Relatório - Classificação de Atividades Físicas

Rodrigo Tornisiello

08/07/2021

Definição do problema

Deseja-se classsificar qual tipo de atividade cada individuo está executando por meio de dados coletados via sensores de *smartphone*.

Definição da base de dados

Utilizou-se a base de dados: *Human Activity Recognition Using Smartphones Data Set*, disponibilizada no repositório de dados para Machine Learning UCI.

Essa base consiste em medições de 561 variáveis de 30 indivíduos ao longo do tempo. As atividades executadas por esses indivíduos foram classificadas em 6 categorias: ANDANDO, SUBINDO ESCADAS, DESCENDO ESCADAS, SENTADO, EM PÉ, DEITADO. As medições foram feitas por celulares do modelo Samsung Galaxy S II.

O próprio dataset já apresenta uma divisão de teste/treino que será respeitada. Mais detalhes sobre as bases podem ser encontrados no link.

Limpeza de dados

É necessário um esforço de organização para esse dataset. Os dados vêm separados em vários arquivos diferentes:

- 'features_info.txt': descreve as variáveis
- 'features.txt': nomes das variáveis
- 'activity_labels.txt': nomes das atividade
- 'train/X_train.txt': dataset de treino
- 'train/y_train.txt': classificação do dataset de treino
- 'test/X_test.txt': dataset de teste
- 'test/y_test.txt': classificação do dataset de teste

Optou-se por trabalhar apenas com as variáveis referentes a média e desvio padrão por motivos de esforço computacional. Caso as previsões se mostrem ineficazes as variáveis escolhidas serão reconsideradas.

Ao final são mostradas as dimensões resultantes após a limpeza e organização dos dados.

```
library(data.table)

# carrega os nomes das variáveis
all_features <- read.table("./UCI HAR Dataset/features.txt")[,2]
```

```r
# cria vetor booleano indicando as variáveis de média ou desvio padrão
my_features <- grepl("mean|std", all_features)

# carrega os nomes das atividades
activity <- read.table("./UCI HAR Dataset/activity_labels.txt")[,2]

# carrega os dados referentes a entrada, saida e indivíduo para teste
X_test <- read.table("./UCI HAR Dataset/test/X_test.txt")
y_test <- read.table("./UCI HAR Dataset/test/y_test.txt")
subject_test <- read.table("./UCI HAR Dataset/test/subject_test.txt")

# carrega os dados referentes a entrada, saida e indivíduo para treino
X_train <- read.table("./UCI HAR Dataset/train/X_train.txt")
y_train <- read.table("./UCI HAR Dataset/train/y_train.txt")
subject_train <- read.table("./UCI HAR Dataset/train/subject_train.txt")

# nomeia as colunas de acordo com os nomes das variáveis
names(X_test) = all_features
names(X_train) = all_features

# filtra apenas as colunas desejadas por meio do vetor booleno
X_test = X_test[,my_features]
X_train = X_train[,my_features]

# nomeia as ativiadades de acordo com os códigos informados
y_test[,2] = activity[y_test[,1]]
y_train[,2] = activity[y_train[,1]]

# nomeia as colunas dos datasets de dados de saída e de indivíduos
names(y_test) = c("Activity_ID", "Activity_name")
names(y_train) = c("Activity_ID", "Activity_name")
names(subject_test) = "Subject"
names(subject_train) = "Subject"

# junta as colunas de saída, indivíduo e entrada em um mesmo conjunto
# o processo é repetido para teste e para treino
test_df <- cbind(y_test,as.data.table(subject_test), X_test)
train_df <- cbind(y_train,as.data.table(subject_train), X_train)

# salva os dados organizados em arquivos .txt
write.table(test_df, file = "./test.txt")
write.table(train_df, file = "./train.txt")

# dimensões da amostra de treino e teste
dim(train_df)
```

```
## [1] 7352   82
```

```r
dim(test_df)
```

```
## [1] 2947   82
```

```
#remove as variáveis
rm(list = ls())
```

Análise exploratória

Durante a análise exploratória nota-se que:

- O ID da atividade e o indivíduo foram interpretados como inteiros, quando na verdade se deseja que sejam variáveis categóricas, por isso essas variáveis são transformadas em `factor`.
- A variável nome da atividade será descartada na modelagem, por isso não há necessidade de transformação.
- Não existem NAs no dataset de treino
- As atividades não estão distribuídas igualmente, sendo assim a métrica Kappa já se mostra mais adequada do que a Acurácia.
- A quantidade de instâncias por indivíduos também não é uniforme.

```
library(ggplot2)
library(RColorBrewer)
library(ellipse)

dataset <- read.table("./train.txt")

# resumo sobre as variáveis
str(dataset)
```

```
## 'data.frame':    7352 obs. of  82 variables:
##  $ Activity_ID                   : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Activity_name                 : chr  "STANDING" "STANDING" "STANDING" "STANDING" ...
##  $ Subject                       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ tBodyAcc.mean...X             : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ tBodyAcc.mean...Y             : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ tBodyAcc.mean...Z             : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ tBodyAcc.std...X              : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ tBodyAcc.std...Y              : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ tBodyAcc.std...Z              : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ tGravityAcc.mean...X          : num  0.963 0.967 0.967 0.968 0.968 ...
##  $ tGravityAcc.mean...Y          : num  -0.141 -0.142 -0.142 -0.144 -0.149 ...
##  $ tGravityAcc.mean...Z          : num  0.1154 0.1094 0.1019 0.0999 0.0945 ...
##  $ tGravityAcc.std...X           : num  -0.985 -0.997 -1 -0.997 -0.998 ...
##  $ tGravityAcc.std...Y           : num  -0.982 -0.989 -0.993 -0.981 -0.988 ...
##  $ tGravityAcc.std...Z           : num  -0.878 -0.932 -0.993 -0.978 -0.979 ...
##  $ tBodyAccJerk.mean...X         : num  0.078 0.074 0.0736 0.0773 0.0734 ...
##  $ tBodyAccJerk.mean...Y         : num  0.005 0.00577 0.0031 0.02006 0.01912 ...
##  $ tBodyAccJerk.mean...Z         : num  -0.06783 0.02938 -0.00905 -0.00986 0.01678 ...
##  $ tBodyAccJerk.std...X          : num  -0.994 -0.996 -0.991 -0.993 -0.996 ...
##  $ tBodyAccJerk.std...Y          : num  -0.988 -0.981 -0.981 -0.988 -0.988 ...
##  $ tBodyAccJerk.std...Z          : num  -0.994 -0.992 -0.99 -0.993 -0.992 ...
##  $ tBodyGyro.mean...X            : num  -0.0061 -0.0161 -0.0317 -0.0434 -0.034 ...
##  $ tBodyGyro.mean...Y            : num  -0.0314 -0.0839 -0.1023 -0.0914 -0.0747 ...
```

```
##  $ tBodyGyro.mean...Z            : num  0.1077 0.1006 0.0961 0.0855 0.0774 ...
##  $ tBodyGyro.std...X             : num  -0.985 -0.983 -0.976 -0.991 -0.985 ...
##  $ tBodyGyro.std...Y             : num  -0.977 -0.989 -0.994 -0.992 -0.992 ...
##  $ tBodyGyro.std...Z             : num  -0.992 -0.989 -0.986 -0.988 -0.987 ...
##  $ tBodyGyroJerk.mean...X        : num  -0.0992 -0.1105 -0.1085 -0.0912 -0.0908 ...
##  $ tBodyGyroJerk.mean...Y        : num  -0.0555 -0.0448 -0.0424 -0.0363 -0.0376 ...
##  $ tBodyGyroJerk.mean...Z        : num  -0.062 -0.0592 -0.0558 -0.0605 -0.0583 ...
##  $ tBodyGyroJerk.std...X         : num  -0.992 -0.99 -0.988 -0.991 -0.991 ...
##  $ tBodyGyroJerk.std...Y         : num  -0.993 -0.997 -0.996 -0.997 -0.996 ...
##  $ tBodyGyroJerk.std...Z         : num  -0.992 -0.994 -0.992 -0.993 -0.995 ...
##  $ tBodyAccMag.mean..            : num  -0.959 -0.979 -0.984 -0.987 -0.993 ...
##  $ tBodyAccMag.std..             : num  -0.951 -0.976 -0.988 -0.986 -0.991 ...
##  $ tGravityAccMag.mean..         : num  -0.959 -0.979 -0.984 -0.987 -0.993 ...
##  $ tGravityAccMag.std..          : num  -0.951 -0.976 -0.988 -0.986 -0.991 ...
##  $ tBodyAccJerkMag.mean..        : num  -0.993 -0.991 -0.989 -0.993 -0.993 ...
##  $ tBodyAccJerkMag.std..         : num  -0.994 -0.992 -0.99 -0.993 -0.996 ...
##  $ tBodyGyroMag.mean..           : num  -0.969 -0.981 -0.976 -0.982 -0.985 ...
##  $ tBodyGyroMag.std..            : num  -0.964 -0.984 -0.986 -0.987 -0.989 ...
##  $ tBodyGyroJerkMag.mean..       : num  -0.994 -0.995 -0.993 -0.996 -0.996 ...
##  $ tBodyGyroJerkMag.std..        : num  -0.991 -0.996 -0.995 -0.995 -0.995 ...
##  $ fBodyAcc.mean...X             : num  -0.995 -0.997 -0.994 -0.995 -0.997 ...
##  $ fBodyAcc.mean...Y             : num  -0.983 -0.977 -0.973 -0.984 -0.982 ...
##  $ fBodyAcc.mean...Z             : num  -0.939 -0.974 -0.983 -0.991 -0.988 ...
##  $ fBodyAcc.std...X              : num  -0.995 -0.999 -0.996 -0.996 -0.999 ...
##  $ fBodyAcc.std...Y              : num  -0.983 -0.975 -0.966 -0.983 -0.98 ...
##  $ fBodyAcc.std...Z              : num  -0.906 -0.955 -0.977 -0.99 -0.992 ...
##  $ fBodyAcc.meanFreq...X         : num  0.252 0.271 0.125 0.029 0.181 ...
##  $ fBodyAcc.meanFreq...Y         : num  0.1318 0.0429 -0.0646 0.0803 0.058 ...
##  $ fBodyAcc.meanFreq...Z         : num  -0.0521 -0.0143 0.0827 0.1857 0.5598 ...
##  $ fBodyAccJerk.mean...X         : num  -0.992 -0.995 -0.991 -0.994 -0.996 ...
##  $ fBodyAccJerk.mean...Y         : num  -0.987 -0.981 -0.982 -0.989 -0.989 ...
##  $ fBodyAccJerk.mean...Z         : num  -0.99 -0.99 -0.988 -0.991 -0.991 ...
##  $ fBodyAccJerk.std...X          : num  -0.996 -0.997 -0.991 -0.991 -0.997 ...
##  $ fBodyAccJerk.std...Y          : num  -0.991 -0.982 -0.981 -0.987 -0.989 ...
##  $ fBodyAccJerk.std...Z          : num  -0.997 -0.993 -0.99 -0.994 -0.993 ...
##  $ fBodyAccJerk.meanFreq...X     : num  0.8704 0.6085 0.1154 0.0358 0.2734 ...
##  $ fBodyAccJerk.meanFreq...Y     : num  0.2107 -0.0537 -0.1934 -0.093 0.0791 ...
##  $ fBodyAccJerk.meanFreq...Z     : num  0.2637 0.0631 0.0383 0.1681 0.2924 ...
##  $ fBodyGyro.mean...X            : num  -0.987 -0.977 -0.975 -0.987 -0.982 ...
##  $ fBodyGyro.mean...Y            : num  -0.982 -0.993 -0.994 -0.994 -0.993 ...
##  $ fBodyGyro.mean...Z            : num  -0.99 -0.99 -0.987 -0.987 -0.989 ...
##  $ fBodyGyro.std...X             : num  -0.985 -0.985 -0.977 -0.993 -0.986 ...
##  $ fBodyGyro.std...Y             : num  -0.974 -0.987 -0.993 -0.992 -0.992 ...
##  $ fBodyGyro.std...Z             : num  -0.994 -0.99 -0.987 -0.989 -0.988 ...
##  $ fBodyGyro.meanFreq...X        : num  -0.2575 -0.0482 -0.2167 0.2169 -0.1533 ...
##  $ fBodyGyro.meanFreq...Y        : num  0.0979 -0.4016 -0.0173 -0.1352 -0.0884 ...
##  $ fBodyGyro.meanFreq...Z        : num  0.5472 -0.0682 -0.1107 -0.0497 -0.1622 ...
##  $ fBodyAccMag.mean..            : num  -0.952 -0.981 -0.988 -0.988 -0.994 ...
##  $ fBodyAccMag.std..             : num  -0.956 -0.976 -0.989 -0.987 -0.99 ...
##  $ fBodyAccMag.meanFreq..        : num  -0.0884 -0.0441 0.2579 0.0736 0.3943 ...
##  $ fBodyBodyAccJerkMag.mean..    : num  -0.994 -0.99 -0.989 -0.993 -0.996 ...
##  $ fBodyBodyAccJerkMag.std..     : num  -0.994 -0.992 -0.991 -0.992 -0.994 ...
##  $ fBodyBodyAccJerkMag.meanFreq.. : num  0.347 0.532 0.661 0.679 0.559 ...
##  $ fBodyBodyGyroMag.mean..       : num  -0.98 -0.988 -0.989 -0.989 -0.991 ...
```

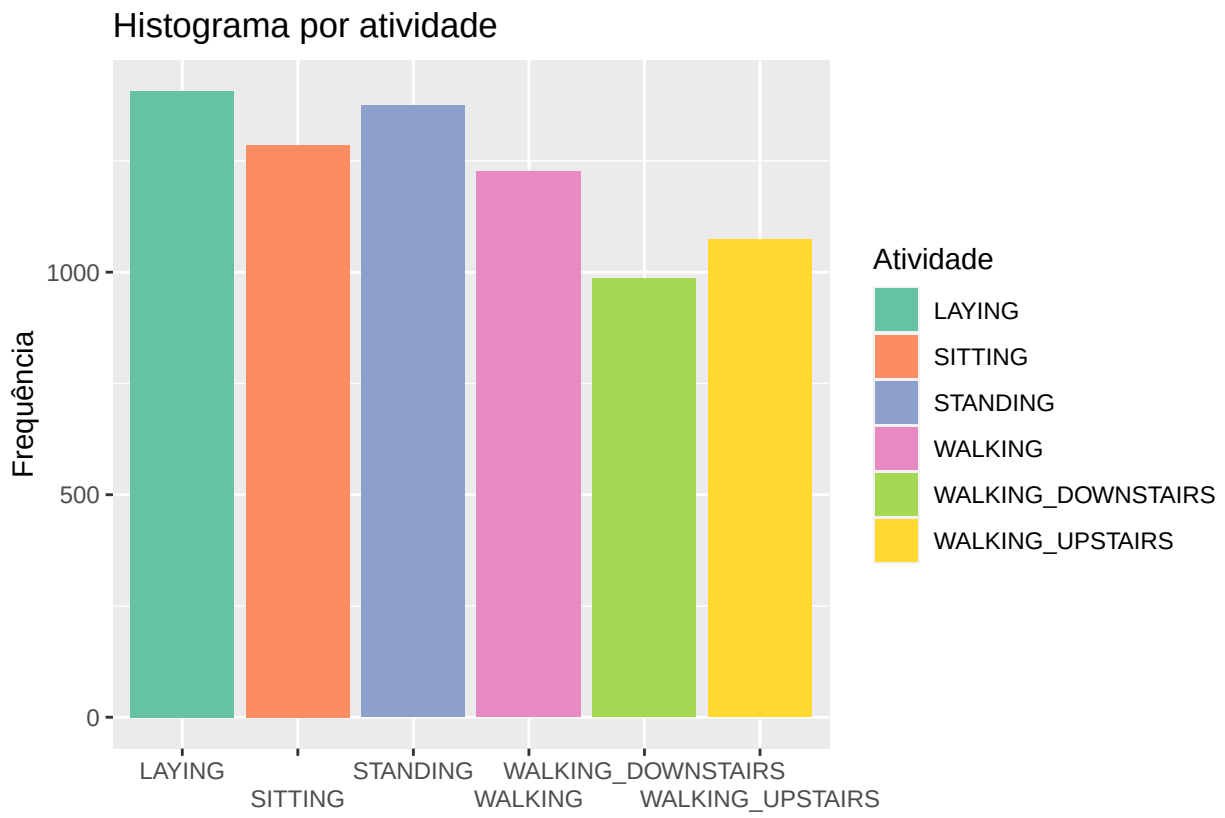
```
##  $ fBodyBodyGyroMag.std..          : num  -0.961 -0.983 -0.986 -0.988 -0.989 ...
##  $ fBodyBodyGyroMag.meanFreq..     : num  -0.129 -0.272 -0.2127 -0.0357 -0.2736 ...
##  $ fBodyBodyGyroJerkMag.mean..     : num  -0.992 -0.996 -0.995 -0.995 -0.995 ...
##  $ fBodyBodyGyroJerkMag.std..      : num  -0.991 -0.996 -0.995 -0.995 -0.995 ...
##  $ fBodyBodyGyroJerkMag.meanFreq..: num  -0.0743 0.1581 0.4145 0.4046 0.0878 ...
```

```
# transformação em factor para ID de atividade e indivíduo
dataset$Activity_ID = as.factor(dataset$Activity_ID)
dataset$Subject = as.factor(dataset$Subject)

# contagem de NAs
colSums(is.na(dataset))
```

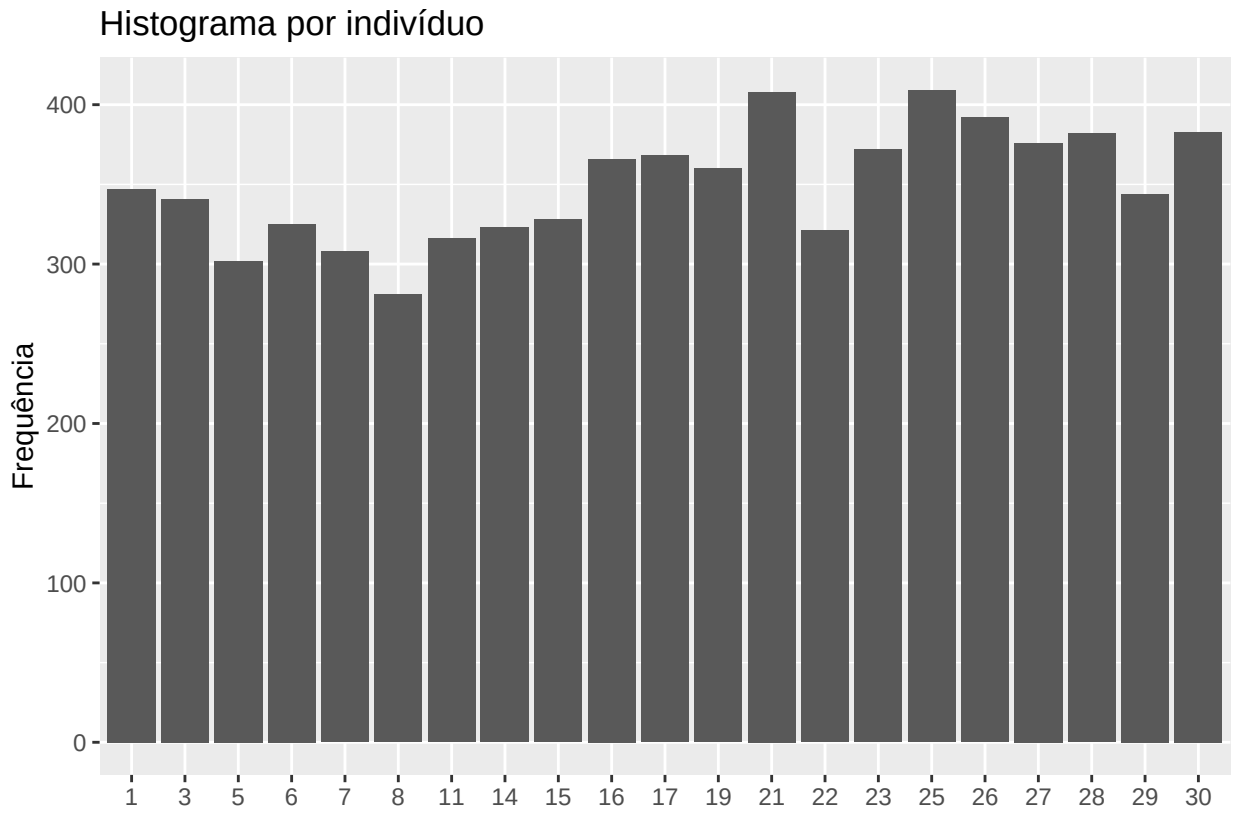
```
##                     Activity_ID                   Activity_name
##                               0                               0
##                         Subject                tBodyAcc.mean...X
##                               0                               0
##                tBodyAcc.mean...Y                tBodyAcc.mean...Z
##                               0                               0
##                 tBodyAcc.std...X                 tBodyAcc.std...Y
##                               0                               0
##                 tBodyAcc.std...Z             tGravityAcc.mean...X
##                               0                               0
##             tGravityAcc.mean...Y             tGravityAcc.mean...Z
##                               0                               0
##              tGravityAcc.std...X              tGravityAcc.std...Y
##                               0                               0
##              tGravityAcc.std...Z            tBodyAccJerk.mean...X
##                               0                               0
##            tBodyAccJerk.mean...Y            tBodyAccJerk.mean...Z
##                               0                               0
##             tBodyAccJerk.std...X             tBodyAccJerk.std...Y
##                               0                               0
##             tBodyAccJerk.std...Z                tBodyGyro.mean...X
##                               0                               0
##               tBodyGyro.mean...Y                tBodyGyro.mean...Z
##                               0                               0
##                tBodyGyro.std...X                 tBodyGyro.std...Y
##                               0                               0
##                tBodyGyro.std...Z           tBodyGyroJerk.mean...X
##                               0                               0
##           tBodyGyroJerk.mean...Y           tBodyGyroJerk.mean...Z
##                               0                               0
##            tBodyGyroJerk.std...X            tBodyGyroJerk.std...Y
##                               0                               0
##            tBodyGyroJerk.std...Z                tBodyAccMag.mean..
##                               0                               0
##                tBodyAccMag.std..             tGravityAccMag.mean..
##                               0                               0
##             tGravityAccMag.std..           tBodyAccJerkMag.mean..
##                               0                               0
##            tBodyAccJerkMag.std..              tBodyGyroMag.mean..
##                               0                               0
##                tBodyGyroMag.std..           tBodyGyroJerkMag.mean..
```

```
##                                0                                0
##          tBodyGyroJerkMag.std..                fBodyAcc.mean...X
##                                0                                0
##                fBodyAcc.mean...Y                fBodyAcc.mean...Z
##                                0                                0
##                 fBodyAcc.std...X                 fBodyAcc.std...Y
##                                0                                0
##                 fBodyAcc.std...Z            fBodyAcc.meanFreq...X
##                                0                                0
##            fBodyAcc.meanFreq...Y            fBodyAcc.meanFreq...Z
##                                0                                0
##            fBodyAccJerk.mean...X            fBodyAccJerk.mean...Y
##                                0                                0
##            fBodyAccJerk.mean...Z             fBodyAccJerk.std...X
##                                0                                0
##             fBodyAccJerk.std...Y             fBodyAccJerk.std...Z
##                                0                                0
##       fBodyAccJerk.meanFreq...X        fBodyAccJerk.meanFreq...Y
##                                0                                0
##       fBodyAccJerk.meanFreq...Z                fBodyGyro.mean...X
##                                0                                0
##                fBodyGyro.mean...Y                fBodyGyro.mean...Z
##                                0                                0
##                 fBodyGyro.std...X                 fBodyGyro.std...Y
##                                0                                0
##                 fBodyGyro.std...Z            fBodyGyro.meanFreq...X
##                                0                                0
##            fBodyGyro.meanFreq...Y            fBodyGyro.meanFreq...Z
##                                0                                0
##                fBodyAccMag.mean..                 fBodyAccMag.std..
##                                0                                0
##            fBodyAccMag.meanFreq..       fBodyBodyAccJerkMag.mean..
##                                0                                0
##       fBodyBodyAccJerkMag.std..   fBodyBodyAccJerkMag.meanFreq..
##                                0                                0
##          fBodyBodyGyroMag.mean..            fBodyBodyGyroMag.std..
##                                0                                0
##     fBodyBodyGyroMag.meanFreq..      fBodyBodyGyroJerkMag.mean..
##                                0                                0
##      fBodyBodyGyroJerkMag.std.. fBodyBodyGyroJerkMag.meanFreq..
##                                0                                0
```

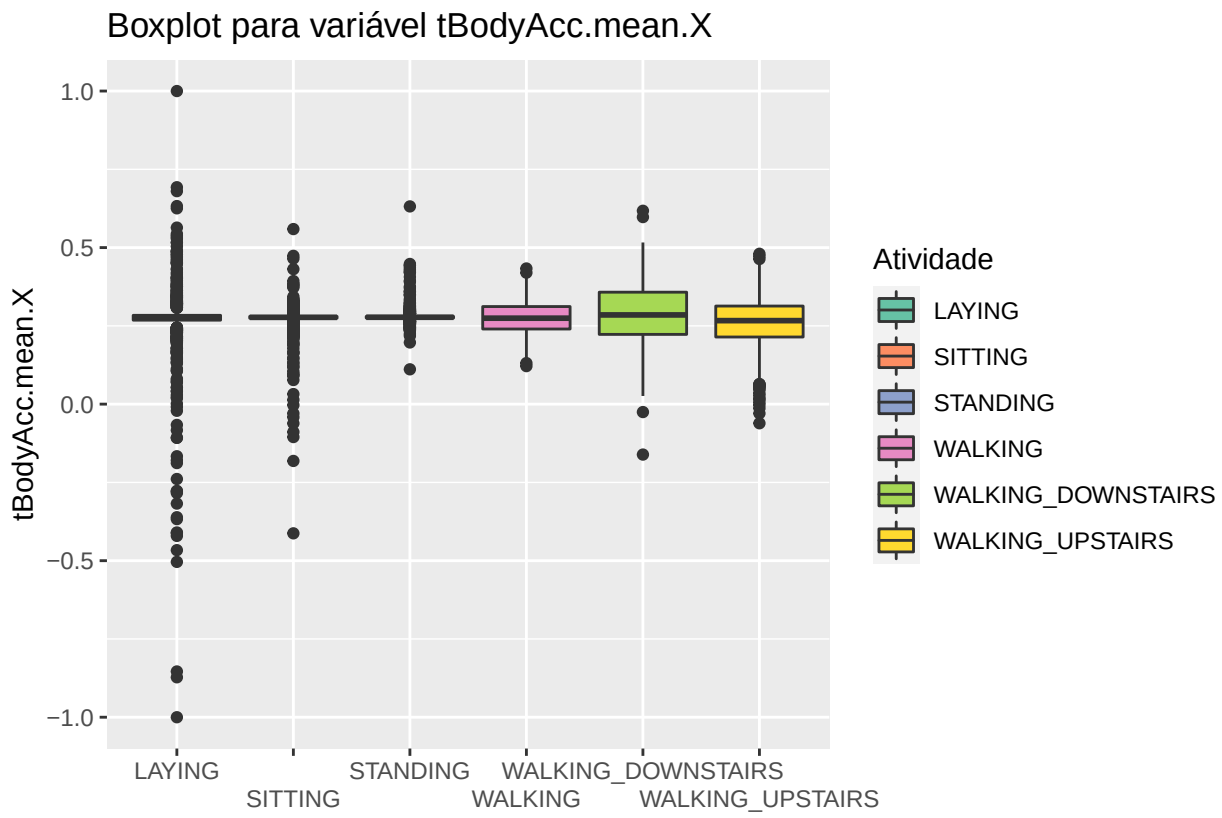
```
# plota histograma por atividade
ggplot(data.frame(dataset$Activity_name),
       aes(x=dataset$Activity_name, fill = dataset$Activity_name)) +
        geom_bar() +
        scale_fill_brewer(palette="Set2") +
        scale_x_discrete(guide = guide_axis(n.dodge=2)) +
        labs(title="Histograma por atividade") +
        xlab("") +
        labs(fill = "Atividade") +
        ylab("Frequência")
```




```
# plota histograma por indivíduo
ggplot(data.frame(dataset$Subject), aes(x=dataset$Subject)) +
        geom_bar() +
        labs(title="Histograma por indivíduo") +
        xlab("") +
        ylab("Frequência")
```




```
# Boxplot para primeira variável
ggplot(dataset, aes(x=dataset$Activity_name, y=dataset$tBodyAcc.mean...X,
                    fill=Activity_name)) +
        geom_boxplot() +
        scale_fill_brewer(palette="Set2") +
        scale_x_discrete(guide = guide_axis(n.dodge=2)) +
        labs(title="Boxplot para variável tBodyAcc.mean.X") +
        xlab("") +
        ylab("tBodyAcc.mean.X") +
        labs(fill = "Atividade")
```



Modelagem

Diversos modelos serão testados utilizando a métrica de performance Kappa. Os resultados são reproduzíveis pois a *seed* foi definida.

Primeiramente busca-se variáveis com baixa variância na tentiva de diminuir a quantidade de variáveis. No entanto, nenhuma variável foi indicada com variância insignificante.

Retira-se a coluna nome da atividade pois essa informação já consta como *factor* na coluna ID de atividade.

Não é necessário pré-processamento do tipo *scaling* pois a database selecionada já possuia essa transformação. Para avaliação de performance foi escolhido um método de *Cross Validation* repetido de 10 dobras e 3 repetições.

Os algoritmos de previsão testados foram: *Linear Discriminant Analysis*, *K-nearest Neighbors*, *Support Vector Machine*, *Random Forest*, C5.0, e *Stochastic Gradient Boosting*.

```
library(caret)

dataset <- read.table("./train.txt")
#dataset <-  read.table(text = rawToChar(obj))    para IBM Cloud

# transformação em factor para ID de atividade e indivíduo
dataset$Activity_ID = as.factor(dataset$Activity_ID)
dataset$Subject = as.factor(dataset$Subject)

# busca variáveis com variância baixa e filtra o dataset
```

```
nzv_names <- nearZeroVar(dataset, names=TRUE)
dataset <- dataset[, setdiff(names(dataset), nzv_names)]

# retira a coluna referente ao nome da atividade e indivíduo
dataset <- dataset[, -2:-3]

# configura Cross Validation repetido de 10 dobras e 3 repetições
# e métrica de desempenho
control <- trainControl(method="repeatedcv", number=10, repeats = 3)
metric <- "Accuracy"
```

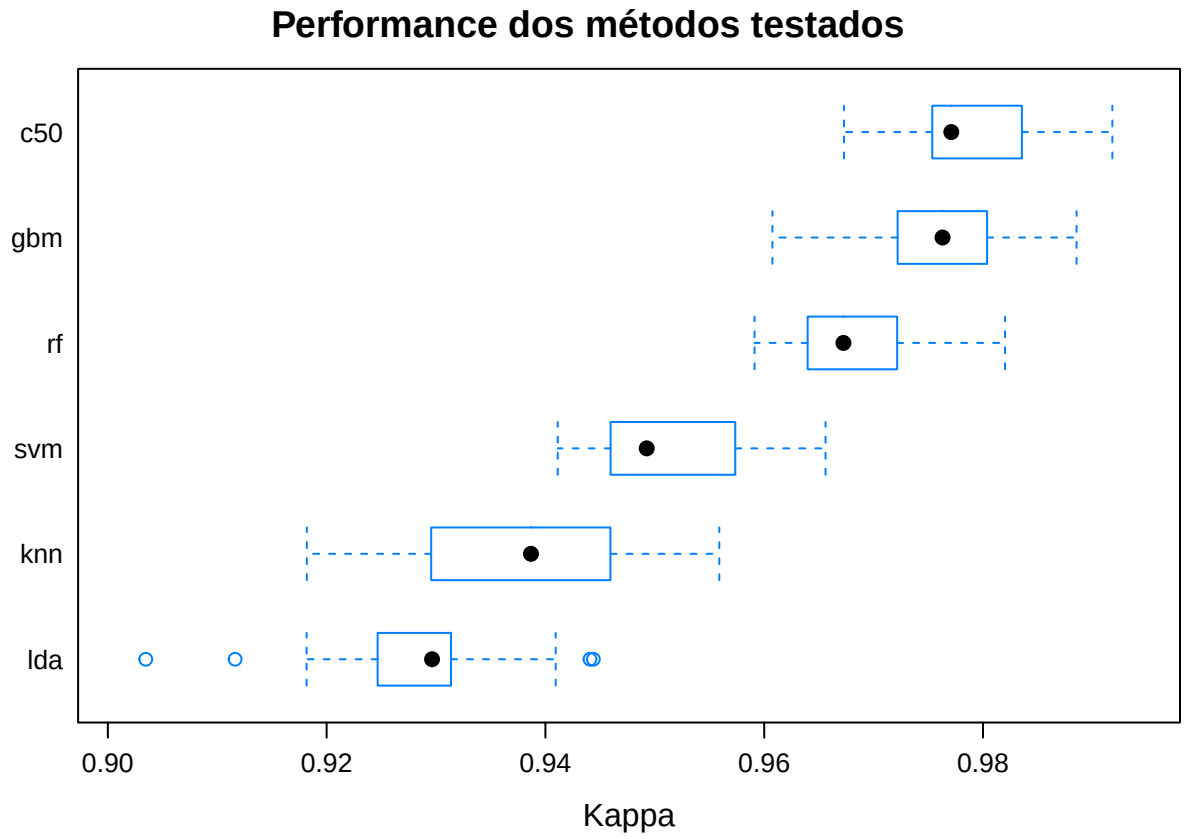
```
# treina um modelo de LDA
set.seed(1234)
fit.lda <- train( Activity_ID ~ ., data = dataset, method = "lda",
              trControl = control, metric=metric)

# treina um modelo de KNN
set.seed(1234)
fit.knn <- train( Activity_ID ~ ., data = dataset, method = "knn",
              trControl = control, metric=metric)

# treina um modelo de SVM
set.seed(1234)
fit.svm <- train( Activity_ID ~ ., data = dataset, method = "svmRadial",
              trControl = control, metric=metric)

# treina um modelo de RF
set.seed(1234)
fit.rf <- train( Activity_ID ~ ., data = dataset, method = "rf",
              trControl = control, metric=metric)

# treina um modelo de C5.0
set.seed(1234)
fit.c50 <- train( Activity_ID ~ ., data = dataset, method = "C5.0",
              trControl = control, metric=metric)

# treina um modelo de GBM
set.seed(1234)
fit.gbm <- train( Activity_ID ~ ., data = dataset, method = "gbm",
              trControl = control, metric=metric, verbose = FALSE)

# salva os modelos treinados
saveRDS(fit.lda, "./fittedLDA.rds")
saveRDS(fit.knn, "./fittedKNN.rds")
saveRDS(fit.svm, "./fittedSVM.rds")
saveRDS(fit.rf, "./fittedRF.rds")
saveRDS(fit.c50, "./fittedC50.rds")
saveRDS(fit.gbm, "./fittedGBM.rds")
```

```
library(lattice)

# carrega os modelos treinados
fit.lda <- readRDS("./Modelos/fittedLDA.rds")
```

```r
fit.knn <- readRDS("./Modelos/fittedKNN.rds")
fit.svm <- readRDS("./Modelos/fittedSVM.rds")
fit.rf  <- readRDS("./Modelos/fittedRF.rds")
fit.c50 <- readRDS("./Modelos/fittedC50.rds")
fit.gbm <- readRDS("./Modelos/fittedGBM.rds")

# cria tabela resumo dos resultados
results <- resamples(list(lda=fit.lda, knn=fit.knn, svm=fit.svm,
                          rf=fit.rf, c50=fit.c50, gbm=fit.gbm))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, knn, svm, rf, c50, gbm
## Number of resamples: 30
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda 0.9198370 0.9377342 0.9415761 0.9407457 0.9429929 0.9538043    0
## knn 0.9320652 0.9418173 0.9490830 0.9479964 0.9547776 0.9633650    0
## svm 0.9510870 0.9551173 0.9578516 0.9594655 0.9642620 0.9714286    0
## rf  0.9660326 0.9700680 0.9727891 0.9735676 0.9768629 0.9850543    0
## c50 0.9728261 0.9795501 0.9809783 0.9823174 0.9860177 0.9932065    0
## gbm 0.9673913 0.9769022 0.9803127 0.9803686 0.9836901 0.9904891    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda 0.9034675 0.9250370 0.9296393 0.9286438 0.9313543 0.9443815    0
## knn 0.9181917 0.9299394 0.9386786 0.9373796 0.9455505 0.9558955    0
## svm 0.9411271 0.9459724 0.9492611 0.9512031 0.9569763 0.9656053    0
## rf  0.9591170 0.9639700 0.9672423 0.9681821 0.9721465 0.9820105    0
## c50 0.9672942 0.9753814 0.9771031 0.9787150 0.9831672 0.9918237    0
## gbm 0.9607557 0.9721969 0.9763014 0.9763692 0.9803674 0.9885521    0
```

```r
# plota boxplot dos resultados
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(results, scales=scales, metric = "Kappa"
       , main = "Performance dos métodos testados")
```



Tuning de hiperparâmetros

Nota-se que os algoritmos de *boosting* se mostraram mais eficazes, registrando Kappa médio de 97,9%. Por isso, o algoritmo selecionado para prosseguir com a modelagem é o *boosting* C5.0.

Para isso é necessário utilizar diretamente a biblioteca C5.0 e customizar a função de treinamento para permitir uma busca extensiva pelos melhores parâmetros.

Para avaliação de performance é mantido o método de *Cross Validation* repetido de 10 dobras e 3 repetições.

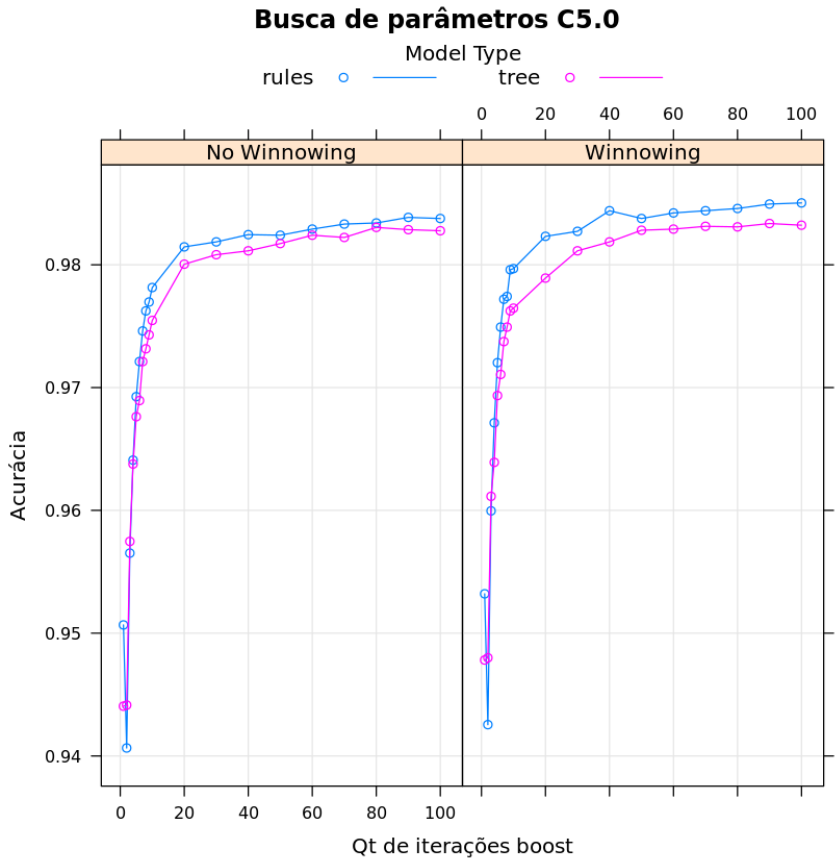
O melhor modelo então é definido com os parâmetros:

- trials: 100
- modelo: Rules
- winnow: TRUE

Nota: o trecho abaixo foi computado no IBM Cloud devido ao esforço computacional necessário. Os resultados são mostrados nesse relatório.

```
library(caret)
library(C50)

ctrl <- trainControl(method = "repeatedcv",
                             number = 10,
                             repeats = 3, returnResamp="all")
```

```
c50Grid <- expand.grid(.trials = c(1:9, (1:10)*10),
                       .model = c("tree", "rules"),
                       .winnow = c(TRUE, FALSE))
#c50Grid
set.seed(1234)
c5Fitvac <- train(Activity_ID ~ .,
                  data = dataset,
                  method = "C5.0",
                  tuneGrid = c50Grid,
                  trControl = ctrl,
                  metric = "Accuracy",
                  importance=TRUE)

# melhor modelo
c5Fitvac$finalModel$tuneValue
# salva busca extensiva
saveRDS(fit.gridC50, "./fittedgridC50.rds")
# plota gráfico
plot(c5Fitvac, main = "Busca de paraâmetros C5.0",
     xlab = "Qt de iterações boost", ylab = "Acurácia")
```



Finalização do modelo

Finalmente, basta treinar um modelo C5.0 com os parâmetros definidos, salvar o modelo e utilizá-lo para prever a amostra de teste, nunca antes vista pelo modelo.

Antes é necessário realizar na amostra de teste o mesmo pré-processamento feito na amostra de treino.

Observa-se que o modelo final tem perfomance de $Kappa = 88,5$ para a amostra de teste, dessa forma considera-se que o objetivo de identificar com precisão a atividade desenvolvida foi alcançado com sucesso.

```
dataset <- read.table("./train.txt")
#dataset <-  read.table(text = rawToChar(obj))    para IBM Cloud

# transformação em factor para ID de atividade e indivíduo
dataset$Activity_ID = as.factor(dataset$Activity_ID)
dataset$Subject = as.factor(dataset$Subject)

# busca variáveis com variância baixa e filtra o dataset
nzv_names <- nearZeroVar(dataset, names=TRUE)
dataset <- dataset[, setdiff(names(dataset), nzv_names)]

# retira a coluna referente ao nome da atividade e indivíduo
dataset <- dataset[, -2:-3]

c50Grid <- expand.grid(.trials = 100,
                       .model = "rules",
                       .winnow = TRUE)
set.seed(1234)
c5Fitvac <- train(Activity_ID ~ .,
                  data = dataset,
                  method = "C5.0",
                  tuneGrid = c50Grid,
                  metric = "Accuracy",
                  importance=TRUE)

saveRDS(c5Fitvac, "./finalC50.rds")
```

```
dataset <- read.table("./train.txt")
test <- read.table("./test.txt")
#dataset <-  read.table(text = rawToChar(obj))    para IBM Cloud

# transformação em factor para ID de atividade e indivíduo
dataset$Activity_ID = as.factor(dataset$Activity_ID)
dataset$Subject = as.factor(dataset$Subject)
test$Activity_ID = as.factor(test$Activity_ID)
test$Subject = as.factor(test$Subject)

# busca variáveis com variância baixa e filtra o dataset
nzv_names <- nearZeroVar(dataset, names=TRUE)
dataset <- dataset[, setdiff(names(dataset), nzv_names)]
test <- test[, setdiff(names(dataset), nzv_names)]

# retira a coluna referente ao nome da atividade e indivíduo
dataset <- dataset[, -2:-3]
test <- test[, -2:-3]
```

```r
finalc50 <- readRDS("./Modelos/finalC50.rds")

predictions <- predict(finalc50, test)
confusionMatrix(predictions, test$Activity_ID)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5   6
##          1 461  38  23   0   0   0
##          2  19 421  44   0   0   0
##          3  16  12 353   0   0   0
##          4   0   0   0 408  46   0
##          5   0   0   0  83 486   0
##          6   0   0   0   0   0 537
##
## Overall Statistics
##
##                Accuracy : 0.9046
##                  95% CI : (0.8935, 0.915)
##     No Information Rate : 0.1822
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8854
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.9294   0.8938   0.8405   0.8310   0.9135   1.0000
## Specificity            0.9751   0.9746   0.9889   0.9813   0.9656   1.0000
## Pos Pred Value         0.8831   0.8698   0.9265   0.8987   0.8541   1.0000
## Neg Pred Value         0.9856   0.9797   0.9739   0.9667   0.9807   1.0000
## Prevalence             0.1683   0.1598   0.1425   0.1666   0.1805   0.1822
## Detection Rate         0.1564   0.1429   0.1198   0.1384   0.1649   0.1822
## Detection Prevalence   0.1771   0.1642   0.1293   0.1541   0.1931   0.1822
## Balanced Accuracy      0.9523   0.9342   0.9147   0.9061   0.9396   1.0000
```

Referências

[1] APA. Kuhn, M., & Johnson, K. (2018). Applied predictive modeling. Springer.

[2] BROWNLEE, J. (2016). Machine Learning Mastery with R.