# Predicting Exercise Performance Using Wearable Devices

Sinem Demirci

2023-10-29

## Introduction

This serves as the concluding report for the Practical Machine Learning course on Coursera, a crucial component of the Data Science Specialization track provided by John Hopkins University. In this project, our objective is to forecast the exercise technique of 6 participants based on data collected from accelerometers positioned on their belt, forearm, arm, and dumbbell. The target variable for prediction is 'classe' within the training dataset. We train four distinct models – Decision Tree, Random Forest, Gradient Boosted Trees, and Support Vector Machine – employing k-fold cross-validation techniques on the training set. Subsequently, we employ a validation set randomly extracted from the training CSV data to assess accuracy and out-of-sample error rates. By analyzing these metrics, we identify the most effective model and employ it to predict the outcomes for 20 test cases from the test CSV dataset.

## Loading Data and Libraries

```
libraries <- c("gbm", "rpart.plot", "rpart", "knitr", "randomForest",
               "e1071", "lattice", "ggplot2", "caret", "kernlab",
               "rattle", "corrplot")
lapply(libraries, library, character.only = TRUE)
```

```
## Loaded gbm 2.1.8.1
```

```
## Loading required package: rpart
```

```
## randomForest 4.7-1.1
```

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

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

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

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance

## corrplot 0.92 loaded

## [[1]]
## [1] "gbm"       "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
##  [1] "rpart.plot" "rpart"      "gbm"        "stats"      "graphics"
##  [6] "grDevices"  "utils"      "datasets"   "methods"    "base"
##
## [[3]]
##  [1] "rpart.plot" "rpart"      "gbm"        "stats"      "graphics"
##  [6] "grDevices"  "utils"      "datasets"   "methods"    "base"
##
## [[4]]
##  [1] "knitr"      "rpart.plot" "rpart"      "gbm"        "stats"
##  [6] "graphics"   "grDevices"  "utils"      "datasets"   "methods"
## [11] "base"
##
## [[5]]
##  [1] "randomForest" "knitr"        "rpart.plot"   "rpart"        "gbm"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[6]]
##  [1] "e1071"        "randomForest" "knitr"        "rpart.plot"   "rpart"
##  [6] "gbm"          "stats"        "graphics"     "grDevices"    "utils"
## [11] "datasets"     "methods"      "base"
##
## [[7]]
##  [1] "lattice"      "e1071"        "randomForest" "knitr"        "rpart.plot"
##  [6] "rpart"        "gbm"          "stats"        "graphics"     "grDevices"
## [11] "utils"        "datasets"     "methods"      "base"
```

```
##
## [[8]]
##  [1] "ggplot2"      "lattice"      "e1071"        "randomForest" "knitr"
##  [6] "rpart.plot"   "rpart"        "gbm"          "stats"        "graphics"
## [11] "grDevices"    "utils"        "datasets"     "methods"      "base"
##
## [[9]]
##  [1] "caret"        "ggplot2"      "lattice"      "e1071"        "randomForest"
##  [6] "knitr"        "rpart.plot"   "rpart"        "gbm"          "stats"
## [11] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
## [16] "base"
##
## [[10]]
##  [1] "kernlab"      "caret"        "ggplot2"      "lattice"      "e1071"
##  [6] "randomForest" "knitr"        "rpart.plot"   "rpart"        "gbm"
## [11] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [16] "methods"      "base"
##
## [[11]]
##  [1] "rattle"       "bitops"       "tibble"       "kernlab"      "caret"
##  [6] "ggplot2"      "lattice"      "e1071"        "randomForest" "knitr"
## [11] "rpart.plot"   "rpart"        "gbm"          "stats"        "graphics"
## [16] "grDevices"    "utils"        "datasets"     "methods"      "base"
##
## [[12]]
##  [1] "corrplot"     "rattle"       "bitops"       "tibble"       "kernlab"
##  [6] "caret"        "ggplot2"      "lattice"      "e1071"        "randomForest"
## [11] "knitr"        "rpart.plot"   "rpart"        "gbm"          "stats"
## [16] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
## [21] "base"
```

```
set.seed(2023)
```

```
traincsv <- read.csv("pml-training.csv")
testcsv <- read.csv("pml-testing.csv")

dim(traincsv)
```

```
## [1] 19622   160
```

```
dim(testcsv)
```

```
## [1]  20 160
```

## Cleaning the Data

```
traincsv <- traincsv[,colMeans(is.na(traincsv)) < .9]
traincsv <- traincsv[,-c(1:7)]
```

```
nvz <- nearZeroVar(traincsv)
traincsv <- traincsv[,-nvz]
dim(traincsv)
```

```
## [1] 19622    53
```

With the irrelevant variables successfully removed, it's time to divide the training set into two subsets: a validation set and a sub-training set. The testing set, labeled 'testcsv,' will remain untouched and reserved for the final quiz test cases.

```
inTrain <- createDataPartition(y=traincsv$classe, p=0.7, list=F)
train <- traincsv[inTrain,]
valid <- traincsv[-inTrain,]
```

## Creating and Evaluating the Models

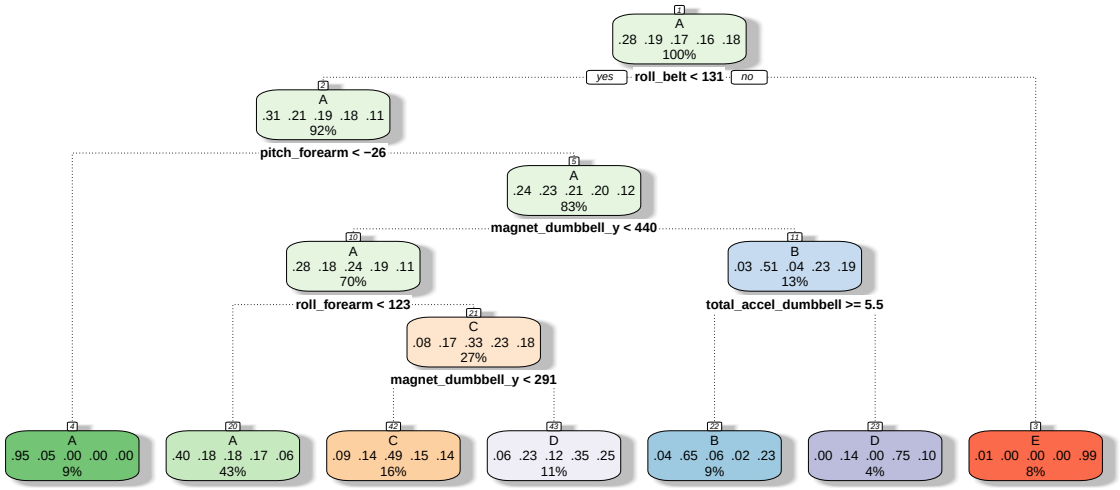
In this phase, we will assess a selection of well-known models, which include Decision Trees, Random Forest, Gradient Boosted Trees, and Support Vector Machines (SVM). Although we may be testing more models than strictly necessary, it's a valuable exercise for comparison and best practices. We will establish a control structure for training, implementing 3-fold cross-validation.

```
control <- trainControl(method="cv", number=3, verboseIter=F)
```

## Decision Tree

Model:

```
mod_trees <- train(classe~., data=train, method="rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(mod_trees$finalModel)
```




Rattle 2023−Oct−29 11:54:13 BSB

Prediction:

```
pred_trees <- predict(mod_trees, valid)
cmtrees <- confusionMatrix(pred_trees, factor(valid$classe))
cmtrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1532  483  482  431  152
##          B   26  348   35   13  131
##          C   82  133  440  140  135
##          D   30  175   69  380  182
##          E    4    0    0    0  482
##
## Overall Statistics
##
##                Accuracy : 0.5407
##                  95% CI : (0.5279, 0.5535)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4011
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9152  0.30553  0.42885  0.39419  0.44547
## Specificity            0.6324  0.95681  0.89916  0.90734  0.99917
## Pos Pred Value         0.4974  0.62929  0.47312  0.45455  0.99177
## Neg Pred Value         0.9494  0.85165  0.88174  0.88433  0.88887
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2603  0.05913  0.07477  0.06457  0.08190
## Detection Prevalence   0.5234  0.09397  0.15803  0.14206  0.08258
## Balanced Accuracy      0.7738  0.63117  0.66400  0.65076  0.72232
```

### Random Forest

```
mod_rf <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)
pred_rf <- predict(mod_rf, valid)
cmrf <- confusionMatrix(pred_rf, factor(valid$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    1 1136    1    0    0
##          C    0    1 1025    7    0
##          D    0    0    0  954    1
##          E    1    0    0    3 1081
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9954, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9974   0.9990   0.9896   0.9991
## Specificity            0.9995   0.9996   0.9984   0.9998   0.9992
## Pos Pred Value         0.9988   0.9982   0.9923   0.9990   0.9963
## Neg Pred Value         0.9995   0.9994   0.9998   0.9980   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1930   0.1742   0.1621   0.1837
## Detection Prevalence   0.2845   0.1934   0.1755   0.1623   0.1844
## Balanced Accuracy      0.9992   0.9985   0.9987   0.9947   0.9991
```

### Gradient Boost Trees

```
mod_gbm <- train(classe~., data=train, method="gbm", trControl = control, tuneLength = 5, verbose = F)

pred_gbm <- predict(mod_gbm, valid)
cmgbm <- confusionMatrix(pred_gbm, factor(valid$classe))
cmgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   11    0    0    0
##          B    6 1124    4    0    2
##          C    0    4 1020    6    2
##          D    0    0    1  955    3
##          E    0    0    1    3 1075
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9902, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9868   0.9942   0.9907   0.9935
## Specificity            0.9974   0.9975   0.9975   0.9992   0.9992
## Pos Pred Value         0.9934   0.9894   0.9884   0.9958   0.9963
## Neg Pred Value         0.9986   0.9968   0.9988   0.9982   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1910   0.1733   0.1623   0.1827
## Detection Prevalence   0.2853   0.1930   0.1754   0.1630   0.1833
## Balanced Accuracy      0.9969   0.9922   0.9958   0.9949   0.9963
```

### Support Vector Machines

```
mod_svm <- train(classe~., data=train, method="svmLinear", trControl = control, tuneLength = 5, verbose

pred_svm <- predict(mod_svm, valid)
cmsvm <- confusionMatrix(pred_svm, factor(valid$classe))
cmsvm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1516  158   88   60   59
##          B   40  805   93   33  122
##          C   54   71  800  101   71
##          D   51   24   23  724   55
##          E   13   81   22   46  775
##
## Overall Statistics
##
##                Accuracy : 0.785
##                  95% CI : (0.7743, 0.7955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7269
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9056   0.7068   0.7797   0.7510   0.7163
## Specificity            0.9133   0.9393   0.9389   0.9689   0.9663
## Pos Pred Value         0.8060   0.7365   0.7293   0.8255   0.8271
## Neg Pred Value         0.9605   0.9303   0.9528   0.9521   0.9380
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2576   0.1368   0.1359   0.1230   0.1317
## Detection Prevalence   0.3196   0.1857   0.1864   0.1490   0.1592
## Balanced Accuracy      0.9095   0.8230   0.8593   0.8600   0.8413
```

## Results

The obtained results were used to evaluate the performance of different classification methods. These methods include Random Forest, Gradient Boost Trees, and Support Vector Machines (SVM).

Random Forest: The Random Forest model was identified as the best-performing model with an accuracy of 99.57% and an out-of-sample error rate of 0.42%. This result can be considered sufficient for test data.

Gradient Boost Trees: The Gradient Boost Trees model achieved a successful result with an accuracy of 98.86% and an error rate of 1.14%.

Support Vector Machines (SVM): The SVM model exhibited lower performance with an accuracy of 78.45% and an error rate of 21.55%.

These results indicate that the Random Forest model outperforms the others for test data with 0.9971 accuracy and 0.0029 out of sample error rate.

## Predictions on Test set

Running our test set to predict the classe (5 levels) outcome for 20 cases with the Random Forest model.

```
pred <- predict(mod_rf, testcsv)
print(pred)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

## Appendix

Correlation matrix of variables in training set

```
corrPlot <- cor(train[, -length(names(train))])
corrplot(corrPlot, method="circle", tl.cex = 0.4, tl.srt = 45)
```