STAT 385 Final Project Report

Katarina Stojanovic, Sara Alaidroos, and Talal Wazir

2025-12-07

Introduction

Sleep is crucial to both physical and mental health, and for that reason it is essential to understand which factors influence sleep disorders. In this project, we analyze a real-world dataset containing information about demographic, lifestyle habits, and cardiovascular health indicators. Our goal is to determine whether we can accurately predict an individual's sleep disorder category (None, Insomnia, or Sleep Apnea) using statistical and machine learning.

The dataset includes variables such as age, sleep duration, quality of sleep, physical activity level, stress level, heart rate, daily steps, BMI category, and blood pressure. Since these factors have know relationships with overall sleep health, they provide a strong basis for prediction. We apply multiple classification models, compare their performance, and interpret key predictors to understand which characteristics are most influential.

Our analysis includes Random Forest, Gradient Boosting (GBM), Support Vector Machines (SVM), Logistic Regression, and LASSO models. By evaluating all of them on the same training and test sets, we showcase a fair comparison and ultimately recommend the most effective method for predicting sleep disorder.

Data Preparation

We begin by loading the dataset and inspecting its structure to understand the types and formats of all variables.

Relevant code:

```
sleep_data <- read.csv("Sleep_health_and_lifestyle_dataset.csv", header = TRUE)

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
```

```
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"
```

One important correction involves the the Blood.Pressure column, which is stored as a single string containing both systolic and diastolic values. Since machine learning models require numeric inputs, we extract these into two separate numering variables (Systolic_BP and Diastolic_BP). This allows blood pressure to contribute meaningfully to the models.

Relevant code:

```
print(str(sleep_data$Blood.Pressure))
```

```
##  chr [1:374] "126/83" "125/80" "125/80" "140/90" "140/90" "140/90" "140/90" ...
## NULL
```

```
sleep_data$Systolic_BP <- NA
sleep_data$Diastolic_BP <- NA

for(i in 1:nrow(sleep_data)) {
  bp_string <- sleep_data$Blood.Pressure[i]

  bp_parts <- strsplit(bp_string, "/")[[1]]

  sleep_data$Systolic_BP[i] <- as.numeric(bp_parts[1])
  sleep_data$Diastolic_BP[i] <- as.numeric(bp_parts[2])
}

print(sleep_data$Systolic_BP)
```

```
##   [1] 126 125 125 140 140 140 140 120 120 120 120 120 120 120 120 120 132 120
##  [19] 132 120 120 120 120 120 120 120 120 120 120 120 130 130 117 125 120 125
##  [37] 125 120 120 120 120 120 120 120 120 120 120 120 120 120 120 120 125 120
##  [55] 125 125 120 125 125 120 125 125 125 125 125 125 118 125 128 128 125 125
##  [73] 125 125 125 125 125 125 125 125 131 131 128 128 120 115 125 125 125 125
##  [91] 125 125 120 135 115 115 115 115 115 115 115 115 115 129 115 129 126 120
## [109] 120 130 115 130 115 130 115 115 115 115 115 115 115 115 115 115 115 120
## [127] 130 115 130 130 115 130 130 115 130 130 115 130 115 130 115 130 115 115
## [145] 130 135 130 132 128 115 115 130 130 130 130 130 130 130 130 130 130 119
## [163] 119 130 130 130 121 125 125 130 130 130 130 130 130 130 130 130 130 130
## [181] 130 130 130 130 130 130 135 130 135 130 135 130 130 130 130 130 130 130
## [199] 130 130 130 130 130 117 122 130 130 130 130 130 130 130 130 130 130 130
## [217] 130 130 130 130 135 130 130 130 135 130 135 130 135 130 135 130 135 130
## [235] 135 130 130 135 130 130 135 130 130 135 130 135 130 130 130 130 135 135
## [253] 135 135 135 135 135 135 135 135 135 135 135 125 142 140 142 140 140 140
## [271] 140 140 140 140 140 140 139 139 140 125 140 140 140 140 140 140 140 140
## [289] 140 140 140 140 140 140 140 140 140 140 125 125 125 125 125 140 140 140
## [307] 130 130 130 130 130 130 125 125 125 125 125 125 125 125 125 125 125 125
## [325] 125 125 125 125 125 125 125 125 125 125 125 125 125 125 125 140 140 118
## [343] 118 140 140 140 140 140 140 140 140 140 140 140 140 140 140 140 140 140
## [361] 140 140 140 140 140 140 140 140 140 140 140 140 140 140
```

```
print(sleep_data$Diastolic_BP)
```

```
##   [1] 83 80 80 90 90 90 90 80 80 80 80 80 80 80 80 80 87 80 87 80 80 80 80 80 80
##  [26] 80 80 80 80 80 86 86 76 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
##  [51] 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 76 80 85 85 80 80 80 80 80
##  [76] 80 80 80 80 80 86 86 84 84 80 75 80 80 80 80 80 80 80 88 75 75 75 75 75 75
## [101] 75 75 75 84 75 84 83 80 80 85 75 85 75 85 75 75 75 75 75 75 75 75 75 75 75
## [126] 80 85 75 85 85 75 85 85 75 85 85 75 85 75 85 75 85 75 75 85 88 85 87 85 78
## [151] 78 85 85 85 85 85 85 85 85 85 85 77 77 85 85 85 79 82 82 85 85 85 85 85 85
## [176] 85 85 85 85 85 85 85 85 85 85 85 90 85 90 85 90 85 85 85 85 85 85 85 85 85
## [201] 85 85 85 76 80 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 90 85 85 85 90
## [226] 85 90 85 90 85 90 85 90 85 90 85 85 90 85 85 90 85 85 90 85 90 85 85 85 85
## [251] 90 90 90 90 90 90 90 90 90 90 90 90 90 82 92 95 92 95 95 95 95 95 95 95 95
## [276] 95 91 91 95 80 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 80 80
## [301] 80 80 82 95 95 95 85 85 85 85 85 85 80 80 80 80 80 80 80 80 80 80 80 80 80
## [326] 80 80 80 80 80 80 80 80 80 80 80 80 80 80 95 95 75 75 95 95 95 95 95 95 95
## [351] 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95
```

We also convert categorical variables such as Gender, Occupation, BMI Category, and Sleep Disorder into factors so that they can be treated correctly during statistical modeling. This is a very necessary step because models like Random Forest and Logistic Regression relu on properly encoded categories.

Relevant code:

```
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)

print(str(sleep_data))
```

```
## 'data.frame':    374 obs. of  15 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 ...
##  $ Blood.Pressure         : chr  "126/83" "125/80" "125/80" "140/90" ...
##  $ Heart.Rate             : int  77 75 75 85 85 85 82 70 70 70 ...
##  $ 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 ...
##  $ Systolic_BP            : num  126 125 125 140 140 140 140 120 120 120 ...
##  $ Diastolic_BP           : num  83 80 80 90 90 90 90 80 80 80 ...
## NULL
```

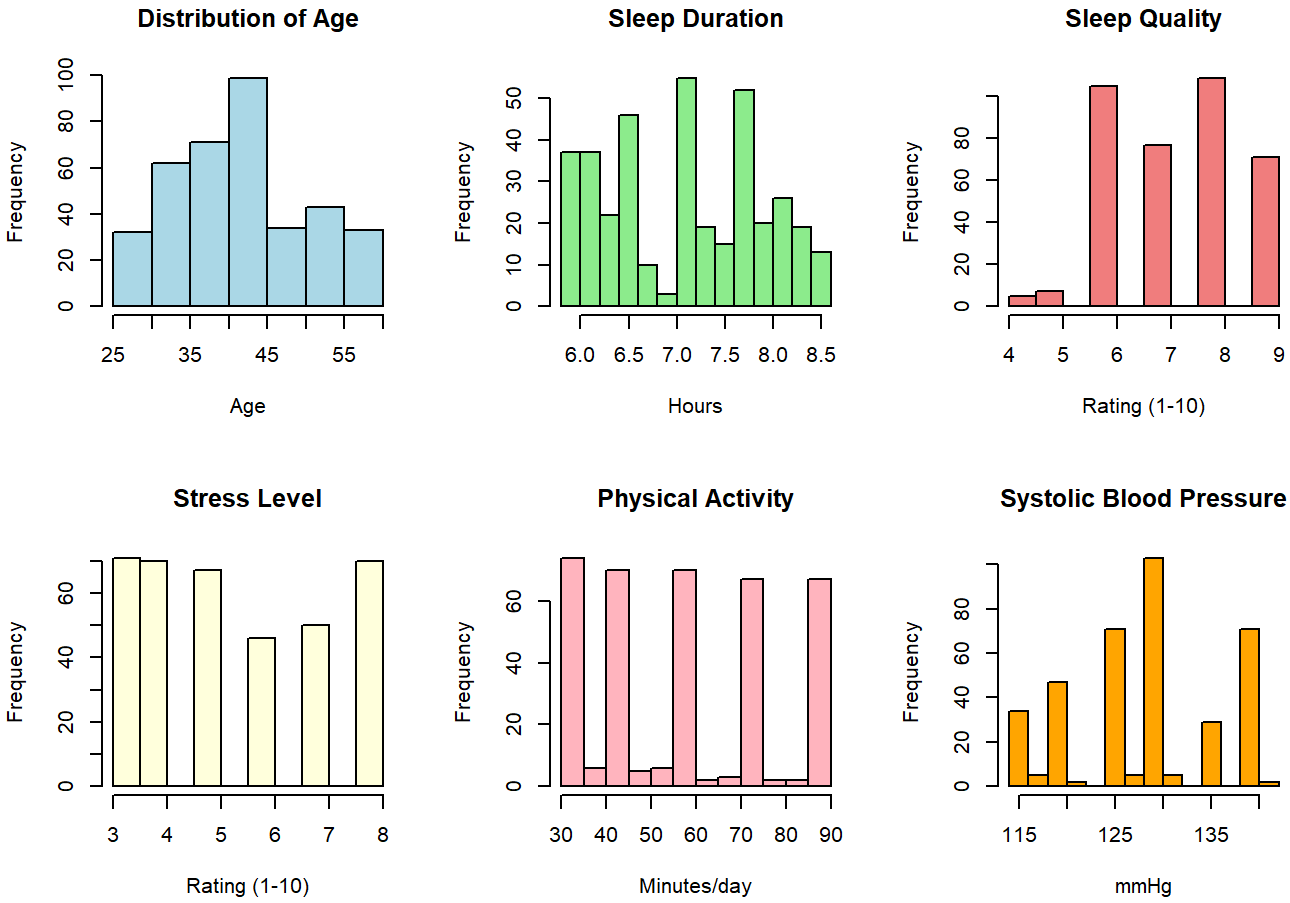
After cleaning, our predictor variables include demographics (Gender, Age, and Occupation), lifestyle factors (Sleep Duration, Physical Activity, Daily Steps, and Stress Level), physiological measurements (Heart Rate, Systolic_BP, and Diastolic_BP), and BMI Category. The response variable for all classification models is Sleep.Disorder, which is a categorical variable with three levels (None, Insomnia, and Sleep Apnea). These preprocessing steps ensure that all variables are in appropriate formats for the machine learning models used later in the analysis.

Exploratory Data Analysis

We begin our analysis by summarizing the numerical variables. Histograms help visualize the distribution of these numerical variables. Age is mostly concentrated around 30-40 years, and most people sleep between 6 and 8 hours per night. Sleep Quality is slightly right-skewed, with many participants rating their sleep 7+ on a 1-10 scale. Physical Activity and Daily Steps show large variation, reflecting diverse lifestyles in the dataset. Sleep Duration, Sleep Quality, and Stress Level show patterns that are consistent with typical population trends. Systolic_BP falls mostly in the 110-130 mmHg range.

Relevant code:

```
par(mfrow = c(2, 3))

hist(sleep_data$Age, main = "Distribution of Age", xlab = "Age", col = "lightblue")
hist(sleep_data$Sleep.Duration, main = "Sleep Duration", xlab = "Hours", col = "lightgreen")
hist(sleep_data$Quality.of.Sleep, main = "Sleep Quality", xlab = "Rating (1-10)", col = "lightco
ral")
hist(sleep_data$Stress.Level, main = "Stress Level", xlab = "Rating (1-10)", col = "lightyello
w")
hist(sleep_data$Physical.Activity.Level, main = "Physical Activity", xlab = "Minutes/day", col =
"lightpink")
hist(sleep_data$Systolic_BP, main = "Systolic Blood Pressure", xlab = "mmHg", col = "orange")
```

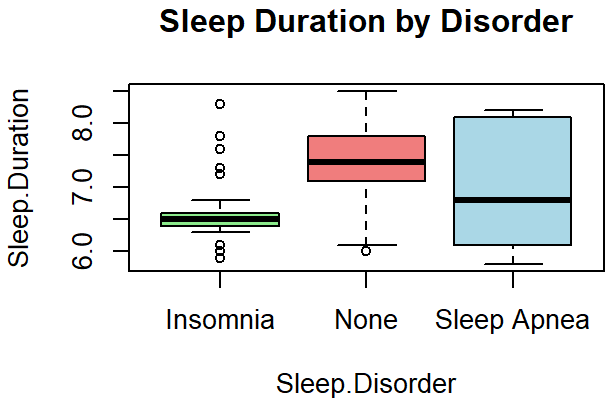


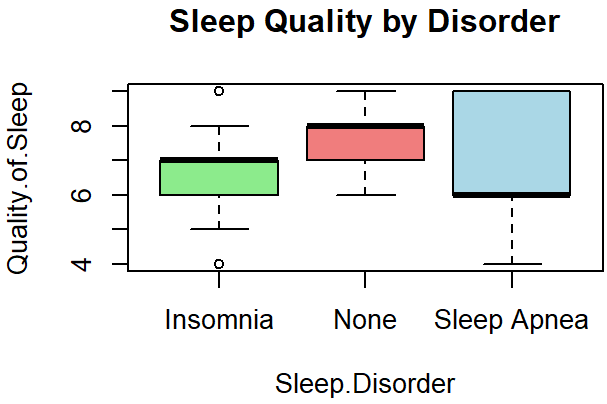

```
par(mfrow = c(1, 1))
```

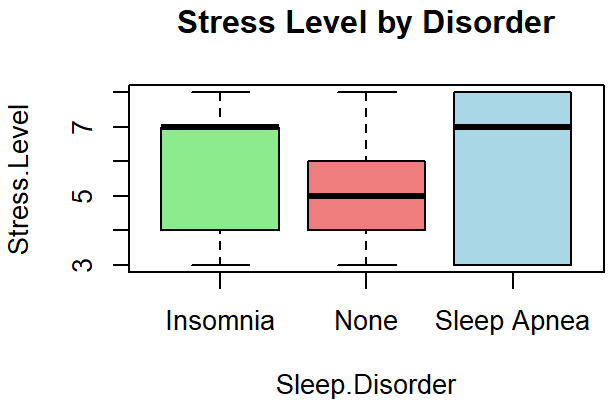
Next, we compare key variables across sleep disorder categories using boxplots. Individuals with Insomnia tend to have the shortest sleep duration, lowest sleep quality, and highest stress levels. Those with None sleep disorder generally sleep longer, report better sleep quality, and show slightly higher physical activity levels. Participants with Sleep Apnea show greater inconsistency in sleep duration and moderately elevated stress.

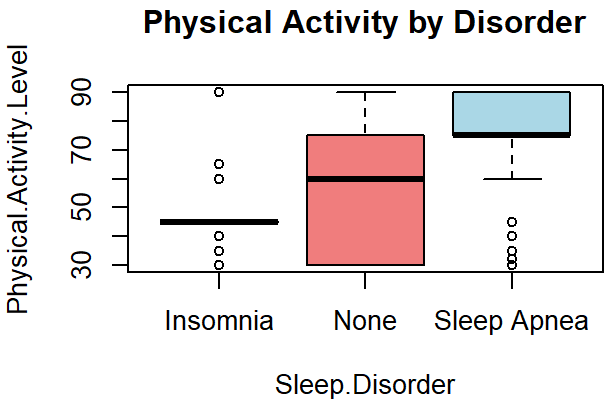
Relevant code:

```
par(mfrow = c(2, 2))
boxplot(Sleep.Duration ~ Sleep.Disorder, data = sleep_data,
        main = "Sleep Duration by Disorder", col = c("lightgreen", "lightcoral", "lightblue"))
boxplot(Quality.of.Sleep ~ Sleep.Disorder, data = sleep_data,
        main = "Sleep Quality by Disorder", col = c("lightgreen", "lightcoral", "lightblue"))
boxplot(Stress.Level ~ Sleep.Disorder, data = sleep_data,
        main = "Stress Level by Disorder", col = c("lightgreen", "lightcoral", "lightblue"))
boxplot(Physical.Activity.Level ~ Sleep.Disorder, data = sleep_data,
        main = "Physical Activity by Disorder", col = c("lightgreen", "lightcoral", "lightblu
e"))
```










```
par(mfrow = c(1, 1))
```

We also examine correlations among the numerical variables. Sleep DUration and Sleep Quality are moderately positively correlated, meaning longer sleep tends to accompany higher self-reported sleep quality. Stress Level has a negative relationship with Sleep Duration and Sleep Quality, which is consistent with our expectations. Systolic_BP and Diastolic_BP are strongly correlated, as expected from paired cardiovascular measurements.

Relevant code:

```
numerical_vars <- sleep_data[, c("Age", "Sleep.Duration", "Quality.of.Sleep",
                                 "Physical.Activity.Level", "Stress.Level",
                                 "Heart.Rate", "Daily.Steps", "Systolic_BP", "Diastolic_BP")]
cor_matrix <- cor(numerical_vars)
print(round(cor_matrix, 3))
```

```
##                            Age Sleep.Duration Quality.of.Sleep
## Age                      1.000          0.345            0.474
## Sleep.Duration           0.345          1.000            0.883
## Quality.of.Sleep         0.474          0.883            1.000
## Physical.Activity.Level  0.179          0.212            0.193
## Stress.Level            -0.422         -0.811           -0.899
## Heart.Rate              -0.226         -0.516           -0.660
## Daily.Steps              0.058         -0.040            0.017
## Systolic_BP              0.606         -0.180           -0.122
## Diastolic_BP             0.594         -0.167           -0.110
##                          Physical.Activity.Level Stress.Level Heart.Rate
## Age                                        0.179       -0.422     -0.226
## Sleep.Duration                             0.212       -0.811     -0.516
## Quality.of.Sleep                           0.193       -0.899     -0.660
## Physical.Activity.Level                    1.000       -0.034      0.137
## Stress.Level                              -0.034        1.000      0.670
## Heart.Rate                                 0.137        0.670      1.000
## Daily.Steps                                0.773        0.187     -0.030
## Systolic_BP                                0.265        0.103      0.294
## Diastolic_BP                               0.383        0.092      0.271
##                          Daily.Steps Systolic_BP Diastolic_BP
## Age                            0.058       0.606        0.594
## Sleep.Duration                -0.040      -0.180       -0.167
## Quality.of.Sleep               0.017      -0.122       -0.110
## Physical.Activity.Level        0.773       0.265        0.383
## Stress.Level                   0.187       0.103        0.092
## Heart.Rate                    -0.030       0.294        0.271
## Daily.Steps                    1.000       0.103        0.242
## Systolic_BP                    0.103       1.000        0.973
## Diastolic_BP                   0.242       0.973        1.000
```

This preliminary exploration highlights clear behavioral and physiological differences between disorder groups, motivating the machine learning classification models used later.

Modeling Strategy

Our task is a multiclass classification problem with three possible outcomes (None, Insomnia, and Sleep Apnea). To get a fair evaluation of all models, we split the dataset into 75% training data and 25% testing data. A random seed is also set to ensure that results can be reproduced.

We choose a diverse collection of models that each have unique strengths:

~Random Forest - captures nonlinear patterns and interactions

~Gradient Boosting (GBM) - builds trees sequentially and often improves predictive accuracy

~SVM - can separate classes using optimal hyperplanes

~Logistic Regression - provides interpretable coefficients and probabilistic outputs

~LASSO - performs variable selection through regularization

By comparing these models, we can determine which model delivers the highest accuracy and identify the predictors that most stringly influence sleep disorder.

Random Forest Model

Random forest is well-suited for this dataset because it handles mixed variable types, captures nonlinear relationships, and provides built-in feature importance measures.

We train two initial models using standard heuristic values for the mtry parameter (one using $\sqrt{p}$ predictors per split and one using p/3 predictors). We evaluate both models using out-of-bag (OOB) error, a reliable internal error estimate. Both models produce an OOB error around 0.10, so we further tune mtry by testing values from 2 to 10. The tuning process shows that mtry = 2 yields the lowest OOB error, indicating that smaller random subsets of predictors at each split improve stability and accuracy.

Using the tuned mtry value, the final Random Forest model achieves a test accuracy of 94.68%. The confusion matrix shows near perfect classification for the None and Sleep Apnea classes, with most errors occuring between Insomnia and None.

The variable importance plot reveals that Systolic_BP, Diastolic_BP, Occupation, and Age are among the most influential predictors. Sleep Duration, Physical Activity, and Stress Level also contribute meaningfully. These results support the idea that both lifestyle habits and cardiovascular measures help distinguish between different types of sleep disorder.

Relevant code:

```
model_data <- sleep_data[, c("Gender", "Age", "Occupation", "Sleep.Duration",
                             "Quality.of.Sleep", "Physical.Activity.Level",
                             "Stress.Level", "BMI.Category", "Systolic_BP",
                             "Diastolic_BP", "Heart.Rate", "Daily.Steps",
                             "Sleep.Disorder")]

set.seed(20251114)
alpha <- 0.75
inTrain <- sample(1:nrow(model_data), alpha * nrow(model_data))
train.set <- model_data[inTrain, ]
test.set <- model_data[-inTrain, ]

nrow(train.set)
```

```
## [1] 280
```

```
nrow(test.set)
```

```
## [1] 94
```

```
### Random Forest Model ###
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.3
```

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

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

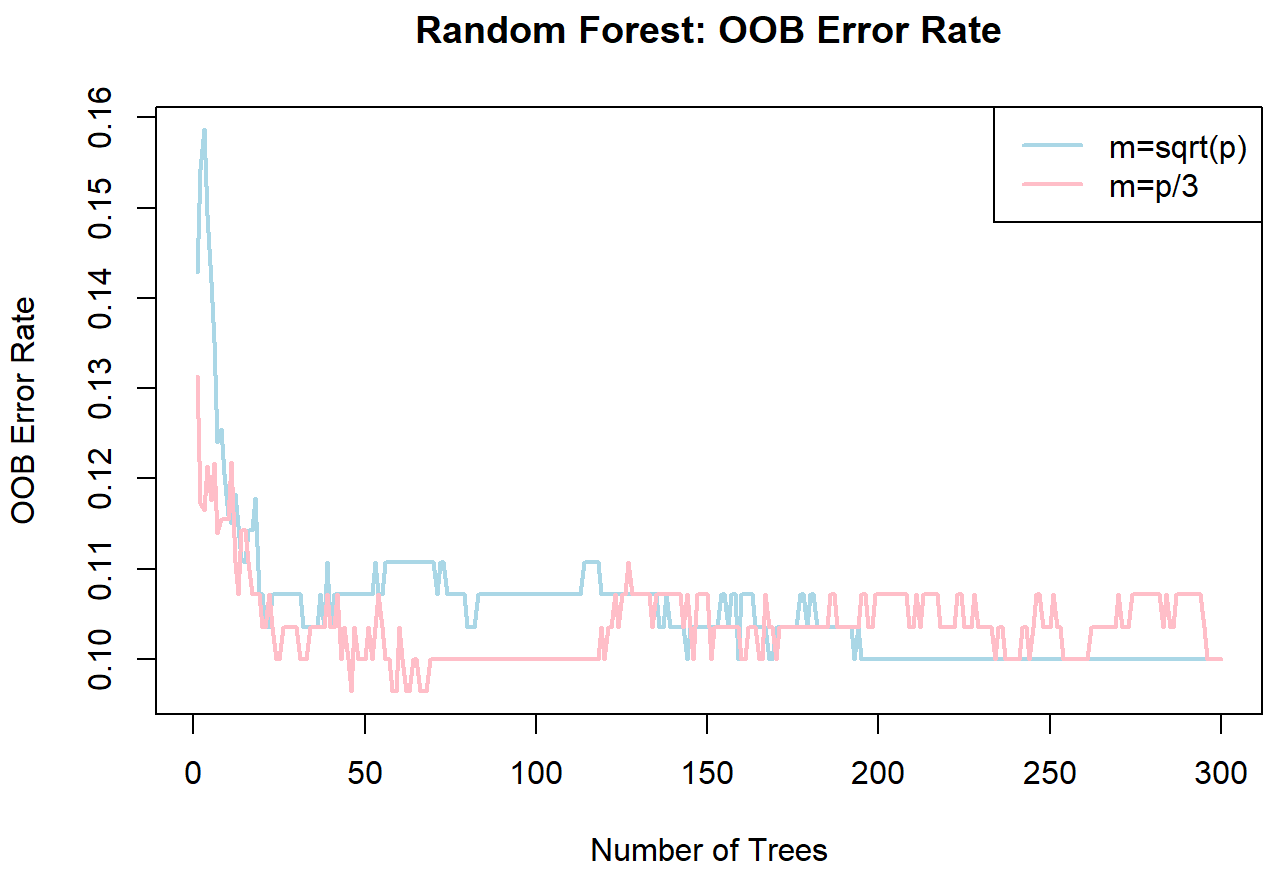
```
num_predictors<-12

set.seed(123)

rf_sqrt <- randomForest(Sleep.Disorder ~ ., data = train.set,
                        mtry = floor(sqrt(num_predictors)),
                        ntree = 300,
                        importance = TRUE)

rf_third <- randomForest(Sleep.Disorder ~ ., data = train.set,
                         mtry = floor(num_predictors/3),
                         ntree = 300,
                         importance = TRUE)

cat("mtry = sqrt(p):", rf_sqrt$err.rate[300, "OOB"], "\n")
```

```
## mtry = sqrt(p): 0.1
```

```
cat("mtry = p/3:", rf_third$err.rate[300, "OOB"], "\n")
```

```
## mtry = p/3: 0.1
```

```
plot(rf_sqrt$err.rate[, "OOB"], type = "l", lwd = 2, col = "lightblue",
     main = "Random Forest: OOB Error Rate",
     xlab = "Number of Trees", ylab = "OOB Error Rate",
     ylim = range(c(rf_sqrt$err.rate[, "OOB"], rf_third$err.rate[, "OOB"])))
lines(rf_third$err.rate[, "OOB"], lwd = 2, col = "pink")
legend("topright", legend = c("m=sqrt(p)", "m=p/3"),
       col = c("lightblue", "pink"), lty = 1, lwd = 2)
```




```
pred_sqrt <- predict(rf_sqrt, test.set)
pred_third <- predict(rf_third, test.set)
conf_matrix <- table(pred_sqrt, test.set$Sleep.Disorder)
print(conf_matrix)
```

```
##
## pred_sqrt     Insomnia None Sleep Apnea
##   Insomnia          17    2           0
##   None               3   48           2
##   Sleep Apnea        0    0          22
```

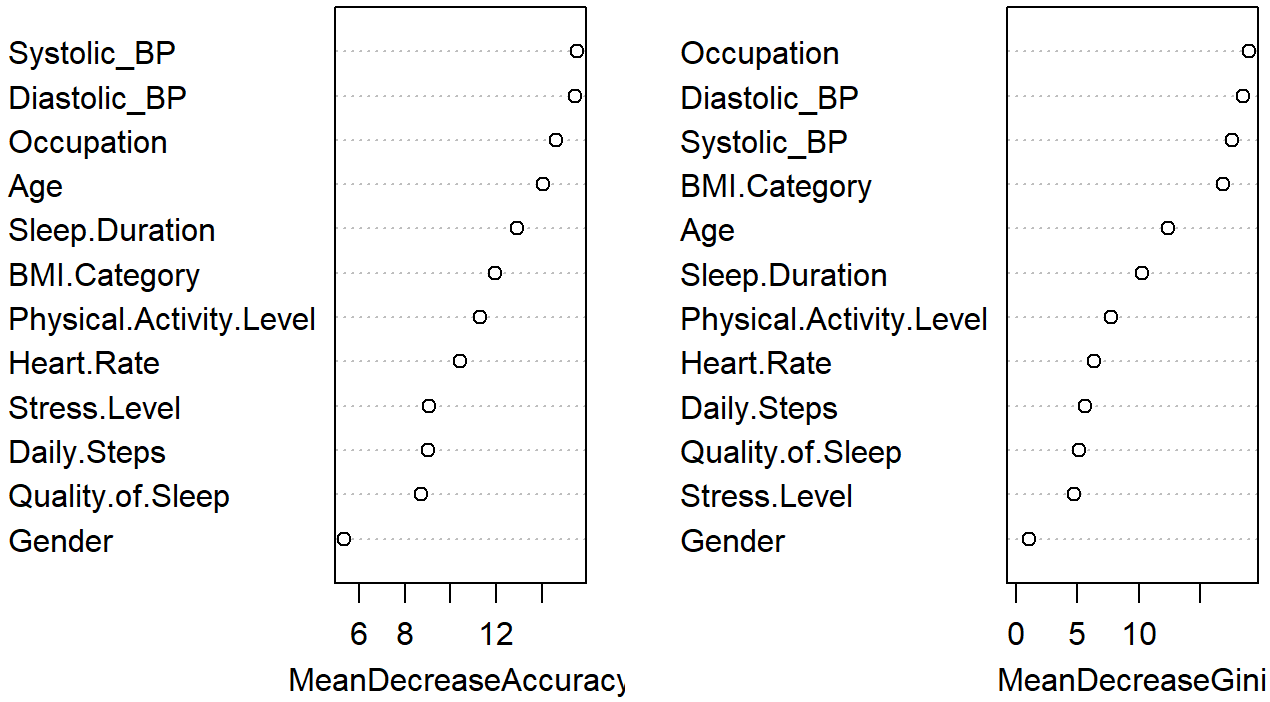
```
mtry_vals <- c(2, 3, 4, 5, 6, 7, 8, 9, 10)
oob_errors <- numeric(length(mtry_vals))

for (i in 1:length(mtry_vals)) {
  set.seed(123)
  rf_temp <- randomForest(Sleep.Disorder ~ ., data = train.set, mtry = mtry_vals[i],ntree = 300)
  oob_errors[i] <- rf_temp$err.rate[300, "OOB"]
  cat("mtry =", mtry_vals[i], "OOB error =", round(oob_errors[i], 4), "\n")
}
```

```
## mtry = 2 OOB error = 0.1
## mtry = 3 OOB error = 0.1036
## mtry = 4 OOB error = 0.1036
## mtry = 5 OOB error = 0.1036
## mtry = 6 OOB error = 0.1071
## mtry = 7 OOB error = 0.1036
## mtry = 8 OOB error = 0.1071
## mtry = 9 OOB error = 0.1071
## mtry = 10 OOB error = 0.1036
```

```
tuning_results <- data.frame(mtry = mtry_vals, OOB_Error = oob_errors)
print(tuning_results)
```

```
##   mtry OOB_Error
## 1    2 0.1000000
## 2    3 0.1035714
## 3    4 0.1035714
## 4    5 0.1035714
## 5    6 0.1071429
## 6    7 0.1035714
## 7    8 0.1071429
## 8    9 0.1071429
## 9   10 0.1035714
```

```
best_mtry <- mtry_vals[which.min(oob_errors)]

### Final Model with best mtry Value ###
set.seed(123)
final_rf <- randomForest(Sleep.Disorder ~ ., data = train.set,
                         mtry = best_mtry,
                         ntree = 300,
                         importance = TRUE)

varImpPlot(final_rf, main = "Random Forest: Variable Importance")
```




```
rf_predictions <- predict(final_rf, newdata = test.set)

conf_matrix <- table(Predicted = rf_predictions, Actual = test.set$Sleep.Disorder)
print(conf_matrix)
```

```
##             Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          17    0           0
##   None               3   50           2
##   Sleep Apnea        0    0          22
```

```
test_error <- 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
test_error
```

```
## [1] 0.05319149
```

```
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
accuracy
```

```
## [1] 0.9468085
```

Gradient Boosting Model (GBM)

Gradient Boosting Model is another tree-based method but it differs from Random Forest because it builds trees sequentially, each one correcting errors made by the previous one. This allows GBM to capture subtle patterns that may not be detected by ensembles of independent trees.

We fit a multinomial GBM model using three levels of interaction depth, shrinkage of 0.01, and 3000 trees. Five-fold cross-validation identifies the optimal number of trees as 172, which helps prevent overfitting and ensures good generalization to unseen data.

On the test set, GBM achieves an accuracy of 92.55%, performing especially well for the Sleep Apnea and None classes. Most misclassifications involve Insomnia, which appears to be the hardest class to seperate consistently across all models.

The variable influence plot shows that Occupation, BMI Category, Systolic_BP, and Age play major roles in classification decisions. A partial dependence plot for Stress Level shows how predicted probabilities change across the stress scale, indicating meaningful nonlinear relationships.

Relevant code:

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.4.3
```

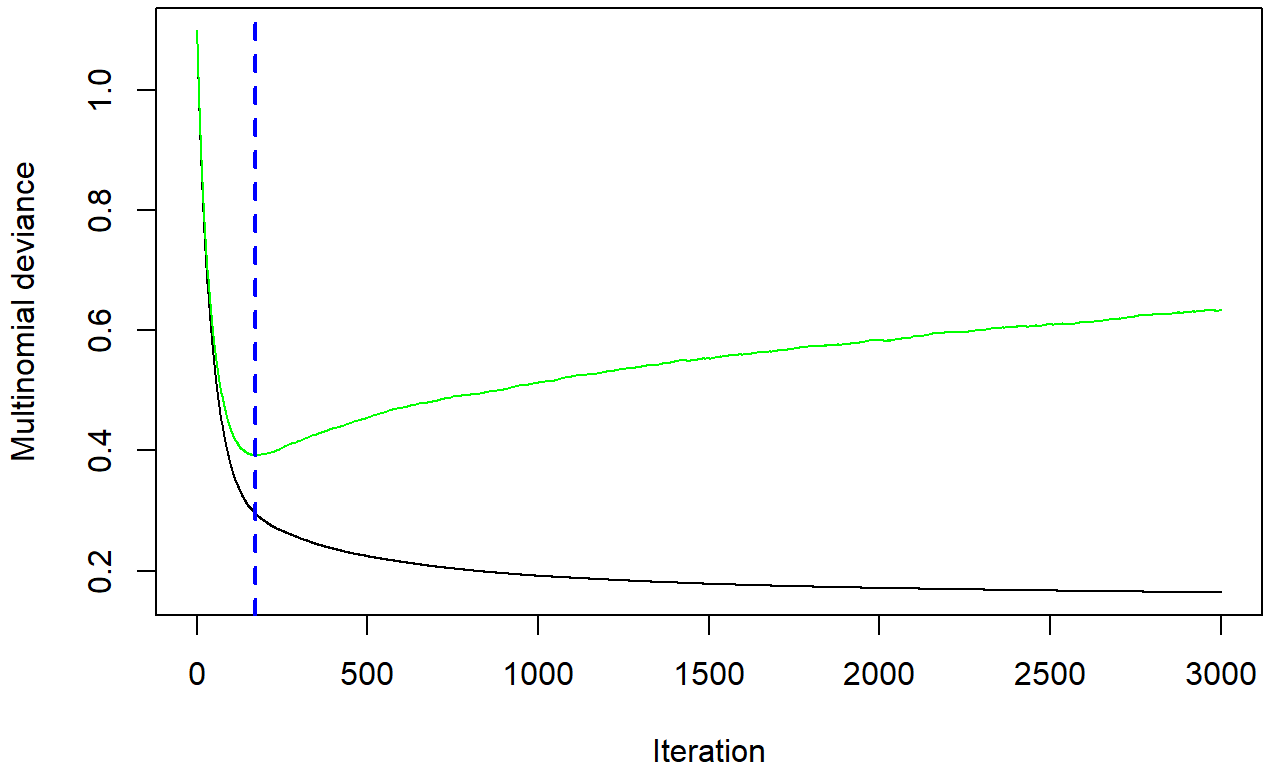
```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://g
ithub.com/gbm-developers/gbm3
```

```
gbm_fit <- gbm(
  Sleep.Disorder ~ .,
  distribution      = "multinomial",
  data              = train.set,
  n.trees           = 3000,
  interaction.depth = 3,
  n.minobsinnode    = 10,
  shrinkage         = 0.01,
  bag.fraction      = 0.8,
  train.fraction    = 1,
  cv.folds          = 5,
  verbose           = FALSE
)
```

```
## Warning: Setting `distribution = "multinomial"` is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
best_iter <- gbm.perf(gbm_fit, method = "cv")
```




```
cat("Optimal number of trees (best_iter):", best_iter, "\n")
```

```
## Optimal number of trees (best_iter): 172
```

```
gbm_prob <- predict(
  gbm_fit,
  newdata = test.set,
  n.trees = best_iter,
  type    = "response"
)

if (length(dim(gbm_prob)) == 3) {
  prob_mat <- gbm_prob[, , 1]
} else {
  prob_mat <- gbm_prob
}

pred_index   <- max.col(prob_mat)
class_levels <- levels(train.set$Sleep.Disorder)

gbm_pred <- factor(
  class_levels[pred_index],
  levels = class_levels
)

gbm_conf_matrix <- table(
  Predicted = gbm_pred,
  Actual    = test.set$Sleep.Disorder
)
gbm_conf_matrix
```

```
##             Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          16    0           1
##   None               3   50           2
##   Sleep Apnea        1    0          21
```

```
gbm_error <- 1 - sum(diag(gbm_conf_matrix)) / sum(gbm_conf_matrix)
cat("Gradient Boosting Test Error Rate:", round(gbm_error, 4), "\n")
```

```
## Gradient Boosting Test Error Rate: 0.0745
```

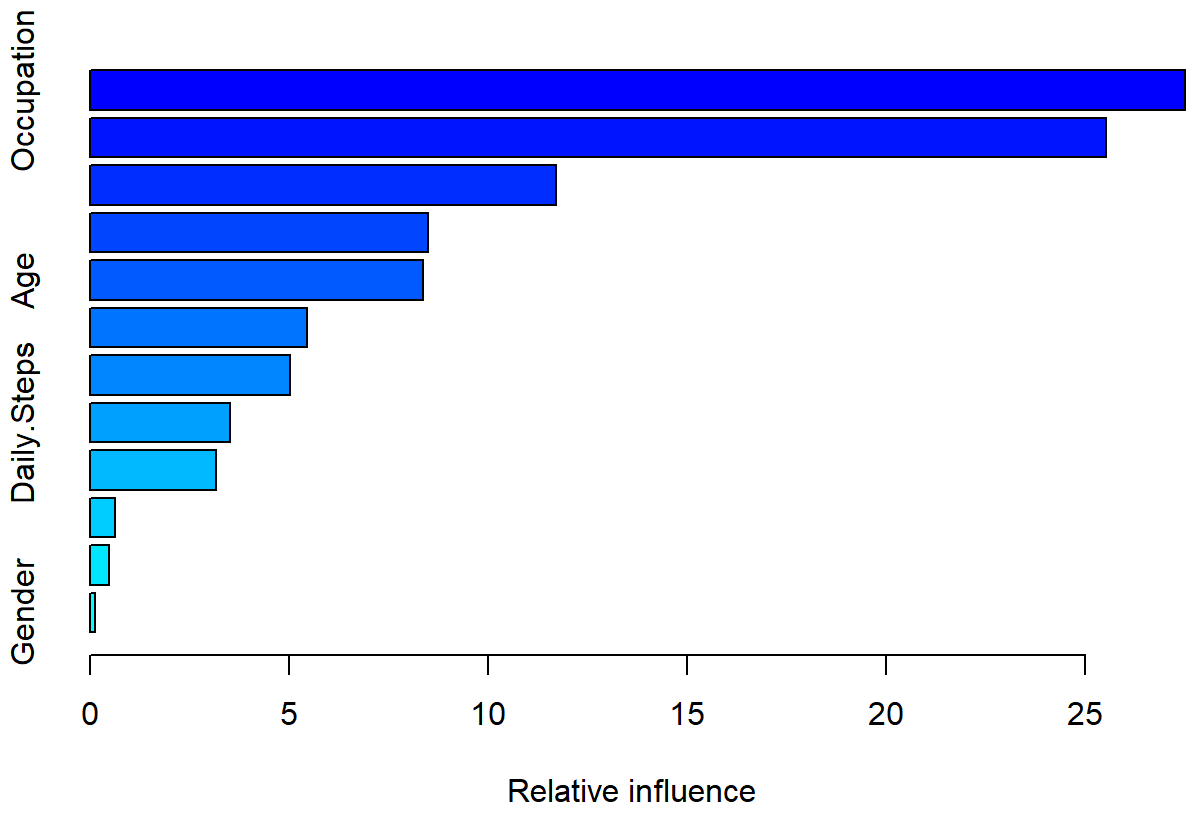
```
gbm_accuracy <- sum(diag(gbm_conf_matrix)) / sum(gbm_conf_matrix)
cat("Gradient Boosting Test Accuracy:", round(gbm_accuracy, 4), "\n")
```

```
## Gradient Boosting Test Accuracy: 0.9255
```

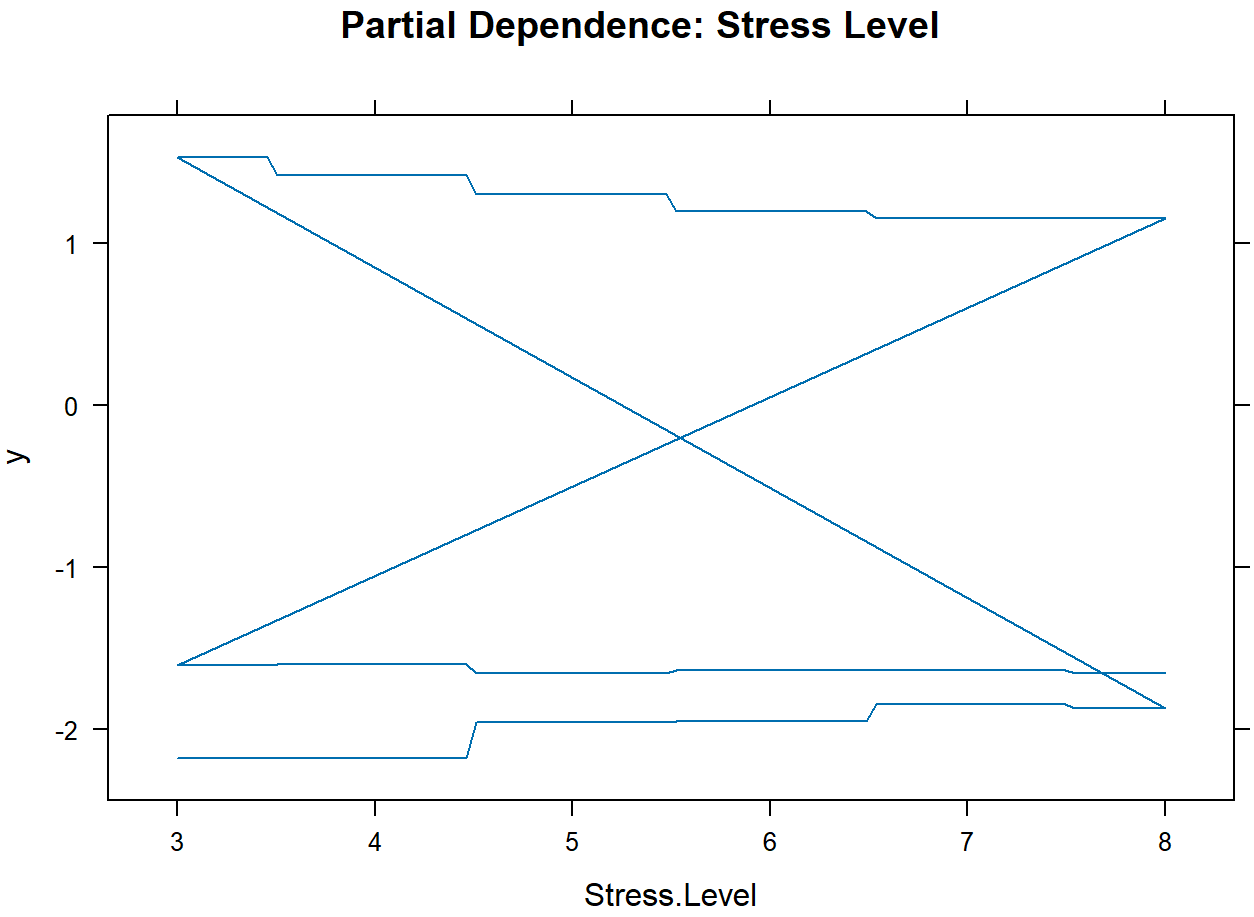
```
cat("\nRelative Influence of Predictors (Gradient Boosting):\n")
```

```
##
## Relative Influence of Predictors (Gradient Boosting):
```

```
gbm_importance <- summary(gbm_fit)
```




```
plot(
  gbm_fit,
  i.var = "Stress.Level",
  main = "Partial Dependence: Stress Level"
)
```



Support Vector Machine (SVM)

SVMs classify data by finding hyperplanes that best separate the classes. Because our predictor set includes many numeric and one-hot encoded variables, a linear SVM is an appropriate and efficient choice.

We fit SVM models using multiple cost values ranging from 0.1 to 50. The cost parameter controls the tolerance for misclassification where low values allow wider margins with more errors, while high values force the model to classify training data more strictly. Through 10-fold cross-validation, we find that cost = 0.5 yields the best performance.

The final SVM model achieves a test accuracy of 93.62%. Like the other models, it performs very well for the Sleep Apnea and None classes, with most confusion occurring between Insomnia and None. A confusion matrix heatmap visually confirms strong separation between classes.

These results show that even a simple linear boundary performs well in distinguishing sleep patterns when enough meaningful predictors are included.

Relevant code:

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.4.3
```

```
train.set$Sleep.Disorder <- as.factor(train.set$Sleep.Disorder)
test.set$Sleep.Disorder  <- as.factor(test.set$Sleep.Disorder)

set.seed(20251114)

## 1. Fit a baseline linear SVM
svm_lin1 <- svm(
  Sleep.Disorder ~ .,
  data   = train.set,
  kernel = "linear",
  cost   = 1,
  scale  = TRUE
)

print(svm_lin1)
```

```
##
## Call:
## svm(formula = Sleep.Disorder ~ ., data = train.set, kernel = "linear",
##     cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  77
```

```
svm_lin1$tot.nSV
```

```
## [1] 77
```

```
svm_lin1$nSV
```

```
## [1] 24 29 24
```

```
svm_lin1_pred <- predict(svm_lin1, newdata = test.set)

svm_lin1_conf <- table(
  Predicted = svm_lin1_pred,
  Actual    = test.set$Sleep.Disorder
)
svm_lin1_conf
```

```
##             Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          17    0           0
##   None               3   49           2
##   Sleep Apnea        0    1          22
```

```
svm_lin1_error <- 1 - sum(diag(svm_lin1_conf)) / sum(svm_lin1_conf)
svm_lin1_acc   <- sum(diag(svm_lin1_conf)) / sum(svm_lin1_conf)

cat("Baseline Linear SVM (cost = 1) - Accuracy:", round(svm_lin1_acc, 4),
    " Error:", round(svm_lin1_error, 4), "\n")
```

```
## Baseline Linear SVM (cost = 1) - Accuracy: 0.9362  Error: 0.0638
```

```
## 2. Increase cost parameter
svm_lin2 <- svm(
  Sleep.Disorder ~ .,
  data   = train.set,
  kernel = "linear",
  cost   = 10,
  scale  = TRUE
)

print(svm_lin2)
```

```
##
## Call:
## svm(formula = Sleep.Disorder ~ ., data = train.set, kernel = "linear",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  77
```

```
svm_lin2$tot.nSV
```

```
## [1] 77
```

```
svm_lin2$nSV
```

```
## [1] 30 22 25
```

```
svm_lin2_pred <- predict(svm_lin2, newdata = test.set)

svm_lin2_conf <- table(
  Predicted = svm_lin2_pred,
  Actual    = test.set$Sleep.Disorder
)
svm_lin2_conf
```

```
##              Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          17    2           0
##   None               3   46           2
##   Sleep Apnea        0    2          22
```

```
svm_lin2_error <- 1 - sum(diag(svm_lin2_conf)) / sum(svm_lin2_conf)
svm_lin2_acc   <- sum(diag(svm_lin2_conf)) / sum(svm_lin2_conf)

cat("Linear SVM (cost = 10) - Accuracy:", round(svm_lin2_acc, 4),
    " Error:", round(svm_lin2_error, 4), "\n")
```

```
## Linear SVM (cost = 10) - Accuracy: 0.9043  Error: 0.0957
```

```
## 3. Choose cost with cross-validation
set.seed(20251114)

tune_lin <- tune.svm(
  Sleep.Disorder ~ .,
  data   = train.set,
  kernel = "linear",
  cost   = c(0.1, 0.5, 1, 5, 10, 20, 50)
)

summary(tune_lin)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.5
##
## - best performance: 0.1035714
##
## - Detailed performance results:
##   cost     error dispersion
## 1  0.1 0.1142857 0.05783313
## 2  0.5 0.1035714 0.06617591
## 3  1.0 0.1035714 0.06399848
## 4  5.0 0.1107143 0.07032884
## 5 10.0 0.1142857 0.07103064
## 6 20.0 0.1107143 0.06828395
## 7 50.0 0.1107143 0.06399848
```

```
best_lin_svm <- tune_lin$best.model
best_lin_svm
```

```
##
## Call:
## best.svm(x = Sleep.Disorder ~ ., data = train.set, cost = c(0.1,
##     0.5, 1, 5, 10, 20, 50), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.5
##
## Number of Support Vectors:  73
```

```
svm_best_pred <- predict(best_lin_svm, newdata = test.set)

svm_best_conf <- table(
  Predicted = svm_best_pred,
  Actual    = test.set$Sleep.Disorder
)
svm_best_conf
```

```
##              Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          17    0           0
##   None               3   49           2
##   Sleep Apnea        0    1          22
```

```
svm_best_error <- 1 - sum(diag(svm_best_conf)) / sum(svm_best_conf)
svm_best_acc   <- sum(diag(svm_best_conf)) / sum(svm_best_conf)

cat("Best Linear SVM (tuned cost) - Accuracy:", round(svm_best_acc, 4),
    " Error:", round(svm_best_error, 4), "\n")
```

```
## Best Linear SVM (tuned cost) - Accuracy: 0.9362  Error: 0.0638
```

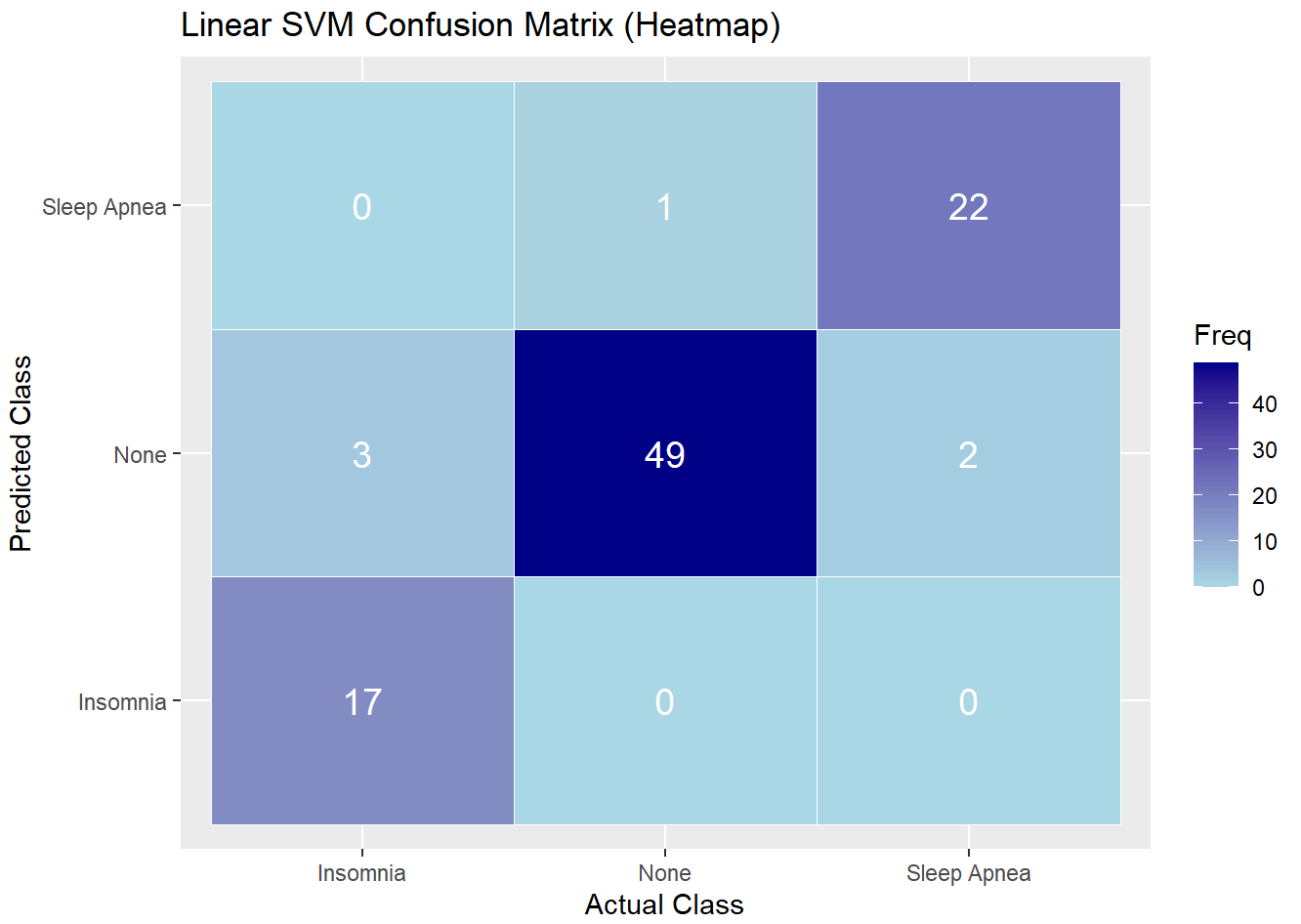
```
## 4. Confusion matrix heatmap
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
##
## Attaching package: 'ggplot2'
```

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

```
cm_df <- as.data.frame(svm_best_conf)
colnames(cm_df) <- c("Predicted", "Actual", "Freq")

ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  geom_text(aes(label = Freq), color = "white", size = 5) +
  labs(
    title = "Linear SVM Confusion Matrix (Heatmap)",
    x     = "Actual Class",
    y     = "Predicted Class"
  )
```



Logistic Regression Model

Multinomial logistic regression provides an interpretable statistical baseline for the classification task. Before fitting the model, we convert all categorical predictors into dummy variables using model.matrix(), which allows the algorithm to operate on numeric inputs.

The logistic regression model achieves a test accuracy of 92.55%, performing consistently across all classes. The output consists of estimated log-odds coefficients for every predictor–class combination. This structure makes it easy to identify which characteristics increase or decrease the likelihood of a person being assigned to a particular sleep disorder category.

Although logistic regression does not capture nonlinear interactions as effectively as tree-based models, its interpretability makes it valuable in contexts where understanding variable effects is essential.

Relevant code:

```
library(nnet)
set.seed(20251114)

train_mm <- model.matrix(Sleep.Disorder ~ ., data = train.set)
test_mm  <- model.matrix(Sleep.Disorder ~ ., data = test.set)

train_df <- as.data.frame(train_mm)
test_df  <- as.data.frame(test_mm)

train_df$Sleep.Disorder <- train.set$Sleep.Disorder
test_df$Sleep.Disorder  <- test.set$Sleep.Disorder

logit_model <- multinom(Sleep.Disorder ~ . , data = train_df)
```

```
## # weights:  78 (50 variable)
## initial  value 307.611441
## iter  10 value 163.628595
## iter  20 value 88.317563
## iter  30 value 68.853959
## iter  40 value 67.549248
## iter  50 value 66.842973
## iter  60 value 66.269250
## iter  70 value 65.489199
## iter  80 value 64.562317
## iter  90 value 62.227265
## iter 100 value 61.701013
## final  value 61.701013
## stopped after 100 iterations
```

```
logit_pred <- predict(logit_model, newdata = test_df)

logit_conf_matrix <- table(Predicted = logit_pred, Actual = test_df$Sleep.Disorder)
print(logit_conf_matrix)
```

```
##              Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          17    1           0
##   None               3   48           2
##   Sleep Apnea        0    1          22
```

```
logit_accuracy <- sum(diag(logit_conf_matrix)) / sum(logit_conf_matrix)
logit_accuracy
```

```
## [1] 0.9255319
```

```
logit_error <- 1 - logit_accuracy
logit_error
```

```
## [1] 0.07446809
```

LASSO Model

LASSO extends logistic regression by applying a penalty that shrinks weak coefficients toward zero. This helps with variable selection and prevents overfitting in high-dimensional settings. We use 10-fold cross-validation to identify the best penalty parameter, resulting in an optimal lambda of 0.00573.

The LASSO model achieves a test accuracy of 90.43%, slightly lower than logistic regression. However, LASSO provides insight into the most influential variables by shrinking insignificant coefficients to zero. Predictors that remain include Sleep Quality, Sleep Duration, Stress Level, and Heart Rate.

Although LASSO sacrifices some accuracy, it produces a simpler and more interpretable model, highlighting key predictors that consistently influence sleep disorder classification.

Relevant code:

```
library("glmnet")
```

```
## Warning: package 'glmnet' was built under R version 4.4.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-10
```

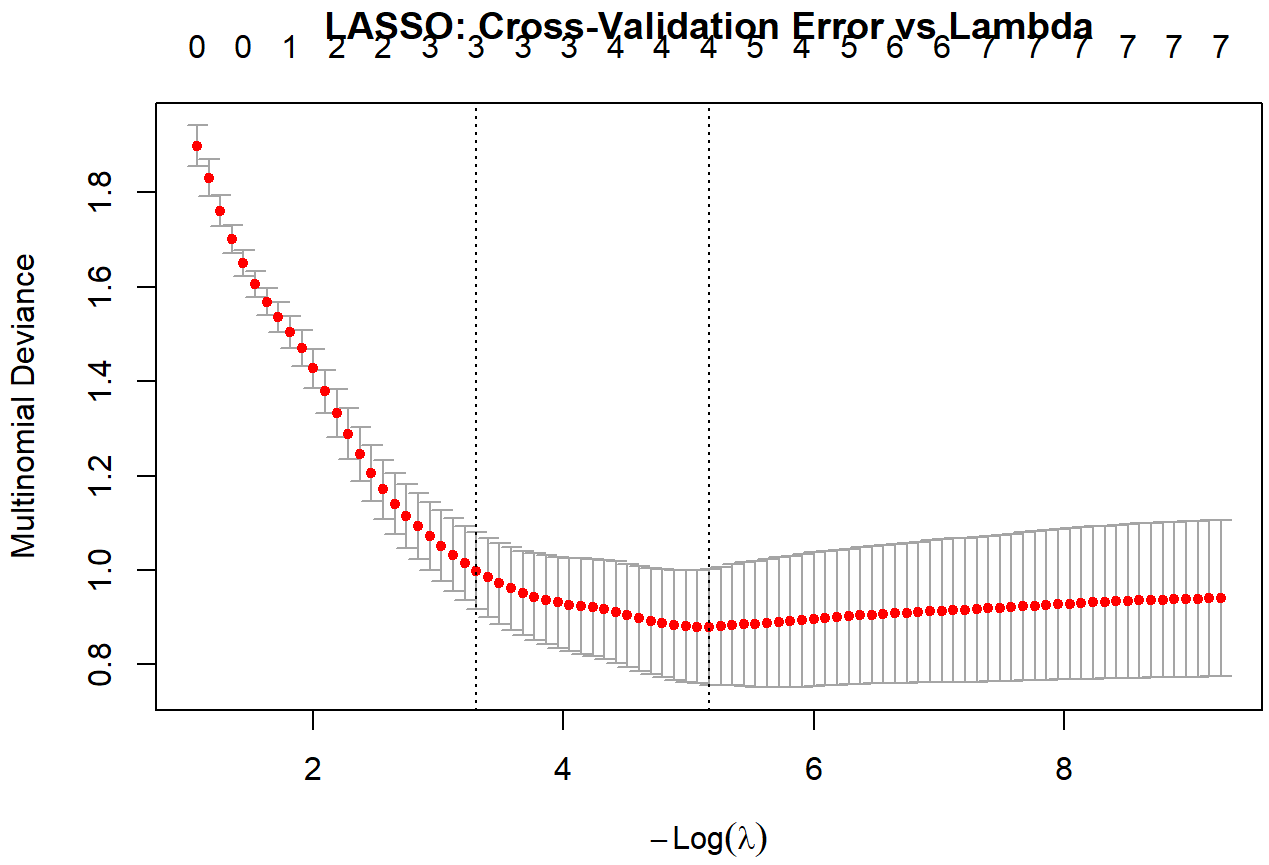
```
xmat_train <- as.matrix(train.set[, !names(train.set) %in% c("Sleep.Disorder")])
y_train <- as.numeric(train.set$Sleep.Disorder)
xmat_test <- as.matrix(test.set[, !names(test.set) %in% c("Sleep.Disorder")])
y_test <- as.numeric(test.set$Sleep.Disorder)

set.seed(20251114)

cvfit <- cv.glmnet(xmat_train, y_train,
                   family = "multinomial",
                   alpha = 1,
                   nfolds = 10)
```

```
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
```

```
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
```

```
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
```

```
lambest_min <- cvfit$lambda.min
lambest_1se <- cvfit$lambda.1se

plot(cvfit, main = "LASSO: Cross-Validation Error vs Lambda")
```




```
lasso_model <- glmnet(xmat_train, y_train,
                      family = "multinomial",
                      alpha = 1,
                      lambda = lambest_min)
```

```
## Warning in storage.mode(x) <- "double": NAs introduced by coercion
```

```
lasso_coef <- coef(lasso_model)

class_names <- levels(train.set$Sleep.Disorder)
for(i in 1:length(lasso_coef)) {
  cat("\nClass:", class_names[i], "\n")

  coef_class <- as.matrix(lasso_coef[[i]])
  non_zero <- which(coef_class != 0)

  if(length(non_zero) > 1) {
    for(j in non_zero) {
      if(rownames(coef_class)[j] != "(Intercept)") {
        cat(sprintf("%-20s: %8.4f\n", rownames(coef_class)[j], coef_class[j]))
      }
    }
  } else {
    cat("(All feature coefficients shrunk to zero)\n")
  }
}
```

```
##
## Class: Insomnia
## Age                 :   0.0378
## Sleep.Duration      :  -0.6039
## Heart.Rate          :  -0.0150
## Daily.Steps         :  -0.0006
##
## Class: None
## Age                 :  -0.1343
## Quality.of.Sleep    :   1.7694
## Physical.Activity.Level:  -0.0041
## Stress.Level        :   0.4183
## Diastolic_BP        :  -0.1888
##
## Class: Sleep Apnea
## Sleep.Duration      :   1.3758
## Diastolic_BP        :   0.1411
## Heart.Rate          :   0.2480
```

```
lasso_pred <- predict(lasso_model,
                      newx = xmat_test,
                      type = "class",
                      s = lambest_min)
```

```
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
## Warning in cbind2(1, newx) %*% (nbeta[[i]]): NAs introduced by coercion
```

```
lasso_pred_factor <- factor(lasso_pred,
                            levels = 1:3,
                            labels = class_names)

conf_matrix <- table(Predicted = lasso_pred_factor,
                     Actual = test.set$Sleep.Disorder)

print(conf_matrix)
```

```
##              Actual
## Predicted     Insomnia None Sleep Apnea
##   Insomnia          16    1           2
##   None               3   49           2
##   Sleep Apnea        1    0          20
```

```
lasso_accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
lasso_accuracy
```

```
## [1] 0.9042553
```

```
lasso_error <- 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
lasso_error
```

```
## [1] 0.09574468
```

Model Comparison and Recommendation

Across all models, accuracy scores fall between 90% and 95%, indicating that sleep disorders can be predicted reliably from lifestyle and health data. Random Forest achieves the highest accuracy at 94.68%, followed closely by SVM at 93.62%. Gradient Boosting and Logistic Regression both achieve 92.55%, while LASSO performs slightly lower at 90.43%.

Random Forest offers the best balance of accuracy, stability, and interpretability through its variable importance measures. GBM is particularly strong in classifying Sleep Apnea, while SVM shows balanced performance across all classes. Logistic Regression provides interpretability, and LASSO identifies the smallest subset of meaningful predictors.

Overall, we recommend Random Forest as the preferred model due to its superior accuracy and robust handling of mixed predictor types. However, GBM and SVM remain strong alternatives, and logistic-based models help support interpretability and feature selection.

Conclusion

Our analysis demonstrates that machine learning methods can effectively predict sleep disorder categories using demographic, lifestyle, and cardiovascular data. All models achieved high accuracy, with Random Forest performing the best overall. Key predictors across models include sleep duration, sleep quality, stress level, heart rate, occupation, and blood pressure.

The dataset has some limitations, including its cross-sectional nature and broad categorical variables such as Occupation and BMI Category. Future improvements could involve collecting time-series sleep data, adding more detailed lifestyle measurements, or experimenting with advanced models like XGBoost or neural networks.

Despite these limitations, our findings support the idea that sleep disorders can be reliably identified based on readily measurable health factors, offering valuable insight for both research and potential clinical applications.