Predict activity quality from activity monitors

Siddhant Sharma

19th October 2020

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement ??? a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

Data description

The outcome variable is `classe` , a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

Initial configuration

The initial configuration consists of loading some required packages and initializing some variables.

library(caret) library(rpart) library(ggplot2) library(corrplot) library(randomForest) library(rattle) set.seed(12345)

Loading the provided data

training_raw <- read.csv(“pml-training.csv”)[,-1] testing <- read.csv(“pml-testing.csv”)[,-1] # check dimension of the training and test dataset dim(training_raw) dim(testing)

Cleaning the loaded data

NZV <- nearZeroVar(training_raw) training <- training_raw[, -NZV] testing <- testing[, -NZV]

NaValues <- sapply(training, function(x) mean(is.na(x))) > 0.9 training <- training[, NaValues == “FALSE”] testing <- testing[, NaValues == “FALSE”]

training <- training[,-c(1:5)] testing <- testing[,-c(1:5)]

dim(training)

dim(testing)

head(training)

Data partition

inTrain <- createDataPartition(y= training$classe, p = 0.7, list = FALSE) training <- training[inTrain,] crossvalidation <- training[-inTrain,]

Training models

model_tree <- train(classe~., data = training, method = “rpart”)

predict_training_tree <- predict(model_tree, training) confusionmatrix_training_tree <- confusionMatrix(predict_training_tree, training$classe)

predict_crossvalidation_tree <- predict(model_tree, crossvalidation) confusionmatrix_cv_tree <- confusionMatrix(predict_crossvalidation_tree, crossvalidation$classe)

print(confusionmatrix_cv_tree)

model_rf <- train(classe~., data = training, method = “rf”)

predict_training_rf <- predict(model_rf, training) confusionmatrix_training_rf <- confusionMatrix(predict_training_rf, training$classe)

predict_crossvalidation_rf <- predict(model_rf, crossvalidation) confusionmatrix_cv_rf <- confusionMatrix(predict_crossvalidation_rf, crossvalidation$classe)

print(confusionmatrix_cv_rf)

Conclusion

As we can we from the result, the random forest algorithem far outperforms the decision tree in terms of accuracy. We are getting 99.25% in sample accuracy, while the decision tree gives us only nearly 50% in sample accuracy

##Prediction predict_testing <- predict(model_rf, testing) predict_testing

Appendix

predictor_factor <- which(sapply(training, class) == “factor”)

predictor_cor <- abs(cor(training[,-predictor_factor]))

predictor_cor[lower.tri(predictor_cor, diag = TRUE)] <- 0

corrplot(predictor_cor, method = “color”, type = “upper”, cl.lim = c(0,1), tl.col = rgb(0, 0, 0)) which(predictor_cor > 0.8, arr.ind = TRUE)