```
library(dplyr)
```

```
     Attaching package: ‘dplyr’


     The following objects are masked from ‘package:stats’:

         filter, lag


     The following objects are masked from ‘package:base’:

         intersect, setdiff, setequal, union
```

```
library(caret)
library(rpart)

# Defina o diretório onde estão os seus arquivos CSV

# Carregue os dados de treinamento
training_data <- read.csv("pml-training.csv")

# Carregue os dados de teste
testing_data <- read.csv("pml-testing.csv")

# Verifique a estrutura dos dados
str(training_data)
str(testing_data)
str(training_data$classe)
str(testing_data$classe)


# Remova linhas com valores ausentes nos dados de treinamento
training_data <- na.omit(training_data)

# Remova linhas com valores ausentes nos dados de teste
testing_data <- na.omit(testing_data)

# Verifique a estrutura dos dados após a remoção de NAs
str(training_data)
str(testing_data)
```

```
training_data <- training_data %>% mutate(classe = as.numeric(factor(classe, levels = c("A", "B", "C", "D", "E"))))
print(training_data$classe)
```

```
       [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
     [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
     [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
     [186] 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
     [223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4
     [260] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
     [297] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5
     [334] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
     [371] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

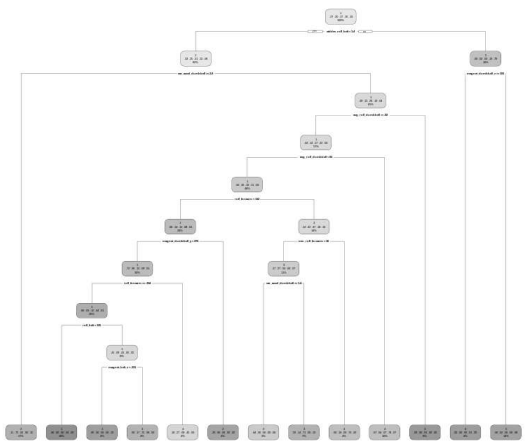
```
training_data1 <- training_data %>% select_if(is.numeric)

testing_data1 <- testing_data %>% select_if(is.numeric)
# Exclua as quatro primeiras linhas dos dados de treinamento
training_data1 <- training_data1[, -c(1:4)]

# Exclua as quatro primeiras colunas (features) dos dados de teste
testing_data1 <- testing_data1[, -c(1:4)]
str(training_data1)
str(testing_data1)
```

```
      $ yaw_belt            : num
      $ total_accel_belt    : int
      $ gyros_belt_x        : num
      $ gyros_belt_y        : num
      $ gyros_belt_z        : num
      $ accel_belt_x        : int
      $ accel_belt_y        : int
      $ accel_belt_z        : int
      $ magnet_belt_x       : int
      $ magnet_belt_y       : int
      $ magnet_belt_z       : int
      $ roll_arm            : num
      $ pitch_arm           : num
      $ yaw_arm             : num
      $ total_accel_arm     : int
      $ gyros_arm_x         : num
      $ gyros_arm_y         : num
      $ gyros_arm_z         : num
      $ accel_arm_x         : int
      $ accel_arm_y         : int
      $ accel_arm_z         : int
      $ magnet_arm_x        : int
      $ magnet_arm_y        : int
      $ magnet_arm_z        : int
      $ roll_dumbbell       : num
      $ pitch_dumbbell      : num
      $ yaw_dumbbell        : num
      $ total_accel_dumbbell: int
      $ gyros_dumbbell_x    : num
      $ gyros_dumbbell_y    : num
      $ gyros_dumbbell_z    : num
      $ accel_dumbbell_x    : int
      $ accel_dumbbell_y    : int
      $ accel_dumbbell_z    : int
      $ magnet_dumbbell_x   : int
      $ magnet_dumbbell_y   : int
      $ magnet_dumbbell_z   : int
      $ roll_forearm        : num
      $ pitch_forearm       : num
      $ yaw_forearm         : num
      $ total_accel_forearm : int
      $ gyros_forearm_x     : num
      $ gyros_forearm_y     : num
      $ gyros_forearm_z     : num
      $ accel_forearm_x     : int
      $ accel_forearm_y     : int
      $ accel_forearm_z     : int
      $ magnet_forearm_x    : int
      $ magnet_forearm_y    : int
      $ magnet_forearm_z    : int
      $ problem_id          : int
```

```
proporcao_treinamento <- 0.7

# Crie um índice aleatório para dividir os dados
indice <- sample(1:nrow(training_data1), floor(proporcao_treinamento * nrow(training_data1)))

# Crie conjuntos de treinamento e validação com base no índice
conjunto_treinamento <- training_data1[indice, ]
conjunto_validacao <- training_data1[-indice, ]

# Crie o modelo de árvore de decisão usando rpart
modelo_arvore <- rpart(classe ~ ., data = conjunto_treinamento, method = "class")

# Faça predições no conjunto de validação
predicoes <- predict(modelo_arvore, newdata = conjunto_validacao, type = "class")

# Avalie o desempenho do modelo (por exemplo, matriz de confusão)
tabela_confusao <- table(predicoes, conjunto_validacao$classe)
print(tabela_confusao)
```

```
     predicoes  1  2  3  4  5
            1 17  1  1  3  0
            2  9 15  4  2  2
            3  2  1 16  2  1
            4  3  4  0 16  1
            5  1  0  0  1 20
```

```
library(rpart.plot)
rpart.plot(modelo_arvore)
```



```
# Calcular a acurácia
acuracia <- sum(diag(tabela_confusao)) / sum(tabela_confusao)
cat("Acurácia: ", acuracia, "\n")

# Calcular a sensibilidade (recall)
sensibilidade <- tabela_confusao[2, 2] / sum(tabela_confusao[2, ])
cat("Sensibilidade (Recall): ", sensibilidade, "\n")

# Calcular a especificidade
especificidade <- tabela_confusao[1, 1] / sum(tabela_confusao[1, ])
cat("Especificidade: ", especificidade, "\n")

# Calcular a precisão (valor preditivo positivo)
precisao <- tabela_confusao[2, 2] / sum(tabela_confusao[, 2])
cat("Precisão: ", precisao, "\n")

# Calcular o F1-score
f1_score <- 2 * (precisao * sensibilidade) / (precisao + sensibilidade)
cat("F1-Score: ", f1_score, "\n")
```

```
Acurácia:  0.6885246
Sensibilidade (Recall):  0.46875
Especificidade:  0.7727273
Precisão:  0.7142857
F1-Score:  0.5660377
```

```
rmarkdown::render("Untitled13.ipynb")
```

```
    studio.com/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML' --include-in-header /tmp/Rtmpa84XLI/rmarkdown-str8fcb291bd439.html
```