Practice Machine Learning Project

LY

4/22/2021

### Background and Goal

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, Six participants at age 20-28 were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and throwing the hips to the front (Class E). More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here: <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here: <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

The goal of this project is to predict the manner (the “classe” variable in the training set) in which the participants did the exercise using data from accelerometers on the belt, forearm, arm, and dumbell. The ‘classe’ variable can be predicted with any of the other variables. Modeling, validation and thoughts of expected out of sample error will be included in this report. The prediction model will be used to predict 20 different test cases.

### Implementation

First step is to load and clean the data.

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(rpart)  
library(rpart.plot)  
library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

##   
## Attaching package: 'rattle'

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

### Read in the downloaded data and check the data   
trainingData <- read.csv("pml-training.csv", header = TRUE, sep = ",", na.strings = c("NA", ""))  
testingData <- read.csv("pml-testing.csv", header = TRUE, sep = ",", na.strings = c("NA", ""))  
  
### Clean the data  
## Clean data step 1: remove variables containing NA’s and missing values.   
trainingData <- trainingData[, which(sapply(trainingData, function(x) {sum(is.na(x))}) == 0)]  
testingData <- testingData[, which(sapply(testingData, function(x) {sum(is.na(x))}) == 0)]  
dim(trainingData)

## [1] 19622 60

## The variables have now been reduced from 160 to 60 in the data set.  
## Clean data step 2: remove variables have near zero variance   
nzv <- nearZeroVar(trainingData, saveMetrics = TRUE)  
trainingData <- trainingData[ , nzv$nzv == "FALSE"]  
trainingData$classe <- as.factor(trainingData$classe)  
nzv <- nearZeroVar(testingData, saveMetrics = TRUE)  
testingData <- testingData[ , nzv$nzv == "FALSE"]  
dim(trainingData)

## [1] 19622 59

## Clean data step 3: remove the first 6 variables, as they have nothing to do with making the predictions  
## There were many observations of each variable (n = 19622). Because random forest processing can be quite time-consuming, I subsetted only 5000 rows for the classification to process more quickly.  
set.seed(123)  
trainingDataSubset <- trainingData[sample(nrow(trainingData), 5000), -c(1:6)]  
testingData <- testingData[ , -c(1:6)]  
dim(trainingDataSubset)

## [1] 5000 53

Secondly, run the cross-validation on the training data set, 70% of the data will be used for training the model and 30% for checking the prediction performance of the model.

## Split training data set into two parts  
set.seed(1234)  
inTrain <- createDataPartition(trainingDataSubset$classe, p = 0.7, list = FALSE)  
training <- trainingDataSubset[inTrain, ]  
testing <- trainingData[-inTrain, ]

Next step is to build the model using the method of Random Forest. The reason for this is that Random Forest is very accurate among other algorithms and it runs very efficiently on large data sets. We will run the set on 5-fold cross validation. In 5-fold cross-validation, the original data-set is randomly partitioned into 5 equal sized sub data-sets. Of the 5 sub data-sets, a single sub data-set is retained as the validation data for testing the model, and the remaining 4 sub data-sets are used as training data. The cross-validation process is then repeated 5 times (the folds), with each of the 5 sub data-sets used exactly once as the validation data. The 5 results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

set.seed(12345)  
rfModel <- train(classe ~., method = "rf", data = training,  
 trControl = trainControl(method = "cv", number = 5),  
 prox = TRUE, allowParallel = TRUE)  
  
rfModel

## Random Forest   
##   
## 3501 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2801, 2801, 2800, 2801, 2801   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9545849 0.9424288  
## 27 0.9602975 0.9496925  
## 52 0.9551575 0.9431759  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 27.

Check performance of model The model will be tested on the validation data (partition of the training data) and a confusion matrix will be used to check the accuracy of the prediction on the validation data.

predictTesting <- predict(rfModel, testing)  
confusionMatrix(testing$classe, predictTesting)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2065 6 4 3 1  
## B 114 3598 73 12 0  
## C 0 65 3316 32 9  
## D 6 7 87 3110 6  
## E 3 20 33 21 3530  
##   
## Overall Statistics  
##   
## Accuracy : 0.9689   
## 95% CI : (0.9661, 0.9715)  
## No Information Rate : 0.2293   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9608   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9438 0.9735 0.9439 0.9786 0.9955  
## Specificity 0.9990 0.9840 0.9916 0.9918 0.9939  
## Pos Pred Value 0.9933 0.9476 0.9690 0.9670 0.9787  
## Neg Pred Value 0.9912 0.9920 0.9845 0.9947 0.9987  
## Prevalence 0.1357 0.2293 0.2179 0.1971 0.2200  
## Detection Rate 0.1281 0.2232 0.2057 0.1929 0.2190  
## Detection Prevalence 0.1290 0.2355 0.2123 0.1995 0.2237  
## Balanced Accuracy 0.9714 0.9787 0.9678 0.9852 0.9947

##Accuracy  
accuracy <- confusionMatrix(testing$classe, predictTesting)$overall[1]  
##Out of sample error  
OOSError <- 1 - confusionMatrix(testing$classe, predictTesting)$overall[1]  
cat("Accuracy: ", accuracy)

## Accuracy: 0.9688605

cat("Out of sample error: ", OOSError)

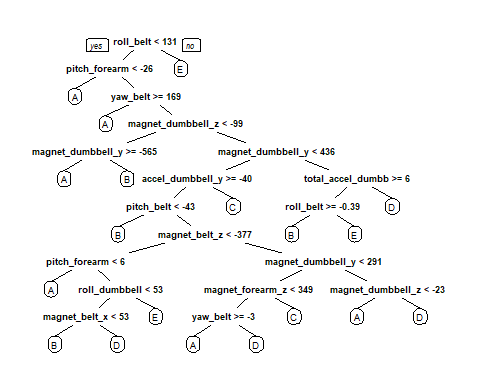
## Out of sample error: 0.03113951

Finally, run the model on the test data The Random Forest model is now applied to the test data to predict the outcome.

answer <- predict(rfModel, testingData)  
answer

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

##Random Forest decision tree  
rfModelTree <- rpart(classe ~., data = training, method = "class")  
prp(rfModelTree)



##Plot of the top 20 variables impact on outcome  
plot(varImp(rfModel), top = 20)

