Machine Learning Project

suszanna

12/08/2020

BACKGROUND

Various devices equipped with embedded accelerometers can be worn on the body during physical exercise to collect data for subsequent analysis of the quality of the exercise after the fact. These devices are used to measure quality of performance of the movements of arms, legs etc. Resulting data is trained by resampling with cross validation to come up with reproducible metrics for accuracy and error rates to classify each performance of the exercise. The data we use was collected from 6 participants who performed one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different variations of the specified exercise: * Class A - exactly according to the specification * Class B - throwing the elbows to the front * Class C - lifting the dumbbell only halfway * Class D - lowering the dumbbell only halfway * Class E - throwing the hips to the front.

GOALS

The goal of this study is to make a reasonable prediction. To do this, we run models that produce predictions typical of each model. We compare the metrics of each (accuracy and error rates) & select the best prediction as our outcome. We focus on the ‘classe’ variable in the above described training set: A,B,C,D,E.

DATA

Load, clean and split

```
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.

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(rpart.plot)
library(corrplot)

## corrplot 0.84 loaded

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

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

library(RColorBrewer)
```

Load and Read CSV Data

```
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"

trainRaw <- read.csv(trainFile)
testRaw <- read.csv(testFile)
```

Clean

```
NZV <- nearZeroVar(trainRaw, saveMetrics = TRUE)
head(NZV, 20)
```

```
##                        freqRatio percentUnique zeroVar   nzv
## X                       1.000000  100.00000000   FALSE FALSE
## user_name               1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1    1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2    1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp          1.000668    0.10192641   FALSE FALSE
## new_window             47.330049    0.01019264   FALSE  TRUE
## num_window              1.000000    4.37264295   FALSE FALSE
## roll_belt               1.101904    6.77810621   FALSE FALSE
## pitch_belt              1.036082    9.37722964   FALSE FALSE
## yaw_belt                1.058480    9.97349913   FALSE FALSE
## total_accel_belt        1.063160    0.14779329   FALSE FALSE
## kurtosis_roll_belt   1921.600000    2.02323922   FALSE  TRUE
## kurtosis_picth_belt   600.500000    1.61553358   FALSE  TRUE
## kurtosis_yaw_belt      47.330049    0.01019264   FALSE  TRUE
## skewness_roll_belt   2135.111111    2.01304658   FALSE  TRUE
## skewness_roll_belt.1  600.500000    1.72255631   FALSE  TRUE
## skewness_yaw_belt      47.330049    0.01019264   FALSE  TRUE
## max_roll_belt           1.000000    0.99378249   FALSE FALSE
## max_picth_belt          1.538462    0.11211905   FALSE FALSE
## max_yaw_belt          640.533333    0.34654979   FALSE  TRUE
```

```
training01 <- trainRaw[, !NZV$nzv]
testing01 <- testRaw[, !NZV$nzv]

regex <- grepl("^X|timestamp|user_name", names(training01))
training <- training01[, !regex]
testing <- testing01[, !regex]
```

```
cond <- (colSums(is.na(training)) == 0)
training <- training[, cond]
testing <- testing[, cond]

dim(training)

## [1] 19622    54

dim(testing)

## [1] 20 54
```

Split

```
set.seed(9573644)
inTrain <- createDataPartition(training$classe, p = 0.70, list = FALSE)
validation <- training[-inTrain, ]
training <- training[inTrain, ]

dim(training)

## [1] 13737    54

dim(validation)

## [1] 5885   54
```

APPROACH

TWO PREDICTIVE MODELS ARE RESAMPLED WITH CROSS VALIDATION TO PROVIDE ACCURACY & OUT OF SAMPLE ERROR RATES

PREDICTION MODEL 1: Recursive Partitioning and Regression Tree (rpart)

0.0538 accuracy/ 0.462 out-of-sample error

Train the model: for the Training data set, find accuracy for rpart with Model 1.

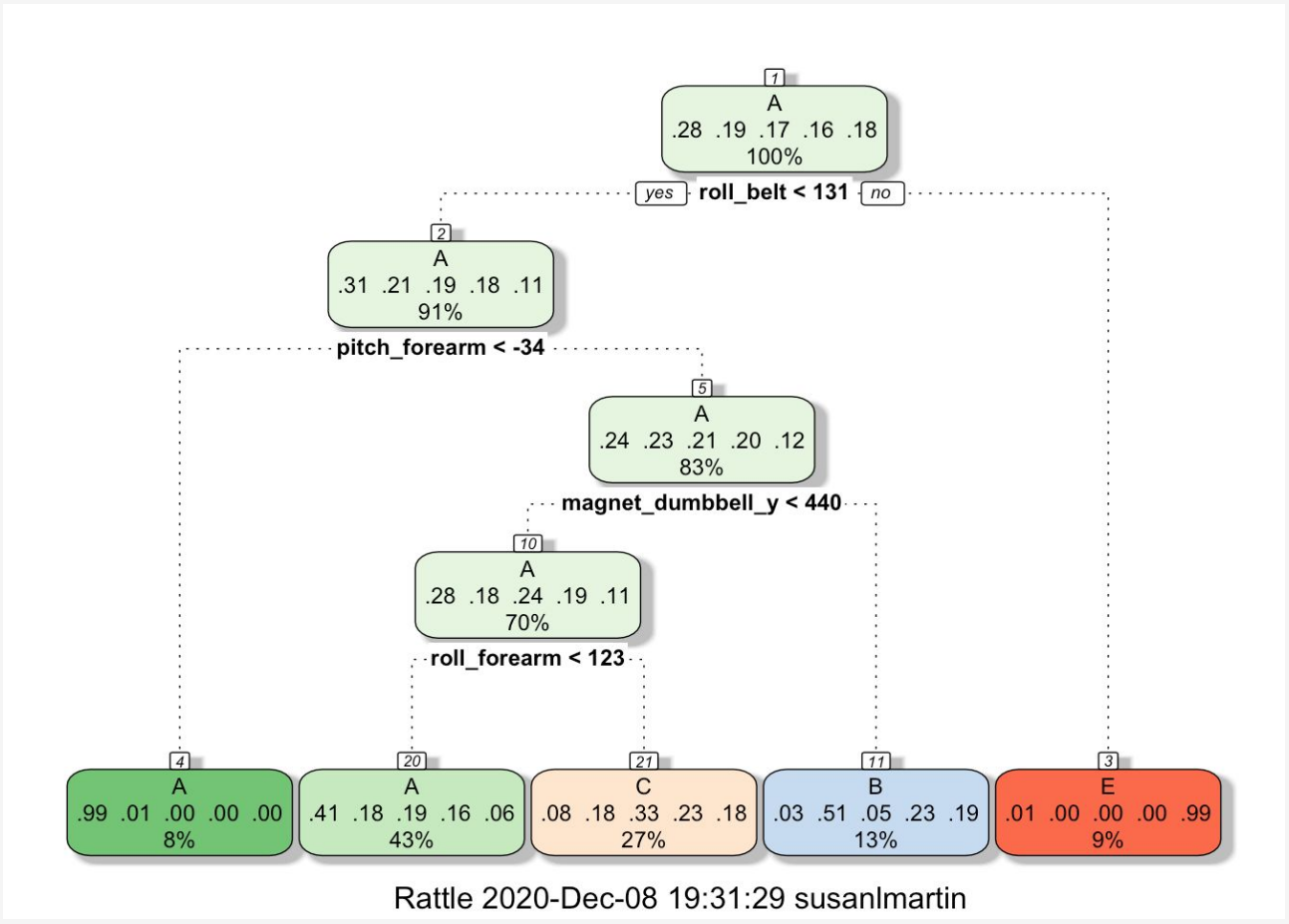
The resampling method ‘cv’ (cross validation) is used to train with the trainControl function. A re-sampling method involves repeatedly drawing samples from a training data set and refitting a

model to obtain additional information about that model.

```
con_trol <- trainControl(method = "cv", number = 5)
model1_rpart <- train(classe ~ ., data = training, method = "rpart", trControl = con_trol)
print(model1_rpart, digits = 4)

## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03901  0.5381    0.40609
##   0.05978  0.3928    0.16952
##   0.11830  0.3167    0.04917
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03901.

fancyRpartPlot(model1_rpart$finalModel)
```



PREDICTION MODEL 2: Random Forest

0.993 accuracy/ 0.007 out-of-sample error

Train the model: for the Training data set, find accuracy of the Random Forest model 2.

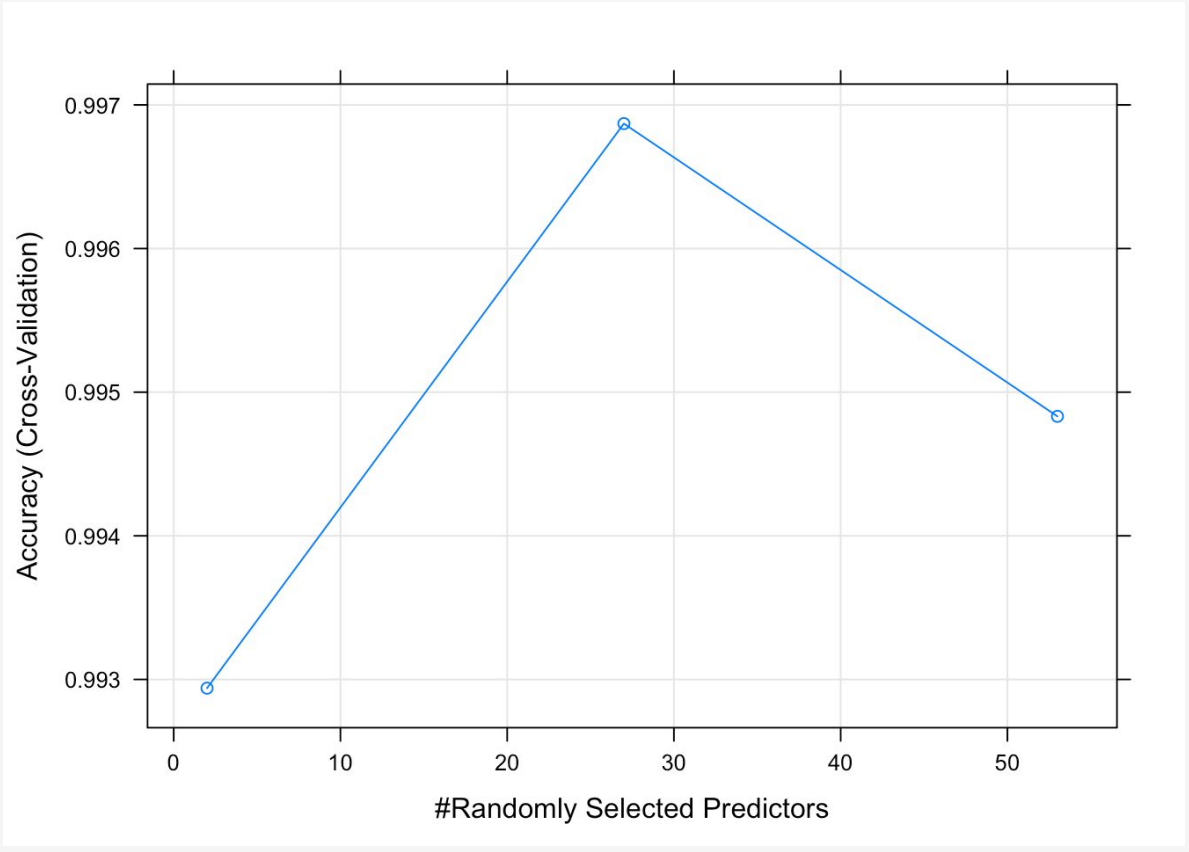
The resampling method ‘cv’ (cross validation) is used again here to train with the trainControl function. As a resampling procedure, cross validation is used to evaluate machine learning models on a limited data sample. It uses a limited sample in order to estimate how the model is expected to perform in general. It is then used to make predictions on data not used during the training of the model.

```
modelRF <- train(classe ~ ., data = training, method = "rf", trControl = trainControl(method = "cv", 5), ntree = 251)
modelRF

## Random Forest
```

```
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10988, 10991, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9929393  0.9910682
##   27    0.9968699  0.9960406
##   53    0.9948318  0.9934618
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.

plot(modelRF, log="y")
```



CONCLUSION

As these models indicate, the performance of the Random Forest (RF) method is superior to that of the Recursive Partitioning and Regression Trees (rpart) method. This is useful information, as recursion is generally known to be slow yet accurate. Our outcome shows that the accuracy of the RF model was 0.993 compared to 0.538 of the rpart Model. The expected out-of-sample error for the RF model is estimated at 0.007, or 0.7%. In contrast, and the recursive rpart Model shows an out-of-sample error rate of 0.462 or 46%. Our outcome is not conclusive, but interesting.

PREDICTION

The following prediction is our research outcome. It is based on the Prediction Model 2 (Random Forest) and is applied against our test data.

```
x <- testing

answers <- predict(modelRF, newdata=x)
answers

##  [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
```

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.