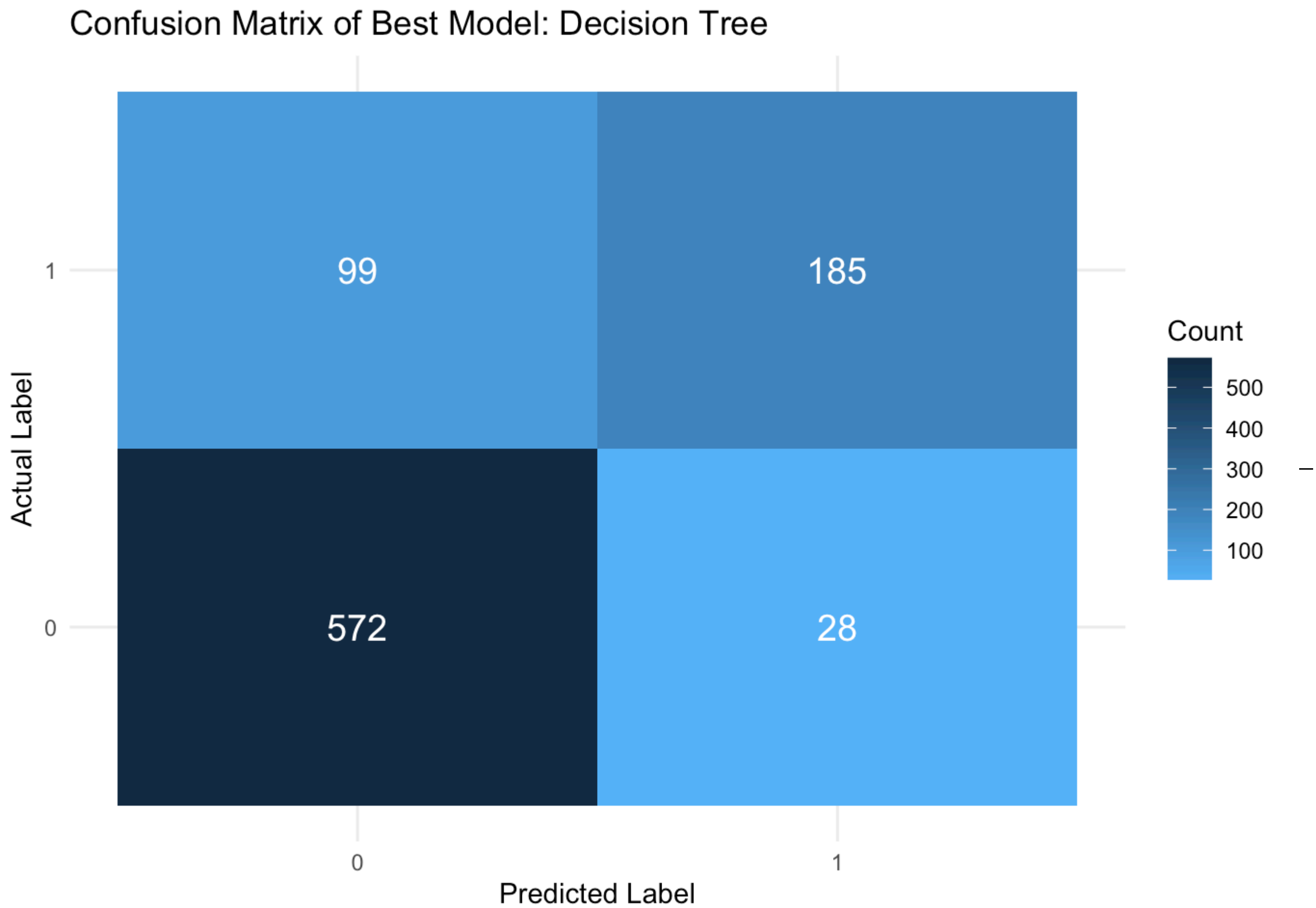
```
##       Predicted
## Actual   0   1
##      0 572  28
##      1  99 185
```

```
# install.packages("reshape2")
# Load libraries
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
# Convert confusion matrix to a data frame for plotting
conf_df <- as.data.frame(conf_matrix)
colnames(conf_df) <- c("Actual", "Predicted", "Count")

# Plot confusion matrix as a heatmap
ggplot(conf_df, aes(x = Predicted, y = Actual, fill = Count)) +
  geom_tile() +
  geom_text(aes(label = Count), color = "white", size = 5) +
  scale_fill_gradient(low = "#56B1F7", high = "#132B43") +
  labs(
    title = paste("Confusion Matrix of Best Model:", best_model_name),
    x = "Predicted Label",
    y = "Actual Label"
  ) +
  theme_minimal()
```



The confusion matrix reveals that the decision tree model performed strongly in identifying students who did not drop out (class 0), with 572 true negatives and only 28 false positives. However, there is some challenge in correctly identifying dropouts (class 1), where the model correctly predicted 185 dropouts but misclassified 99 actual dropouts. _ This suggests the model is conservative in flagging dropout risk, prioritising precision over recall. While it excels at ruling out students unlikely to drop out, further tuning or hybrid methods may improve sensitivity to at-risk cases.

```
best_accuracy <- accuracy_scores[best_model_name] * 100

# Extract key parameters
if (best_model_name == "Random Forest") {
  param_summary <- sprintf("ntree=%d, mtry=%d", best_model$ntree, best_model$mtry)
} else if (best_model_name == "Logistic Regression") {
  param_summary <- "default glm settings"
} else if (best_model_name == "SVM") {
  param_summary <- paste("kernel =", best_model@kernel)
} else if (best_model_name == "Decision Tree") {
  param_summary <- paste("maxdepth =", best_depth)
} else {
  param_summary <- "parameters not available"
}

# Print out the conclusion of result
cat(sprintf("From the results above we can see that %s with parameters %s performs be
st with the highest accuracy of %.2f%%.",
            best_model_name, param_summary, best_accuracy))
```

```
## From the results above we can see that Decision Tree with parameters maxdepth = 3
performs best with the highest accuracy of 85.63%.
```