Predicting Sleep Quality

Analyzing the Impact of Lifestyle Factors on Sleep Health

Jiajia Feng

UCSB Spring 2024

Contents

1. Introduction . . . 3
2. Data Description . . . 3
3. Exploratory Data Analysis (EDA) . . . 7
4. Model Setting up . . . 17
5. Model Building . . . 18
6. Model Result . . . 24
7.Fit the Model to the Training set. . . . 25
8. Test the Model . . . 26
9. Error Analysis . . . 26
9. Conclusion . . . 29
Sources . . . 31

11
12
1
10
2
9
3
8
7

1. Introduction

Background and Motivation:

During my busy college life, I often find that anxiety and stress negatively impact my sleep. In addition to academic pressures, I also notice that exercise affects my sleep duration and quality. Consequently, I usually have to choose between focusing on my studies or on physical activity. I frequently struggle with issues such as insomnia, or sleeping too little or too much. These sleep problems further contribute to my anxiety. This has led me to seek a better understanding of the relationship between sleep and various lifestyle factors. I aim to use a machine learning model to analyze these lifestyle factors and predict sleep quality, thereby gaining a deeper insight into my own sleep patterns. I found a dataset on Kaggle called "Sleep Health and Lifestyle Dataset" that perfectly covers the lifestyle factors I believe are important.

Objective:

In this project I plan to develop a predictive model that assesses sleep quality from various personal health and lifestyle factors. This model aims to identify significant predictors of good sleep and help individuals improve their sleep by understanding how different factors affect their sleep quality.

2. Data Description

Data Loading

```r
sleep_data <- read.csv("Sleep.csv")
head(sleep_data)
```

```
##   Person.ID Gender Age           Occupation Sleep.Duration Quality.of.Sleep
## 1         1   Male  27    Software Engineer            6.1                6
## 2         2   Male  28               Doctor            6.2                6
## 3         3   Male  28               Doctor            6.2                6
## 4         4   Male  28 Sales Representative            5.9                4
## 5         5   Male  28 Sales Representative            5.9                4
## 6         6   Male  28    Software Engineer            5.9                4
##   Physical.Activity.Level Stress.Level BMI.Category Blood.Pressure Heart.Rate
## 1                      42            6   Overweight         126/83         77
## 2                      60            8       Normal         125/80         75
## 3                      60            8       Normal         125/80         75
## 4                      30            8        Obese         140/90         85
## 5                      30            8        Obese         140/90         85
## 6                      30            8        Obese         140/90         85
##   Daily.Steps Sleep.Disorder
## 1        4200           None
## 2       10000           None
## 3       10000           None
## 4        3000    Sleep Apnea
## 5        3000    Sleep Apnea
## 6        3000       Insomnia
```

```r
summary(sleep_data)
```

```
##    Person.ID        Gender               Age         Occupation
##  Min.   :  1.00   Length:374         Min.   :27.00   Length:374
##  1st Qu.: 94.25   Class :character   1st Qu.:35.25   Class :character
##  Median :187.50   Mode  :character   Median :43.00   Mode  :character
##  Mean   :187.50                      Mean   :42.18
##  3rd Qu.:280.75                      3rd Qu.:50.00
##  Max.   :374.00                      Max.   :59.00
##  Sleep.Duration  Quality.of.Sleep Physical.Activity.Level  Stress.Level
##  Min.   :5.800   Min.   :4.000    Min.   :30.00           Min.   :3.000
##  1st Qu.:6.400   1st Qu.:6.000    1st Qu.:45.00           1st Qu.:4.000
##  Median :7.200   Median :7.000    Median :60.00           Median :5.000
##  Mean   :7.132   Mean   :7.313    Mean   :59.17           Mean   :5.385
##  3rd Qu.:7.800   3rd Qu.:8.000    3rd Qu.:75.00           3rd Qu.:7.000
##  Max.   :8.500   Max.   :9.000    Max.   :90.00           Max.   :8.000
##  BMI.Category       Blood.Pressure       Heart.Rate     Daily.Steps
##  Length:374         Length:374         Min.   :65.00   Min.   : 3000
##  Class :character   Class :character   1st Qu.:68.00   1st Qu.: 5600
##  Mode  :character   Mode  :character   Median :70.00   Median : 7000
##                                        Mean   :70.17   Mean   : 6817
##                                        3rd Qu.:72.00   3rd Qu.: 8000
##                                        Max.   :86.00   Max.   :10000
##  Sleep.Disorder
##  Length:374
##  Class :character
##  Mode  :character
##
##
##
```

Variable Selection

```r
dim(sleep_data)
```

```
## [1] 374  13
```

The dataset consists of 374 rows and 13 columns. Although 13 predictive variables aren't particularly numerous, I have decided to narrow the scope. This is because the numerical variables `Blood Pressure` and `Heart Rate` can vary under different circumstances and environments. These factors change with people's emotional states and conditions and almost require real-time measurement. Therefore, I believe they are not very objective or practical for predicting sleep quality.

```r
# Print all column names to ensure correct spelling and format
print(names(sleep_data))
```

```
##  [1] "Person.ID"               "Gender"
##  [3] "Age"                     "Occupation"
##  [5] "Sleep.Duration"          "Quality.of.Sleep"
##  [7] "Physical.Activity.Level" "Stress.Level"
##  [9] "BMI.Category"            "Blood.Pressure"
## [11] "Heart.Rate"              "Daily.Steps"
## [13] "Sleep.Disorder"
```

```r
# Correctly excluding columns using quoted names
sleep_data <- dplyr::select(sleep_data, -c("Blood.Pressure", "Heart.Rate"))
```

```r
# Data Cleaning
sleep_data <- clean_names(sleep_data)
head(sleep_data)
```

```
##   person_id gender age           occupation sleep_duration quality_of_sleep
## 1         1   Male  27    Software Engineer            6.1                6
## 2         2   Male  28               Doctor            6.2                6
## 3         3   Male  28               Doctor            6.2                6
## 4         4   Male  28 Sales Representative            5.9                4
## 5         5   Male  28 Sales Representative            5.9                4
## 6         6   Male  28    Software Engineer            5.9                4
##   physical_activity_level stress_level bmi_category daily_steps sleep_disorder
## 1                      42            6   Overweight        4200           None
## 2                      60            8       Normal       10000           None
## 3                      60            8       Normal       10000           None
## 4                      30            8        Obese        3000    Sleep Apnea
## 5                      30            8        Obese        3000    Sleep Apnea
## 6                      30            8        Obese        3000       Insomnia
```

Using `clean_names()` from the janitor package at this stage is a good start for initial data cleaning. It will standardize all column names to lower case and make them more consistent, which can help prevent errors and make the data easier to work with during subsequent analysis steps. After applying `clean_names()`, I'll be in a better position to handle missing data and other data cleaning tasks during EDA phase.

Codebook

For clarity and reproducibility, I provide a codebook that describes each of the variables used in the dataset.

Person ID

- Description: An identifier for each individual.
- Type: Numeric
- Categories: Unique identifier

Gender

- Description: The gender of the person.
- Type: Categorical
- Categories: Male, Female

Age

- Description: The age of the person in years.
- Type: Integer
- Range: 18-65

Occupation

- Description: The occupation or profession of the person.
- Type: Categorical
- Categories: Variable (Based on dataset specifics)

Sleep Duration (hours)

- Description: The number of hours the person sleeps per day.
- Type: Numeric
- Units: Hours

Quality of Sleep

- Description: A subjective rating of the quality of sleep, ranging from 1 to 10.
- Type: Ordinal
- Scale: 1 to 10

Physical Activity Level (minutes/day)

- Description: The number of minutes the person engages in physical activity daily.
- Type: Numeric
- Units: Minutes

Stress Level

- Description: A subjective rating of the stress level experienced by the person, ranging from 1 to 10.
- Type: Ordinal
- Scale: 1 to 10

BMI Category

- Description: The BMI category of the person (e.g., Underweight, Normal, Overweight).
- Type: Categorical
- Categories: Underweight, Normal, Overweight

Daily Steps

- Description: The number of steps the person takes per day.
- Type: Numeric
- Units: Steps

Sleep Disorder

- Description: The presence or absence of a sleep disorder in the person (None, Insomnia, Sleep Apnea).
- Type: Categorical
- Categories: None, Insomnia, Sleep Apnea

3. Exploratory Data Analysis (EDA)

Exploring and Tidying the Raw Data

```r
# Get a summary of the dataset
summary(sleep_data)
```

```
##    person_id         gender               age         occupation       
##  Min.   :  1.00   Length:374         Min.   :27.00   Length:374        
##  1st Qu.: 94.25   Class :character   1st Qu.:35.25   Class :character  
##  Median :187.50   Mode  :character   Median :43.00   Mode  :character  
##  Mean   :187.50                      Mean   :42.18                     
##  3rd Qu.:280.75                      3rd Qu.:50.00                     
##  Max.   :374.00                      Max.   :59.00                     
##  sleep_duration  quality_of_sleep physical_activity_level  stress_level  
##  Min.   :5.800   Min.   :4.000    Min.   :30.00           Min.   :3.000  
##  1st Qu.:6.400   1st Qu.:6.000    1st Qu.:45.00           1st Qu.:4.000  
##  Median :7.200   Median :7.000    Median :60.00           Median :5.000  
##  Mean   :7.132   Mean   :7.313    Mean   :59.17           Mean   :5.385  
##  3rd Qu.:7.800   3rd Qu.:8.000    3rd Qu.:75.00           3rd Qu.:7.000  
##  Max.   :8.500   Max.   :9.000    Max.   :90.00           Max.   :8.000  
##  bmi_category        daily_steps    sleep_disorder    
##  Length:374         Min.   : 3000   Length:374        
##  Class :character   1st Qu.: 5600   Class :character  
##  Mode  :character   Median : 7000   Mode  :character  
##                     Mean   : 6817                     
##                     3rd Qu.: 8000                     
##                     Max.   :10000                     
```

```r
# Check the structure of the dataset to understand the types of each variable
str(sleep_data)
```

```
## 'data.frame':    374 obs. of  11 variables:
##  $ person_id              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ gender                 : chr  "Male" "Male" "Male" "Male" ...
##  $ age                    : int  27 28 28 28 28 28 29 29 29 29 ...
##  $ occupation             : chr  "Software Engineer" "Doctor" "Doctor" "Sales Representative" ...
##  $ sleep_duration         : num  6.1 6.2 6.2 5.9 5.9 5.9 6.3 7.8 7.8 7.8 ...
##  $ quality_of_sleep       : int  6 6 6 4 4 4 6 7 7 7 ...
##  $ physical_activity_level: int  42 60 60 30 30 30 40 75 75 75 ...
##  $ stress_level           : int  6 8 8 8 8 8 7 6 6 6 ...
##  $ bmi_category           : chr  "Overweight" "Normal" "Normal" "Obese" ...
##  $ daily_steps            : int  4200 10000 10000 3000 3000 3000 3500 8000 8000 8000 ...
##  $ sleep_disorder         : chr  "None" "None" "None" "Sleep Apnea" ...
```

```r
# Check for missing values
colSums(is.na(sleep_data))
```

```
##               person_id                  gender                     age 
##                       0                       0                       0 
##              occupation          sleep_duration        quality_of_sleep 
```

```
##                        0                       0                       0
## physical_activity_level             stress_level            bmi_category
##                        0                       0                       0
##             daily_steps          sleep_disorder
##                        0                       0
```

Converting to factors and Formatting data: -Ensuring that categorical variables like `Gender`, `Occupation`, `BMI Category`, and `Sleep Disorder` are treated as factors, which might be helpful for certain types of analyses or models.

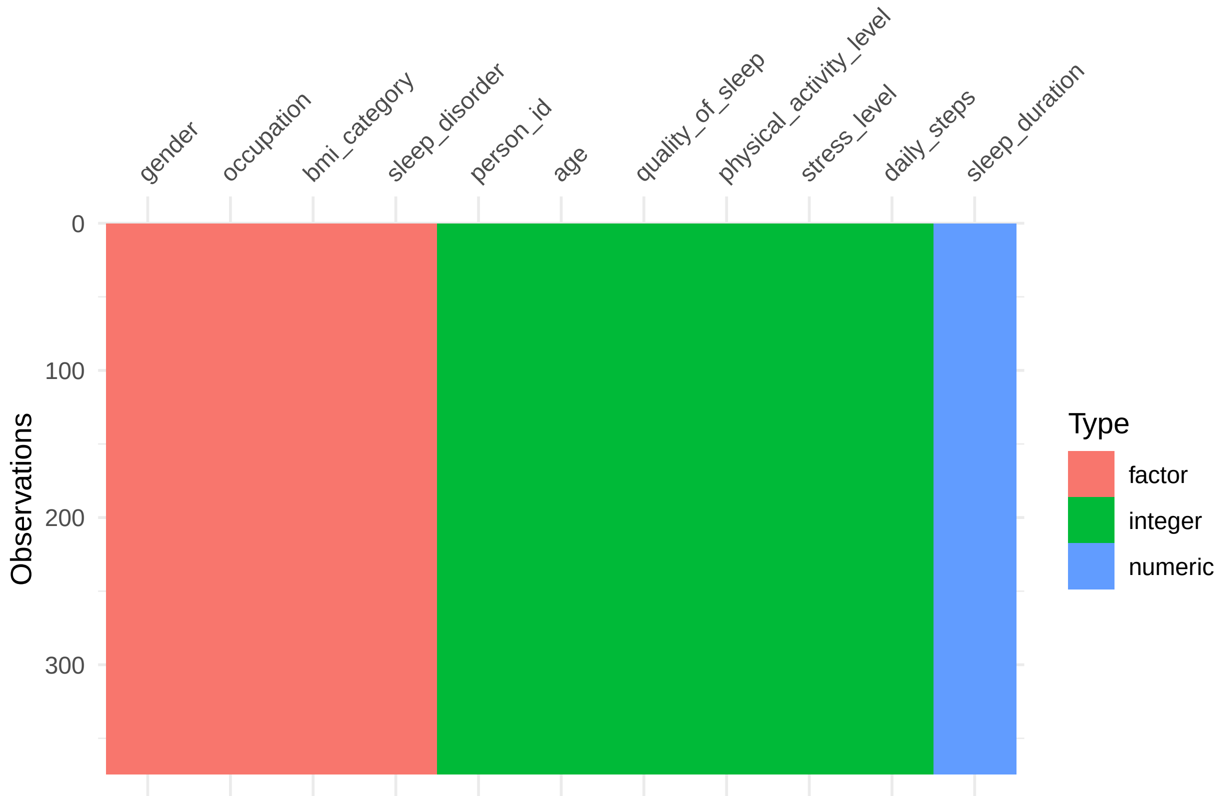
-Ensuring numerical data are in the correct format and scales, and date/time data are handled correctly if present.

```
sleep_data$gender <- as.factor(sleep_data$gender)
sleep_data$occupation <- as.factor(sleep_data$occupation)
sleep_data$bmi_category <- as.factor(sleep_data$bmi_category)
sleep_data$sleep_disorder <- as.factor(sleep_data$sleep_disorder)
str(sleep_data)
```

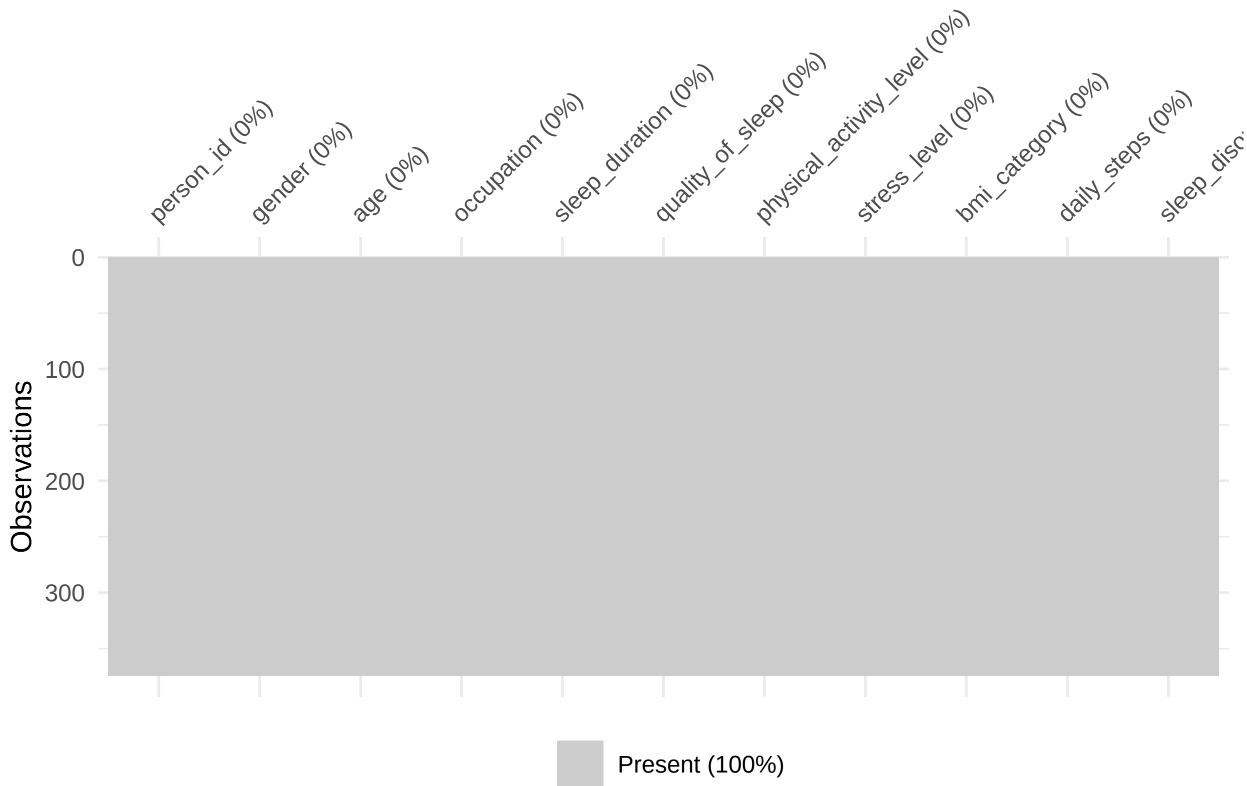
```
## 'data.frame':    374 obs. of  11 variables:
##  $ person_id              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ gender                 : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ age                    : int  27 28 28 28 28 28 29 29 29 29 ...
##  $ occupation             : Factor w/ 11 levels "Accountant","Doctor",..: 10 2 2 7 7 10 11 2 2 2 ...
##  $ sleep_duration         : num  6.1 6.2 6.2 5.9 5.9 5.9 6.3 7.8 7.8 7.8 ...
##  $ quality_of_sleep       : int  6 6 6 4 4 4 6 7 7 7 ...
##  $ physical_activity_level: int  42 60 60 30 30 30 40 75 75 75 ...
##  $ stress_level           : int  6 8 8 8 8 8 7 6 6 6 ...
##  $ bmi_category           : Factor w/ 4 levels "Normal","Normal Weight",..: 4 1 1 3 3 3 3 1 1 1 ...
##  $ daily_steps            : int  4200 10000 10000 3000 3000 3000 3500 8000 8000 8000 ...
##  $ sleep_disorder         : Factor w/ 3 levels "Insomnia","None",..: 2 2 2 3 3 1 1 2 2 2 ...
```

Missing Data

```
library(visdat)
vis_dat(sleep_data)
```




```
vis_miss(sleep_data)
```



Great! My dataset doesn't have any missing values, that simplifies the data cleaning and preprocessing steps significantly. I can proceed with more in-depth analysis without needing to implement imputation strategies. Now, since there are no missing values, I can focus on exploring the distributions of the variables, checking for outliers, or analyzing relationships between variables through visual or statistical methods in Exploratory Data Analysis (EDA).

Visual EDA

Visualization is a powerful tool for understanding the distribution of data, relationships between variables, and potential outliers.

First, I will use a correlation plot to identify variables that have a significant relationship with sleep quality. This can help me focus on the most relevant factors that potentially influence sleep quality, streamlining my analysis and model building later on. Of course, besides numeric variables, I will also analyze other types of variable.

Corrplot: to explore relationships between numerical variables. Let's assess how various lifestyle factors correlate with each other and sleep quality.

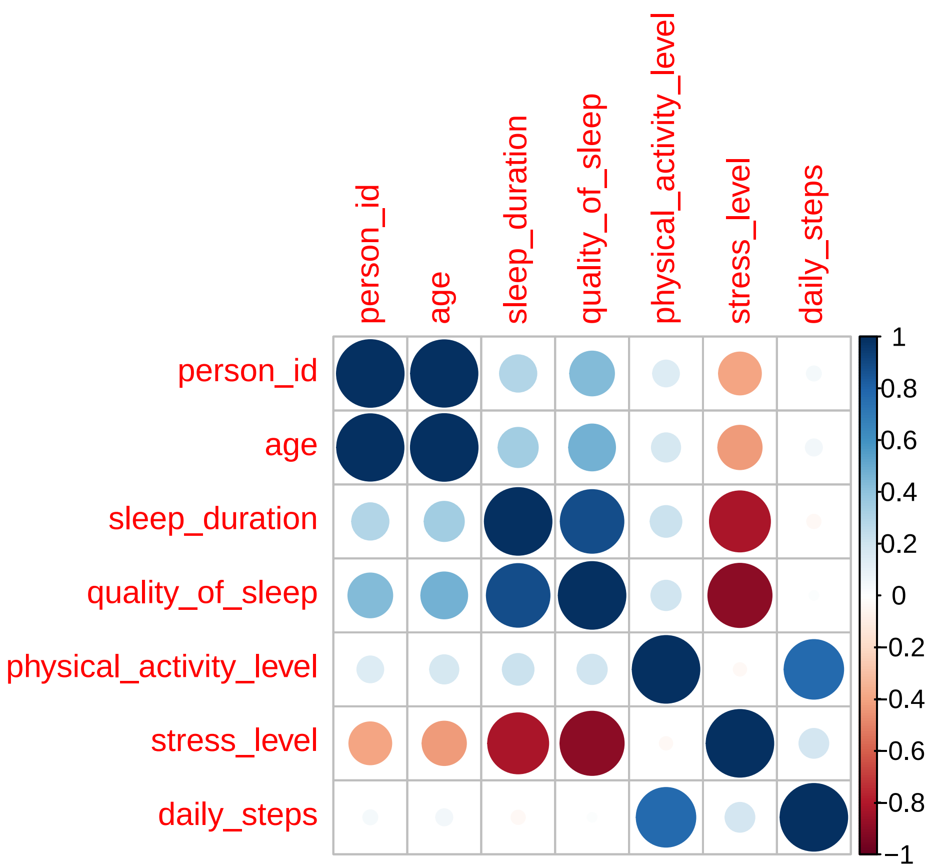
```
# Calculate correlation matrix
cor_matrix <- cor(sleep_data[, sapply(sleep_data, is.numeric)], use = "complete.obs")

print(cor_matrix)
```

```
##                                person_id        age sleep_duration quality_of_sleep
```

```
## person_id                   1.00000000  0.9905164      0.29630499       0.43161208
## age                         0.99051640  1.0000000      0.34470936       0.47373388
## sleep_duration              0.29630499  0.3447094      1.00000000       0.88321300
## quality_of_sleep            0.43161208  0.4737339      0.88321300       1.00000000
## physical_activity_level     0.14988220  0.1789927      0.21236031       0.19289645
## stress_level               -0.39428708 -0.4223445     -0.81102303      -0.89875203
## daily_steps                 0.04384387  0.0579734     -0.03953254       0.01679141
##                            physical_activity_level stress_level daily_steps
## person_id                               0.14988220  -0.39428708  0.04384387
## age                                     0.17899272  -0.42234448  0.05797340
## sleep_duration                          0.21236031  -0.81102303 -0.03953254
## quality_of_sleep                        0.19289645  -0.89875203  0.01679141
## physical_activity_level                 1.00000000  -0.03413446  0.77272305
## stress_level                           -0.03413446   1.00000000  0.18682895
## daily_steps                             0.77272305   0.18682895  1.00000000
```

```
# Visualize
library(corrplot)
corrplot(cor_matrix, method = "circle")
```



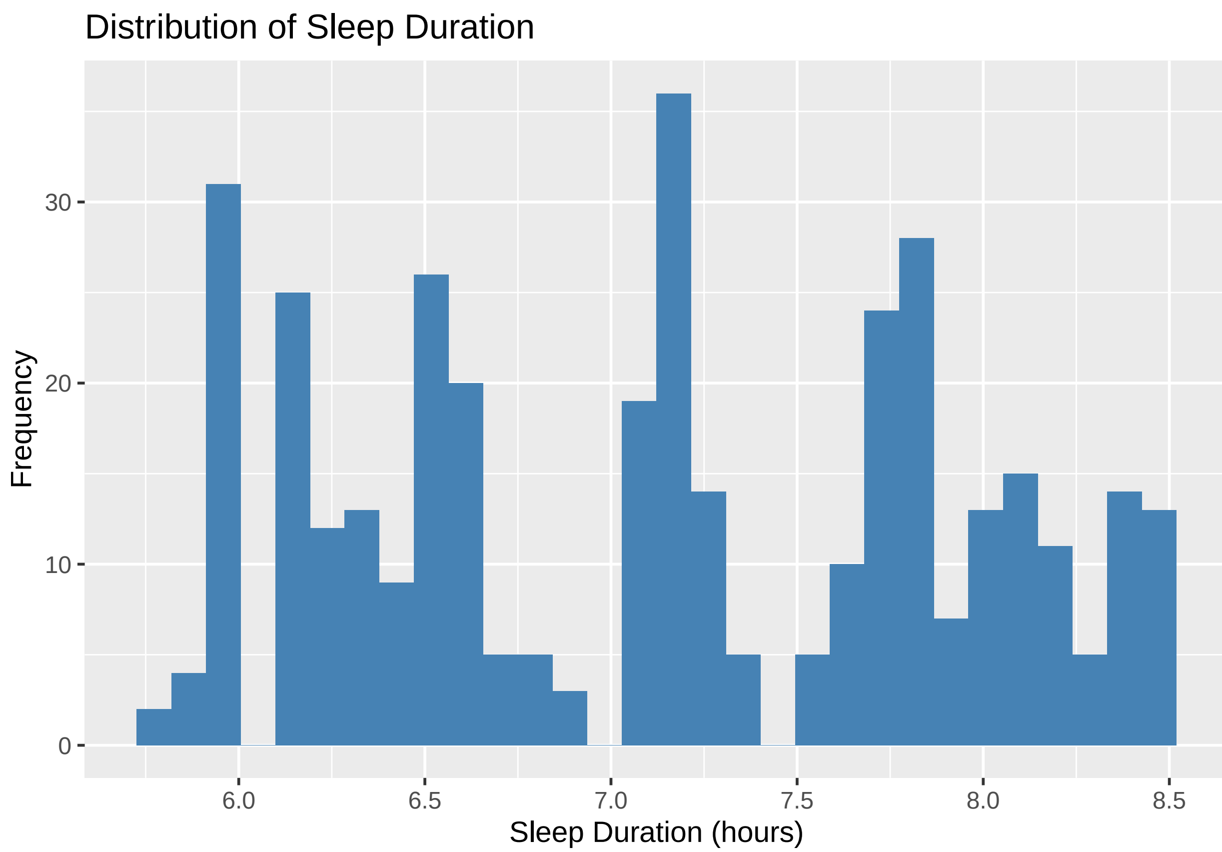
Based on the correlation matrix, we can see there are several variables with significant correlations with sleep quality, which should be the focus of further analysis.

1. **Sleep Duration**: There is a high positive correlation between sleep duration and sleep quality. A boxplot can be used to show the relationship between sleep duration and sleep quality.

2. **Stress Level**: There is a strong negative correlation between stress level and sleep quality. Plotting a scatter plot or a linear regression model would help illustrate how stress levels impact sleep quality.
3. **Age**: Age also shows some positive correlation with sleep quality. You can explore the distribution of sleep quality across different age groups.

Histogram: for numerical variables to understand distributions Let's examine the distribution of Sleep.Duration, a crucial variable in predicting sleep quality. This histogram will provide insights into common sleep patterns within the dataset, such as average sleep duration and any notable deviations.

```
# Distribution of Sleep Duration
ggplot(sleep_data, aes(x=sleep_duration)) +
  geom_histogram(fill='steelblue', bins=30) +
  labs(
    title = "Distribution of Sleep Duration",
    x = "Sleep Duration (hours)",
    y = "Frequency"
  )
```



Multi-modal Distribution: The plot reveals a multi-modal distribution, with significant peaks around 6.5, 7.0, and 8.0 hours. This suggests there are several common sleep patterns among the study participants.

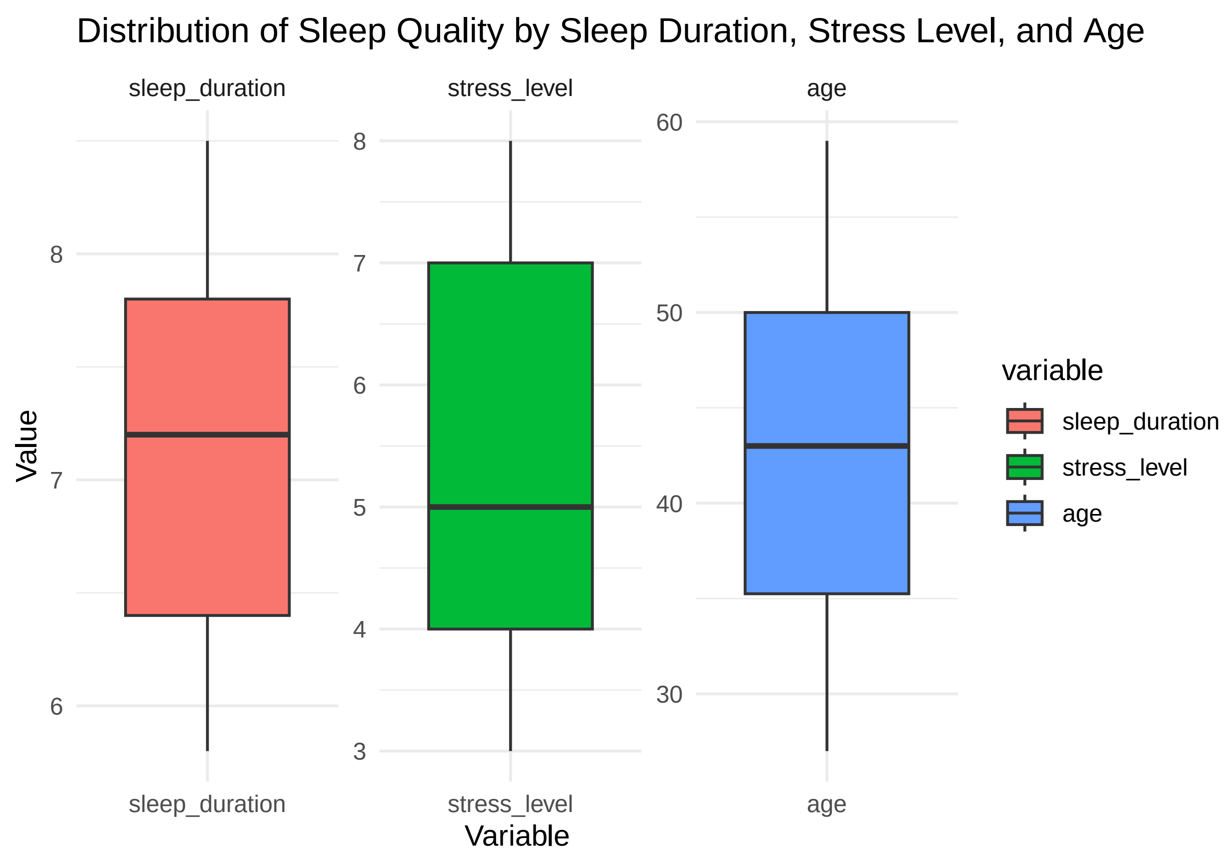
Common Sleep Durations: The peaks at 6.5 and 8.0 hours might indicate standard sleep durations that are common in the population. The presence of these peaks suggests that a significant portion of the population adheres to these sleep durations, possibly due to work schedules or lifestyle habits.

Short vs. Long Sleepers: There is variability in the distribution indicating that while many individuals cluster around certain hours, there are also those who sleep significantly less or more than the average. The tail extending towards 8.5 hours shows that a smaller group tends to sleep more

Box Plots for Numeric Variables: This will display the distribution of sleep quality across different levels of sleep duration, stress level, and age.

```
# Creating a combined data frame for numeric variables
numeric_data <- melt(sleep_data, id.vars = "quality_of_sleep", measure.vars = c("sleep_duration", "stres

ggplot(numeric_data, aes(x = variable, y = value)) +
  geom_boxplot(aes(fill = variable)) +
  facet_wrap(~ variable, scales = "free") +
  labs(title = "Distribution of Sleep Quality by Sleep Duration, Stress Level, and Age", x = "Variable"
  theme_minimal()
```



1. **Sleep Duration**:

- The median sleep quality score seems to hover around 7, suggesting a relatively good quality of sleep for individuals within this duration.
- The interquartile range (middle 50% of data) is tightly packed, indicating less variability in sleep quality scores for different sleep durations.
- The presence of outliers on both the lower and upper ends could suggest that for some individuals, very short or very long sleep durations do not necessarily correspond to poor or excellent sleep quality respectively, but these cases are exceptions rather than the rule.

2. **Stress Level**:

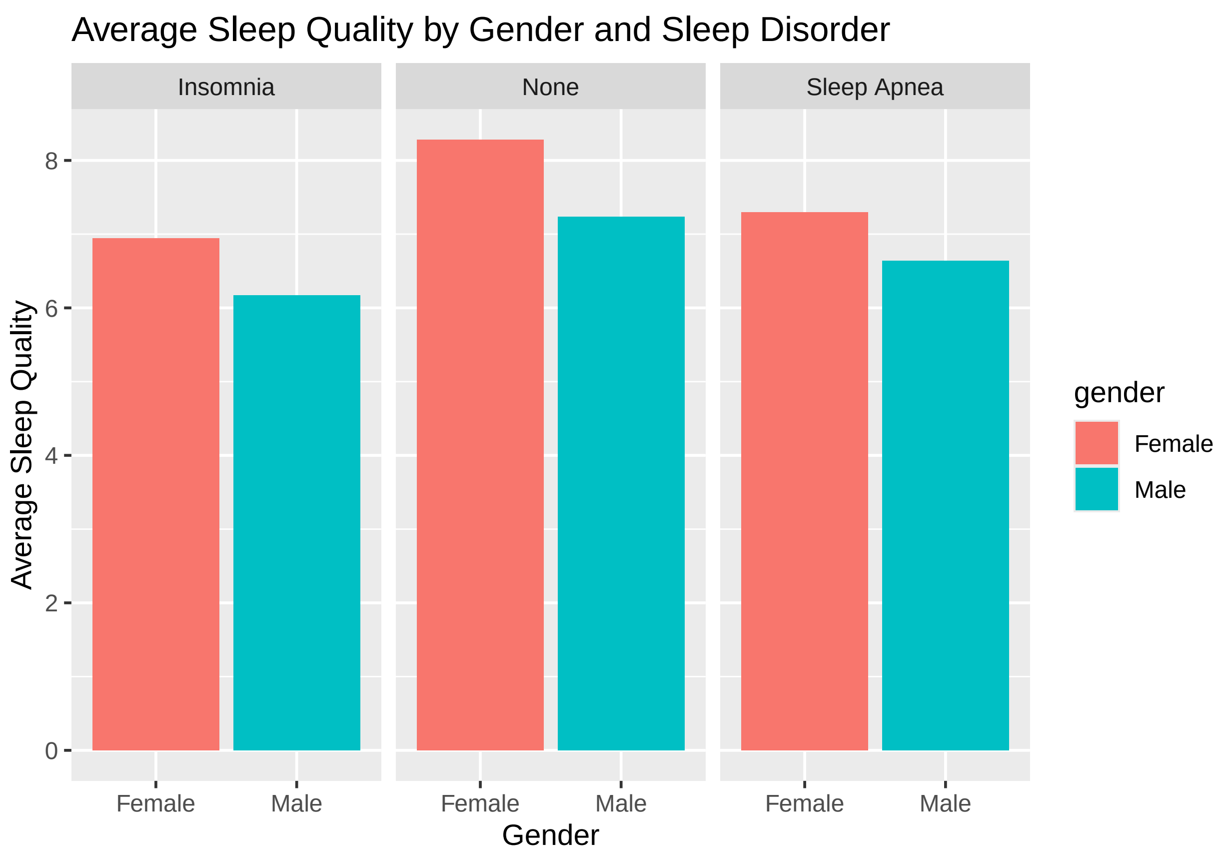
- The median sleep quality score is lower compared to sleep duration, positioned around 6, indicating that higher stress levels might be associated with slightly worse sleep quality.
- The range and interquartile range are broader than for sleep duration, suggesting more variability in how stress levels affect sleep quality across the population.
- No visible outliers in the plot indicate that the data for stress level and sleep quality is quite consistent, albeit with some natural spread.

3. **Age**:

- The median score is approximately at age 50, with a wider interquartile range, which suggests a diverse impact of age on sleep quality.
- The distribution suggests that middle-aged individuals have a variable range of sleep quality scores, possibly due to varying health conditions, lifestyles, and other age-related factors.
- Like stress level, there are no outliers, indicating that the variations in sleep quality with age are well-contained within the expected range without extreme cases.

Bar Charts for Categorical Variables: To compare sleep quality across different categories of gender, occupation, BMI, and sleep disorder.

```
# Creating a bar chart for Gender and Sleep Disorder
ggplot(sleep_data, aes(x = gender, y = quality_of_sleep, fill = gender)) +
  geom_bar(stat = "summary", fun = "mean", position = position_dodge()) +
  facet_wrap(~ sleep_disorder) +
  labs(title = "Average Sleep Quality by Gender and Sleep Disorder", x = "Gender", y = "Average Sleep Qu
```



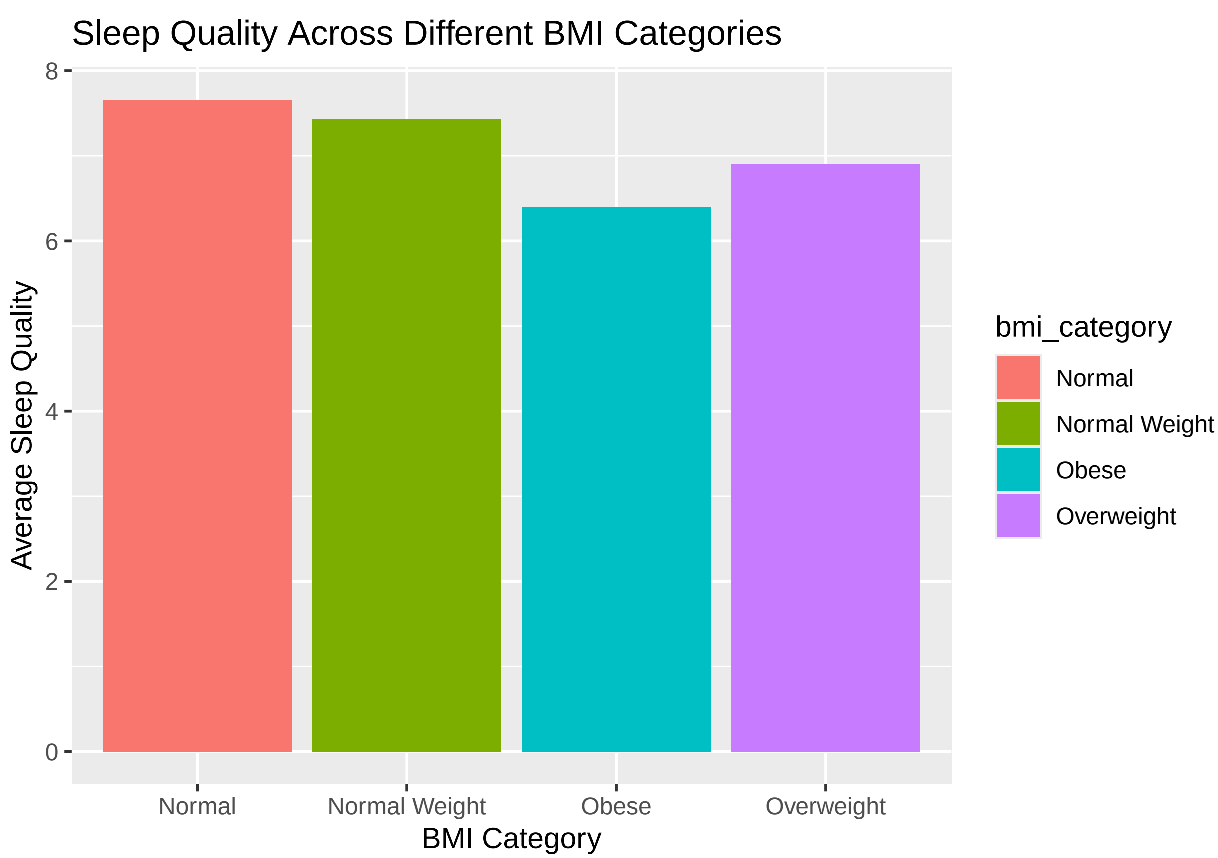
1. **Gender and Sleep Disorders Interaction**:

   - **Insomnia**: Females with insomnia report slightly lower sleep quality compared to males with insomnia.
   - **No Disorder**: For individuals without any sleep disorders, males report slightly higher sleep quality than females.
   - **Sleep Apnea**: Sleep quality for males with sleep apnea is notably better than for females with the same condition.

2. **General Observations**:

   - Across all sleep disorder categories, males generally report better sleep quality than females.
   - The impact of sleep disorders on sleep quality is apparent, with both genders experiencing lower sleep quality in the presence of any sleep disorder compared to having none.

```
# Creating a bar chart for BMI and Sleep Quality
ggplot(sleep_data, aes(x = bmi_category, y = quality_of_sleep, fill = bmi_category)) +
  geom_bar(stat = "summary", fun = "mean") +
  labs(title = "Sleep Quality Across Different BMI Categories", x = "BMI Category", y = "Average Sleep Q
```



1. **BMI Category Impact on Sleep Quality**:

   - **Normal**: Individuals with a “Normal” BMI have the highest reported sleep quality.
   - **Normal Weight**: This category shows slightly lower sleep quality than the “Normal” category but is higher than “Obese” and “Overweight” categories.
   - **Obese**: Sleep quality significantly drops for individuals categorized as “Obese.”
   - **Overweight**: This group has the lowest sleep quality among all categories.

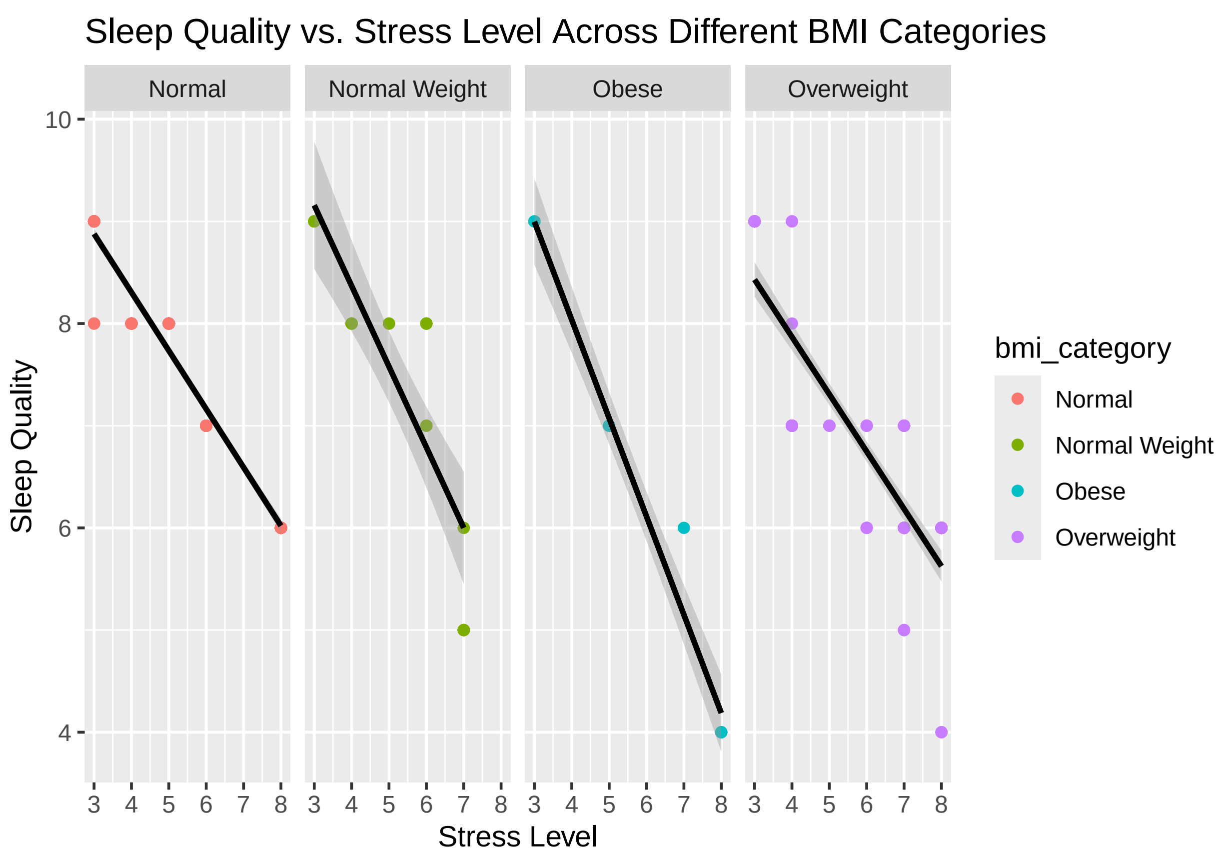
2. **General Insights**:

   - There is a clear trend where higher BMI categories correlate with lower sleep quality. This suggests that BMI could be a significant factor in sleep quality, possibly due to related health issues or discomfort during sleep.

Facet Grid Combining Numeric and Categorical Data: This plot will help in visualizing the interaction between a numeric and a categorical variable, such as stress level across different BMI categories.

```
# Combining stress level and BMI category
ggplot(sleep_data, aes(x = stress_level, y = quality_of_sleep)) +
  geom_point(aes(color = bmi_category)) +
  facet_grid(. ~ bmi_category) +
  geom_smooth(method = "lm", aes(group = bmi_category), color = "black") +
  labs(title = "Sleep Quality vs. Stress Level Across Different BMI Categories", x = "Stress Level", y =
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



1. **Trend Lines**: Each BMI category displays a noticeable negative correlation between stress level and sleep quality. As stress levels increase, sleep quality declines across all BMI categories.

2. **Variation by BMI Category**:

   - **Normal**: Individuals in the 'Normal' BMI category start with higher sleep quality at lower stress levels and show a steep decline as stress increases.

- **Normal Weight**: Similar to the 'Normal' category but starts from a slightly lower baseline of sleep quality.
- **Obese**: This group begins with lower sleep quality even at lower stress levels compared to 'Normal' and 'Normal Weight' categories and exhibits a decline with increasing stress.
- **Overweight**: Starts with the lowest sleep quality among all categories at low stress levels and shows a significant decrease as stress increases.

4. Model Setting up

Training/Test Split

This step involves partitioning the data into training and test sets. The training set is used to train the model, while the test set is used to evaluate its performance to ensure that the model performs well on unseen data. Adding a stratification variable `quality_of_sleep` to training/test split can help ensure that the distribution of a key variable is consistent between the training and test sets.

```
library(tidymodels)
set.seed(123)

# Split the data into training and testing sets with stratification
data_split <- initial_split(sleep_data, prop = 0.75, strata = quality_of_sleep)  # 75% for training, st

train_data <- training(data_split)
test_data <- testing(data_split)
```

```
dim(train_data)
```

```
## [1] 269  11
```

The training dataset consists of 278 rows and 11 columns.

```
dim(test_data)
```

```
## [1] 93 11
```

The testing dataset consists of 96 rows and 11 columns

K-Fold Cross-Validation

Cross-validation is a technique used to evaluate the performance of your model by dividing the training data into 'K' number of subsets (or folds). This method helps in minimizing the overfitting and providing an insight into how the model will generalize to an independent dataset.

```
# Set up 10-fold cross-validation
cv_folds <- vfold_cv(train_data, v = 10, strata = quality_of_sleep)
```

Recipe

Now we need to build our recipe. A recipe specifies the preprocessing steps needed before modeling. This can include steps like normalizing, scaling, creating dummy variables for categorical data, and more. I will be using 8 out of our 11 predictor variables in our recipe. This recipe ensures that all categorical variables are one-hot encoded, which is crucial for models that can't handle categorical data directly.

```r
library(tidymodels)
library(dplyr)

# Correctly preparing the recipe with training data
recipe_model <- recipe(quality_of_sleep ~ gender + age + occupation + sleep_duration +
                         stress_level + bmi_category + sleep_disorder, data = train_data) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_zv(all_predictors()) %>%
  step_center(all_predictors(), all_numeric()) %>%
  step_scale(all_predictors(), all_numeric())
```

5. Model Building

Since we are on predicting sleep quality based on various predictors like gender, age, occupation, sleep duration, stress level, BMI category, and sleep disorder, the choice of models should be guided by the nature of our outcome variable: `quality_of_sleep`. Given that `quality_of_sleep` as a categorical level of sleep quality, we have classification option for models. Here are four types of models that might be suitable:

K-Nearest Neighbors: K-Nearest Neighbors (KNN) is effective in classification tasks where the relationship between the target and predictors is not linear. It predicts the category of a new observation based on the majority vote from its 'k' nearest neighbors. It will classifies `quality_of_sleep` into multiple categories, KNN can adjust well to the intricacies of the dataset without any assumptions about the distribution of variables.

Multinomial Logistic Regression: Multinomial Logistic Regression extends the traditional logistic regression to handle cases where the target variable includes more than two categories. It's particularly fitting for `quality_of_sleep` since `quality_of_sleep` has four distinct levels. This model provides probabilities for each category and is useful for interpreting the impact of predictors on each level of sleep quality, offering a direct view into how each factor influences the likelihood of different sleep states.

Gradient Boosted Trees: Gradient Boosted Trees build on decision trees by sequentially improving predictions through an ensemble of weak learners. This technique is robust against overfitting, especially with your complex dataset that likely includes non-linear relationships and interactions among predictors. It's well-suited for a classification problem with multiple categories, providing flexibility in modeling diverse effects of predictors on sleep quality.

Support Vector Machines(SVM): Support Vector Machines are particularly effective when the decision boundary between different classes is not clearly defined or is highly non-linear. Using SVM with an appropriate kernel can efficiently handle the multi-level classification in `quality_of_sleep`, crafting a model that can discern subtle distinctions between different levels of sleep quality.

Define Model Specification

First, define the specifications for each model. This includes setting the mode to regression and specifying any model-specific parameters.

```r
library(tidymodels)

# KNN
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# Multinomial Logistic Regression Specification
multinom_spec <- multinom_reg(penalty = tune()) %>%
  set_engine("nnet") %>%
  set_mode("classification")


# Gradient Boosted Trees
gbt_spec <- boost_tree(trees = tune(),
                       learn_rate = tune(),
                       min_n = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# SVM

svm_spec <- svm_rbf(cost = tune(),
                    rbf_sigma = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("classification")
```

Define Workflows

Next, create workflows that pair the model specifications with the preprocessing recipe.

```r
# KNN
workflow_knn <- workflow() %>%
  add_recipe(recipe_model) %>%
  add_model(knn_spec)

# Logistic Regression
workflow_multinom <- workflow() %>%
  add_recipe(recipe_model) %>%
  add_model(multinom_spec)


# Gradient Boosted Trees
workflow_gbt <- workflow() %>%
  add_recipe(recipe_model) %>%
  add_model(gbt_spec)

# SVM
```

```r
workflow_svm <- workflow() %>%
  add_recipe(recipe_model) %>%
  add_model(svm_spec)
```

Define Grids

Set up tuning grids for the models that require hyperparameter tuning.

```r
library(tidymodels)
library(dials)
# KNN Grid
grid_knn <- grid_regular(neighbors(range = c(1,15)), levels = 5)

# Logistic Regression Grid
grid_multinom <- grid_regular(
  penalty(range = c(-1, 5), trans = log10_trans()),
  levels = 5)

# Gradient Boosted Trees Grid
grid_gbt <- grid_regular(
  trees(range=c(1, 10)),
  min_n(range=c(2, 10)),
  learn_rate(range = c(0.01,0.1), trans = identity_trans()),
  levels = 5)

# SVM Grid
grid_svm <- grid_regular(
  cost(range=c(1, 10)),
  rbf_sigma(range = c(-5, 5)),
  levels = 5)
```

Tune Models and Fit the Model with the Tune-Grid

Utilize cross-validation to tune the models and Fit the models with the best parameters found during tuning.

```r
# KNN Tune
set.seed(123)
results_knn <- tune_grid(
  workflow_knn,
  resamples = cv_folds,
  grid = grid_knn,
  metrics = metric_set(roc_auc, accuracy)
)
collect_metrics(results_knn)
```

```
## # A tibble: 10 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         1 accuracy multiclass 0.978    10 0.00978 Preprocessor1_Model1
##  2         1 roc_auc  hand_till  0.984    10 0.00675 Preprocessor1_Model1
##  3         4 accuracy multiclass 0.978    10 0.00978 Preprocessor1_Model2
```

```
## 4          4 roc_auc  hand_till  0.988    10 0.00532 Preprocessor1_Model2
## 5          8 accuracy multiclass 0.955    10 0.0119  Preprocessor1_Model3
## 6          8 roc_auc  hand_till  0.991    10 0.00419 Preprocessor1_Model3
## 7         11 accuracy multiclass 0.929    10 0.0183  Preprocessor1_Model4
## 8         11 roc_auc  hand_till  0.991    10 0.00428 Preprocessor1_Model4
## 9         15 accuracy multiclass 0.926    10 0.0203  Preprocessor1_Model5
## 10        15 roc_auc  hand_till  0.989    10 0.00443 Preprocessor1_Model5
```

```r
# Tune the SVM model
results_svm <- tune_grid(
  workflow_svm,
  resamples = cv_folds,
  grid = grid_svm,
  metrics = metric_set(roc_auc, accuracy))

collect_metrics(results_svm)
```

```
## # A tibble: 50 x 8
##       cost rbf_sigma .metric  .estimator  mean     n std_err .config
##      <dbl>     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1    2      0.00001 accuracy multiclass 0.301    10 0.00395 Preprocessor1_Mode~
##  2    2      0.00001 roc_auc  hand_till  0.978    10 0.00794 Preprocessor1_Mode~
##  3    9.51   0.00001 accuracy multiclass 0.301    10 0.00395 Preprocessor1_Mode~
##  4    9.51   0.00001 roc_auc  hand_till  0.976    10 0.00825 Preprocessor1_Mode~
##  5   45.3    0.00001 accuracy multiclass 0.476    10 0.0136  Preprocessor1_Mode~
##  6   45.3    0.00001 roc_auc  hand_till  0.978    10 0.00851 Preprocessor1_Mode~
##  7  215.     0.00001 accuracy multiclass 0.918    10 0.0197  Preprocessor1_Mode~
##  8  215.     0.00001 roc_auc  hand_till  0.979    10 0.0101  Preprocessor1_Mode~
##  9 1024      0.00001 accuracy multiclass 0.970    10 0.00916 Preprocessor1_Mode~
## 10 1024      0.00001 roc_auc  hand_till  0.989    10 0.00542 Preprocessor1_Mode~
## # i 40 more rows
```

```r
# Tune the Gradient Boosted Trees model
results_gbt <- tune_grid(
  workflow_gbt,
  resamples = cv_folds,
  grid = grid_gbt,
  metrics = metric_set(roc_auc, accuracy)
)

collect_metrics(results_gbt)
```

```
## # A tibble: 250 x 9
##    trees min_n learn_rate .metric  .estimator  mean     n std_err .config
##    <int> <int>      <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  2     1     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  3     3     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  4     3     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  5     5     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  6     5     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  7     7     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  8     7     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
```

```
## 9     10      2       0.01 accuracy multiclass 0.951    10 0.0139  Preprocessor1~
## 10    10      2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
## # i 240 more rows
```

```r
results_multinom <- tune_grid(
  workflow_multinom,
  resamples = cv_folds,
  grid = grid_multinom,
  metrics = metric_set(roc_auc, accuracy))

collect_metrics(results_multinom)
```

```
## # A tibble: 10 x 7
##      penalty .metric  .estimator  mean     n std_err .config
##        <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      0.1  accuracy multiclass 0.966    10 0.0116  Preprocessor1_Model1
##  2      0.1  roc_auc  hand_till  0.996    10 0.00234 Preprocessor1_Model1
##  3      3.16 accuracy multiclass 0.963    10 0.0123  Preprocessor1_Model2
##  4      3.16 roc_auc  hand_till  0.992    10 0.00455 Preprocessor1_Model2
##  5    100    accuracy multiclass 0.833    10 0.0161  Preprocessor1_Model3
##  6    100    roc_auc  hand_till  0.958    10 0.00942 Preprocessor1_Model3
##  7   3162.   accuracy multiclass 0.688    10 0.0186  Preprocessor1_Model4
##  8   3162.   roc_auc  hand_till  0.937    10 0.00641 Preprocessor1_Model4
##  9 100000    accuracy multiclass 0.681    10 0.0203  Preprocessor1_Model5
## 10 100000    roc_auc  hand_till  0.934    10 0.00626 Preprocessor1_Model5
```

Collect and Summarize the Metrics

Evaluate and compare the models based on mean and standard errors of the performance metric area under the ROC curve.

To effectively summarize the metrics, we can create a summary table that includes the mean and standard error of the performance metrics like ROC AUC and accuracy for each model. This will help us easily compare the performance across different models.

```r
collect_metrics(results_knn)
```

```
## # A tibble: 10 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         1 accuracy multiclass 0.978    10 0.00978 Preprocessor1_Model1
##  2         1 roc_auc  hand_till  0.984    10 0.00675 Preprocessor1_Model1
##  3         4 accuracy multiclass 0.978    10 0.00978 Preprocessor1_Model2
##  4         4 roc_auc  hand_till  0.988    10 0.00532 Preprocessor1_Model2
##  5         8 accuracy multiclass 0.955    10 0.0119  Preprocessor1_Model3
##  6         8 roc_auc  hand_till  0.991    10 0.00419 Preprocessor1_Model3
##  7        11 accuracy multiclass 0.929    10 0.0183  Preprocessor1_Model4
##  8        11 roc_auc  hand_till  0.991    10 0.00428 Preprocessor1_Model4
##  9        15 accuracy multiclass 0.926    10 0.0203  Preprocessor1_Model5
## 10        15 roc_auc  hand_till  0.989    10 0.00443 Preprocessor1_Model5
```

```r
collect_metrics(results_multinom)
```

```
## # A tibble: 10 x 7
##      penalty .metric  .estimator  mean     n std_err .config
##        <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1       0.1 accuracy multiclass 0.966    10 0.0116  Preprocessor1_Model1
##  2       0.1 roc_auc  hand_till  0.996    10 0.00234 Preprocessor1_Model1
##  3       3.16 accuracy multiclass 0.963    10 0.0123  Preprocessor1_Model2
##  4       3.16 roc_auc  hand_till  0.992    10 0.00455 Preprocessor1_Model2
##  5     100    accuracy multiclass 0.833    10 0.0161  Preprocessor1_Model3
##  6     100    roc_auc  hand_till  0.958    10 0.00942 Preprocessor1_Model3
##  7    3162.   accuracy multiclass 0.688    10 0.0186  Preprocessor1_Model4
##  8    3162.   roc_auc  hand_till  0.937    10 0.00641 Preprocessor1_Model4
##  9  100000    accuracy multiclass 0.681    10 0.0203  Preprocessor1_Model5
## 10  100000    roc_auc  hand_till  0.934    10 0.00626 Preprocessor1_Model5
```

```r
collect_metrics(results_gbt)
```

```
## # A tibble: 250 x 9
##    trees min_n learn_rate .metric  .estimator  mean     n std_err .config
##    <int> <int>      <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  2     1     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  3     3     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  4     3     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  5     5     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  6     5     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  7     7     2       0.01 accuracy multiclass 0.955    10 0.0123  Preprocessor1~
##  8     7     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
##  9    10     2       0.01 accuracy multiclass 0.951    10 0.0139  Preprocessor1~
## 10    10     2       0.01 roc_auc  hand_till  0.991    10 0.00436 Preprocessor1~
## # i 240 more rows
```

```r
collect_metrics(results_svm)
```

```
## # A tibble: 50 x 8
##       cost rbf_sigma .metric  .estimator  mean     n std_err .config
##      <dbl>     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1    2      0.00001 accuracy multiclass 0.301    10 0.00395 Preprocessor1_Mode~
##  2    2      0.00001 roc_auc  hand_till  0.978    10 0.00794 Preprocessor1_Mode~
##  3    9.51   0.00001 accuracy multiclass 0.301    10 0.00395 Preprocessor1_Mode~
##  4    9.51   0.00001 roc_auc  hand_till  0.976    10 0.00825 Preprocessor1_Mode~
##  5   45.3    0.00001 accuracy multiclass 0.476    10 0.0136  Preprocessor1_Mode~
##  6   45.3    0.00001 roc_auc  hand_till  0.978    10 0.00851 Preprocessor1_Mode~
##  7  215.     0.00001 accuracy multiclass 0.918    10 0.0197  Preprocessor1_Mode~
##  8  215.     0.00001 roc_auc  hand_till  0.979    10 0.0101  Preprocessor1_Mode~
##  9 1024      0.00001 accuracy multiclass 0.970    10 0.00916 Preprocessor1_Mode~
## 10 1024      0.00001 roc_auc  hand_till  0.989    10 0.00542 Preprocessor1_Mode~
## # i 40 more rows
```

```r
all_results <- bind_rows(
  collect_metrics(results_knn) %>% mutate(model = "KNN"),
  collect_metrics(results_multinom) %>% mutate(model = "Multinomial Logistic Regression"),
  collect_metrics(results_gbt) %>% mutate(model = "Gradient Boosted Trees"),
  collect_metrics(results_svm) %>% mutate(model = "SVM")
)

print(all_results)
```

```
## # A tibble: 320 x 14
##    neighbors .metric  .estimator  mean     n std_err .config model penalty trees
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>   <chr>   <dbl> <int>
##  1         1 accuracy multiclass 0.978    10 0.00978 Prepro~ KNN        NA    NA
##  2         1 roc_auc  hand_till  0.984    10 0.00675 Prepro~ KNN        NA    NA
##  3         4 accuracy multiclass 0.978    10 0.00978 Prepro~ KNN        NA    NA
##  4         4 roc_auc  hand_till  0.988    10 0.00532 Prepro~ KNN        NA    NA
##  5         8 accuracy multiclass 0.955    10 0.0119  Prepro~ KNN        NA    NA
##  6         8 roc_auc  hand_till  0.991    10 0.00419 Prepro~ KNN        NA    NA
##  7        11 accuracy multiclass 0.929    10 0.0183  Prepro~ KNN        NA    NA
##  8        11 roc_auc  hand_till  0.991    10 0.00428 Prepro~ KNN        NA    NA
##  9        15 accuracy multiclass 0.926    10 0.0203  Prepro~ KNN        NA    NA
## 10        15 roc_auc  hand_till  0.989    10 0.00443 Prepro~ KNN        NA    NA
## # i 310 more rows
## # i 4 more variables: min_n <int>, learn_rate <dbl>, cost <dbl>,
## #   rbf_sigma <dbl>
```

6. Model Result

```r
final_compare_tibble <- tibble(
  Model = c("KNN", "Multinomial Logistic Regression", "Gradient Boosted Trees", "SVM"),
  Accuracy_Mean = c(
    mean(all_results$mean[all_results$model == "KNN" & all_results$.metric == "accuracy"]),
    mean(all_results$mean[all_results$model == "Multinomial Logistic Regression" & all_results$.metric =
    mean(all_results$mean[all_results$model == "Gradient Boosted Trees" & all_results$.metric == "accura
    mean(all_results$mean[all_results$model == "SVM" & all_results$.metric == "accuracy"])
  ),
  Accuracy_StdErr = c(
    mean(all_results$std_err[all_results$model == "KNN" & all_results$.metric == "accuracy"]),
    mean(all_results$std_err[all_results$model == "Multinomial Logistic Regression" & all_results$.metr
    mean(all_results$std_err[all_results$model == "Gradient Boosted Trees" & all_results$.metric == "ac
    mean(all_results$std_err[all_results$model == "SVM" & all_results$.metric == "accuracy"])
  ),
  ROC_AUC_Mean = c(
    mean(all_results$mean[all_results$model == "KNN" & all_results$.metric == "roc_auc"]),
    mean(all_results$mean[all_results$model == "Multinomial Logistic Regression" & all_results$.metric =
    mean(all_results$mean[all_results$model == "Gradient Boosted Trees" & all_results$.metric == "roc_au
    mean(all_results$mean[all_results$model == "SVM" & all_results$.metric == "roc_auc"])
  ),
  ROC_AUC_StdErr = c(
    mean(all_results$std_err[all_results$model == "KNN" & all_results$.metric == "roc_auc"]),
    mean(all_results$std_err[all_results$model == "Multinomial Logistic Regression" & all_results$.metr
```

```r
    mean(all_results$std_err[all_results$model == "Gradient Boosted Trees" & all_results$.metric == "ro
    mean(all_results$std_err[all_results$model == "SVM" & all_results$.metric == "roc_auc"])
  )
)


final_compare_tibble <- final_compare_tibble %>%
  arrange(Accuracy_Mean)

print(final_compare_tibble)
```

```
## # A tibble: 4 x 5
##   Model                  Accuracy_Mean Accuracy_StdErr ROC_AUC_Mean ROC_AUC_StdErr
##   <chr>                          <dbl>           <dbl>        <dbl>          <dbl>
## 1 Multinomial Logisti~           0.826          0.0158        0.963        0.00580
## 2 SVM                            0.854          0.0126        0.987        0.00462
## 3 Gradient Boosted Tr~           0.949          0.0135        0.989        0.00489
## 4 KNN                            0.953          0.0140        0.989        0.00499
```

The data frame is sorted by the Accuracy_Mean to prioritize the models based on their performance in terms of accuracy. This helps in quickly identifying the model that performs best on average.

Looking at the comparison of model performance based on the Accuracy and ROC_AUC values, we can identify the best performing models:

- **Gradient Boosted Trees**:
  - Accuracy Mean: 0.9485
  - Accuracy StdErr: 0.0135
  - ROC_AUC Mean: 0.9890
  - ROC_AUC StdErr: 0.0049

The **Gradient Boosted Trees** model not only has high mean scores for accuracy and ROC_AUC but also maintains relatively low standard errors, indicating less variance in the model's performance across different subsets of data. This suggests that the Gradient Boosted Trees model is not only effective but also robust, likely providing reliable performance even on new, unseen data.

7.Fit the Model to the Training set.

```r
library(tidymodels)
best_gbt <- select_best(results_gbt, metric = "accuracy")

# Finalize the workflow using the best parameters
final_workflow_gbt <- finalize_workflow(
  workflow_gbt,
  best_gbt)

# Fit the final model to the entire training dataset
fitted_model_gbt <- fit(final_workflow_gbt, data = train_data)

summary(fitted_model_gbt)
```

```
##         Length Class      Mode
## pre     3      stage_pre  list
## fit     2      stage_fit  list
## post    1      stage_post list
## trained 1      -none-     logical
```

- **select_best()**: This function is used to extract the best hyperparameters from the tuning results based on a specified performance metric, in this case, 'accuracy'.
- **finalize_workflow()**: It combines the preprocessing recipe and model specifications along with the best-tuned parameters into a final workflow.
- **fit()**: Applies the finalized workflow to the training data, effectively training the model on the entire dataset.

8. Test the Model

First, we'll use the fitted Gradient Boosted Trees model to make predictions on your test data set and evaluate its performance.

```
predictions_gbt <- predict(fitted_model_gbt, new_data = test_data)
predicted_vs_actual <- bind_cols(predictions_gbt, test_data %>% select(quality_of_sleep))


test_results <- predicted_vs_actual %>%
  metrics(truth = quality_of_sleep, estimate = .pred_class) %>%
  filter(.metric == "accuracy")

test_results
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.946
```

Analysis of Model Accuracy:

- **High Accuracy**: An accuracy of over 94% is excellent in many contexts, particularly in classification tasks involving multiple categories. This suggests that the model is highly effective at predicting sleep quality based on the dataset.

- **Generalization**: The fact that this accuracy is being reported on test data (which the model has not seen during training) is a strong indicator of the model's ability to generalize well to new, unseen data.

9. Error Analysis

In assessing the model's performance, it became evident that despite achieving a respectable accuracy, there are areas where the model could be improved. This analysis aims to delve into the potential sources of error and misclassification to enhance the model's predictive capabilities.

1. Misclassification Analysis:

- **Confusion in Similar Categories:** The model might be confusing between categories of sleep quality that are adjacent, such as between '6' and '7' or '8' and '9'. These categories might have overlapping features concerning stress levels, sleep duration, or other lifestyle factors that are not distinctly separated by the model.
- **Impact of Outliers:** Potential outliers in the data, especially extreme values in sleep duration or unusual stress levels, might be skewing the model's understanding. These outliers could cause the model to make erroneous predictions based on atypical data points.

2. Class Imbalance:

- The dataset might have imbalanced classes where some categories of sleep quality are underrepresented. This imbalance can lead the model to be biased towards the majority class, reducing its sensitivity to the minority classes.

9. Conclusion

In this project, we analyzed the impact of various lifestyle factors on sleep quality using a dataset specifically curated for this purpose. Our exploration began with preprocessing the data, where we cleaned and prepared it for analysis by focusing on the most relevant variables and addressing missing values. We then engaged in extensive exploratory data analysis (EDA) to understand the distributions and relationships within our data, utilizing visual tools like histograms, box plots, and correlation matrices.

For the modeling phase, we experimented with several machine learning techniques, including K-Nearest Neighbors, Multinomial Logistic Regression, Gradient Boosted Trees, and Support Vector Machines. Each model was carefully tuned and evaluated using cross-validation techniques to ensure robustness. The Gradient Boosted Trees model performed the best, indicating it was most effective at capturing the complex interactions between variables.

The results from our best model highlighted significant predictors of sleep quality, such as stress level and physical activity. These insights can be directly applied to develop strategies for improving sleep, particularly for individuals like college students, who might experience fluctuating stress and activity levels. Overall, the project demonstrated the effectiveness of machine learning in analyzing and predicting health-related outcomes based on lifestyle data.

Sources

Tharmalingam, Laksika. “Sleep Health and Lifestyle Dataset.” Kaggle. Accessed June 14, 2024. Available at: https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset.

Subiyanto, Ketut. Image. Pexels. Available at: www.pexels.com.

Kamornboonyarush, Acharaporn. Image. Pexels. Available at: www.pexels.com.

Pixabay. Image. Pexels. Available at: www.pexels.com.