Projeto - Qualidade de Sono e Análises de Distúrbios

Luiza Moreira Salgado - 11202021683
João Hugo Martins da Luz - 11202021919
Amanda Cruz Francesconi - 11020315
Eduardo Teixeira de Aguiar Veiga - 11202021505

15 August, 2023

Introdução

Importância do sono

O sono é essencial para a saúde e o bem-estar geral. Ele ajuda a reparar o corpo, a consolidar a memória e o aprendizado, e a regular o humor e a emoções. Quando não dormimos o suficiente, podemos sentir-nos cansados, irritados, com dificuldade de concentração e de tomar decisões. Também podemos ser mais propensos a ter acidentes e a cometer erros.

O sono é dividido em quatro estágios:

Estágio 1: é um estágio leve de sono, durante o qual ainda estamos conscientes do nosso entorno. Estágio 2: é um estágio mais profundo de sono, durante o qual nossa respiração e batimentos cardíacos começam a abrandar. Estágio 3: é um estágio ainda mais profundo de sono, durante o qual é difícil acordar. Estágio 4: é o estágio mais profundo de sono, durante o qual nosso corpo está em repouso completo.

O sono REM (movimento rápido dos olhos) é um estágio de sono caracterizado por movimentos rápidos dos olhos, sonhos vívidos e atividade cerebral elevada. Este estágio é importante para a consolidação da memória e do aprendizado.

Distúrbios do Sono

Os distúrbios do sono são problemas que interferem na qualidade do sono. Eles podem ser causados por uma variedade de fatores, incluindo estresse, ansiedade, depressão, doenças físicas, uso de medicamentos e hábitos de vida inadequados. Os distúrbios do sono podem ter um impacto significativo na qualidade de vida, afetando o trabalho, os relacionamentos e a saúde geral.

Alguns dos distúrbios do sono mais comuns incluem:

Insônia: é a dificuldade de iniciar ou manter o sono. Apneia do sono: é uma condição na qual a respiração é interrompida durante o sono.

Análise de Dados com Machine Learning

A análise de dados com Machine Learning pode ser usada para identificar as causas dos distúrbios do sono e para desenvolver tratamentos mais eficazes. Os algoritmos de machine learning podem ser usados para analisar grandes quantidades de dados, como registros médicos, registros de sono e hábitos de vida. Isso pode ajudar os pesquisadores a identificar padrões e tendências que não seriam visíveis a olho nu.

Base de Dados

A base de dados a ser analisada foi retirada do site Kaggle, no seguinte endereço:

https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset?resource=download

Dados apresentados

Os dados que são apresentados e que serão utilizados em nossas análises são:

- ID (que será retirado pois não auxilia em nossas análises e previsões);
- Gênero (variável categórica);
- Idade;
- Ocupação (variável categórica);
- Duração do sono;
- Qualidade do sono (em uma escala de 1-10);
- Tempo de atividade física (em minutos/dia);
- Nível de estresse (em uma escala de 1-10);
- IMC (variável categórica);
- Pressão sanguínea;
- Média de batimentos cardíacos em repouso;
- Quantidade de passos diários;

A variável a ser prevista (ou seja, a variável alvo) é "Distúrbio do sono", uma variável categórica que diz se aquela pessoa não possui nenhum distúrbio, se possui "Insônia" ou se possui "Apneia". Para algumas análises futuras binárias, resumiremos em alguns modelos "Insônia"e "Apneia" em apenas uma variável, "Positive", indicando que há um distúrbio de sono naquele determinado indivíduo.

Análises iniciais

A partir de análises inciais, começaremos a trabalhar com os dados filtrando e modificando algumas variáveis necessárias.

A coluna de ID, por exemplo, sabe-se que não irá influenciar nas previsões - pelo contrário, pode acabar atrapalhando na modelagem. por isso, retiraremos essa característica.

Além disso, as colunas com variáveis categóricas serão tratadas utilizando a função "set_dummy", ou seja, transformadas em variáveis Dummy.

Bibliotecas necessárias

```
library(tidymodels)
library(tidyverse)
```

Importando o dataframe

```
# Coluna a ser predita: "Sleep.Disorder"

df <- read.csv("/cloud/project/assets/Sleep_health_and_lifestyle_dataset.csv") %>%
    na.omit() %>%
    mutate(Sleep.Disorder = ifelse(
                    Sleep.Disorder %in% c("Sleep Apnea", "Insomnia"),
                    "Positive",
                    Sleep.Disorder
                )
```

```
            )
table(df$Sleep.Disorder)["Positive"]
```

```
## Positive
##      155
```

```
table(df$Sleep.Disorder)["None"]
```

```
## None
##  219
```

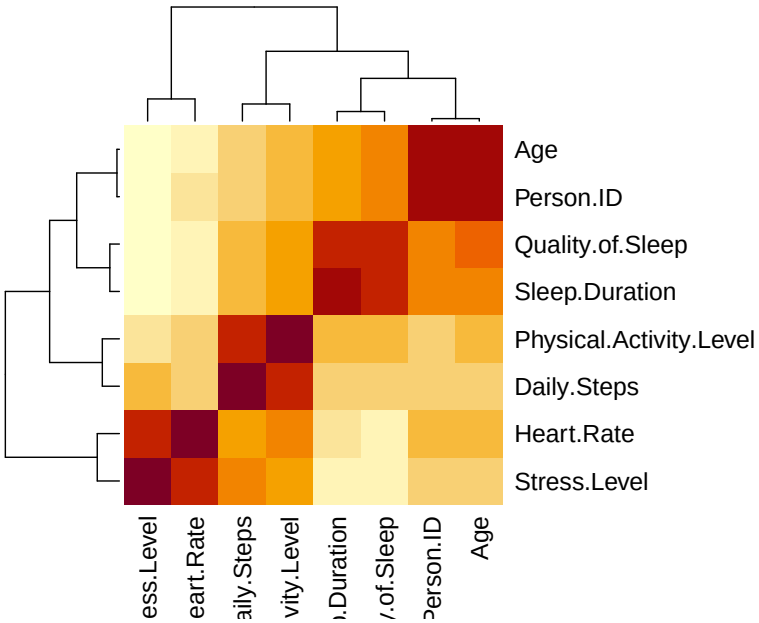
Modificamos a coluna "Sleep.Disorder", unificando os distúrbios de sono em uma classe só. Uma vez que a proporção da quantidade de quem possui qualquer um dos distúrbios de sono (155) é parecida com a quantidade dos que não têm (219), podemos fazer essa junção das classes em uma só e tratar este problema como Classificação Binária - inclusive, nos trará melhores resultados, uma vez que os dados estão mais balanceados desta maneira.

Gráficos iniciais

Mapa de calor - correlação

Podemos analisar o quanto as variáveis estão correlacionadas entre si a partir de um mapa de calor. Para isso, selecionamos apenas as variáveis numéricas (pois apenas elas podem entrar no cálculo de correlação):

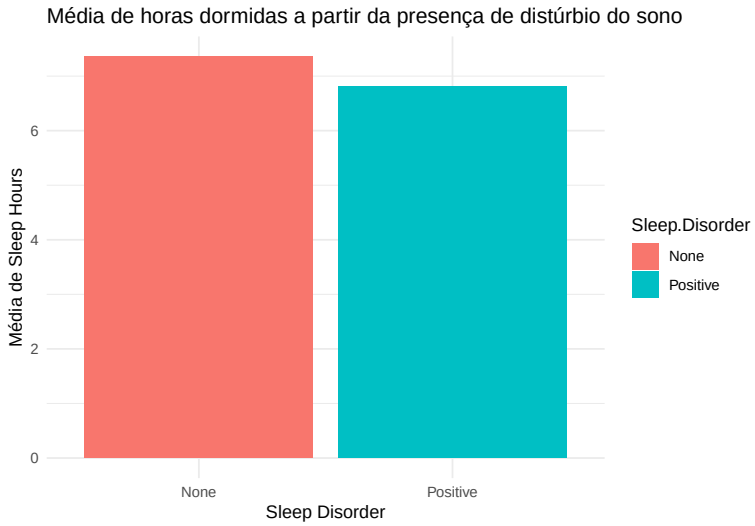
```
data_cor <- df %>% select_if(is.numeric)

cor_matrix <- cor(data_cor)

heatmap(cor_matrix)
```



Podemos tirar algumas análises deste gráfico, como a relação forte que há entre a média de batimentos cardíacos e nível de estresse. Isso indica que quanto maior a média de batimentos, maior o nível de estresse que aquela pessoa possui. Além de outras relações fortes, como a qualidade de sono e a duração do sono, uma vez que quanto maior a duração em horas, melhor é o sono daquele indivíduo.

Gráfico média de sono x distúrbio de sono

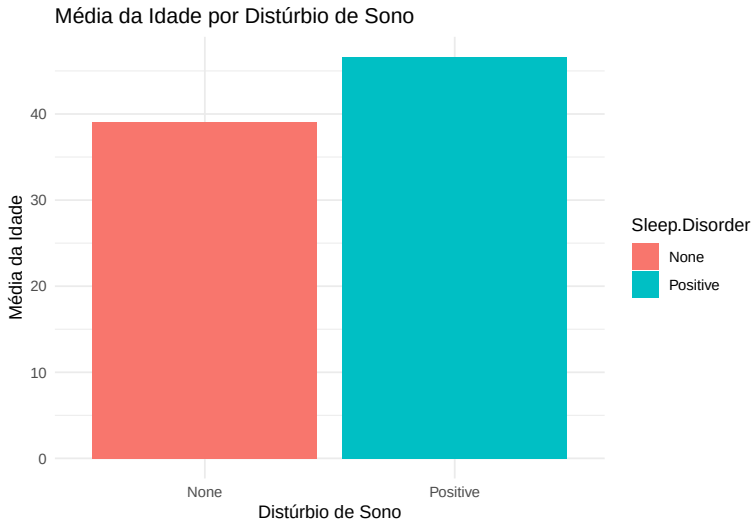
```
df %>%
  group_by(Sleep.Disorder) %>%
  summarise(avg_sleep_hours = mean(Sleep.Duration, na.rm = TRUE)) %>%
  ggplot(aes(x = Sleep.Disorder, y = avg_sleep_hours, fill = Sleep.Disorder)) +
  geom_bar(stat = "identity") +
  labs(title = "Média de horas dormidas a partir da presença de distúrbio do sono",
       x = "Sleep Disorder",
       y = "Média de Sleep Hours") +
  theme_minimal()
```



A média de horas dormidas em quem possui distúrbio de sono é menor.

Gráfico média de idade x distúrbio do sono

```
df %>%
  group_by(Sleep.Disorder) %>%
  summarise(avg_age = mean(Age, na.rm = TRUE)) %>%
  ggplot(aes(x = Sleep.Disorder, y = avg_age, fill = Sleep.Disorder)) +
  geom_bar(stat = "identity") +
  labs(title = "Média da Idade por Distúrbio de Sono",
       x = "Distúrbio de Sono",
       y = "Média da Idade") +
  theme_minimal()
```

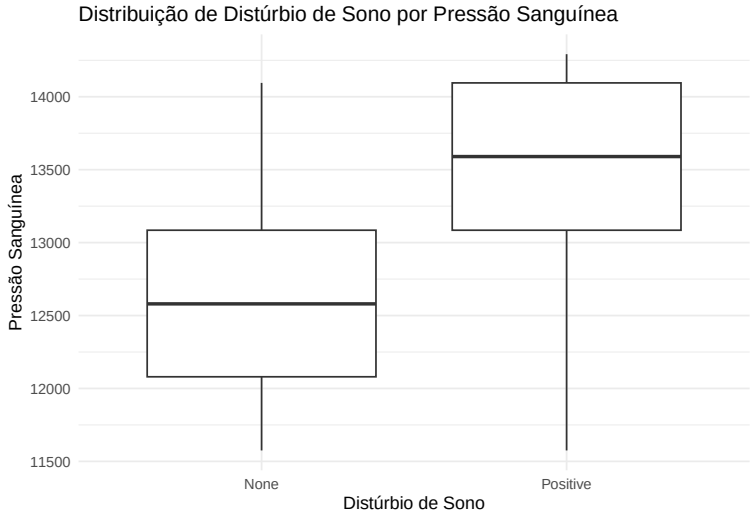


A média de idade de pessoas que possuem distúrbio de sono é maior em comparação às pessoas que não têm.

Gráfico pressão arterial x distúrbio de sono

Tiramos a "/" do dado da pressão sanguínea para poder realizar o "box plot".

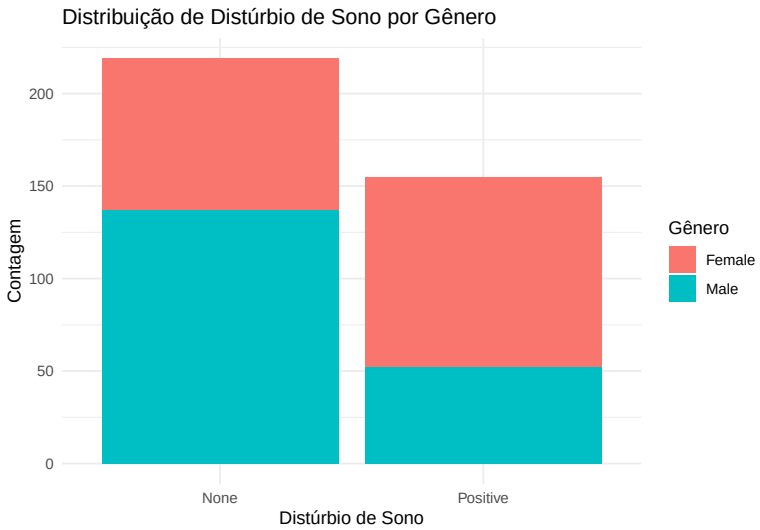
```
data <- df %>% mutate(Blood.Pressure = gsub("/", "", Blood.Pressure))
ggplot(data, aes(x = Sleep.Disorder, y = as.numeric(Blood.Pressure))) +
  geom_boxplot() +
  labs(title = "Distribuição de Distúrbio de Sono por Pressão Sanguínea",
       x = "Distúrbio de Sono",
       y = "Pressão Sanguínea") +
  theme_minimal()
```



Com uma análise superficial da relação entre pressão e distúrbio de sono, nota-se que a tendência maior é que pessoas com algum tipo de distúrbio de sono possuem uma média de pressão sanguínea mais elevada comparada à pessoas que não têm nenhum distúrbio.

Gráfico gênero x distúrbio de sono

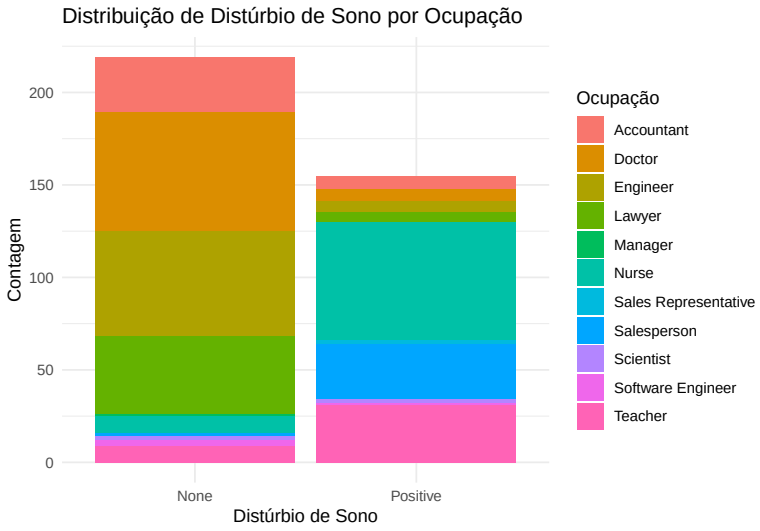
```
ggplot(df, aes(x = Sleep.Disorder, fill = Gender)) +
  geom_bar() +
  labs(title = "Distribuição de Distúrbio de Sono por Gênero",
       x = "Distúrbio de Sono",
       fill = "Gênero",
       y = "Contagem") +
  theme_minimal()
```



Em proporção, o genêro que mais têm problemas de sono é o gênero feminino.

Gráfico ocupação x distúrbio de sono

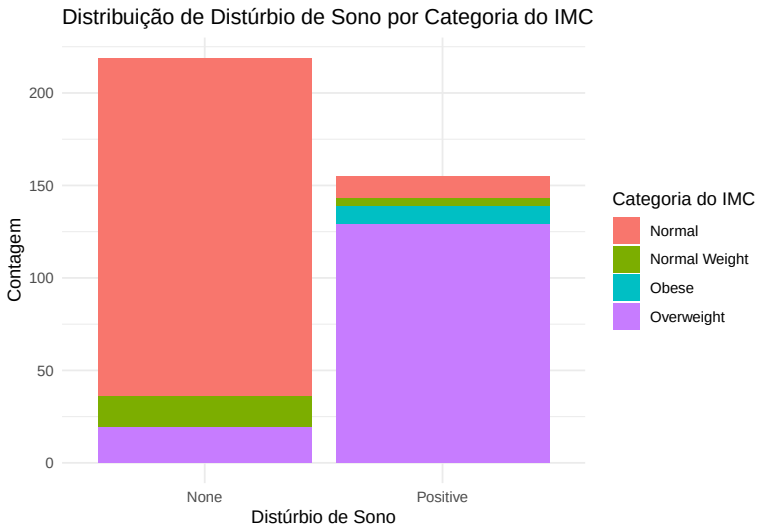
```
ggplot(data, aes(x = Sleep.Disorder, fill = Occupation)) +
  geom_bar() +
  labs(title = "Distribuição de Distúrbio de Sono por Ocupação",
       x = "Distúrbio de Sono",
       fill = "Ocupação",
       y = "Contagem") +
  theme_minimal()
```



Por curiosidade, fizemos essa análise. Em proporção, enfermeiros e vendedores são os que mais possuem distúrbios de sono, seguido de professores.

Gráfico IMC x distúrbio de sono

```
ggplot(data, aes(x = Sleep.Disorder, fill = `BMI.Category`)) +
  geom_bar() +
  labs(title = "Distribuição de Distúrbio de Sono por Categoria do IMC",
       x = "Distúrbio de Sono",
       fill = "Categoria do IMC",
       y = "Contagem") +
  theme_minimal()
```



Pessoas na categoria “sobrepeso” no IMC geralmente possuem mais distúrbios de sono.

Modelos

Iniciamos, então, a modelagem com os dados para buscar a melhor alternativa na hora de prever distúrbios de sono entre os dados propostos.

Análises Discriminantes (LDA, QDA e Naive Bayes)

Realizamos as análises discriminantes a partir dos algoritmos LDA, QDA e Naive Bayes a fim de encontrar a fronteira de decisão que melhor separa as classes no espaço de atributos (no nosso caso, Sleep.Disorder) com base nos rótulos de classe conhecidos.

```
library(tidymodels)
library(discrim)
library(ggplot2)

folds <- vfold_cv(df, v = 10)
rec <- recipe(Sleep.Disorder ~ ., df) %>%
    # Remover a Coluna Id que não vai nos ajudar
    step_rm(Person.ID) %>%
    # Converte as variáveis nominais para dummy variables
    # exceto pela saída, que não queremos converter.
    step_dummy(all_nominal(), -all_outcomes()) %>%
    # Vamos agora centrar e escalonar nossas
    # variáveis
    step_center(all_predictors()) %>%
    step_scale(all_predictors())
LDA <- discrim_regularized(frac_common_cov = 1) %>%
    set_engine("klaR")
res_lda <- fit_resamples(
  LDA,
  rec,
  folds,
  metrics = NULL,
  control = control_resamples()
)
metrics_lda <- res_lda %>%
  collect_metrics()
print(metrics_lda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.930    10  0.0123 Preprocessor1_Model1
## 2 roc_auc  binary     0.946    10  0.0128 Preprocessor1_Model1
```

```
QDA <- discrim_regularized(frac_common_cov = 0) %>%
    set_engine("klaR")
res_qda <- fit_resamples(
  QDA,
  rec,
  folds,
  metrics = NULL,
  control = control_resamples()
)
metrics_qda <- res_qda %>%
  collect_metrics()
  print(metrics_qda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
```

```
## 1 accuracy binary     0.933    10  0.0135 Preprocessor1_Model1
## 2 roc_auc  binary     0.942    10  0.0107 Preprocessor1_Model1
```

```
naive <- naive_Bayes() %>%
set_engine("klaR")
res_naive <- fit_resamples(
  naive,
  rec,
  folds,
  metrics = NULL,
  control = control_resamples()
)
metrics_naive <- res_naive %>%
  collect_metrics()
  print(metrics_naive)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.914    10  0.0139 Preprocessor1_Model1
## 2 roc_auc  binary     0.926    10  0.0185 Preprocessor1_Model1
```

Com o resultado das métricas da acurácia e ROC, podemos analisar que o QDA foi o que melhor obteve resultado. Isso pode significar que o relacionamento entre as variáveis preditoras (atributos) e a variável de classe é mais complexo e não pode ser bem modelado por uma fronteira de decisão linear ou por uma suposição de covariância comum entre as classes. Como a diferença não foi muito significativo, ficamos apenas com a suposição de que o problema não é linearmente descrito.

Regressão Logística

```
library(tidymodels)
library(glmnet)
library(tidyverse)
library(yardstick)

rec <- recipe(Sleep.Disorder ~ ., df) %>%
    # Remover a Coluna Id que não vai nos ajudar
    step_rm(Person.ID) %>%
    # Converte as variáveis nominais para dummy variables
    # exceto pela saída, que não queremos converter.
    step_dummy(all_nominal(), -all_outcomes()) %>%
    # Vamos agora centrar e escalonar nossas
    # variáveis
    step_center(all_predictors()) %>%
    step_scale(all_predictors())

lr.model <- logistic_reg(penalty = 0,
    mixture = NULL
    ) %>% set_engine("glmnet")

set.seed(42)
ind <- sample(2, 374, replace = TRUE, prob = c(0.7,0.3))
treino <- df[ind == 1,]
teste  <- df[ind == 2,]
```

```r
rec.prep <- rec %>% prep(treino)

train.prep <- juice(rec.prep)
test.prep <- bake(rec.prep, teste)

lr.fit <- lr.model %>% fit(Sleep.Disorder ~ ., train.prep)

test.pred <- test.prep %>%
bind_cols(lr.fit %>% predict(new_data = test.prep))

matrix_result <- conf_mat(test.pred,Sleep.Disorder,.pred_class)$table

print(matrix_result)
```

```
##           Truth
## Prediction None Positive
##   None       60        6
##   Positive    1       41
```

```r
recall_value <- recall(test.pred, truth = Sleep.Disorder, estimate = .pred_class, event_level="second")
print(recall_value)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 recall  binary         0.872
```

```r
precision_value <- precision(test.pred, truth = Sleep.Disorder, estimate = .pred_class, event_level="se
print(precision_value)
```

```
## # A tibble: 1 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.976
```

```r
sensitivity_value <- sensitivity(test.pred, truth = Sleep.Disorder, estimate = .pred_class, event_level=
print(sensitivity_value)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity binary         0.872
```

```r
specificity_value <- specificity(test.pred, truth = Sleep.Disorder, estimate = .pred_class, event_level=
print(specificity_value)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 specificity binary         0.984
```

Podemos interpretar os valores da matriz da seguinte forma:

Verdadeiros negativos (TN): 60 - São as instâncias que foram corretamente classificadas como “None” (classe negativa).

Falsos positivos (FP): 6 - São as instâncias que foram erroneamente classificadas como “Positive” (classe positiva), mas que na verdade pertencem à classe “None”.

Falsos negativos (FN): 1 - São as instâncias que foram erroneamente classificadas como “None”, mas que na verdade pertencem à classe “Positive”.

Verdadeiros positivos (TP): 41 - São as instâncias que foram corretamente classificadas como “Positive”.

Uma alta acurácia, sensibilidade, especificidade e precisão indicam que o modelo está fazendo boas previsões e é capaz de discriminar corretamente entre as classes. Em resumo, esses resultados sugerem que o modelo de regressão logística tem um bom desempenho na detecção de casos positivos, com um recall respeitável de 87,2%. Além disso, a alta precisão de 97,6% indica que a grande maioria das previsões positivas feitas pelo modelo está correta. A especificidade alta de 98,6% indica que o modelo também é eficaz em identificar corretamente os casos negativos.

No entando, deve-se lembrar que o conjunto de amostras não é consideravelmente grande, e modelos geralmente acabam se “viciando” com os poucos dados apresentados. Talvez estas métricas não sejam os melhores apoios para conferir a modelagem.

Perceptron

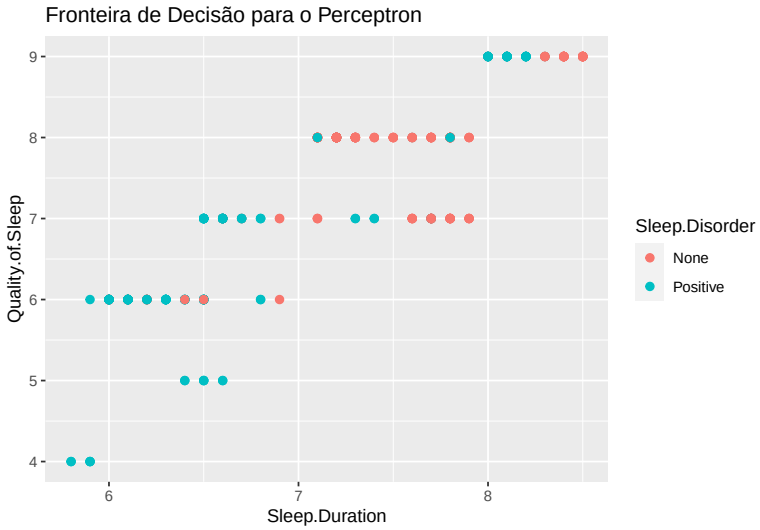
```r
library(mlr3)
library(mlr3learners)
library(mlr3viz)
library(tidymodels)
library(modelr)

# Definindo a função
perceptron.fit <- function(X, y, eta = 0.01, epochs = 100) {
  n <- nrow(X)
  p <- ncol(X)
  w <- matrix(rnorm(p), ncol = 1)
  b <- rnorm(1)

  for (epoch in 1:epochs) {
    e <- 0
    for (i in 1:n) {
      x <- as.numeric(X[i, ])
      x[is.na(x)] <- 0

      pred <- sum(x * w) + b
      target <- ifelse(y[i] == "Positive", 1, -1)

      if (!is.na(pred) && target * pred <= 0) {
        w <- w + eta * target * x
        b <- b + eta * target
        e <- e + 1
      }
    }

    if (e == 0) {
      cat("Converged after", epoch, "epochs.\n")
      break
    }
  }

  return(list("weights" = w, "intercept" = b))
}
```

```r
# Função para as predições
perceptron.predict <- function(model, X) {
  w <- model$weights
  b <- model$intercept

  n <- nrow(X)
  pred <- vector("character", length = n)

  for (i in 1:n) {
    x <- as.numeric(X[i, ])
    x[is.na(x)] <- 0
    result <- sum(x * w) + b

    if (result > 0) {
      pred[i] <- "Positive"
    } else {
      pred[i] <- "Negative"
    }
  }

  return(pred)
}

# Separação que estamos utilizando entre treino e teste
set.seed(42)
ind <- sample(2, 374, replace = TRUE, prob = c(0.7,0.3))
train_data <- df[ind == 1,]
test_data  <- df[ind == 2,]

# Treino do modelo
X_train <- as.matrix(train_data[, -which(names(train_data) == "Sleep.Disorder")])
y_train <- train_data$Sleep.Disorder
model <- perceptron.fit(X_train, y_train)

# Fazendo as predições com os dados de teste
X_test <- as.matrix(test_data[, -which(names(test_data) == "Sleep.Disorder")])
predictions <- perceptron.predict(model, X_test)

# Avaliando acurácia
accuracy <- sum(predictions == test_data$Sleep.Disorder) / length(predictions)
cat("Acurácia:", accuracy, "\n")
```

```
## Acurácia: 0.4434783
```

```r
# Gerando a malha de pontos para o plot
df_plot <- df %>% na.omit()

plot_grid <- expand_grid(
  Sleep.Duration = seq_range(df_plot$Sleep.Duration, 50),
  Quality.of.Sleep = seq_range(df_plot$Quality.of.Sleep, 50)
)

# Adicionando a predição das classes à malha de pontos
X_plot <- as.matrix(plot_grid)
```

```r
predictions_plot <- perceptron.predict(model, X_plot)

plot_grid_pred <- plot_grid %>%
  mutate(pred = predictions_plot)

# Fazendo o gráfico da fronteira de decisão
ggplot(plot_grid_pred, aes(Sleep.Duration, Quality.of.Sleep)) +
  geom_contour(aes(z = as.integer(pred)),
               alpha = 0.5, show.legend = FALSE, breaks = c(1L, 2L),
               size = 1.2, color = "red") +
  geom_point(data = df_plot, aes(color = Sleep.Disorder), size = 2) +
  labs(title = "Fronteira de Decisão para o Perceptron")
```



A acurácia é uma medida de desempenho que indica a proporção de previsões corretas em relação ao total de previsões feitas pelo modelo. No caso do perceptron com 44% de acurácia, isso sugere que o modelo não está conseguindo capturar os padrões e relações nos dados de maneira eficaz, levando a previsões imprecisas.

Com isso, podemos concluir que o Perceptron não é o mais indicado para nosso problema, uma vez que avaliamos já nas análises discriminantes que a fronteira de decisão do problema não é próxima de ser descrita linearmente. O Perceptron é um modelo linear simples e pode não ser adequado para problemas complexos com relacionamentos não lineares entre as variáveis preditoras e a variável de classe. Nesses casos, modelos mais complexos e flexíveis podem ser necessários para obter uma melhor acurácia.

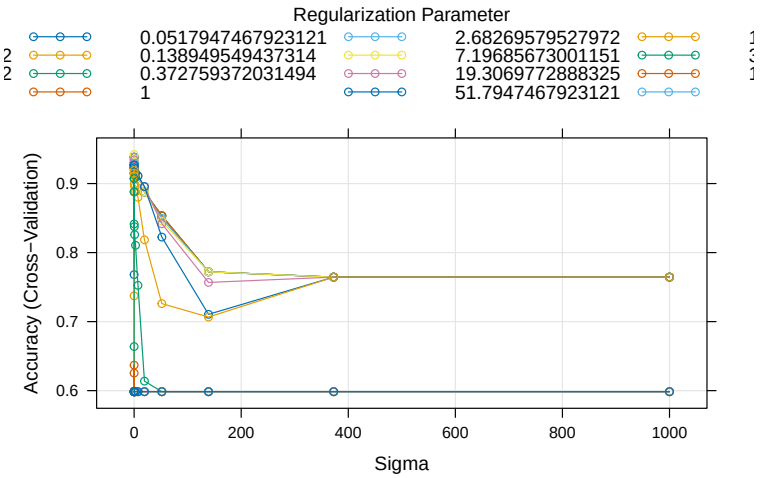
SVM

Extendendo o perceptron, podemos tentar treinar um SVM com fronteira de decisão mais complexa, não linear. Adicionalmente, vamos introduzir o uso de um novo pacote que facilita o processo de treinamento e validação dos modelos: o pacote Caret.

```r
library(caret)
library(kernlab)

modelo_svm <- train(Sleep.Disorder ~ ., data = train.prep, method = "lssvmRadial", trControl = trainCon
```

O pacote Caret já cuida do processo de treinamento e seleção de hiperparâmetros a partir de validação cruzada. Podemos ver a acurácia do modelo com diferentes valores desses hiperparâmetros. No gráfico abaixo, notamos a perda de flexibilidade do modelo conforme o valor de sigma aumenta, com diferentes valores de tau.

```
plot(modelo_svm)
```



Dentre as opções mostradas acima, o melhor modelo é:

```
print(modelo_svm$finalModel)
```

```
## Least Squares Support Vector Machine object of class "lssvm"
##
## problem type : classification
##  parameter : tau = 0.138949549437314
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.001
##
## Number of data points used for training : 35
## Training error : 0.114286
```

Por fim, podemos avaliar a performance de generalização desse SVM utilizando uma matriz de confusão e analisando sua acurácia. Apesar da boa performance do modelo no treino, vemos que sua matriz de confusão (e acurácia) não é tão boa, possivelmente sinalizando que mesmo com a validação, o modelo sobreajustou os dados, sendo necessário buscar novas combinações de hiperparâmetros, ou mais dados, já que o conjunto é bem pequeno.

```
previsoes_svm <- predict(modelo_svm, test.prep)
confusao_svm <- confusionMatrix(previsoes_svm, test.prep$Sleep.Disorder[0:108])
print(confusao_svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction None Positive
##   None       44       22
##   Positive   20       22
##
##                Accuracy : 0.6111
##                  95% CI : (0.5125, 0.7034)
##     No Information Rate : 0.5926
##     P-Value [Acc > NIR] : 0.3867
##
##                   Kappa : 0.1888
```

```
##
##  Mcnemar's Test P-Value : 0.8774
##
##             Sensitivity : 0.6875
##             Specificity : 0.5000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5238
##              Prevalence : 0.5926
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.6111
##       Balanced Accuracy : 0.5938
##
##        'Positive' Class : None
##
```

Árvore de decisão

Como último modelo, foi escolhida a árvore de decisão, por conta da sua interpretabilidade nos resultados. Para treinar a árvore de decisão não precisamos de variáveis dummy, por esse motivo é realizado uma preparação de dados distinta dos demais modelos já apresentados. A árvore é treinada com os dados de treino e utilizando essa biblioteca já é realizada a validação cruzada, escolhendo o melhor resultado para cada fold.

```
treino$Person.ID = NULL
teste$Person.ID = NULL
# Realizar treinamento e validação cruzada
modelo_arvore <- train(Sleep.Disorder ~ ., data = train.prep, method = "rpart", trControl = trainControl
print(modelo_arvore$results)
```

```
##       cp  Accuracy     Kappa AccuracySD    KappaSD
## 1   0.01 0.9111614 0.8144853 0.02210727 0.04592171
## 2   0.04 0.8957014 0.7835552 0.02607215 0.05448078
## 3   0.07 0.8957014 0.7835552 0.02607215 0.05448078
## 4   0.10 0.8957014 0.7835552 0.02607215 0.05448078
## 5   0.13 0.8957014 0.7835552 0.02607215 0.05448078
## 6   0.16 0.8957014 0.7835552 0.02607215 0.05448078
## 7   0.19 0.8957014 0.7835552 0.02607215 0.05448078
## 8   0.22 0.8957014 0.7835552 0.02607215 0.05448078
## 9   0.25 0.8957014 0.7835552 0.02607215 0.05448078
## 10 0.28 0.8957014 0.7835552 0.02607215 0.05448078
## 11 0.31 0.8957014 0.7835552 0.02607215 0.05448078
## 12 0.34 0.8957014 0.7835552 0.02607215 0.05448078
## 13 0.37 0.8957014 0.7835552 0.02607215 0.05448078
## 14 0.40 0.8957014 0.7835552 0.02607215 0.05448078
## 15 0.43 0.8957014 0.7835552 0.02607215 0.05448078
## 16 0.46 0.8957014 0.7835552 0.02607215 0.05448078
## 17 0.49 0.8957014 0.7835552 0.02607215 0.05448078
## 18 0.52 0.8957014 0.7835552 0.02607215 0.05448078
## 19 0.55 0.8957014 0.7835552 0.02607215 0.05448078
## 20 0.58 0.8957014 0.7835552 0.02607215 0.05448078
## 21 0.61 0.8957014 0.7835552 0.02607215 0.05448078
## 22 0.64 0.8957014 0.7835552 0.02607215 0.05448078
## 23 0.67 0.8957014 0.7835552 0.02607215 0.05448078
## 24 0.70 0.8957014 0.7835552 0.02607215 0.05448078
## 25 0.73 0.7649321 0.4469736 0.15425194 0.40824216
## 26 0.76 0.5984917 0.0000000 0.00522761 0.00000000
```

```
## 27 0.79 0.5984917 0.0000000 0.00522761 0.00000000
## 28 0.82 0.5984917 0.0000000 0.00522761 0.00000000
## 29 0.85 0.5984917 0.0000000 0.00522761 0.00000000
## 30 0.88 0.5984917 0.0000000 0.00522761 0.00000000
```

Na tabela acima mostramos os primeiros resultados para cada CP testado.

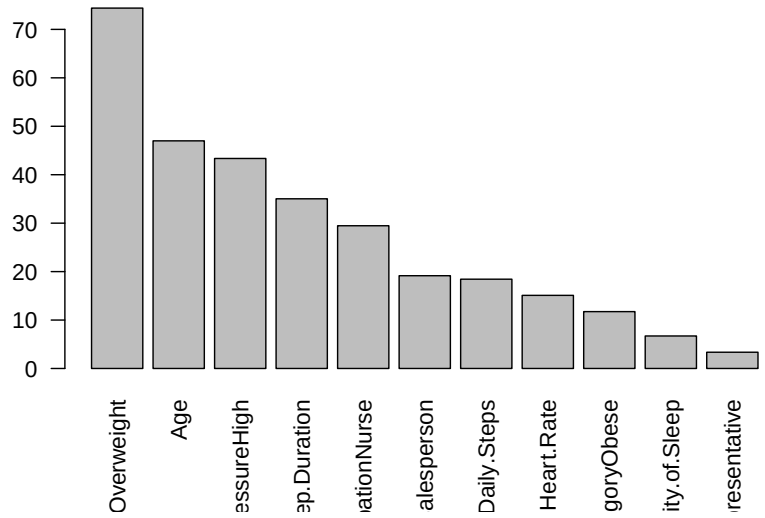
Importância das variáveis

Podemos também obter a informação do modelo treinado da importância das variáveis, sendo as principais: BMI Acima do Peso, Idade e Pressão Sanguínea alta.

```
print(modelo_arvore$finalModel$variable.importance)
```

```
##         BMI.CategoryOverweight                            Age
##                      74.395157                      46.991284
##             Blood.PressureHigh                 Sleep.Duration
##                      43.353616                      35.027395
##                OccupationNurse          OccupationSalesperson
##                      29.463429                      19.151229
##                    Daily.Steps                     Heart.Rate
##                      18.431207                      15.093942
##              BMI.CategoryObese                Quality.of.Sleep
##                      11.739732                       6.708419
## OccupationSales Representative
##                       3.354209
```

```
barplot(modelo_arvore$finalModel$variable.importance, las = 2)
```



CP de cada fold realizado

```
print(head(modelo_arvore$resampledCM))
```

```
##     cp cell1 cell2 cell3 cell4 Resample
## 1 0.01    30     1     3    18    Fold1
## 2 0.02    30     1     3    18    Fold1
## 3 0.03    30     1     3    18    Fold1
## 4 0.04    30     1     2    19    Fold1
## 5 0.05    30     1     2    19    Fold1
## 6 0.06    30     1     2    19    Fold1
```

Melhor acurária escolhida para cada fold

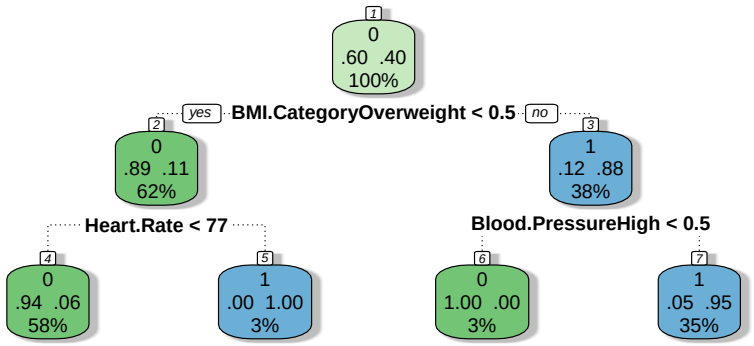
```
print(head(modelo_arvore$resample))
```

```
##    Accuracy     Kappa Resample
## 1 0.9230769 0.8377535    Fold1
## 2 0.9245283 0.8422619    Fold3
## 3 0.8867925 0.7757405    Fold4
## 4 0.9433962 0.8787185    Fold5
## 5 0.9444444 0.8841202    Fold2
```

Resultado da árvore

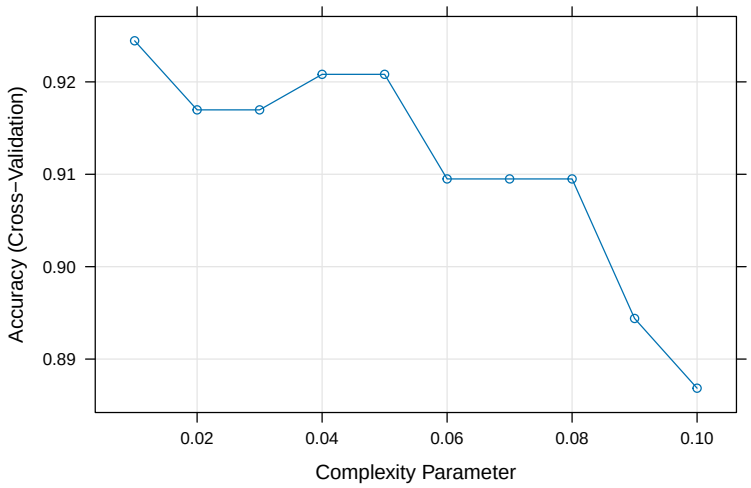
Como resultado obteve-se uma árvore de altura 2 e 4 folhas. Onde as folhas azuis indicam a presença de um distúrbio de sono. Para uma entrada de dados onde a pessoa tenha sobrepeso e pressão sanguínea, a árvore retorna 95% de probabilidade de que a variável resposta indique a presença de um distúrbio de sono.

```
library(rattle)
fancyRpartPlot(modelo_arvore$finalModel)
```



Curva erro Cross Validation

```
plot(modelo_arvore)
```



Teste da árvore

Observando-se a matriz de confusão é possível notar que a árvore obteve bons resultados, com poucos erros de predição, tanto falso positivo quanto falso negativo. Além de apresentar uma ótima acurácia de 91,74%

```r
previsoes_arvore <- predict(modelo_arvore, test.prep)
confusao_arvore <- confusionMatrix(data=previsoes_arvore, reference = test.prep$Sleep.Disorder[0:108])
print(confusao_arvore)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction None Positive
##   None       44       23
##   Positive   20       21
##
##                Accuracy : 0.6019
##                  95% CI : (0.5032, 0.6948)
##     No Information Rate : 0.5926
##     P-Value [Acc > NIR] : 0.4634
##
##                   Kappa : 0.1665
##
##  Mcnemar's Test P-Value : 0.7604
##
##             Sensitivity : 0.6875
##             Specificity : 0.4773
##          Pos Pred Value : 0.6567
##          Neg Pred Value : 0.5122
##              Prevalence : 0.5926
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.6204
##       Balanced Accuracy : 0.5824
##
##        'Positive' Class : None
##
```

A árvore de decisão modelada foi capaz de classificar corretamente as instâncias do conjunto de dados em suas respectivas categorias cerca de 91,74% das vezes. Isso significa que a complexidade do problema pôde ser transcrito através da robustez da árvore, uma vez que árvores deste tipo são eficazes em lidar com dados complexos, incluindo características categóricas e numéricas, sem muita necessidade de pré-processamento intenso. Além disso, neste meio de dados vindos da medicina, a árvore é particularmente eficaz devido à sua capacidade de explicar as decisões tomadas.

As variáveis apresentadas mais importantes são: estar acima do peso, idade, pressão sanguínea e duração do sono.

Random Forest

Após o teste em uma única árvore, foi testada uma Random Forest, com o intuito de testar se diversas árvores melhorariam a acurácia do resultado anterior. A random forest foi treinada e foi realizada a validação cruzada.

```r
library(caret)
library(randomForest)
formula <- Sleep.Disorder ~ .
grid <- expand.grid(
  mtry = c(2,3,4,5,6,7,8,9,10,11,12,13,14,15),
  ntree = c(100, 200, 300, 400, 500)
)
```

```
modelo_forest <- train(formula, data = train.prep, method = "rf", trControl = trainControl(method = "cv
print(modelo_forest$results)
```

```
##   mtry  Accuracy     Kappa AccuracySD    KappaSD
## 1    2 0.9266968 0.8470283 0.04969270 0.10518537
## 2   21 0.9343891 0.8647257 0.03487577 0.07100178
## 3   41 0.9190045 0.8320367 0.02481736 0.04992372
```

Importância das variáveis

Como no modelo de uma única árvore as variáveis com maior importância se mantiveram as mesmas: BMI Acima do Peso, Pressão Sanguínea alta e Idade.

```
print(modelo_forest$finalModel$importance)
```

```
##                                 MeanDecreaseGini
## GenderMale                            2.33999865
## Age                                  11.95556305
## OccupationDoctor                      2.77424266
## OccupationEngineer                    1.97958341
## OccupationLawyer                      1.44038284
## OccupationManager                     0.11477089
## OccupationNurse                       4.95946672
## OccupationSales Representative        0.17659797
## OccupationSalesperson                 2.47682586
## OccupationScientist                   0.13671391
## OccupationSoftware Engineer           0.01952514
## OccupationTeacher                     1.32092499
## Sleep.Duration                        7.96856133
## Quality.of.Sleep                      4.58227247
## Physical.Activity.Level               3.66397138
## Stress.Level                          4.50391342
## BMI.CategoryNormal Weight             0.46159706
## BMI.CategoryObese                     0.98036395
## BMI.CategoryOverweight               15.99155492
## Blood.PressureHigh                   14.61405392
## Heart.Rate                            5.28131015
## Daily.Steps                           5.90845729
```

Matriz de confusão

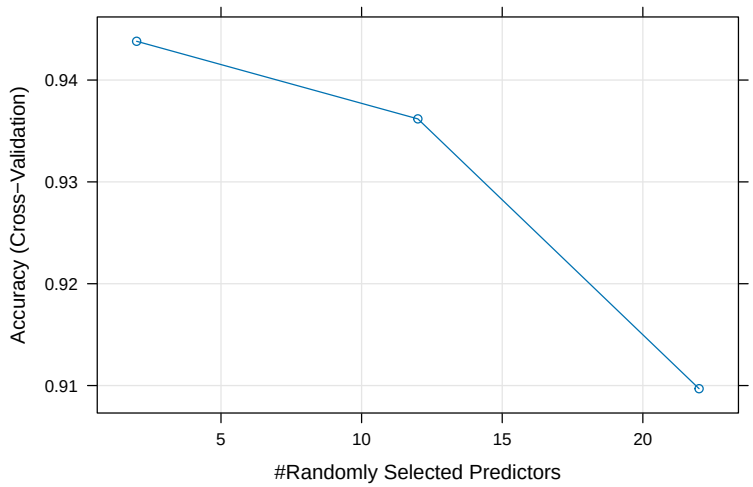
A matriz de confusão mostra que novamente obteve-se poucos erros em cada classe para os dados de treinamento.

```
print(modelo_forest$finalModel$confusion)
```

```
##     0  1 class.error
## 0 152  6  0.03797468
## 1   9 98  0.08411215
```

Curva erro cross validation

```
plot(modelo_forest)
```



Teste da Árvore

```
previsoes_forest <- predict(modelo_forest, test.prep)
confusao_forest <- confusionMatrix(data=previsoes_forest, reference = test.prep$Sleep.Disorder[0:108])
print(confusao_forest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction None Positive
##   None       44       22
##   Positive   20       22
##
##                Accuracy : 0.6111
##                  95% CI : (0.5125, 0.7034)
##     No Information Rate : 0.5926
##     P-Value [Acc > NIR] : 0.3867
##
##                   Kappa : 0.1888
##
##  Mcnemar's Test P-Value : 0.8774
##
##             Sensitivity : 0.6875
##             Specificity : 0.5000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5238
##              Prevalence : 0.5926
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.6111
##       Balanced Accuracy : 0.5938
##
##        'Positive' Class : None
##
```

O algoritmo do Random Forest teve um bom desempenho, com mais de 94% de acurácia. Isso sugere que os dados podem possuir características que permitem que o algoritmo Random Forest capture padrões complexos e tome decisões precisas, indicando também a não linearidade do modelo como já previsto anteriormente.

Conclusões

Após uma análise detalhada dos modelos aplicados aos dados, fica evidente que a regressão logística apresentou o melhor desempenho em termos de acurácia, sensibilidade, especificidade e precisão. Esses resultados sugerem que o modelo foi capaz de realizar previsões precisas e discriminar corretamente entre as classes, especialmente no que diz respeito à detecção de casos positivos. No entanto, é importante considerar o contexto em que esses resultados foram obtidos.

Embora a regressão logística tenha se destacado, é essencial reconhecer que o conjunto de amostras utilizado não é consideravelmente grande. Isso levanta a possibilidade de que os modelos possam ter sido influenciados pela falta de dados, resultando em um potencial viés devido à escassez de exemplos. Além disso, a complexidade intrínseca dos dados, com padrões e relações não lineares, pode ter limitado a capacidade da regressão logística de capturar completamente as nuances do problema.

Diante dessas limitações, o modelo Random Forest surge como uma alternativa atraente. Seus bons resultados, com uma alta acurácia de mais de 94%, indicam que é capaz de lidar com a complexidade dos dados e capturar padrões complexos, incluindo relações não lineares. Essa capacidade do Random Forest de criar um conjunto diversificado de árvores de decisão e combiná-las para chegar a uma decisão final o torna uma escolha promissora para esse cenário.

Portanto, apesar dos resultados favoráveis da regressão logística, é razoável considerar que, dadas as limitações do conjunto de dados e a complexidade das relações subjacentes, o Random Forest poderia ser uma opção mais sólida para abordar o problema em questão, oferecendo uma melhor adaptação à natureza não linear dos dados e uma maior resiliência diante do tamanho limitado do conjunto de amostras.