Practical Machine Learning Course | Peer Review Project

Evgeniy Paskin

21 october 2016

Synopsis

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the quality (class) of excercise execution

The goal of the project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We’ll use all of the other variables to predict the class. This report describes the model building process as well as cross validation procedures, estimation of the expected out of sample error.

The resulting model will be used to predict 20 different test cases.

The Data Description

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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

The data comes in two files: training and testing.

Executive summary & Results highlight:

Based on the analyzed statistics and models it turns out that:

- The data allows us to build very accurate model
- Out of 3 used models: Random Forest, SVM Radial and GBM, the RF model showed the best accuracy (>>99%) and was used for further predictions
- The selected model also showed very zero out-of-sample error rate when checked on testing data (even smaller, than for training data) reaching accuracy of 100%
- The RF model applied to testing data resulted in 100% correct predictions

---

Loading and preparing data, loading required packages

Including required packages

```
library(caret)
library(ggplot2)
library(rattle)
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```
set.seed(20161021) # setting seed for reproducibility
```

Download raw data files

```
# Download data
trainUrl<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

trainCsv <- "pml-training.csv"
testCsv <- "pml-testing.csv"

if (!file.exists(trainCsv)) {download.file(trainUrl, destfile=trainCsv, method="cu
rl")}
if (!file.exists(testCsv))  {download.file(testUrl, destfile=testCsv, method="curl
")}

# Read files into R
train <- read.table(trainCsv, sep=",", dec=".", header = TRUE, na.strings=c("NA","
"))
test <- read.table(testCsv, sep=",", dec=".", header = TRUE, na.strings=c("NA","")
)

# Removing unnecessary variables
rm(testCsv, testUrl, trainCsv, trainUrl)
```

Cleaning data

In order to feed the algorithms with the correct data we'll clean up our data set

Cleaning nero zero variance predictors (NZV)

We'll use the 'nearZeroVar' from caret package to remove predictors' which variance is close to 0

```
NZV <- nearZeroVar(train, saveMetrics = TRUE)
train <- train[, !NZV$nzv]

# Let's check how many variables were droped
sum(NZV$nzv)
```

```
## [1] 43
```

With this pricedure we've totally dropped 43 variables

Cleaning predictors which do not relate to the excercise

The first columns contains descriptional data (e.g. individual name) and are not usefull to predict the excercise classe

```
train <- train[, -(1:6)]
```

Because Random Forest model doesn't work well with NA's, we'll drop all the variables, with number of NAs > 1300

```
train <- train[,colSums(is.na(train)) < 1300 ]
```

Let's check what are the dimensuions of the resulting train data frame

```
dim(train)
```

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

Spliting training data set 80/20

As a next step we'll split the training data set into training (80%) and testing (20%, to check out-of-sample errors rate for our model) The method for splitting is caret's function "createDataPartition".

```
inTrain <- createDataPartition(train$classe, p=0.80, list=FALSE)
training <- train[inTrain, ]
testing <- train[-inTrain, ]
```

Model Parameters and preprocessing

We'll gather modeling options into 'trainctrl' variable via trainControl function of the caret package. The function controls the computational nuances of the train function.

- To remove noise and redundant (highly correlated variables) we'll use PCA preprocessing
- To avoid overfittigd we'll used cross validation (5 times per each model)

```
trainctrl <- trainControl(method = "cv", number = 5 , preProcOptions="pca")
```

Modeling and models comparison

The full list of caret's algorithms is presented here (https://topepo.github.io/caret/modelList.html (https://topepo.github.io/caret/modelList.html)). For our puproses we'll use and compare only several of them: Random Forest (rf), Gradient Boosting Machine (GBM) and Support Vector Machine (SVM) These models are very widely used and are considered quite good

We'll use cache=TRUE Knitr option for models training to save time on rebuilds

```
rf <- train(classe ~ ., data = train, method = "rf", trControl= trainctrl)
svmr <- train(classe ~ ., data = train, method = "svmRadial", trControl= trainctrl
)
gbm <- train(classe ~ ., data = train, method = "gbm", trControl= trainctrl)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1244
##      2        1.5233             nan     0.1000    0.0870
##      3        1.4661             nan     0.1000    0.0681
##      4        1.4223             nan     0.1000    0.0549
##      5        1.3868             nan     0.1000    0.0513
##      6        1.3543             nan     0.1000    0.0429
##      7        1.3262             nan     0.1000    0.0359
##      8        1.3028             nan     0.1000    0.0321
##      9        1.2825             nan     0.1000    0.0315
##     10        1.2627             nan     0.1000    0.0322
##     20        1.1052             nan     0.1000    0.0184
##     40        0.9352             nan     0.1000    0.0090
##     60        0.8278             nan     0.1000    0.0065
##     80        0.7478             nan     0.1000    0.0043
##    100        0.6856             nan     0.1000    0.0036
##    120        0.6350             nan     0.1000    0.0033
##    140        0.5915             nan     0.1000    0.0023
##    150        0.5737             nan     0.1000    0.0025
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1839
##      2        1.4893             nan     0.1000    0.1301
##      3        1.4061             nan     0.1000    0.1053
##      4        1.3385             nan     0.1000    0.0823
##      5        1.2849             nan     0.1000    0.0738
##      6        1.2376             nan     0.1000    0.0639
##      7        1.1979             nan     0.1000    0.0630
##      8        1.1588             nan     0.1000    0.0528
##      9        1.1250             nan     0.1000    0.0457
##     10        1.0965             nan     0.1000    0.0416
##     20        0.8946             nan     0.1000    0.0296
##     40        0.6775             nan     0.1000    0.0131
##     60        0.5546             nan     0.1000    0.0118
##     80        0.4674             nan     0.1000    0.0053
##    100        0.4031             nan     0.1000    0.0036
##    120        0.3498             nan     0.1000    0.0033
##    140        0.3082             nan     0.1000    0.0026
##    150        0.2905             nan     0.1000    0.0027
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2357
##      2        1.4585             nan     0.1000    0.1590
##      3        1.3581             nan     0.1000    0.1280
##      4        1.2776             nan     0.1000    0.1070
##      5        1.2105             nan     0.1000    0.0901
##      6        1.1545             nan     0.1000    0.0705
##      7        1.1081             nan     0.1000    0.0769
##      8        1.0606             nan     0.1000    0.0680
##      9        1.0185             nan     0.1000    0.0474
##     10        0.9884             nan     0.1000    0.0590
##     20        0.7472             nan     0.1000    0.0213
##     40        0.5248             nan     0.1000    0.0120
```

```
##     60        0.4051             nan     0.1000    0.0081
##     80        0.3277             nan     0.1000    0.0063
##    100        0.2693             nan     0.1000    0.0029
##    120        0.2265             nan     0.1000    0.0012
##    140        0.1937             nan     0.1000    0.0020
##    150        0.1778             nan     0.1000    0.0017
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1276
##      2        1.5236             nan     0.1000    0.0880
##      3        1.4655             nan     0.1000    0.0686
##      4        1.4206             nan     0.1000    0.0526
##      5        1.3858             nan     0.1000    0.0444
##      6        1.3565             nan     0.1000    0.0436
##      7        1.3284             nan     0.1000    0.0397
##      8        1.3031             nan     0.1000    0.0358
##      9        1.2805             nan     0.1000    0.0304
##     10        1.2605             nan     0.1000    0.0313
##     20        1.1082             nan     0.1000    0.0155
##     40        0.9388             nan     0.1000    0.0091
##     60        0.8328             nan     0.1000    0.0053
##     80        0.7531             nan     0.1000    0.0057
##    100        0.6889             nan     0.1000    0.0032
##    120        0.6385             nan     0.1000    0.0028
##    140        0.5955             nan     0.1000    0.0030
##    150        0.5766             nan     0.1000    0.0025
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1867
##      2        1.4894             nan     0.1000    0.1292
##      3        1.4053             nan     0.1000    0.1039
##      4        1.3394             nan     0.1000    0.0850
##      5        1.2856             nan     0.1000    0.0748
##      6        1.2384             nan     0.1000    0.0704
##      7        1.1940             nan     0.1000    0.0560
##      8        1.1579             nan     0.1000    0.0553
##      9        1.1231             nan     0.1000    0.0424
##     10        1.0966             nan     0.1000    0.0446
##     20        0.8967             nan     0.1000    0.0229
##     40        0.6869             nan     0.1000    0.0099
##     60        0.5616             nan     0.1000    0.0052
##     80        0.4750             nan     0.1000    0.0071
##    100        0.4077             nan     0.1000    0.0041
##    120        0.3575             nan     0.1000    0.0043
##    140        0.3144             nan     0.1000    0.0018
##    150        0.2980             nan     0.1000    0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2397
##      2        1.4609             nan     0.1000    0.1612
##      3        1.3588             nan     0.1000    0.1321
##      4        1.2761             nan     0.1000    0.1053
##      5        1.2103             nan     0.1000    0.0910
##      6        1.1529             nan     0.1000    0.0732
```

```
##      7        1.1064             nan     0.1000    0.0651
##      8        1.0660             nan     0.1000    0.0577
##      9        1.0296             nan     0.1000    0.0610
##     10        0.9922             nan     0.1000    0.0602
##     20        0.7652             nan     0.1000    0.0232
##     40        0.5353             nan     0.1000    0.0151
##     60        0.4109             nan     0.1000    0.0077
##     80        0.3302             nan     0.1000    0.0060
##    100        0.2714             nan     0.1000    0.0043
##    120        0.2289             nan     0.1000    0.0029
##    140        0.1954             nan     0.1000    0.0030
##    150        0.1814             nan     0.1000    0.0018
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1258
##      2        1.5243             nan     0.1000    0.0879
##      3        1.4667             nan     0.1000    0.0643
##      4        1.4236             nan     0.1000    0.0546
##      5        1.3881             nan     0.1000    0.0524
##      6        1.3551             nan     0.1000    0.0424
##      7        1.3273             nan     0.1000    0.0351
##      8        1.3045             nan     0.1000    0.0383
##      9        1.2799             nan     0.1000    0.0325
##     10        1.2597             nan     0.1000    0.0303
##     20        1.1091             nan     0.1000    0.0189
##     40        0.9345             nan     0.1000    0.0121
##     60        0.8284             nan     0.1000    0.0073
##     80        0.7499             nan     0.1000    0.0050
##    100        0.6869             nan     0.1000    0.0039
##    120        0.6344             nan     0.1000    0.0031
##    140        0.5916             nan     0.1000    0.0024
##    150        0.5727             nan     0.1000    0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1833
##      2        1.4909             nan     0.1000    0.1287
##      3        1.4084             nan     0.1000    0.1044
##      4        1.3415             nan     0.1000    0.0816
##      5        1.2885             nan     0.1000    0.0750
##      6        1.2404             nan     0.1000    0.0706
##      7        1.1969             nan     0.1000    0.0573
##      8        1.1601             nan     0.1000    0.0527
##      9        1.1264             nan     0.1000    0.0414
##     10        1.0993             nan     0.1000    0.0459
##     20        0.8961             nan     0.1000    0.0221
##     40        0.6854             nan     0.1000    0.0126
##     60        0.5630             nan     0.1000    0.0090
##     80        0.4758             nan     0.1000    0.0044
##    100        0.4060             nan     0.1000    0.0043
##    120        0.3553             nan     0.1000    0.0030
##    140        0.3116             nan     0.1000    0.0015
##    150        0.2916             nan     0.1000    0.0026
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.6094             nan     0.1000    0.2309
##      2        1.4620             nan     0.1000    0.1604
##      3        1.3621             nan     0.1000    0.1248
##      4        1.2840             nan     0.1000    0.1043
##      5        1.2182             nan     0.1000    0.1004
##      6        1.1569             nan     0.1000    0.0819
##      7        1.1048             nan     0.1000    0.0614
##      8        1.0654             nan     0.1000    0.0608
##      9        1.0272             nan     0.1000    0.0645
##     10        0.9885             nan     0.1000    0.0521
##     20        0.7631             nan     0.1000    0.0227
##     40        0.5296             nan     0.1000    0.0128
##     60        0.4049             nan     0.1000    0.0070
##     80        0.3274             nan     0.1000    0.0034
##    100        0.2698             nan     0.1000    0.0038
##    120        0.2257             nan     0.1000    0.0020
##    140        0.1926             nan     0.1000    0.0018
##    150        0.1777             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1278
##      2        1.5230             nan     0.1000    0.0873
##      3        1.4652             nan     0.1000    0.0674
##      4        1.4201             nan     0.1000    0.0523
##      5        1.3852             nan     0.1000    0.0522
##      6        1.3523             nan     0.1000    0.0433
##      7        1.3240             nan     0.1000    0.0350
##      8        1.3012             nan     0.1000    0.0355
##      9        1.2793             nan     0.1000    0.0351
##     10        1.2565             nan     0.1000    0.0297
##     20        1.1044             nan     0.1000    0.0163
##     40        0.9341             nan     0.1000    0.0106
##     60        0.8273             nan     0.1000    0.0054
##     80        0.7482             nan     0.1000    0.0049
##    100        0.6850             nan     0.1000    0.0037
##    120        0.6344             nan     0.1000    0.0033
##    140        0.5898             nan     0.1000    0.0013
##    150        0.5699             nan     0.1000    0.0036
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1893
##      2        1.4881             nan     0.1000    0.1287
##      3        1.4058             nan     0.1000    0.1087
##      4        1.3380             nan     0.1000    0.0821
##      5        1.2852             nan     0.1000    0.0685
##      6        1.2412             nan     0.1000    0.0742
##      7        1.1958             nan     0.1000    0.0574
##      8        1.1597             nan     0.1000    0.0526
##      9        1.1270             nan     0.1000    0.0489
##     10        1.0954             nan     0.1000    0.0441
##     20        0.8933             nan     0.1000    0.0193
##     40        0.6893             nan     0.1000    0.0131
##     60        0.5579             nan     0.1000    0.0089
##     80        0.4684             nan     0.1000    0.0053
```

```
##    100        0.4033             nan     0.1000    0.0030
##    120        0.3511             nan     0.1000    0.0027
##    140        0.3082             nan     0.1000    0.0019
##    150        0.2892             nan     0.1000    0.0030
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2378
##      2        1.4589             nan     0.1000    0.1586
##      3        1.3573             nan     0.1000    0.1317
##      4        1.2757             nan     0.1000    0.1055
##      5        1.2100             nan     0.1000    0.0812
##      6        1.1571             nan     0.1000    0.0685
##      7        1.1133             nan     0.1000    0.0751
##      8        1.0671             nan     0.1000    0.0673
##      9        1.0260             nan     0.1000    0.0533
##     10        0.9930             nan     0.1000    0.0532
##     20        0.7649             nan     0.1000    0.0275
##     40        0.5348             nan     0.1000    0.0184
##     60        0.4059             nan     0.1000    0.0075
##     80        0.3259             nan     0.1000    0.0020
##    100        0.2686             nan     0.1000    0.0031
##    120        0.2236             nan     0.1000    0.0015
##    140        0.1907             nan     0.1000    0.0018
##    150        0.1776             nan     0.1000    0.0012
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1302
##      2        1.5223             nan     0.1000    0.0879
##      3        1.4638             nan     0.1000    0.0667
##      4        1.4203             nan     0.1000    0.0538
##      5        1.3847             nan     0.1000    0.0507
##      6        1.3520             nan     0.1000    0.0403
##      7        1.3262             nan     0.1000    0.0413
##      8        1.3007             nan     0.1000    0.0351
##      9        1.2792             nan     0.1000    0.0331
##     10        1.2568             nan     0.1000    0.0304
##     20        1.1037             nan     0.1000    0.0160
##     40        0.9366             nan     0.1000    0.0112
##     60        0.8317             nan     0.1000    0.0070
##     80        0.7502             nan     0.1000    0.0039
##    100        0.6872             nan     0.1000    0.0040
##    120        0.6370             nan     0.1000    0.0029
##    140        0.5921             nan     0.1000    0.0028
##    150        0.5734             nan     0.1000    0.0019
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1901
##      2        1.4866             nan     0.1000    0.1271
##      3        1.4048             nan     0.1000    0.1059
##      4        1.3367             nan     0.1000    0.0815
##      5        1.2841             nan     0.1000    0.0746
##      6        1.2364             nan     0.1000    0.0636
##      7        1.1955             nan     0.1000    0.0552
##      8        1.1611             nan     0.1000    0.0507
```

```
##      9        1.1294             nan     0.1000    0.0486
##     10        1.0981             nan     0.1000    0.0395
##     20        0.8949             nan     0.1000    0.0262
##     40        0.6820             nan     0.1000    0.0117
##     60        0.5594             nan     0.1000    0.0090
##     80        0.4703             nan     0.1000    0.0068
##    100        0.4051             nan     0.1000    0.0031
##    120        0.3545             nan     0.1000    0.0024
##    140        0.3142             nan     0.1000    0.0017
##    150        0.2957             nan     0.1000    0.0033
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2336
##      2        1.4596             nan     0.1000    0.1561
##      3        1.3600             nan     0.1000    0.1255
##      4        1.2815             nan     0.1000    0.0981
##      5        1.2188             nan     0.1000    0.1024
##      6        1.1553             nan     0.1000    0.0721
##      7        1.1090             nan     0.1000    0.0655
##      8        1.0674             nan     0.1000    0.0638
##      9        1.0278             nan     0.1000    0.0653
##     10        0.9879             nan     0.1000    0.0533
##     20        0.7522             nan     0.1000    0.0252
##     40        0.5327             nan     0.1000    0.0140
##     60        0.4072             nan     0.1000    0.0099
##     80        0.3277             nan     0.1000    0.0048
##    100        0.2719             nan     0.1000    0.0030
##    120        0.2277             nan     0.1000    0.0017
##    140        0.1956             nan     0.1000    0.0017
##    150        0.1810             nan     0.1000    0.0016
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2361
##      2        1.4597             nan     0.1000    0.1572
##      3        1.3604             nan     0.1000    0.1258
##      4        1.2805             nan     0.1000    0.1084
##      5        1.2134             nan     0.1000    0.0877
##      6        1.1594             nan     0.1000    0.0752
##      7        1.1120             nan     0.1000    0.0682
##      8        1.0697             nan     0.1000    0.0606
##      9        1.0313             nan     0.1000    0.0612
##     10        0.9941             nan     0.1000    0.0528
##     20        0.7579             nan     0.1000    0.0252
##     40        0.5369             nan     0.1000    0.0170
##     60        0.4125             nan     0.1000    0.0082
##     80        0.3320             nan     0.1000    0.0048
##    100        0.2712             nan     0.1000    0.0060
##    120        0.2274             nan     0.1000    0.0020
##    140        0.1952             nan     0.1000    0.0016
##    150        0.1818             nan     0.1000    0.0012
```

Let’s take a look into models accuracy results:

```
max(rf$results$Accuracy) # Random forest
```

```
## [1] 0.9946998
```

```
max(svmr$results$Accuracy) # SVM Radial
```

```
## [1] 0.9337985
```

```
max(gbm$results$Accuracy) # Gradient Boosting
```

```
## [1] 0.9619816
```

Although all models shows quite solid results (>90%), the Random Forest is true leader with 99.4% accuracy. Confusion matrix from the model summary below shows all details

```
print(rf)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15697, 15698, 15698, 15697, 15698
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9941392  0.9925859
##   27    0.9946998  0.9932952
##   52    0.9890428  0.9861382
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(rf$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.41%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5574    5    1    0    0 0.001075269
## B   19 3773    4    1    0 0.006320780
## C    0    7 3405   10    0 0.004967855
## D    1    1   21 3189    4 0.008395522
## E    0    0    3    4 3600 0.001940671
```

We'll use this model to predict results on the testing part of the data (20% of the initial dataset) to check for out-of-sample error behavior

```
# Predicting the classes
prediction1 <- predict(rf, testing)

# Comparing classes with the actual testing dataset
confusionMatrix(prediction1, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    0    0    0
##          C    0    0  684    0    0
##          D    0    0    0  643    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Accuracy on the testing data reached 100%. The out-of-sample error rate (zero) is even lower than the in-sample. That means that chances are our model is quite good.

Prediction of TEST data

We'll use our model to predict classes for test data:

```
prediction2 <- predict(rf, test)
as.data.frame(prediction2)
```

```
##    prediction2
## 1            B
## 2            A
## 3            B
## 4            A
## 5            A
## 6            E
## 7            D
## 8            B
## 9            A
## 10           A
## 11           B
## 12           C
## 13           B
## 14           A
## 15           E
## 16           E
## 17           A
## 18           B
## 19           B
## 20           B
```

The prediction seems correct, as the final score is 100% (20/20)