

Math 133: Statistical Learning Methods - Module 1 Assessment

Pranav Jayakumar

February 3, 2025

Abstract

In this module assessment, we will draw from survey data and fit simple linear models to understand the effects of various predictors on students' stress.

1 Data Analysis

For the following analysis, we will fit simple linear models of the form

$$Y = \beta_0 + \beta_1 X$$

where $Y = \text{stress_score}$ and X is one of the following numeric features: `maximum_alcohol_consumed`, `gpa`, or `screen_time`.

1.1 Data Visualization

We will begin by creating scatterplots of students' stress scores vs. each of the three aforementioned predictors. We will use a 70-30 training-testing split for our models.

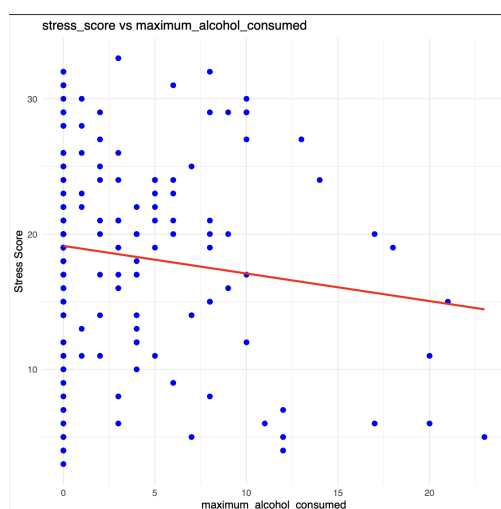


Figure 1: Scatterplot of Stress Score vs. Maximum Alcohol Consumed

The linear model for stress score vs. maximum alcohol consumed is represented by:

$$\hat{y} = 18.8543 - 0.2999x$$

We observe an R^2 score of 0.0417.

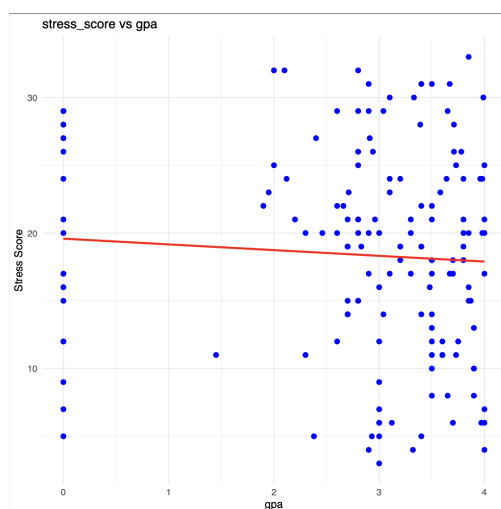


Figure 2: Scatterplot of Stress Score vs. GPA

The linear model for stress score vs. GPA is represented by:

$$\hat{y} = 19.8029 - 0.3343x$$

We observe an R^2 score of 0.0027.

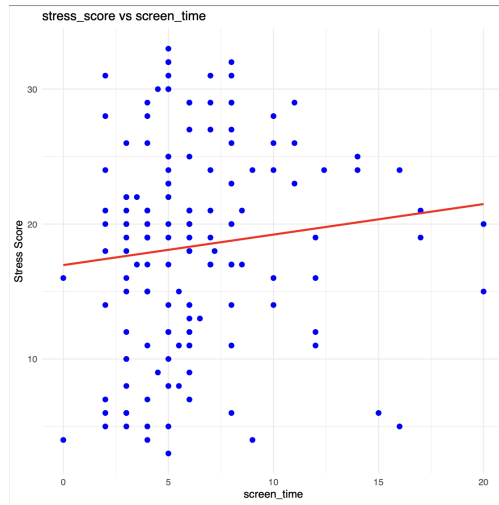


Figure 3: Scatterplot of Stress Score vs. Screen Time

The linear model for stress score vs. screen time is represented by:

$$\hat{y} = 16.7664 + 0.3193x$$

We observe an R^2 score of 0.0222.

1.2 Insights

We observe that the linear model `stress_score~screen_time` has the highest R^2 score. Therefore, there is evidence to suggest that screen time is the best out of the three predictors of stress.

Plotting screen time data in a histogram shows that the data is approximately normally distributed.

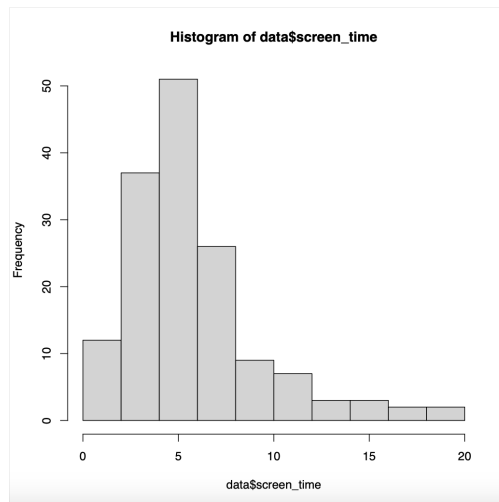


Figure 4: Histogram of Screen Time

2 R Coding

```

1  #!/usr/bin/env Rscript
2  library(ggplot2)
3
4  set.seed(123)
5
6  fit_linear_model <- function(y, x, raw_data) {
7    # Train-test split
8    n <- nrow(raw_data)
9    trainIndex <- sample(n, round(0.7 * n, 0))
10   train <- raw_data[trainIndex, ]
11   test <- raw_data[-trainIndex, ]
12
13   # Construct formula
14   formula <- as.formula(paste(y, "~", x))
15
16   # Fit model
17   model <- lm(formula, data = train)
18
19   overview = summary(model)
20   print(overview)
21
22   # Predict on testing data
23   y_test <- test[[y]]
24   y_hat <- predict(model, newdata = test)
25
26   # Analyze accuracy
27   SSE <- sum((y_test - y_hat)^2)
28   MSE <- SSE / nrow(test)
29   RMSE <- sqrt(MSE)

```

```

30 SST <- sum((y_test - mean(y_test))^2)
31 R2 <- 1 - SSE / SST
32
33 return(list(SSE = SSE, MSE = MSE, RMSE = RMSE, SST = SST, R2 = R2))
34 }
35
36 main <- function() {
37   # Load data
38   data <- read.csv("../data/survey_fall2023.csv")
39
40   # Define predictors
41   predictors <- c("maximum_alcohol_consumed", "gpa", "screen_time")
42   results <- list()
43
44   # Open PDF device for saving all plots
45   pdf("Rplots.pdf") # This will save all plots sequentially to Rplots.
46   pdf
47
48   # Iterate over predictors
49   for (predictor in predictors) {
50     if (!predictor %in% colnames(data)) {
51       cat("\nWarning: Predictor", predictor, "not found in dataset.\n")
52       Skipping...\n")
53       next
54     }
55
56     cat("\nLinear model for stress score ~", predictor, "\n")
57     result <- fit_linear_model("stress_score", predictor, data)
58
59     # Store results for later comparison
60     results[[predictor]] <- result
61
62     # Format and print results
63     formatted_results <- lapply(result[1:5], function(x) format(round(x,
64       4), nsmall = 4))
65     print(formatted_results)
66
67     # Create scatterplot for the predictor
68     ggplot(data, aes_string(x = predictor, y = "stress_score")) +
69       geom_point(color = "blue", size = 2) +
70       geom_smooth(method = "lm", color = "red", se = FALSE) +
71       ggtitle(paste("stress_score vs", predictor)) +
72       xlab(predictor) +
73       ylab("Stress Score") +
74       theme_minimal() -> plot # Assign ggplot to a variable
75
76     print(plot) # Ensure the plot is sent to the PDF file
77   }
78
79   hist(data$screen_time, breaks = "Freedman-Diaconis") -> plot1
80   print(plot1)
81
82   # Close PDF device

```

```
80 dev.off()
81
82 cat("\nAll plots saved to Rplots.pdf.\n")
83 }
84
85 # Run the script if executed directly
86 if (interactive() || identical(Sys.getenv("R_SCRIPT"), "")) {
87   main()
88 }
```