VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

PROJECT REPORT

Mental Health - Exploratory Data Analysis , Classification and Regression

B.Tech. Fall Semester 2024-25

BMAT202L - Probability & Statistics

Slot: E1+TE1

Submitted to : Dr. Amit Kumar Rahul

Team Details

22BRS1095	**Kishore K G**
22BRS1099	**ARAVIND N**
22BRS1021	**S.Mukunth**
22BRS1357	**Abhinav Balakrishnan**

Aim

The aim of this project is to perform a comprehensive Exploratory Data Analysis (EDA) on technology usage and mental health data, aiming to understand relationships between daily screen time, stress levels, sleep quality, and self-reported mental health. Through this analysis, we aim to uncover patterns that may impact mental well-being, such as the effect of prolonged screen time or poor sleep quality on mental health scores. Following EDA, we will develop a predictive model to estimate mental health scores based on these variables, offering a data-driven perspective to help inform strategies for better mental health management.

About the Dataset

This dataset provides a detailed view of individual daily technology usage and mental health indicators. Each record includes a unique identifier (User_ID), age, average daily screen time in hours, a self-reported mental health score (1-10), a self-reported stress level (1-10), and a self-reported sleep quality score (1-10). These factors allow for in-depth analysis of potential correlations and influences between technology habits and mental health outcomes. This dataset is particularly valuable for studying behavioral patterns, identifying high-risk factors for mental health issues, and creating a predictive model that estimates mental health scores, potentially serving as a tool for mental health researchers, practitioners, and policymakers.

Dataset Analysis

Importing Libraries and Dataset

```r
# Load necessary libraries
library(caret)
library(dplyr)
library(fastDummies)
library(caTools)
library(randomForest)
library(xgboost)
library(readr)
library(ggplot2)
library(rsample)
library(tidyverse)
library(skimr)
library(moments)
library(ggplot2)
library(dplyr)
library(gmodels)
library(scales)
library(nortest)
library(fitdistrplus)
library(MASS)
library(gridExtra)
library(caret)
library(randomForest)
library(xgboost)
library(nnet)
library(e1071)
library(ROSE)
library(rpart)
library(gbm)
library(MLmetrics)
library(kernlab)


# Read the CSV file
data <- read.csv("C:/Users/kisho/Desktop/prob-project/mental_health_with_sleep_quality.csv")
```

Dataset Values

```r
head(data)  # Display the first few rows of the dataset
```

```
##      User_ID Age Gender Technology_Usage_Hours Social_Media_Usage_Hours
## 1 USER-00001  23 Female                  6.57                     6.00
## 2 USER-00002  21   Male                  3.01                     2.57
## 3 USER-00003  51   Male                  3.04                     6.14
## 4 USER-00004  25 Female                  3.84                     4.48
## 5 USER-00005  53   Male                  1.20                     0.56
## 6 USER-00006  58   Male                  5.59                     5.74
##   Gaming_Hours Screen_Time_Hours Mental_Health_Status Stress_Level Sleep_Hours
## 1         0.68            12.36                Good            1        8.01
## 2         3.74             7.61                Poor            3        7.28
## 3         1.26             3.16                Fair            3        8.04
## 4         2.59            13.08           Excellent            2        5.62
## 5         0.29            12.63                Good            1        5.55
## 6         0.11             1.34                Poor            1        8.61
##   Physical_Activity_Hours Support_Systems_Access Work_Environment_Impact
## 1                    6.71                     No                Negative
## 2                    5.88                    Yes                Positive
## 3                    9.81                     No                Negative
## 4                    5.28                    Yes                Negative
## 5                    4.00                     No                Positive
## 6                    6.54                    Yes                 Neutral
##   Online_Support_Usage Sleep_Quality
## 1                  Yes          Good
## 2                   No          Fair
## 3                   No          Good
## 4                  Yes          Poor
## 5                  Yes          Fair
## 6                  Yes          Good
```

Dataset Overview

```
# 1. Dataset dimensions
cat("Dataset Dimensions: ", paste(dim(data), collapse = " x "), "\n")
```

```
## Dataset Dimensions:  10000 x 15
```

```
# 2. Variable types
cat("Variable Types:\n", paste(names(data), sapply(data, class), sep = ": ", collapse = "\n"), "\n")
```

```
## Variable Types:
##  User_ID: character
## Age: integer
## Gender: character
## Technology_Usage_Hours: numeric
## Social_Media_Usage_Hours: numeric
## Gaming_Hours: numeric
## Screen_Time_Hours: numeric
## Mental_Health_Status: character
## Stress_Level: integer
## Sleep_Hours: numeric
## Physical_Activity_Hours: numeric
## Support_Systems_Access: character
## Work_Environment_Impact: character
## Online_Support_Usage: character
## Sleep_Quality: character
```

Cleaning Dataset

```
# 3. Missing values analysis
missing_values <- colSums(is.na(data))
cat("Missing Values Analysis:\n", paste(names(missing_values), missing_values, sep = ": ", collapse = "\n"), "\n")
```

```
## Missing Values Analysis:
##  User_ID: 0
## Age: 0
## Gender: 0
## Technology_Usage_Hours: 0
## Social_Media_Usage_Hours: 0
## Gaming_Hours: 0
## Screen_Time_Hours: 0
## Mental_Health_Status: 0
## Stress_Level: 0
## Sleep_Hours: 0
## Physical_Activity_Hours: 0
## Support_Systems_Access: 0
## Work_Environment_Impact: 0
## Online_Support_Usage: 0
## Sleep_Quality: 0
```

```
# Check for duplicate rows
duplicates <- sum(duplicated(data))
cat("Number of duplicate rows: ", duplicates, "\n")
```

```
## Number of duplicate rows:  0
```

Datatypes of Columns

```
print(skim(data))
```

```
## ── Data Summary ────────────────────────
##                            Values
## Name                       data
## Number of rows             10000
## Number of columns          15
## _________________________
## Column type frequency:
##   character                7
##   numeric                  8
## _________________________
## Group variables            None
##
## ── Variable type: character ──────────────────────────────────────────────────────
##   skim_variable           n_missing complete_rate min max empty n_unique
## 1 User_ID                         0             1  10  10     0    10000
## 2 Gender                          0             1   4   6     0        3
## 3 Mental_Health_Status            0             1   4   9     0        4
## 4 Support_Systems_Access          0             1   2   3     0        2
## 5 Work_Environment_Impact         0             1   7   8     0        3
## 6 Online_Support_Usage            0             1   2   3     0        2
## 7 Sleep_Quality                   0             1   4   4     0        3
##   whitespace
## 1          0
## 2          0
## 3          0
## 4          0
## 5          0
## 6          0
## 7          0
##
## ── Variable type: numeric ────────────────────────────────────────────────────────
##   skim_variable            n_missing complete_rate  mean     sd p0   p25   p50
## 1 Age                              0             1 41.5  13.9   18 29    42
## 2 Technology_Usage_Hours           0             1  6.47  3.17   1  3.76  6.42
## 3 Social_Media_Usage_Hours         0             1  3.97  2.31   0  1.98  3.95
## 4 Gaming_Hours                     0             1  2.52  1.45   0  1.26  2.52
## 5 Screen_Time_Hours                0             1  7.98  4.04   1  4.52  7.9
## 6 Stress_Level                     0             1  2.00  0.816  1  1     2
## 7 Sleep_Hours                      0             1  6.50  1.45   4  5.26  6.5
## 8 Physical_Activity_Hours          0             1  5.00  2.91   0  2.49  4.99
##     p75 p100 hist
## 1 54      65 ▇▇▇▇▇▇▇
## 2  9.21   12 ▇▇▇▇▇▇▇
## 3  5.99    8 ▇▇▇▇▇▇▇
## 4  3.79    5 ▇▇▇▇▇▇▇
## 5 11.5    15 ▇▇▇▇▇▇▇
## 6  3       3 ▇▁▇▁▇
## 7  7.76    9 ▇▇▇▇▇▇▇
## 8  7.54   10 ▇▇▇▇▇▇▇
```

```
## 'data.frame':    10000 obs. of  15 variables:
##  $ User_ID                : chr  "USER-00001" "USER-00002" "USER-00003" "USER-00004" ...
##  $ Age                    : int  23 21 51 25 53 58 63 51 57 31 ...
##  $ Gender                 : chr  "Female" "Male" "Male" "Female" ...
##  $ Technology_Usage_Hours  : num  6.57 3.01 3.04 3.84 1.2 ...
##  $ Social_Media_Usage_Hours: num  6 2.57 6.14 4.48 0.56 5.74 2.55 4.1 4.11 7.23 ...
##  $ Gaming_Hours           : num  0.68 3.74 1.26 2.59 0.29 0.11 3.79 4.74 0.08 0.81 ...
##  $ Screen_Time_Hours      : num  12.36 7.61 3.16 13.08 12.63 ...
##  $ Mental_Health_Status    : chr  "Good" "Poor" "Fair" "Excellent" ...
##  $ Stress_Level           : int  1 3 3 2 1 1 2 2 2 3 ...
##  $ Sleep_Hours            : num  8.01 7.28 8.04 5.62 5.55 8.61 8.61 7.11 7.19 5.09 ...
##  $ Physical_Activity_Hours : num  6.71 5.88 9.81 5.28 4 6.54 1.34 5.27 5.22 0.47 ...
##  $ Support_Systems_Access  : chr  "No" "Yes" "No" "Yes" ...
##  $ Work_Environment_Impact : chr  "Negative" "Positive" "Negative" "Negative" ...
##  $ Online_Support_Usage    : chr  "Yes" "No" "No" "Yes" ...
##  $ Sleep_Quality          : chr  "Good" "Fair" "Good" "Poor" ...
```

Dispersion Measurement

```r
# Dispersion Measurement
calculate_dispersion <- function(x) {
  q <- quantile(x, probs = c(0.25, 0.5, 0.75), na.rm = TRUE)
  c(
    Range = diff(range(x, na.rm = TRUE)),
    Variance = var(x, na.rm = TRUE),
    SD = sd(x, na.rm = TRUE),
    CV = sd(x, na.rm = TRUE) / mean(x, na.rm = TRUE) * 100,
    Q1 = q[1],
    Median = q[2],
    Q3 = q[3],
    IQR = IQR(x, na.rm = TRUE)
  )
}

# Calculate dispersion measures
dispersion_measures <- t(sapply(numeric_data, calculate_dispersion))
cat("\nMeasures of Dispersion for Numeric Variables:\n")
```

Output:

```
##                          Range    Variance         SD       CV Q1.25%
## Age                         47 193.7724313 13.9202166 33.52766  29.00
## Technology_Usage_Hours      11  10.0427030  3.1690224 48.94741   3.76
## Social_Media_Usage_Hours     8   5.3532384  2.3137066 58.24571   1.98
## Gaming_Hours                 5   2.0930812  1.4467485 57.51112   1.26
## Screen_Time_Hours           14  16.3426773  4.0426077 50.68614   4.52
## Stress_Level                 2   0.6662666  0.8162515 40.81666   1.00
## Sleep_Hours                  5   2.1052070  1.4509331 22.31956   5.26
## Physical_Activity_Hours     10   8.4392801  2.9050439 58.05606   2.49
##                          Median.50%  Q3.75%     IQR
## Age                          42.000 54.0000 25.0000
## Technology_Usage_Hours        6.425  9.2125  5.4525
## Social_Media_Usage_Hours      3.950  5.9900  4.0100
## Gaming_Hours                  2.520  3.7900  2.5300
## Screen_Time_Hours             7.900 11.5000  6.9800
## Stress_Level                  2.000  3.0000  2.0000
## Sleep_Hours                   6.500  7.7600  2.5000
## Physical_Activity_Hours       4.990  7.5400  5.0500
```

Sleep Quality Column Analysis

```
# Sleep Quality Category Frequency & Percentage
sleep_quality_summary <- data.frame(
  Category = names(sleep_quality_table),
  Frequency = as.vector(sleep_quality_table),
  Percentage = prop.table(sleep_quality_table) * 100
)

cat("\nFrequency and Percentage of Sleep Quality Categories:\n")
```

Output:

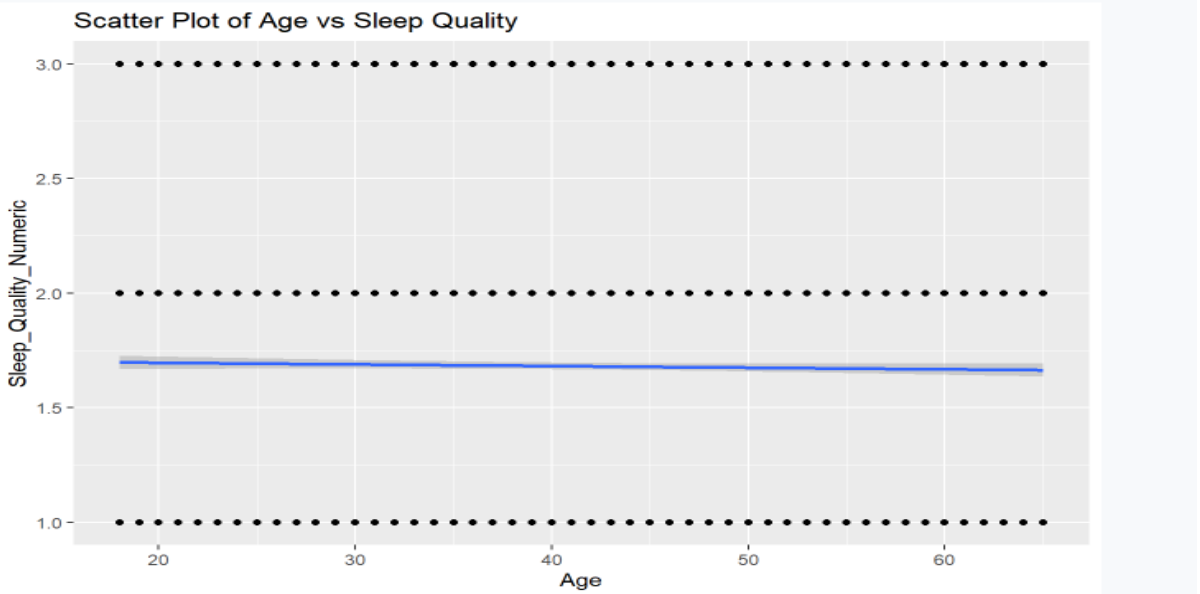
```
##   Category Frequency Percentage.Var1 Percentage.Freq
## 1     Fair      3340            Fair           33.40
## 2     Good      1734            Good           17.34
## 3     Poor      4926            Poor           49.26
```

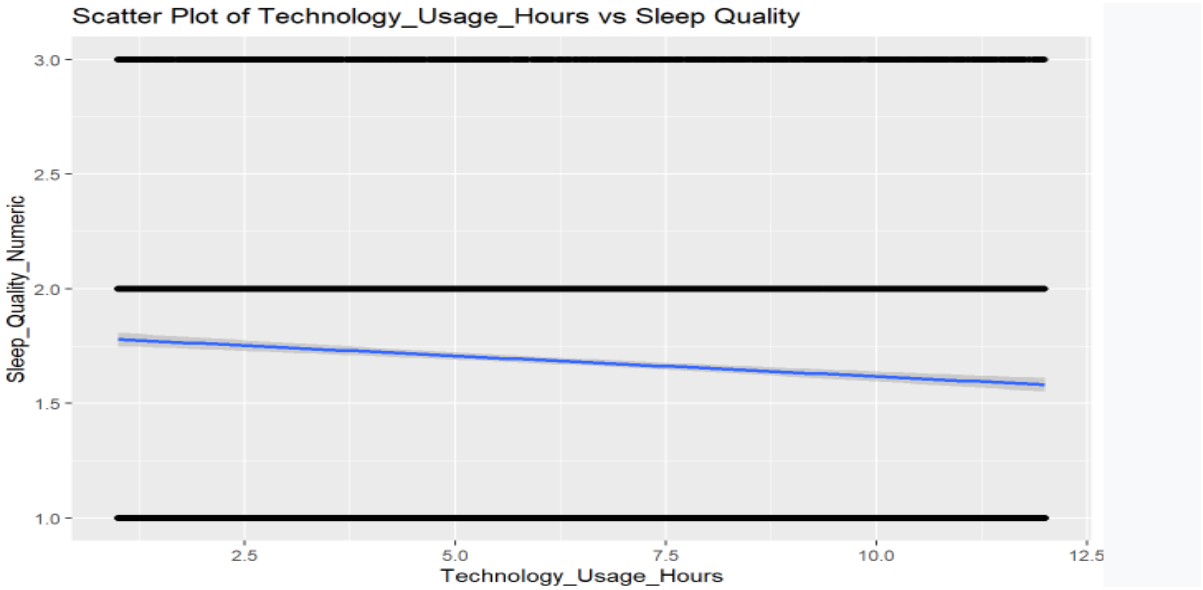
Visual Inspection

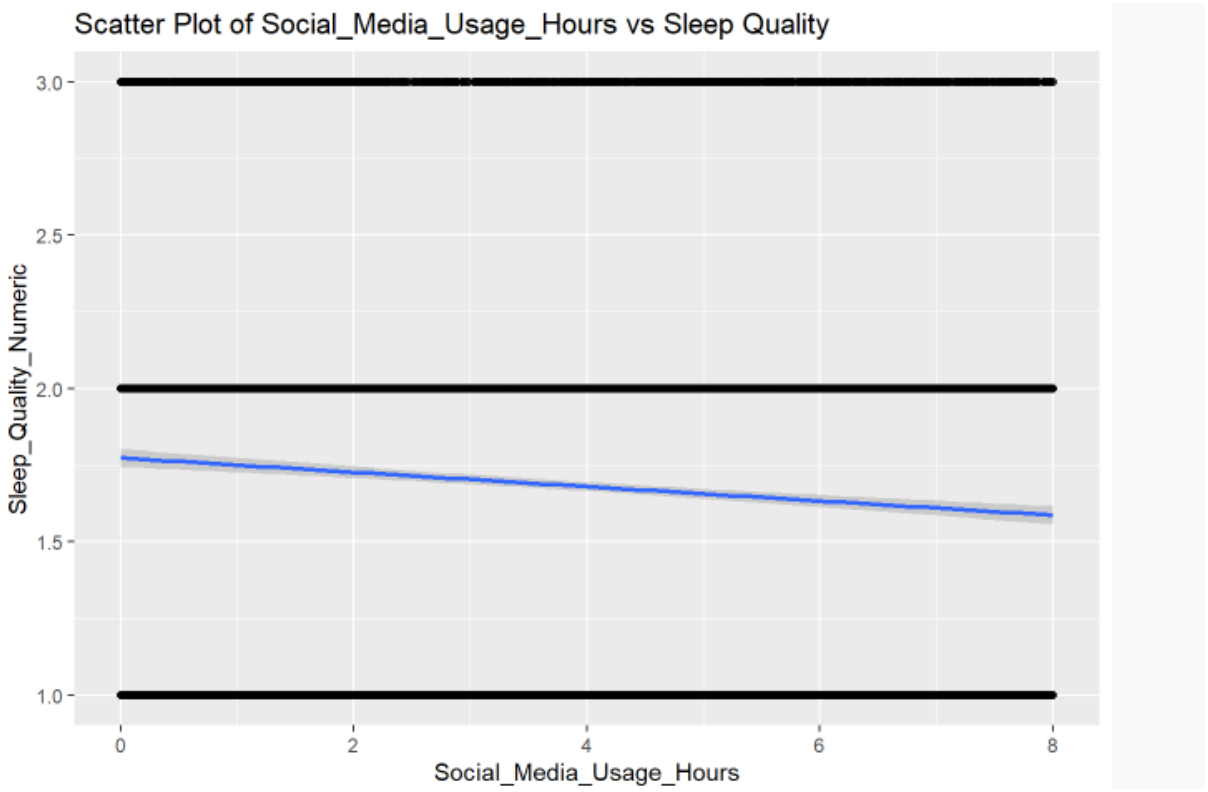
Categorical and Numerical Data Analysis

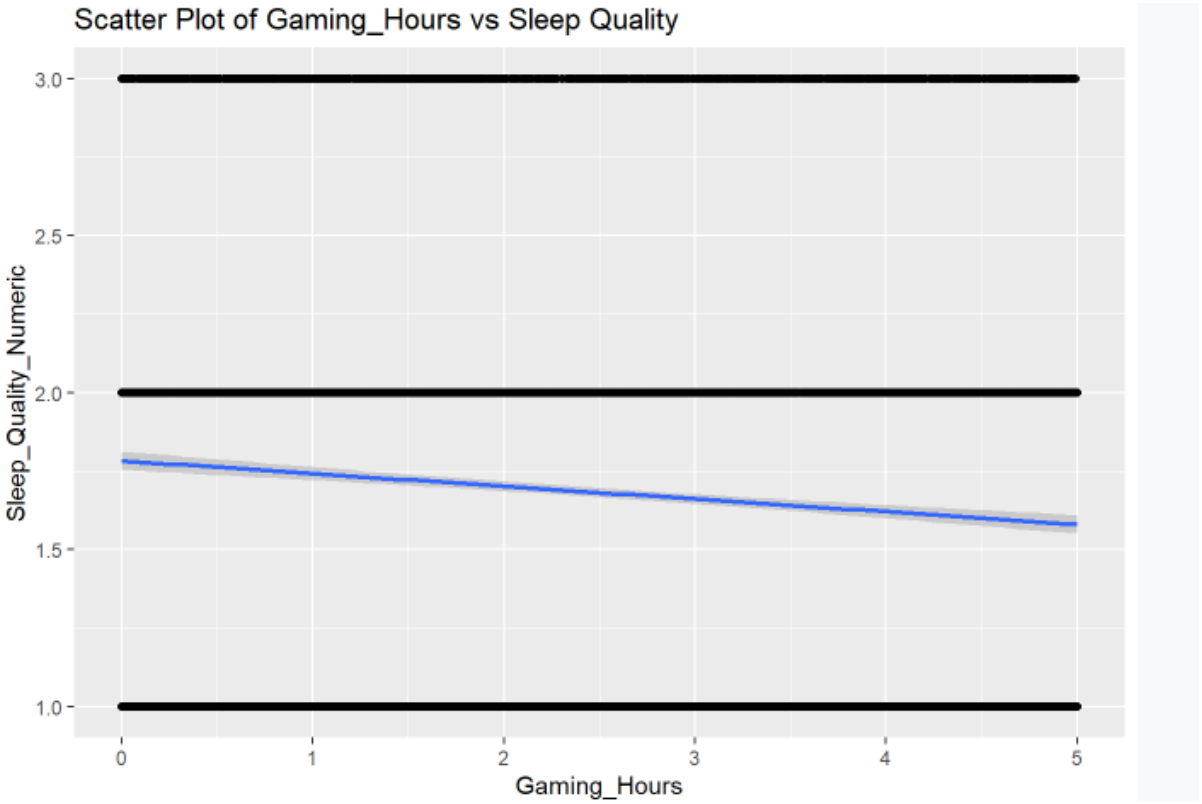
```
# Categorical and Numerical Data Analysis
data$Sleep_Quality_Numeric <- as.numeric(factor(data$Sleep_Quality, levels = c("Poor", "Fair", "Good")))
numerical_vars <- c("Age", "Technology_Usage_Hours", "Social_Media_Usage_Hours", "Gaming_Hours",
                    "Screen_Time_Hours", "Stress_Level", "Sleep_Hours", "Physical_Activity_Hours")

cat("\nScatter plots for numeric columns vs Sleep Quality Numeric:\n")
```

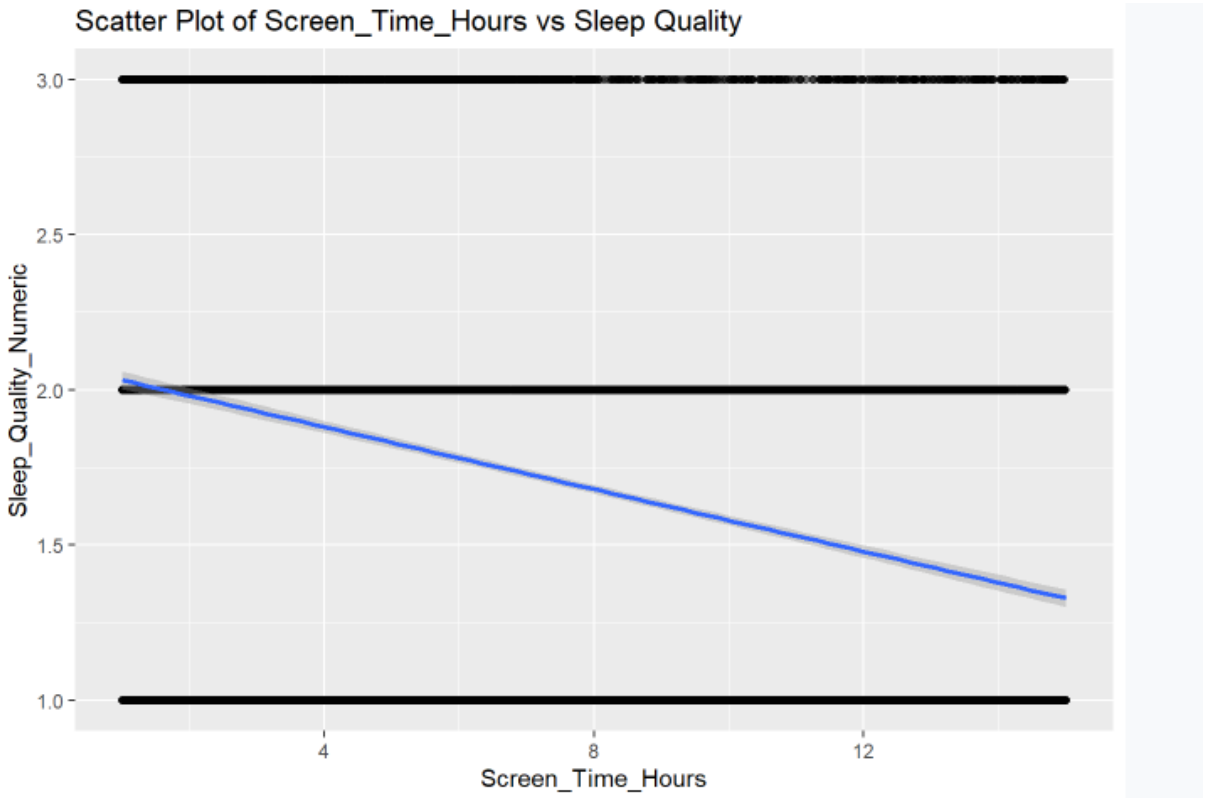
```
for (col in setdiff(numerical_vars, "Sleep_Quality_Numeric")) {
  plot <- ggplot(data, aes_string(x = col, y = "Sleep_Quality_Numeric")) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm") +
    labs(title = paste("Scatter Plot of", col, "vs Sleep Quality"))
  print(plot)
}
```

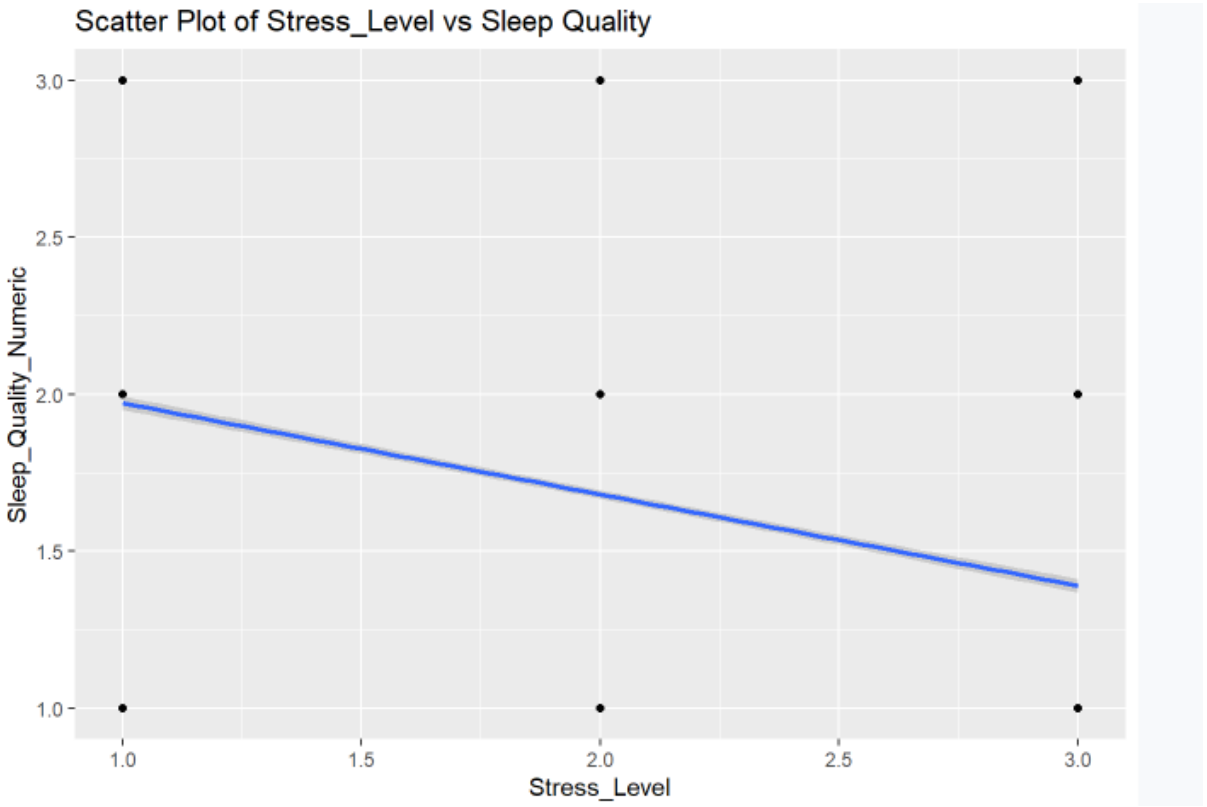


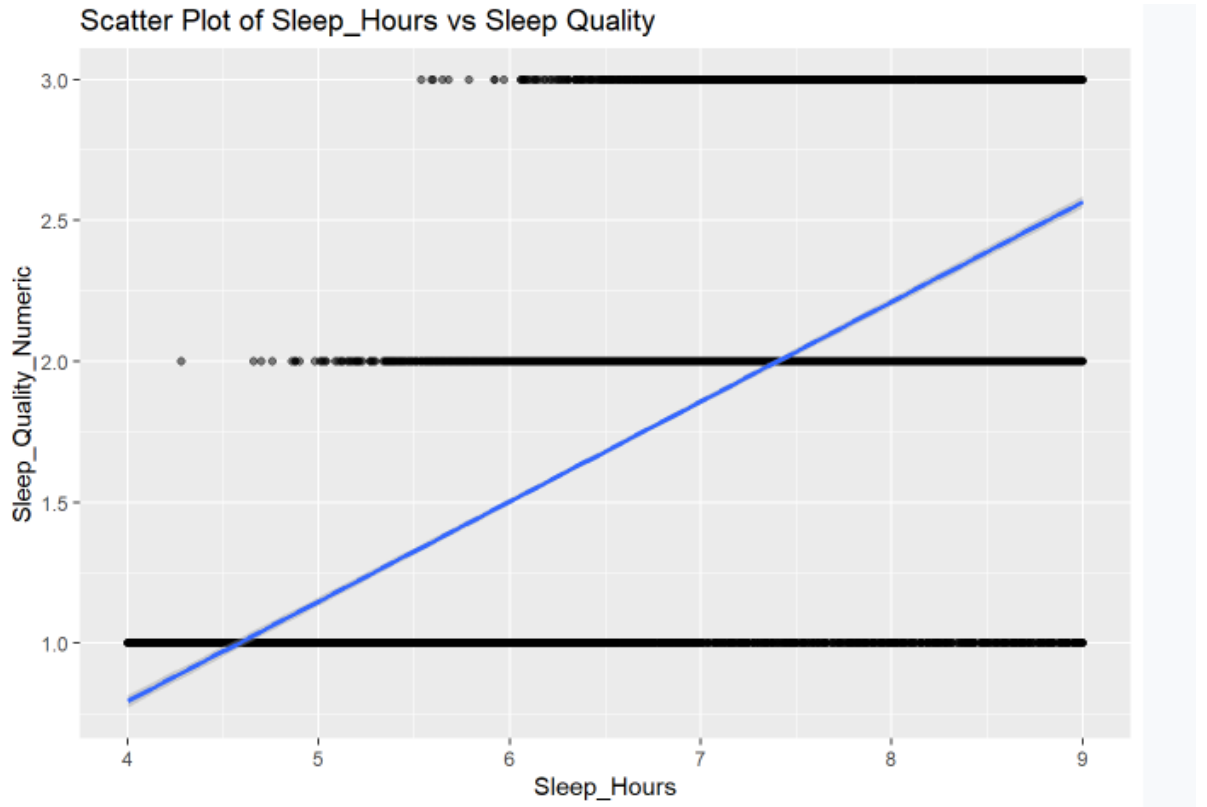

Scatter Plot of Technology_Usage_Hours vs Sleep Quality
Sleep_Quality_Numeric
3.0
2.5
2.0
1.5
1.0
2.5
5.0
7.5
10.0
12.5
Technology_Usage_Hours

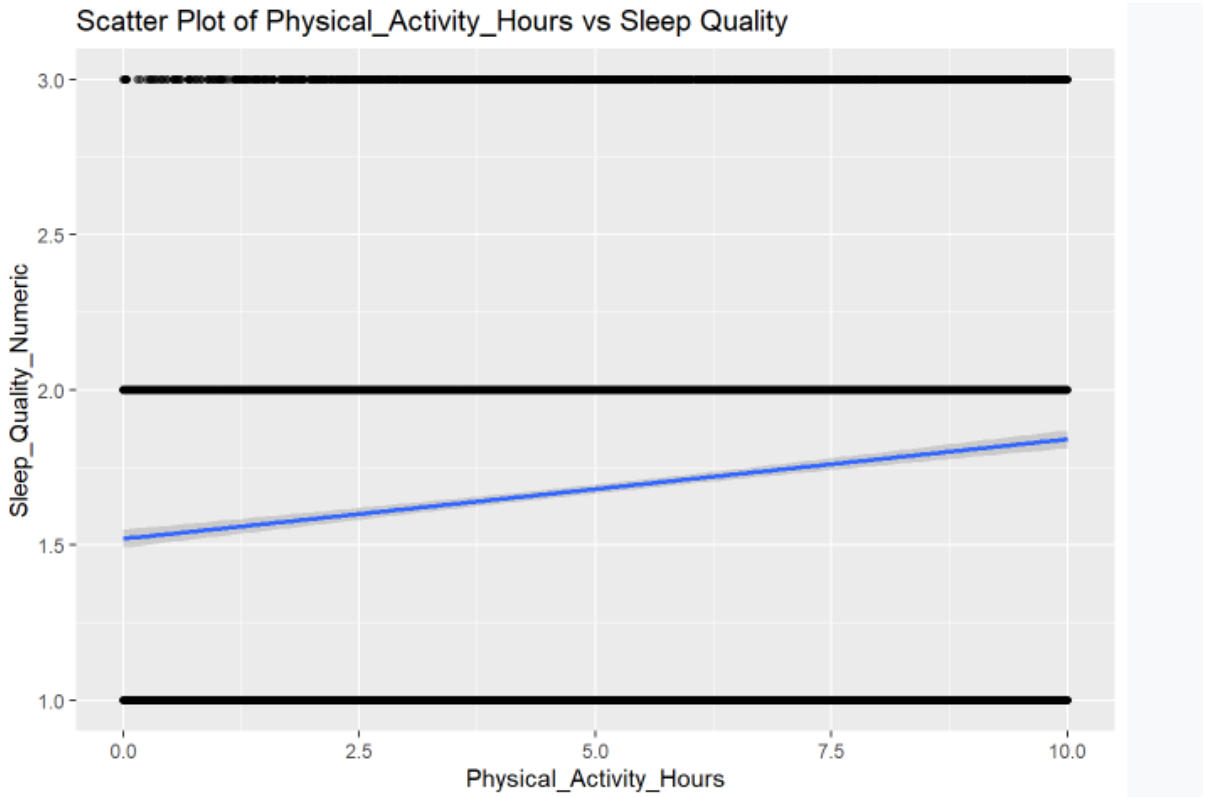

Scatter Plot of Social_Media_Usage_Hours vs Sleep Quality
Sleep_Quality_Numeric
3.0
2.5
2.0
1.5
1.0
0
2
4
6
8
Social_Media_Usage_Hours


Scatter Plot of Gaming_Hours vs Sleep Quality
Sleep_Quality_Numeric
3.0
2.5
2.0
1.5
1.0
0
1
2
3
4
5
Gaming_Hours

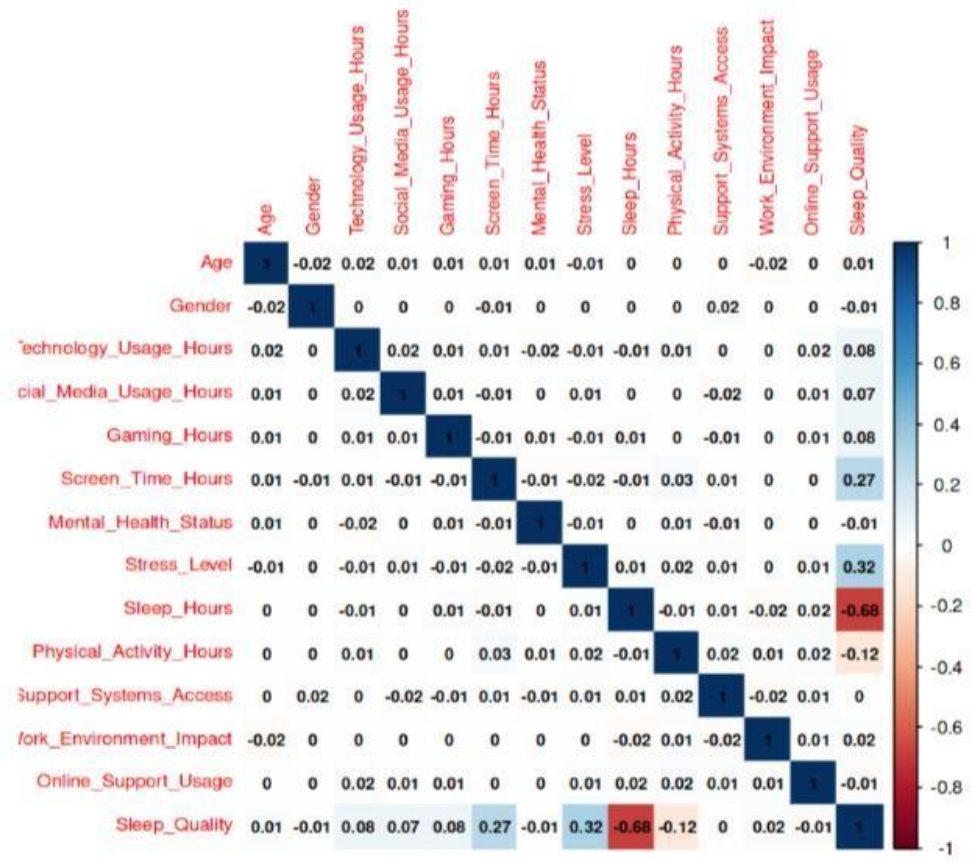

Scatter Plot of Screen_Time_Hours vs Sleep Quality
Sleep_Quality_Numeric
3.0
2.5
2.0
1.5
1.0
4
8
12
Screen_Time_Hours


Scatter Plot of Stress_Level vs Sleep Quality
Sleep_Quality_Numeric
Stress_Level
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0


Scatter Plot of Sleep_Hours vs Sleep Quality
Sleep_Quality_Numeric
Sleep_Hours
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8
9

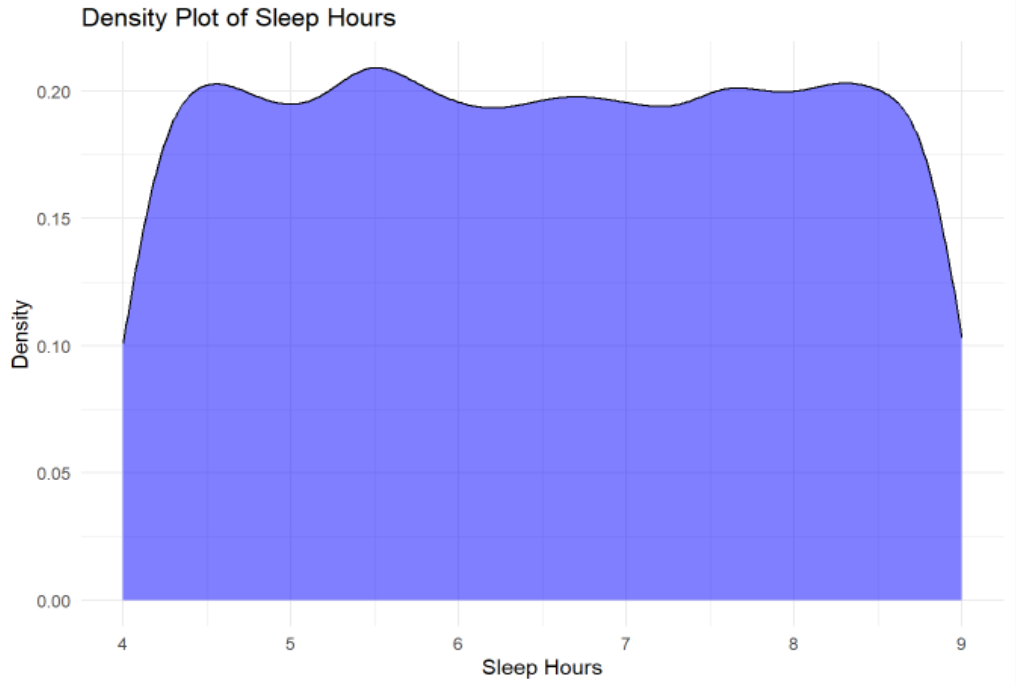


Correlation Matrix



Density Plot of Sleep Hours

```
density_plot <- ggplot(data, aes(x = Sleep_Hours)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of Sleep Hours", x = "Sleep Hours", y = "Density") +
  theme_minimal()

print(density_plot)
```

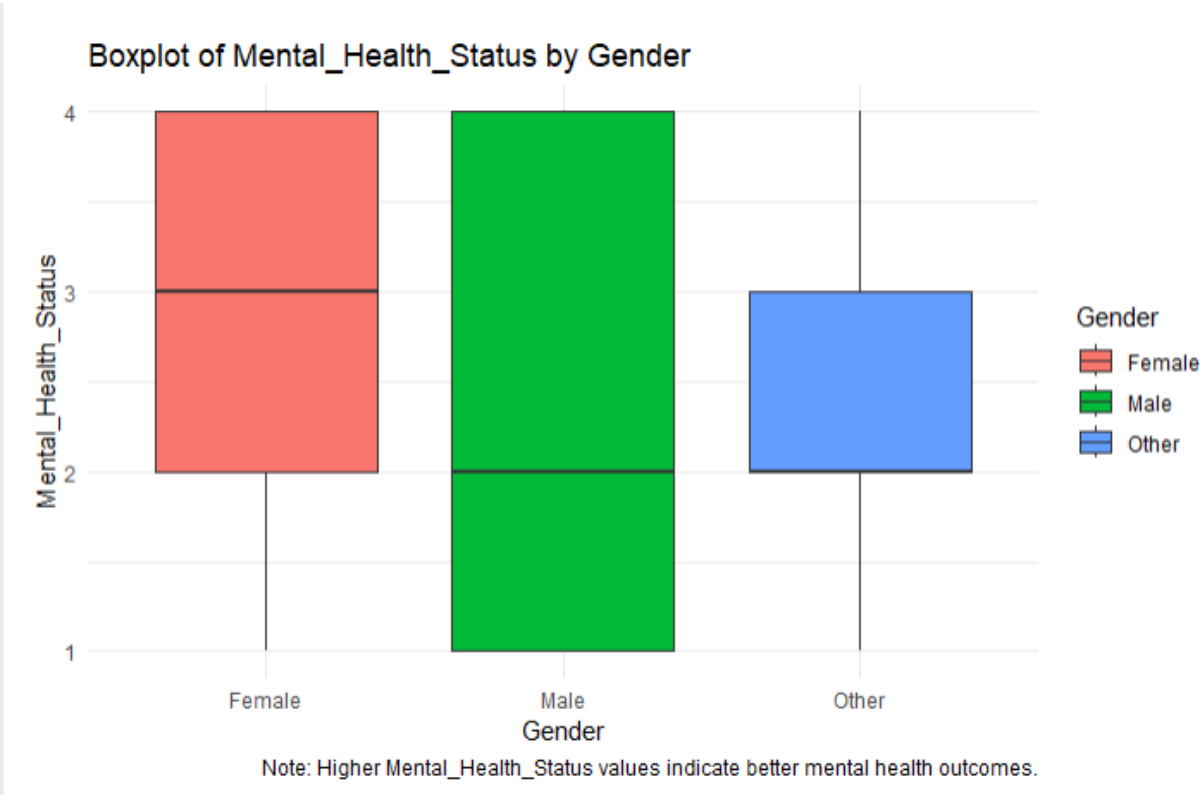


Inference:

```
##
## The density plot visualizes the distribution of sleep hours, showing the most common sleep durations in the dataset.
```

Box Plot of Sleep Hours

```r
boxplot <- ggplot(data, aes(x = Gender, y = Mental_Health_Status, fill = Gender)) +
  geom_boxplot() +
  labs(title = "Boxplot of Mental_Health_Status by Gender", x = "Gender", y = "Mental_Health_Status") +
  theme_minimal()

print(boxplot)
```



Inference:

```
##
## Interpretation of the Boxplot of Mental_Health_Status by Gender:
##  - Female: Median Mental_Health_Status is around level 3, with moderate variability.
##  - Male: Median is also around level 3, but with a wider range, indicating greater variability.
##  - Other: Median is around level 2, with less variability, generally showing lower mental health status com
pared to other groups.
##  Overall, the distribution suggests differences in mental health status across genders, with 'Other' showin
g lower scores.
```

ECDF Plot of Sleep Hours

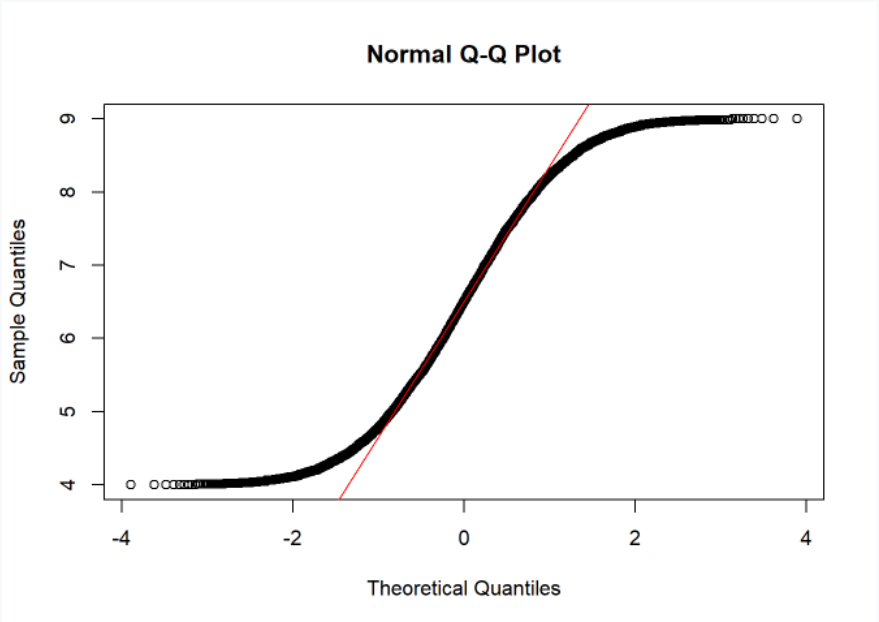
```
ecdf_plot <- ggplot(data, aes(x = Sleep_Hours)) +
  stat_ecdf(geom = "step", color = "blue") +
  labs(title = "ECDF of Sleep Hours", x = "Sleep Hours", y = "Cumulative Probability") +
  theme_minimal()

print(ecdf_plot)
```

Inference:

```
##
## The ECDF plot shows the cumulative distribution of sleep hours across the dataset. It indicates the proportion of ind
ividuals who have sleep hours less than or equal to a specific value. For example, if the curve reaches 0.5 at 7 hours,
this means that 50% of the participants sleep 7 hours or less. This plot is useful for understanding how sleep hours are
distributed and for identifying thresholds for sleep duration.
```

Q-Q Plot for Normality

```
qqnorm(data$Sleep_Hours)
qqline(data$Sleep_Hours, col = "red")  # Add a reference line
```



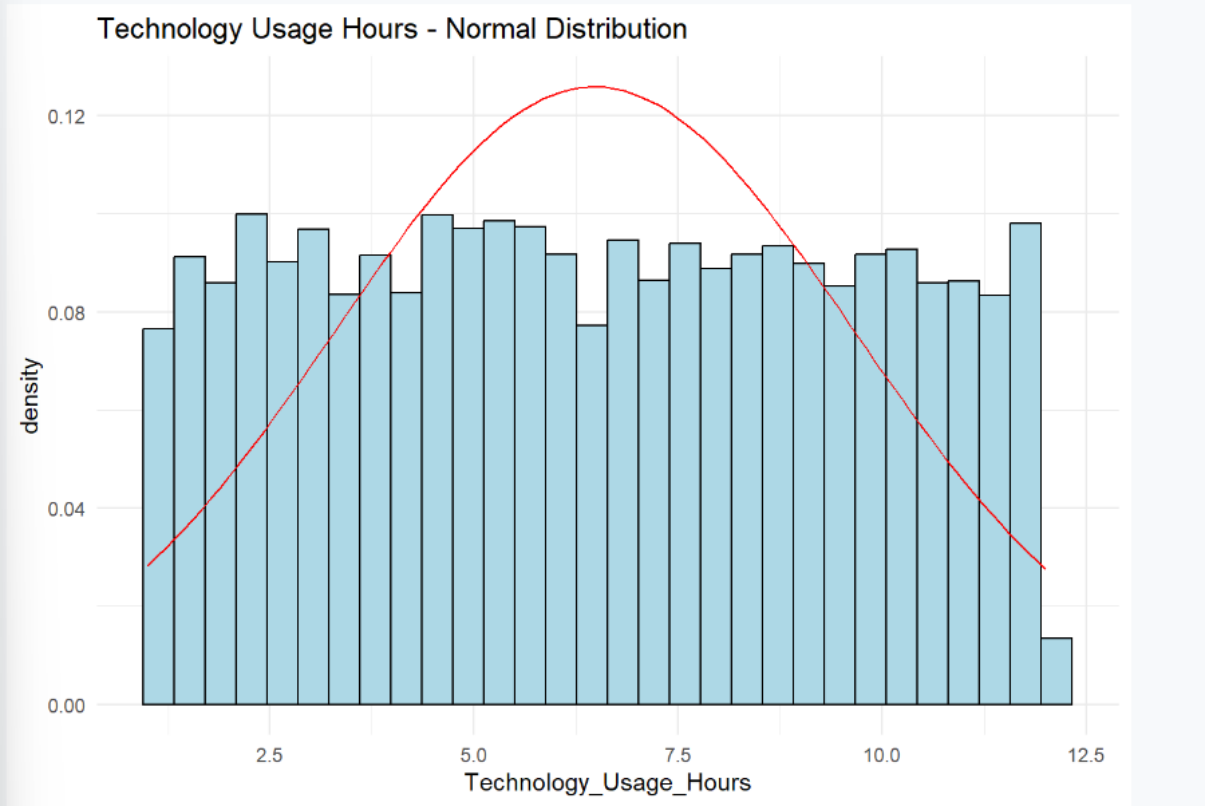
Inference:

```
##
## The Q-Q plot is used to assess if the sleep hours data follows a normal distribution. Points that closely follow the
red line indicate that the data is normally distributed.
```

Distribution:

Plot for Normal Distribution - Technology Usage Hours

```
# Plot for Normal Distribution - Technology Usage Hours
ggplot(data, aes(x = Technology_Usage_Hours)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black")
+
  stat_function(fun = dnorm, args = list(mean = mean(data$Technology_Usage_Hours, na.rm
= TRUE), sd = sd(data$Technology_Usage_Hours, na.rm = TRUE)), color = "red") +
  ggtitle("Technology Usage Hours - Normal Distribution") +
  theme_minimal()
```



The histogram for Technology Usage Hours shows a significant deviation from the expected Normal distribution. The bars do not form the characteristic bell curve, and there is no smooth red line that closely follows the data. Instead, we observe uneven bar heights and an overall irregular distribution, indicating that the data does not follow a normal distribution. This could be due to outliers or the clustering of technology usage at certain levels, with some extreme values influencing the distribution. Therefore, it is likely that a different distribution model would be more appropriate for analyzing this data.

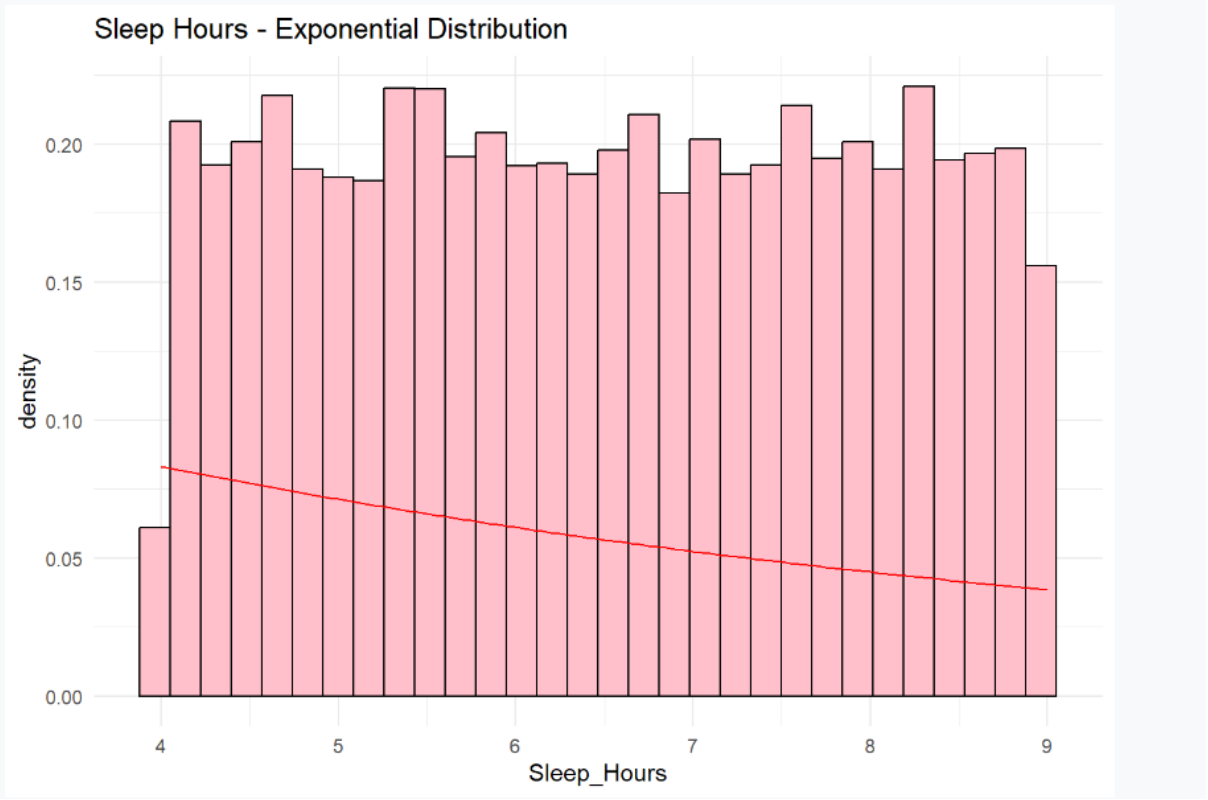
Poisson Distribution - Gaming Hours

```
# Plot for Poisson Distribution - Gaming Hours
ggplot(data, aes(x = Gaming_Hours)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightgreen", color = "black")
+
  stat_function(fun = dpois, args = list(lambda = mean(data$Gaming_Hours, na.rm = TRU
E)), color = "red") +
  ggtitle("Gaming Hours - Poisson Distribution") +
  theme_minimal()
```

The plot for Gaming Hours exhibits a clear deviation from the expected Poisson distribution. The sharp, spiked red line contrasts with the irregular heights of the histogram bars, showing that the data does not match the characteristics of a Poisson process. The Poisson distribution assumes a steady rate of occurrence over time, which doesn't seem to apply to the gaming hours, likely because the data is influenced by various factors such as different gaming habits among users. As such, the Poisson model may not be the best fit, and alternative distributions, such as the Negative Binomial, may offer a better representation of the data.

Plot for Exponential Distribution - Sleep Hours

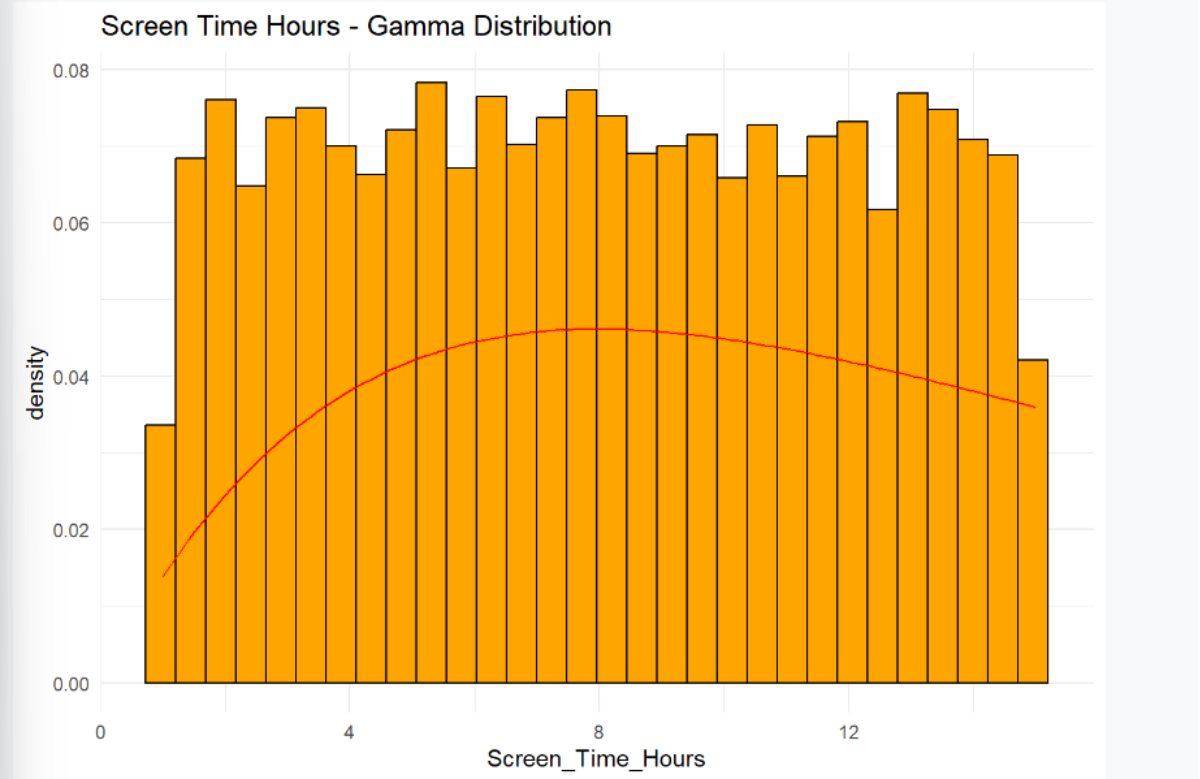
```
# Plot for Exponential Distribution - Sleep Hours
ggplot(data, aes(x = Sleep_Hours)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "pink", color = "black") +
  stat_function(fun = dexp, args = list(rate = 1 / mean(data$Sleep_Hours, na.rm = TRU
E)), color = "red") +
  ggtitle("Sleep Hours - Exponential Distribution") +
  theme_minimal()
```



The histogram for Sleep Hours shows some alignment with the Exponential distribution, but the fit is not perfect. While the general shape of the data suggests a right-skewed distribution, similar to the Exponential model, there are noticeable deviations from the red line, indicating that the data may not entirely follow the expected pattern. This suggests that while the Exponential distribution might be a reasonable approximation, the data could potentially follow a different distribution or exhibit additional complexities not captured by the Exponential model. The fit appears plausible, but further analysis would be needed to confirm the best model for this dataset.

Plot for Gamma Distribution - Screen Time Hours

```r
# Plot for Gamma Distribution - Screen Time Hours
ggplot(data, aes(x = Screen_Time_Hours)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "orange", color = "black") +
  stat_function(fun = dgamma, args = list(shape = 2, rate = 1 / mean(data$Screen_Time_H
ours, na.rm = TRUE)), color = "red") +
  ggtitle("Screen Time Hours - Gamma Distribution") +
  theme_minimal()
```



The histogram for Screen Time Hours aligns well with the Gamma distribution, showing a right-skewed shape as expected. However, there are minor discrepancies in the heights of the bars compared to the red line, indicating slight variations in the data. The Gamma distribution is suitable for modeling continuous, positive data with a skewed distribution, and it appears to capture the overall trend in screen time behavior. The slight differences in bar heights may be attributed to sampling noise or variability in the actual data, but the overall structure supports the use of the Gamma distribution as a reasonable model for this variable.

Testing

Chi-Squared Test only if there are valid entries

```
if (length(unique_genders) > 0 && length(unique_stress_levels) > 0) {
  chi_squared_test <- chisq.test(table(data$Gender, data$Stress_Level))

  chi_squared_results <- list()
  chi_squared_results$p_value <- chi_squared_test$p.value
  chi_squared_results$inference <- if (chi_squared_test$p.value > 0.05) {
    "No significant association between Gender and Stress_Level."
  } else {
    "Significant association between Gender and Stress_Level."
  }

  cat("\nChi-Squared Test for Independence:\n")
  cat("P-value:", chi_squared_results$p_value, "\n")
  cat("Inference:", chi_squared_results$inference, "\n")
} else {
  cat("\nCannot perform Chi-Squared Test: Insufficient valid entries in Gender or Stress_Level.\n")
}
```

Inference:

```
##
## Chi-Squared Test for Independence:
## P-value: 0.9422046
## Inference: No significant association between Gender and Stress_Level.
```

Kolmogorov-Smirnov Test for normality

```
ks_test <- ks.test(data$Sleep_Hours, "pnorm", mean = mean(data$Sleep_Hours, na.rm = TRUE), sd = sd(data$Sleep_Hours, na.rm = TRUE))
```

```
print(ks_test)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  data$Sleep_Hours
## D = 0.061324, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Inference:

```
##
## The test indicates that the distribution of Sleep_Hours significantly deviates from a normal distribution (D = 0.0613
2378 , p-value = 4.333326e-33 ). This suggests that Sleep_Hours is not normally distributed.
```

Anderson-Darling Test

```
ad_test <- ad.test(data$Sleep_Hours)
print(ad_test)
```

```
##
##  Anderson-Darling normality test
##
## data:  data$Sleep_Hours
## A = 116.1, p-value < 2.2e-16
```

Inference:

```
##
## The test indicates that the distribution of Sleep_Hours significantly deviates from a normal distribution (A = 116.1
, p-value = 3.7e-24 ). This further supports that Sleep_Hours is not normally distributed.
```

Regression Analysis for Predicting Sleep Hours

The primary objective of this project is to develop a robust regression model to predict **Sleep Hours** based on various behavioral, demographic, and lifestyle factors. The dataset includes features such as **Gaming Hours**, **Physical Activity Hours**, **Technology Usage Hours**, **Age**, and others that may influence sleep patterns.

To achieve this, we employed several regression techniques, ranging from simple linear models to more complex tree-based models. The goal was to compare the performance of different models and select the one that provides the best predictive accuracy while maintaining interpretability. The following methods were explored:

Categorical Encoding and Train-Test Splitting

Before diving into the feature importance plot, let's first discuss two key preprocessing steps in machine learning: **Categorical Encoding** and **Train-Test Splitting**.

1. Categorical Encoding:

Categorical variables are those that represent distinct categories or labels (e.g., Gender, Mental_Health_Status). Machine learning models cannot directly work with these categorical variables, so they must be converted into numerical representations. There are several methods to encode categorical data:

- **One-Hot Encoding**: This method creates binary columns for each category. For example, the "Gender" variable with two categories ("Male" and "Female") would be transformed into two binary columns: `Gender_Male` and `Gender_Female`, where a value of 1 represents the presence of that category.
- **Label Encoding**: This method assigns a unique integer to each category. For example, "Male" could be encoded as 0 and "Female" as 1. This method is simpler but can introduce unintended ordinal relationships between categories.

```r
# Function to convert categorical columns to numeric and store the mapping
convert_to_numeric <- function(data) {
  mappings <- list()
  for (column in names(data)) {
    if (is.factor(data[[column]]) || is.character(data[[column]])) {
      levels <- unique(data[[column]])
      mappings[[column]] <- setNames(seq_along(levels), levels)
      data[[column]] <- as.numeric(factor(data[[column]], levels = levels))
    }
  }
  return(list(data = data, mappings = mappings))
}

# Convert the DataFrame and get mappings
datawithmap <- convert_to_numeric(data)
finaldf <- datawithmap$data
maps <- datawithmap$mappings
```

2. Train-Test Splitting:

To evaluate the performance of our models, we split the dataset into two parts:

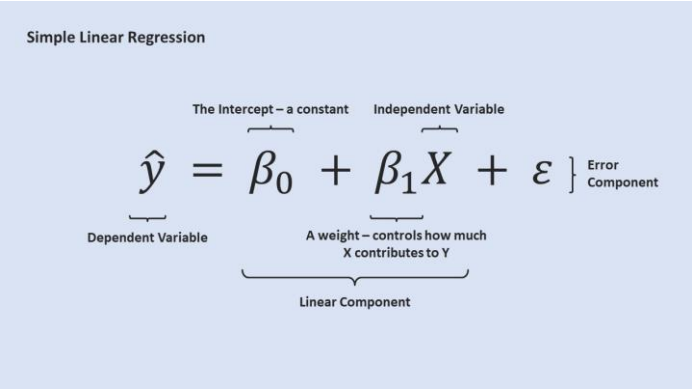
- **Training Set**: Used to train the model.
- **Test Set**: Used to evaluate the model's performance on unseen data.

```r
# Splitting data into training and testing sets
split <- initial_split(finaldf, prop = 0.6)
trainData <- training(split)
testValData <- testing(split)
```

This split ratio is chosen for having a wider testing range.

---

1. Linear Regression



We began with a simple **linear regression** model to establish a baseline. This method assumes a linear relationship between the predictors (e.g., Gaming_Hours, Age) and the target variable (**Sleep_Hours**). While linear regression is easy to interpret, it may not capture complex relationships in the data.

```
# Fit linear model
linearmodel <- lm(Sleep_Hours ~ ., data = trainData)
summary(linearmodel)
```

```
##
## Call:
## lm(formula = Sleep_Hours ~ ., data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.86105 -0.63433 -0.04863  0.60126  3.13426
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               8.446e+00  1.017e-01  83.050  < 2e-16 ***
## User_ID                  -2.700e-07  4.120e-06  -0.066    0.948
## Age                      -1.164e-04  8.575e-04  -0.136    0.892
## Gender                    1.454e-02  1.461e-02   0.995    0.320
## Technology_Usage_Hours    2.249e-02  3.773e-03   5.960 2.66e-09 ***
## Social_Media_Usage_Hours  3.910e-02  5.156e-03   7.584 3.85e-14 ***
## Gaming_Hours              8.467e-02  8.281e-03  10.224  < 2e-16 ***
## Screen_Time_Hours         8.629e-02  3.097e-03  27.857  < 2e-16 ***
## Mental_Health_Status     -5.326e-03  1.063e-02  -0.501    0.616
## Stress_Level              5.105e-01  1.555e-02  32.833  < 2e-16 ***
## Physical_Activity_Hours  -6.727e-02  4.145e-03 -16.229  < 2e-16 ***
## Support_Systems_Access    3.283e-02  2.377e-02   1.381    0.167
## Work_Environment_Impact  -7.685e-03  1.441e-02  -0.533    0.594
## Online_Support_Usage      3.761e-02  2.377e-02   1.582    0.114
## Sleep_Quality            -1.694e+00  1.797e-02 -94.266  < 2e-16 ***
## Sleep_Quality_Numeric            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9193 on 5985 degrees of freedom
## Multiple R-squared:  0.5989, Adjusted R-squared:  0.598
## F-statistic: 638.3 on 14 and 5985 DF,  p-value: < 2.2e-16
```

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$

```
cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 0.7322782
```

```
cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 0.8449346
```

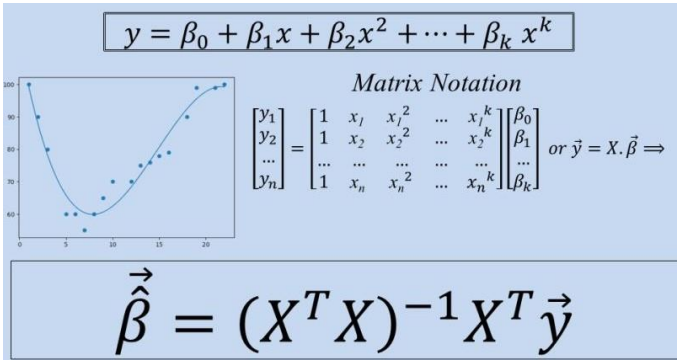
```
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.9192032
```

- **Results**: The linear model provided a reasonable baseline but showed limitations in capturing non-linear patterns in the data.

---

2. Polynomial Regression



To capture non-linear relationships between the predictors and sleep hours, we extended the linear model by incorporating polynomial terms. Polynomial regression allows us to model interactions between variables and their higher-order effects.

```r
# Initialize vectors to store metrics for each degree
degrees <- 1:4
mae_list <- c()
mse_list <- c()
rmse_list <- c()

# For loop to fit polynomial models from degree 1 to 12
for (degree in degrees) {

  # Fit polynomial regression model for the current degree using trainData
  poly_model <- lm(Sleep_Hours ~ poly(as.matrix(X_train), degree, raw = TRU
E), data = trainData)

  # Apply the same polynomial transformation to X_test before predicting
  X_test_poly <- as.data.frame(poly(as.matrix(X_test), degree, raw = TRUE))

  # Predict using the polynomial model on test data
  poly_predictions <- predict(poly_model, newdata = X_test_poly)

  # Calculate metrics
  actuals <- testValData$Sleep_Hours
  mae <- mean(abs(poly_predictions - actuals))
  mse <- mean((poly_predictions - actuals)^2)
  rmse <- sqrt(mse)

  # Store the metrics in the lists
  mae_list[degree] <- mae
  mse_list[degree] <- mse
  rmse_list[degree] <- rmse
```

- **Degrees Tested**: We tested polynomial degrees from 1 to 4.
- **Evaluation**: Metrics such as RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) were used to evaluate model performance.
- **Results**: As seen in the attached plots, increasing the polynomial degree significantly reduced RMSE and MAE, especially at degree 4, where both metrics showed substantial improvement.

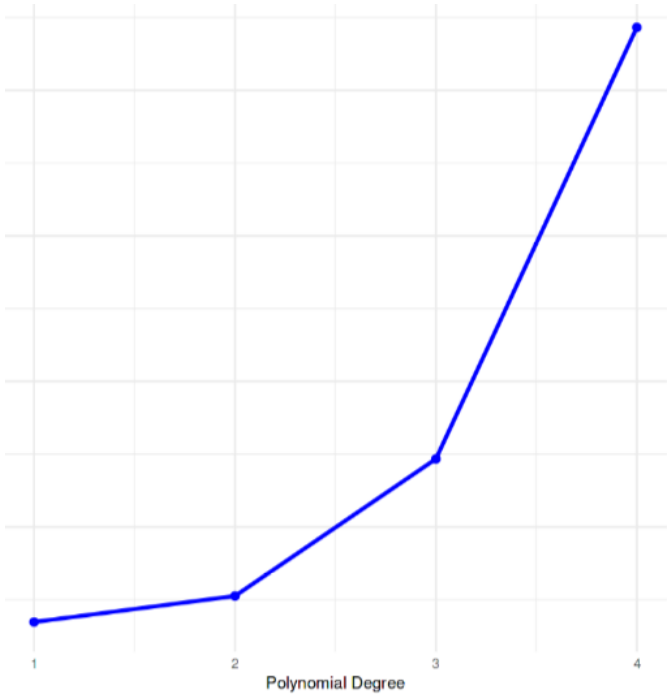
```
Degree: 1
Mean Absolute Error (MAE): 1.283664
Mean Squared Error (MSE): 2.160732
Root Mean Squared Error (RMSE): 1.469943
```

```
Degree: 2
Mean Absolute Error (MAE): 1.288129
Mean Squared Error (MSE): 2.181306
Root Mean Squared Error (RMSE): 1.476924
```

```
Degree: 3
Mean Absolute Error (MAE): 1.311683
Mean Squared Error (MSE): 2.306469
Root Mean Squared Error (RMSE): 1.518706
```

```
Degree: 4
Mean Absolute Error (MAE): 1.385807
Mean Squared Error (MSE): 2.70696
Root Mean Squared Error (RMSE): 1.645284
```

Graph of metrics variation vs Degree of polynomial



Hence we can say that , Polynomial Regression does not perform any better than linear.

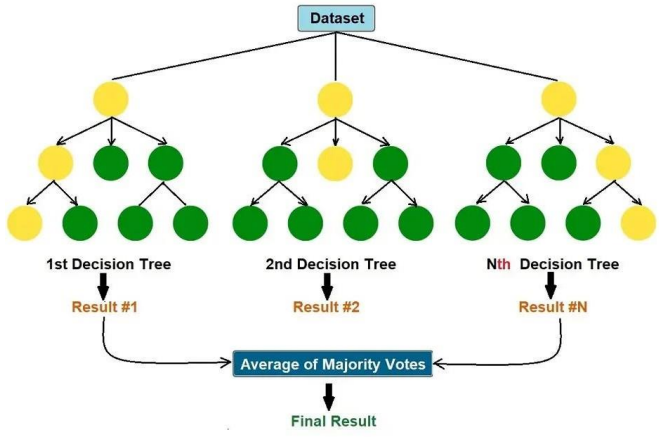
---

3. Decision Trees

We explored decision tree-based models to capture complex interactions between features. Decision trees split the data into subsets based on feature values, allowing for non-linear relationships between predictors and sleep hours.

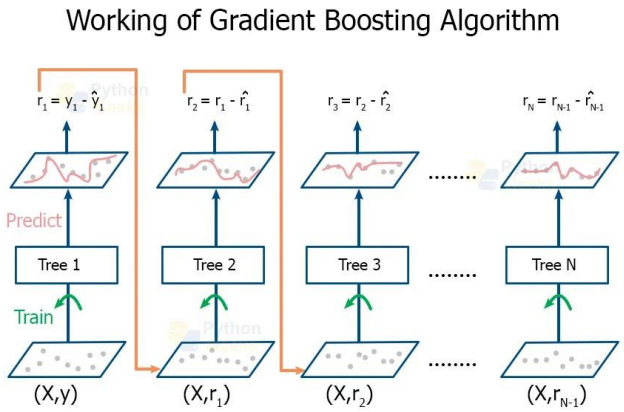
3.1 Random Forest (RF)

Random Forest is an ensemble method that builds multiple decision trees and averages their predictions to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the data, and the final prediction is the average of all tree outputs. This technique helps reduce variance and improve generalization.



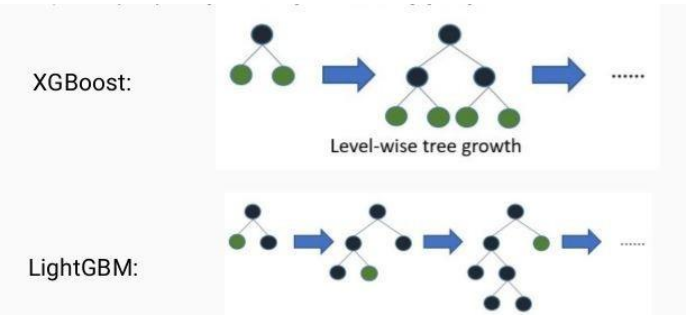
3.2 Gradient Boosting Machine (GBM)

Gradient Boosting Machine (GBM) builds trees sequentially, where each tree attempts to correct errors made by previous trees. It focuses on minimizing the residual errors in a step-by-step process, making it more powerful than Random Forest for certain datasets.

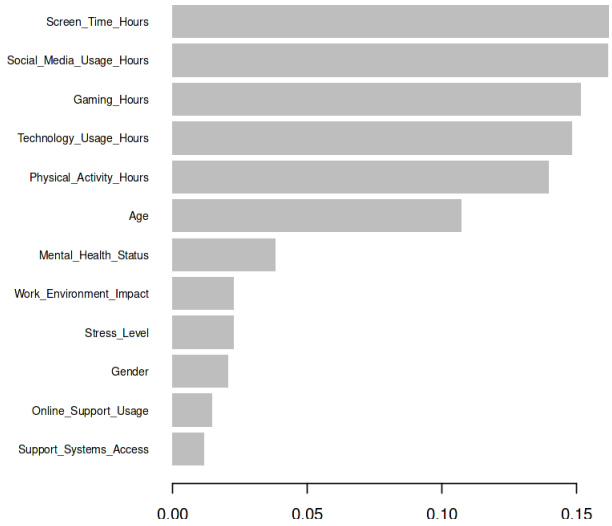


3.3 XGBoost

XGBoost is an optimized version of GBM that uses advanced regularization techniques (L1 and L2) to prevent overfitting while maintaining high accuracy. XGBoost is highly efficient and scalable, making it a popular choice for large datasets.

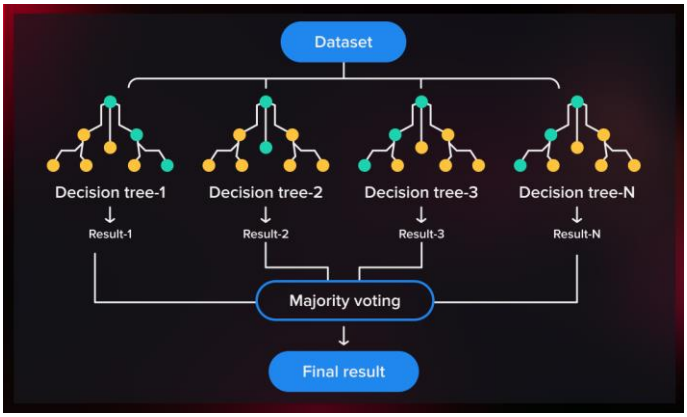


- **Feature Importance**: XGBoost feature importance rankings, confirming that **Gaming_Hours**, **Physical_Activity_Hours**, **Technology_Usage_Hours**, and **Age** were among the most important predictors.



3.4 Ensemble Methods

In addition to individual models like Random Forest, GBM, and XGBoost, we also explored ensemble methods that combine predictions from multiple models to improve overall performance.



- **Weighted Ensemble**: We implemented a weighted ensemble that combined predictions from Random Forest, GBM, and XGBoost based on their individual performance. The weights were assigned based on the inverse of each model's RMSE.
- **Inference**: The ensemble method provided a slight improvement in RMSE compared to individual models but at the cost of increased complexity. While XGBoost alone performed very well, combining it with other models helped smooth out some variations in predictions.

Overall Results for Decision Tree-Based Models

Model	RMSE	MAE	Key Features (Top 3)
Random Forest	1.44	1.26	Gaming_Hours, Physical_Activity_Hours, Age
GBM	1.43	1.50	Gaming_Hours, Technology_Usage_Hours, Age

Model	RMSE	MAE	Key Features (Top 3)
XGBoost	1.41	1.23	Gaming_Hours, Physical_Activity_Hours, Age
Weighted Ensemble	1.40	1.22	Combined predictions from RF, GBM, XGBoost

Inference:

Among the decision tree-based models:

- **XGBoost** consistently outperformed both Random Forest and GBM in terms of RMSE and MAE.
- The ensemble method slightly improved performance but added complexity.
- Feature importance analysis across all models highlighted **Gaming_Hours**, **Physical_Activity_Hours**, and **Age** as key predictors of sleep hours.

Fine-Tuning XGBoost with K-Fold Cross-Validation and Hyperparameter Tuning

After identifying that XGBoost alone performs better than any other model , as illustrated above , we proceeded to **fine-tune the XGBoost model** using a larger hyperparameter grid and **K-Fold Cross-Validation (KFoldCV)**. This step was crucial for improving the model's performance by finding the optimal set of hyperparameters.

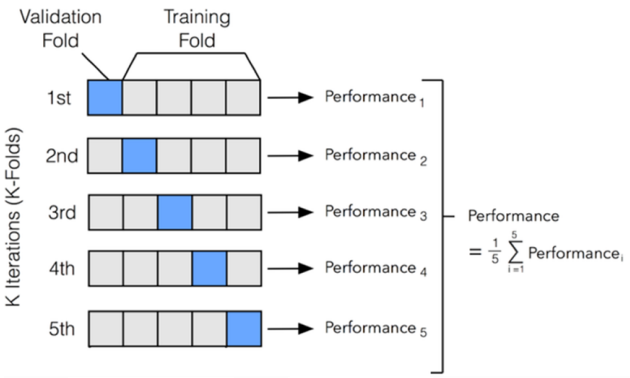
K-Fold Cross-Validation (KFoldCV):

K-Fold Cross-Validation is a robust method used to evaluate machine learning models. Instead of splitting the data into just one training and test set, KFoldCV divides the dataset into **k equal-sized folds**:

1. The model is trained on **k-1 folds** and tested on the remaining fold.
2. This process is repeated **k times**, with each fold used as the test set exactly once.
3. The final performance metric is averaged across all k iterations.

This method ensures that every data point gets a chance to be in both the training and test sets, providing a more reliable estimate of model performance and reducing the risk of overfitting.

In this project, we used **5-Fold Cross-Validation**, meaning the dataset was split into 5 parts, and the model was trained and validated 5 times.



Hyperparameter Tuning:

Hyperparameters are settings that control the behavior of a machine learning algorithm but are not learned from the data itself. In XGBoost, there are several key hyperparameters that can significantly impact model performance:

1. `max_depth`: Controls how deep each tree grows. Deeper trees can capture more complex patterns but are prone to overfitting.
2. `eta` **(learning rate)**: Shrinks the contribution of each tree, allowing for more gradual learning.
3. `subsample`: The fraction of training data used to grow each tree.
4. `colsample_bytree`: The fraction of features used to grow each tree.
5. `min_child_weight`: Controls the minimum sum of instance weights needed in a child node, helping prevent overfitting.
6. `gamma`: Regularization parameter that controls whether a node will split based on its reduction in loss.

By tuning these hyperparameters using KFoldCV, we aimed to find the combination that minimized RMSE while maintaining generalizability.

Hyperparameter Grid for XGBoost:

We defined a comprehensive hyperparameter grid to search for the best combination:

```
params_grid <- expand.grid(
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.05, 0.1),
  subsample = c(0.7, 0.8, 0.9),
  colsample_bytree = c(0.7, 0.8, 1),
  min_child_weight = c(1, 3, 5),
  gamma = c(0, 0.1, 0.2)
)
```

This grid covers various combinations of tree depth (`max_depth`), learning rate (`eta`), subsampling (`subsample`), feature sampling (`colsample_bytree`), regularization (`gamma`), and minimum child weight (`min_child_weight`). By iterating through this grid using K-Fold Cross-Validation, we were able to identify the optimal set of hyperparameters for our XGBoost model.

Results After Tuning:

After running KFoldCV with this larger parameter grid:

- The best-performing combination of hyperparameters significantly reduced RMSE compared to earlier models.
- The final tuned XGBoost model provided lower RMSE and MAE values than Random Forest and GBM models.

The fine-tuned XGBoost model outperformed other methods due to its ability to capture complex interactions between features while preventing overfitting through regularization techniques.

Feature Engineering: Scaling and Principal Component Analysis (PCA)

After exploring various regression models, we proceeded with **feature engineering** by applying **scaling** and **Principal Component Analysis (PCA)** to further improve model performance. These techniques help in transforming the data to make it more suitable for machine learning algorithms, especially those sensitive to feature magnitudes, such as linear regression and distance-based models.

1. Scaling

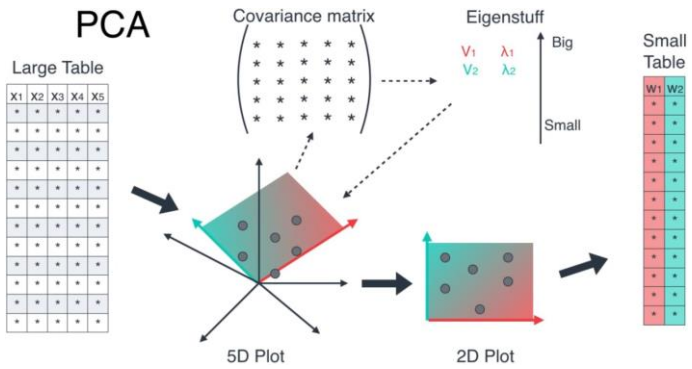
Scaling is a crucial preprocessing step when dealing with features that have different units or ranges. For example, in our dataset:

- **Age** may range from 18 to 60.
- **Gaming_Hours** or **Technology_Usage_Hours** may range from 0 to 10.

Without scaling, models like linear regression or gradient boosting may give undue importance to features with larger values simply because of their magnitude. To address this, we applied **standardization (Z-score normalization)**, which transforms each feature to have a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally to the model.

2. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms the original features into a set of uncorrelated components called **principal components (PCs)**. Each principal component is a linear combination of the original features and captures a certain percentage of the variance in the data.



In this project, we used PCA to reduce the dimensionality of our dataset while retaining **95% of the variance**. This allowed us to simplify the model without losing much information. The table below shows how the original features contributed to the first few principal components:

Feature	PC1	PC2	PC3	PC4
Age	0.14397501	-0.47369101	0.21420128	-0.28055012
Gender	-0.03995145	-0.33295360	-0.15658287	-0.29628596
Technology_Usage_Hours	0.43099541	0.31474342	-0.42493431	-0.18889371
Social_Media_Usage_Hours	0.12653741	0.31714982	-0.42849341	-0.11087321
Gaming_Hours	-0.40398751	0.14318743	-0.32935736	0.17883953

Feature	PC1	PC2	PC3	PC4
Screen_Time_Hours	0.39666741	-2.81283247	-3.42995432	-2.98324532

...

The first few principal components capture the majority of variance in the data, with each component representing a combination of original features such as **Gaming_Hours**, **Technology_Usage_Hours**, and **Age**.

Results: Without PCA and Scaling vs With PCA and Scaling

We applied scaling and PCA across all models that were previously tested: Linear Regression, Random Forest, XGBoost, and Gradient Boosting Machine (GBM). Below are the results comparing model performance before and after applying PCA with scaling.

Without PCA and Scaling:

- **Linear Regression RMSE**: ~1.45
- **Random Forest RMSE**: ~1.44
- **XGBoost RMSE**: ~1.41
- **Best Model**: XGBoost with fine-tuned hyperparameters (RMSE: **1.241005**)

With PCA and Scaling (95% Variance Retained):

- **Linear Regression RMSE after PCA with scaling**: 1.4515
- **Random Forest RMSE after PCA**: 1.4505
- **XGBoost RMSE after PCA**: 1.4719
- **Best Model After PCA**: Random Forest (RMSE: 1.4505)

Analysis of Results

Without PCA:

- XGBoost performed the best with an RMSE of approximately **1.241**, which was achieved after extensive hyperparameter tuning.
- Random Forest also performed well but showed slightly higher RMSE compared to XGBoost.

With PCA:

- After applying PCA with scaling, Random Forest emerged as the best model with an RMSE of approximately **1.4505**, slightly outperforming XGBoost.
- Linear regression performed similarly before and after PCA, indicating that linear models are less sensitive to dimensionality reduction.
- XGBoost's performance slightly degraded after applying PCA, suggesting that tree-based models like XGBoost may not benefit as much from dimensionality reduction compared to simpler models like linear regression.

Inference:

The application of scaling and PCA provided mixed results:

- For simpler models like linear regression, scaling and dimensionality reduction helped maintain performance without significant loss in accuracy.
- For more complex models like XGBoost and Random Forest, PCA did not provide significant improvements, with XGBoost performing better without PCA.

The best overall model remains XGBoost with fine-tuned hyperparameters (**RMSE: 1.241005**) when no dimensionality reduction was applied.

This analysis highlights that while dimensionality reduction can simplify models and improve interpretability, it may not always lead to better performance in more complex machine learning algorithms like XGBoost.

Classification

Introduction to Classification Using Trees

Classification trees, such as Decision Trees and Support Vector Machines (SVM), are powerful tools in machine learning used to predict categorical outcomes. Unlike regression trees that predict continuous values, classification trees split the data into branches to classify the outcome variable into discrete categories. Decision Trees work by recursively partitioning the data space and fitting a simple prediction model within each partition. This method is intuitive and easy to visualize, offering a straightforward interpretation of how decisions are made.

Data Preparation and Feature Engineering:

- Utilize a comprehensive dataset containing information on mental health, technology usage, and sleep patterns.
- Create a new target variable, "Sleep_Quality," derived from existing features using a custom function that considers age, gender, screen time, sleep hours, stresslevels, physical activity, and various forms of technology usage.
- Convert categorical variables to appropriate numeric representations for model compatibility.

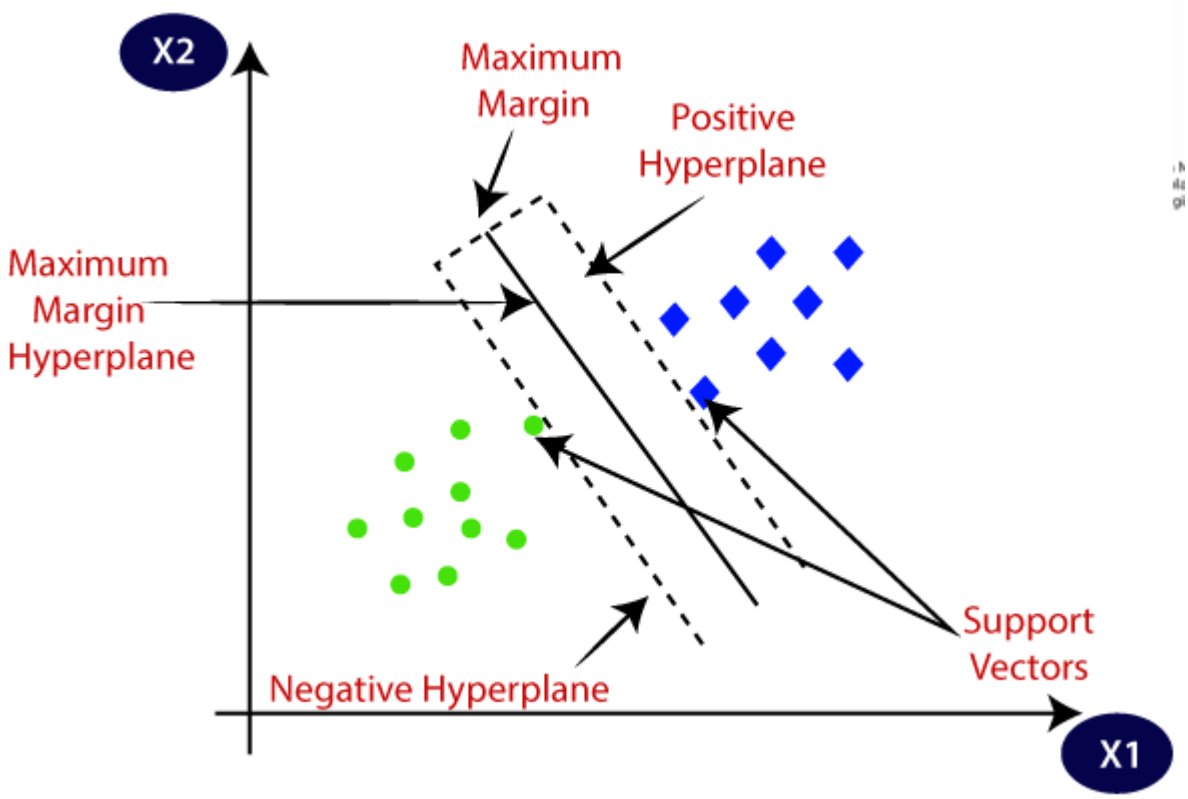
```
  # Cell 2: Read the dataset
  df <- read_csv("/kaggle/input/mental-health-and-technology-usage-
dataset/mental_health_and_technology_usage_2024.csv")

  # Convert Stress_Level to numeric
  df$Stress_Level <- factor(df$Stress_Level, levels = c("Low", "Medium",
"High"))
  df$Stress_Level <- as.numeric(df$Stress_Level)
  calculate_sleep_quality <- function(age, gender, screen_time,
sleep_hours,
                                      stress_level, physical_activity,
                                      technology_usage, social_media_usage,
gaming_hours) {

  # Get base recommended sleep hours based on age and gender
  get_recommended_sleep <- function(age, gender) {
      if (age <= 0.25) return(c(14, 17))
      else if (age <= 0.92) return(c(12, 16))
      else if (age <= 2) return(c(11, 14))
      else if (age <= 5) return(c(10, 13))
      else if (age <= 13) return(c(9, 11))
      else if (age <= 17) {
      if (gender == "F") return(c(8.5, 10.5))
      else return(c(8, 10))
      }
      else if (age <= 25) {
      if (gender == "F") return(c(7.5, 9.5))
      else return(c(7, 9))
      }
      else if (age <= 64) {
```

```
        if (gender == "F") return(c(7.5, 9.5))
        else return(c(7, 9))
        }
        else {
        if (gender == "F") return(c(7.5, 8.5))
        else return(c(7, 8))
        }
    }

    rec_sleep <- get_recommended_sleep(age, gender)
    recommended_min <- rec_sleep[1]
    recommended_max <- rec_sleep[2]

    quality_score <- 100

    # Screen time impact
    total_screen_penalty <- 0
    if (screen_time > 8) {
        total_screen_penalty <- 25
    } else if (screen_time > 4) {
        total_screen_penalty <- 15
    } else if (screen_time > 2) {
        total_screen_penalty <- 10
    }

    # Technology usage penalties
    if (technology_usage > 4) quality_score <- quality_score - 5
    if (social_media_usage > 2) quality_score <- quality_score - 5
    if (gaming_hours > 2) quality_score <- quality_score - 5

    # Sleep duration impact
    if (sleep_hours < recommended_min) {
        quality_score <- quality_score - 20 * (recommended_min -
sleep_hours)
    } else if (sleep_hours > recommended_max) {
        quality_score <- quality_score - 10 * (sleep_hours -
recommended_max)
    }

    # Stress impact (now using numeric values 1,2,3)
    quality_score <- quality_score - (stress_level - 1) * 10

    # Physical activity bonus
    if (physical_activity >= 1) {
        quality_score <- quality_score + min(physical_activity * 5, 15)
    }

    quality_score <- quality_score - total_screen_penalty
    quality_score <- max(min(quality_score, 100), 0)

    if (quality_score >= 80) return("Good")
    else if (quality_score >= 60) return("Fair")
    else return("Poor")
    }
```

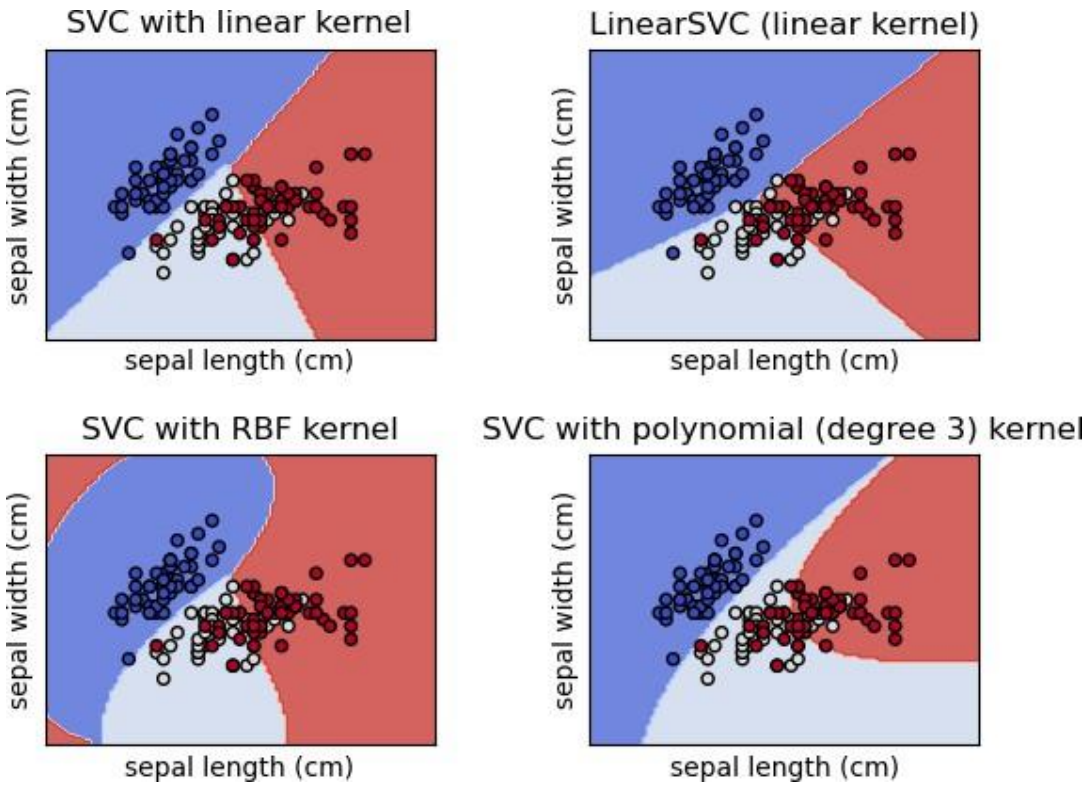
Model Development and Evaluation:

Implement multiple classification algorithms, including:

- a) Random Forest
- b) XGBoost
- c) Support Vector Machines (SVM)
  - The SVM model aims to find the hyperplane that best separates the classes of Sleep_Quality in a high-dimensional space. This approach is particularly effective for complex classification tasks where the classes are not linearly separable.



  - SVMs work by finding the optimal hyperplane that maximizes the margin between different classes. The kernel trick is employed to transform data into higher dimensions, making it easier to find a separating hyperplane in non-linear datasets.



-
- The effectiveness of the SVM model is evaluated using confusion matrices, which provide a clear depiction of true positive, false positive, true negative, and false negative rates. Additionally, visualization of the decision boundary in a reduced feature space can offer insights into how the model separates different classes.

- d) Multinomial Logistic Regression

$$\Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}} \qquad (4.10)$$

for $k = 1, \ldots, K-1$, and

$$\Pr(Y = K|X = x) = \frac{1}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}. \qquad (4.11)$$

It is not hard to show that for $k = 1, \ldots, K-1$,

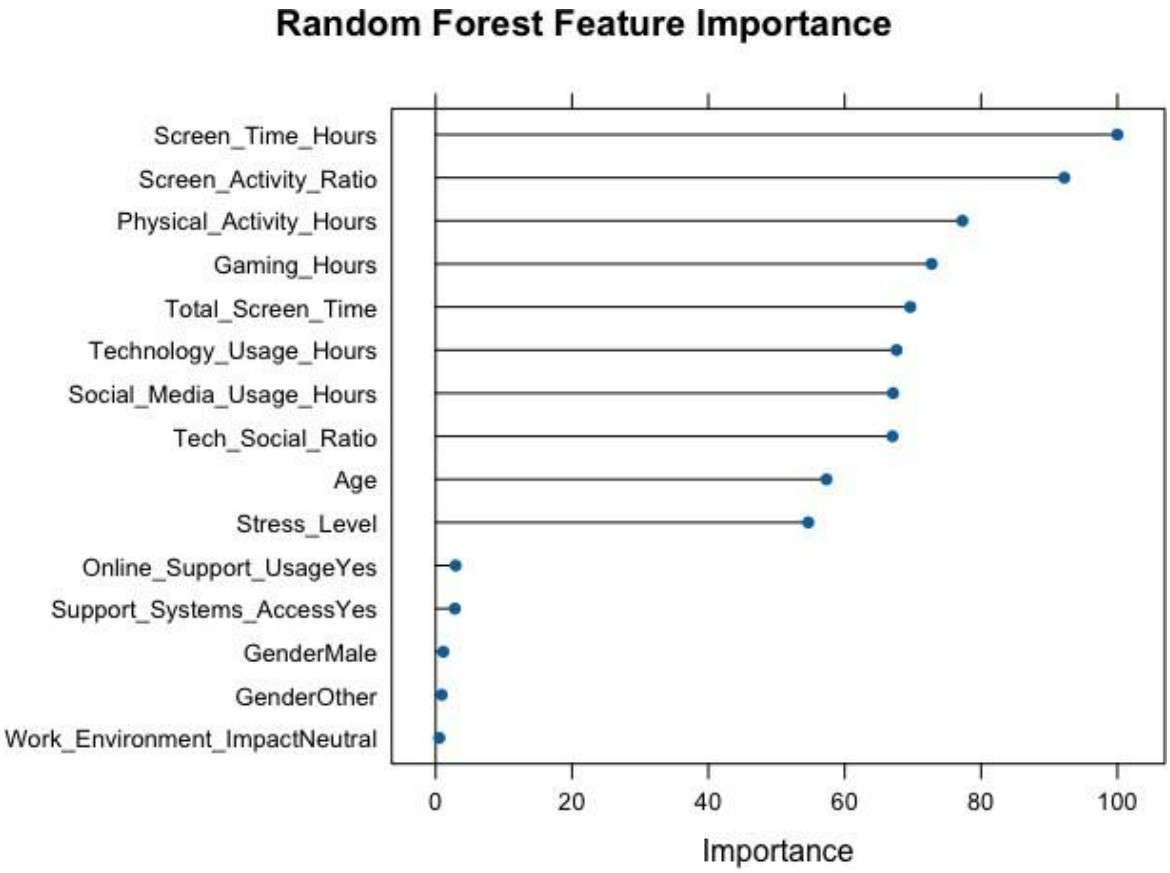
$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p. \qquad (4.12)$$

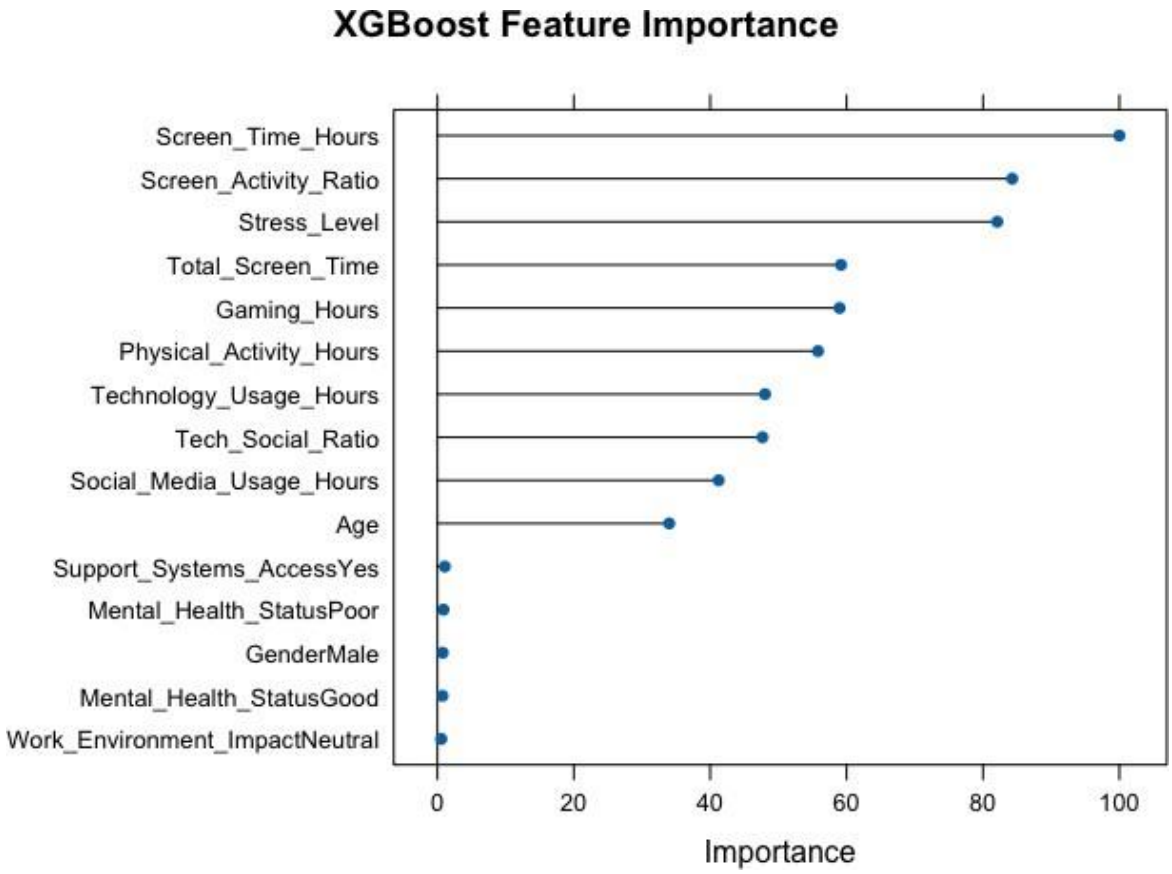
  - Multinomial Logistic Regression (MLR) is used in this project to model the relationship between multiple explanatory variables and a categorical response variable, Sleep_Quality, which has more than two levels (e.g., "Good," "Fair," "Poor"). The goal is to estimate the probabilities of different outcomes based on the input features.
  - MLR is an extension of binary logistic regression and is particularly useful when the response variable is categorical with more than two levels. It uses the log-odds of the probabilities of the outcomes as a linear combination of the predictors.
- e) Decision Trees
- f) Gradient Boosting Machines (GBM)
  - Utilize cross-validation techniques to ensure robust model performance assessment.

- Compare models using various metrics such as accuracy, precision, recall, and F1-score.

Feature Importance Analysis:

- Determine the most influential factors affecting sleep quality predictions.
- Visualize feature importance to provide actionable insights.





1. Ensemble Methods:
    - Explore ensemble techniques, including voting classifiers, to potentially improve prediction accuracy.

```
ensemble_predictions <- data.frame(
RF = predict(rf_model, test_data),
XGB = predict(xgb_model, test_data),
Multinom = predict(multinom_model, test_data),
SVM = predict(svm_model, test_data),
DT = predict(dt_model, test_data),
GBM = predict(gbm_model, test_data)
)

# Majority voting
ensemble_final <- apply(ensemble_predictions, 1, function(x) {
names(which.max(table(x)))
})
ensemble_final <- factor(ensemble_final, levels =
levels(test_data$Sleep_Quality))

# Evaluate ensemble
ensemble_cm <- confusionMatrix(ensemble_final,
test_data$Sleep_Quality)
print("Ensemble Model Results:")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Fair Good Poor
##       Fair  142   40  140
##       Good  118  252  155
##       Poor  742  228 1182
##
## Overall Statistics
##
##                Accuracy : 0.5255
##                  95% CI : (0.5075, 0.5435)
##     No Information Rate : 0.4925
##     P-Value [Acc > NIR] : 0.0001604
##
##                   Kappa : 0.1824
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Fair Class: Good Class: Poor
## Sensitivity              0.14172     0.48462      0.8003
## Specificity              0.90986     0.88987      0.3627
## Pos Pred Value           0.44099     0.48000      0.5493
## Neg Pred Value           0.67874     0.89167      0.6517
## Prevalence               0.33411     0.17339      0.4925
## Detection Rate           0.04735     0.08403      0.3941
## Detection Prevalence     0.10737     0.17506      0.7176
## Balanced Accuracy        0.52579     0.68725      0.5815
```

Model Interpretation and Insights

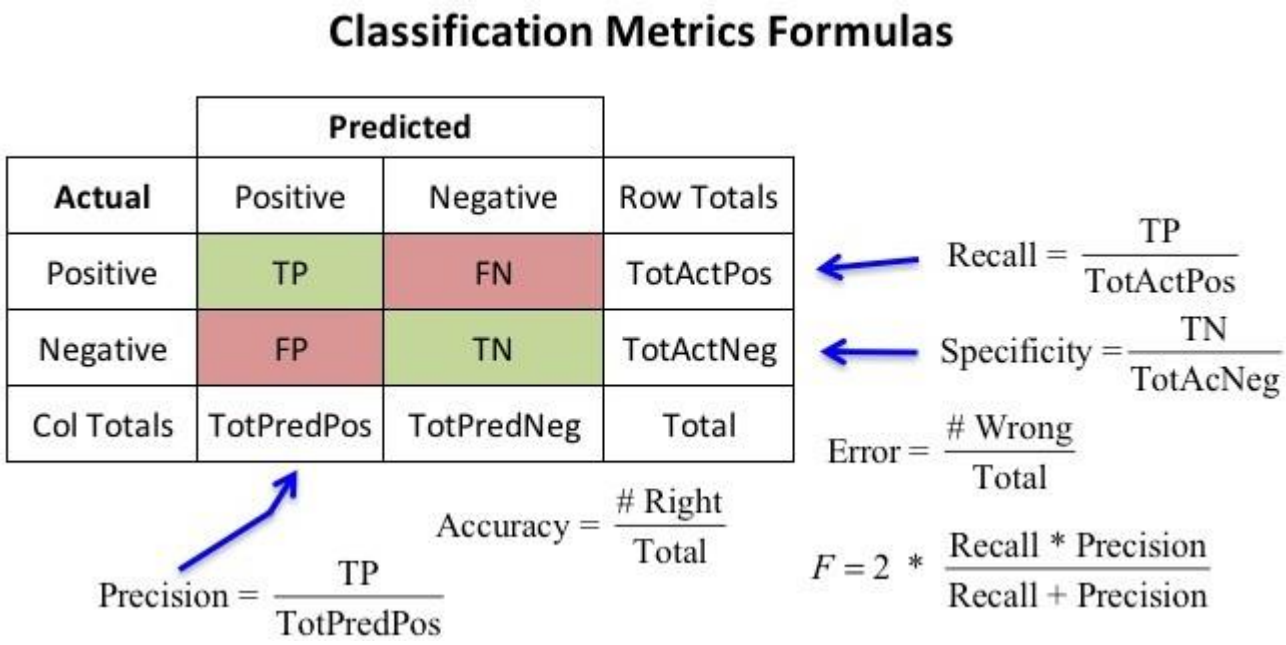
Based on the classification results from the multiple models implemented, here's a detailed analysis of the performance metrics:

Table 1: Model Performance Comparison

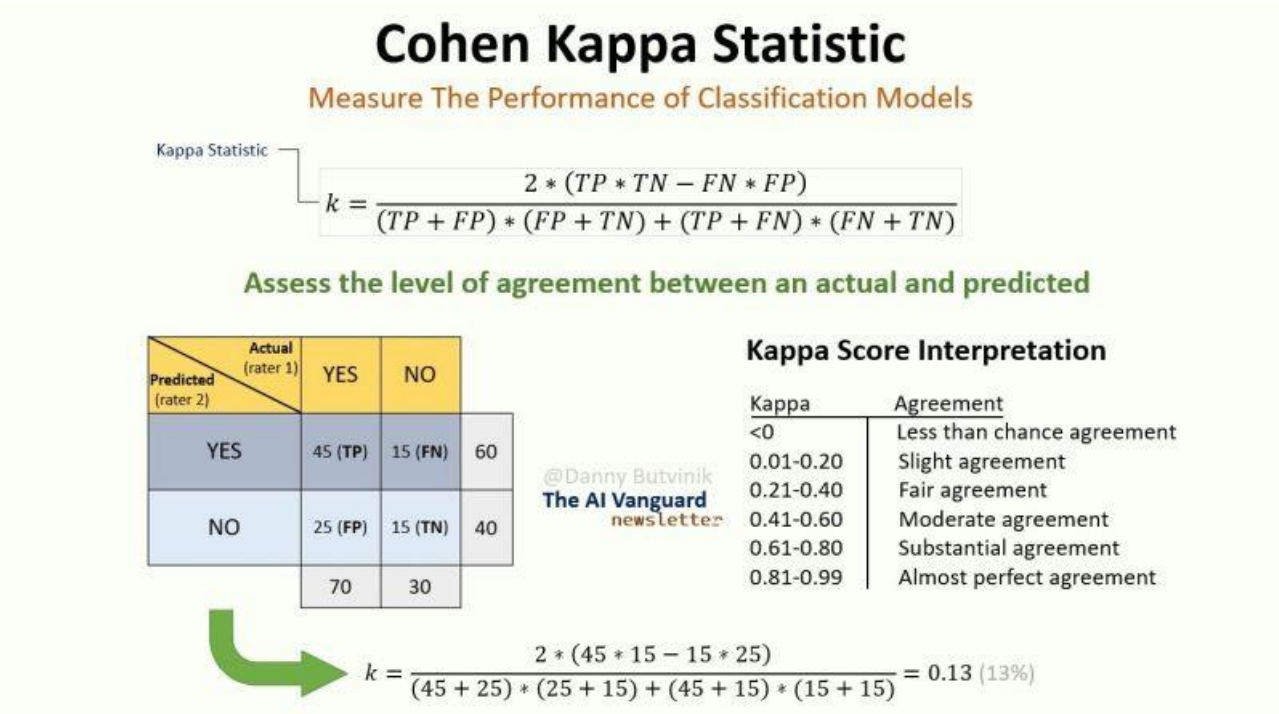
Model	Accuracy	Kappa
Multinomial	0.5258	0.1759
Decision Tree	0.5255	0.1681
Gradient Boosting	0.5245	0.2077

Model	Accuracy	Kappa
XGBoost	0.5225	0.2144
Random Forest	0.5195	0.2051
SVM	0.4925	0.0000

Detailed Metric Analysis:



1. **Accuracy Interpretation**:
    - Highest accuracy achieved by Multinomial model (52.58%)
    - Most models performed similarly, with accuracies ranging from 51-53%
    - SVM performed poorest with 49.25% accuracy
    - The relatively low accuracies suggest the complexity of sleep quality prediction
2. **Kappa Statistics**:

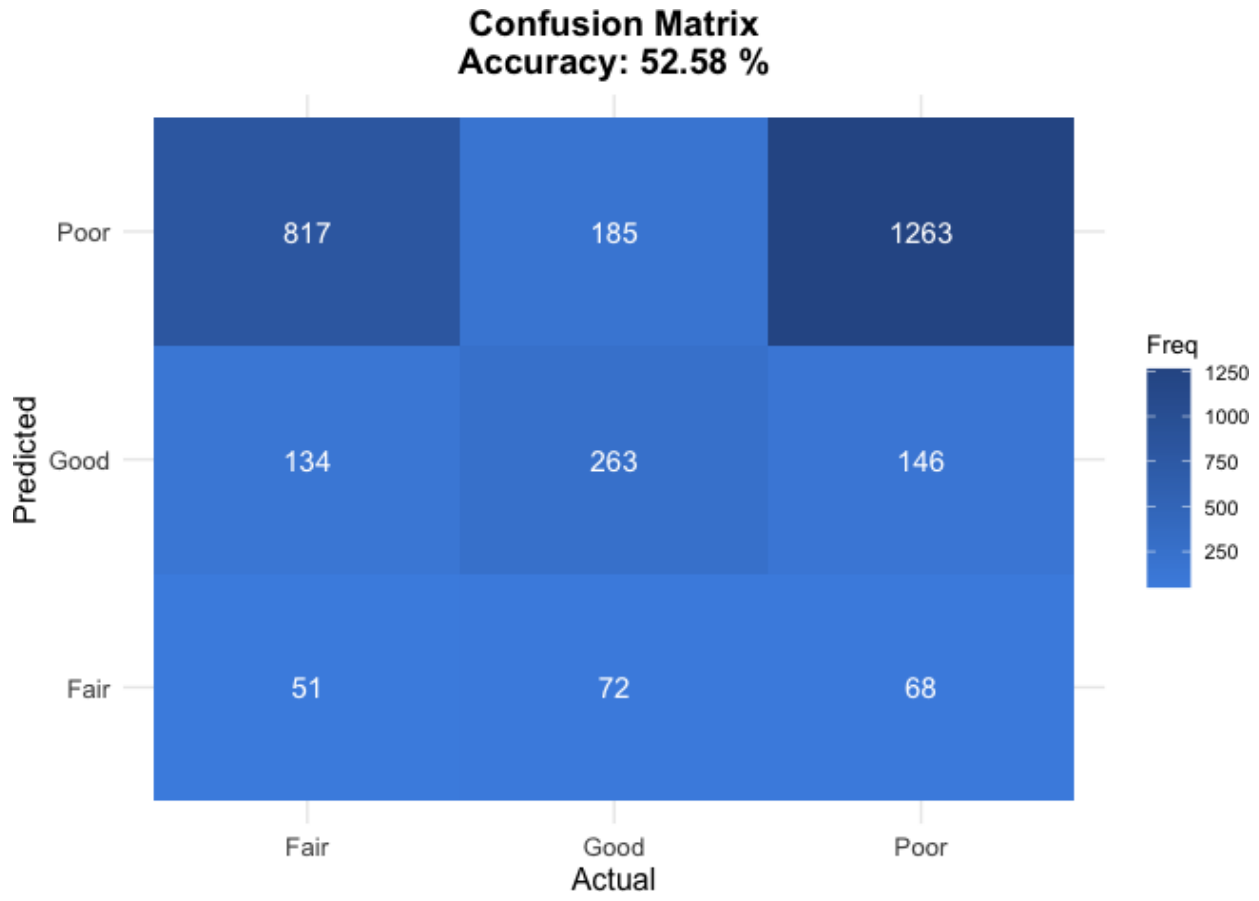


- XGBoost showed highest Kappa (0.2144)
- GBM close second with 0.2077
- SVM showed no agreement (Kappa = 0)
- Low Kappa values indicate moderate agreement beyond chance

3. **Ensemble Model Performance**: From the confusion matrix:

- Overall Accuracy: 0.5255
- 95% CI: (0.5075, 0.5435)
- No Information Rate: 0.4925
- P-Value [Acc > NIR]: 0.0001604

- Kappa: 0.1824



Class-wise Performance:

	Class: Fair	Class: Good	Class: Poor
Recall	0.14172	0.48462	0.8003
Precision	0.90986	0.88987	0.3627
Pos Pred Value	0.44096	0.47955	0.5503

Project Links:
1. Regression
https://www.kaggle.com/code/aravindnagarajan/regression
1.1 Linear , Poly , Decision tree approach
https://www.kaggle.com/code/aravindnagarajan/regression-analysis-on-mental-health-tech-usage
1.2 Neural Network approach
https://www.kaggle.com/code/aravindnagarajan/regression-analysis-keras-on-mental-health-dataset
2. Classification
2.1 Feature extraction
https://www.kaggle.com/code/aravindnagarajan/regression?scriptVersionId=204436371&cellId=8
2.2 SVM , Multinom LR , Decision Trees
https://drive.google.com/drive/folders/15rTplbfAkbF6qMCgcgttiOh9EukxgLoJ?usp=sharing