Machine Learning Project

suszanna

11/25/2020

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.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
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(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

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

```
set.seed(101)
```

DATA

LOAD, CLEAN & SPLIT

```
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

```
dim(training)
```

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

```
dim(testing)
```

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

```
table(training$classe)
```

```
## 
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

DATA: remove variables with missing values from the training set

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

#x <- training$classe
#x[1:10]; x[5581:5591]; x[9379:9389]; x[12817:12827]; x[16021:16031]; tail(x)
table(training$classe)
```

```
## 
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

DATA: remove irrelevant predictors

```
trainData <- training[, -c(1:7)]
testData <- testing[, -c(1:7)]

dim(trainData)
```

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

```
dim(testData)
```

```
## [1] 20 53
```

```
table(trainData$classe)
```

```
## 
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

```
table(testData$classe)
```

```
## < table of extent 0 >
```

DATA: split data for cross validation

In order to get out-of-sample errors, we split the cleaned training set trainData into a training set (train, 70%) for prediction and a validation set (validate, 30%) to be used to assess model performance.

```
set.seed(7826)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
train <- trainData[inTrain, ]
validate <- trainData[-inTrain, ]

dim(train)
```

```
## [1] 13737    53
```

```
dim(validate)
```

```
## [1] 5885   53
```

```
table(train$classe)
```

```
##
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

```
table(validate$classe)
```

```
##
##    A    B    C    D    E
## 1674 1139 1026  964 1082
```

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)

0.0526 accuracy/ 0.9470 out-of-sample error.

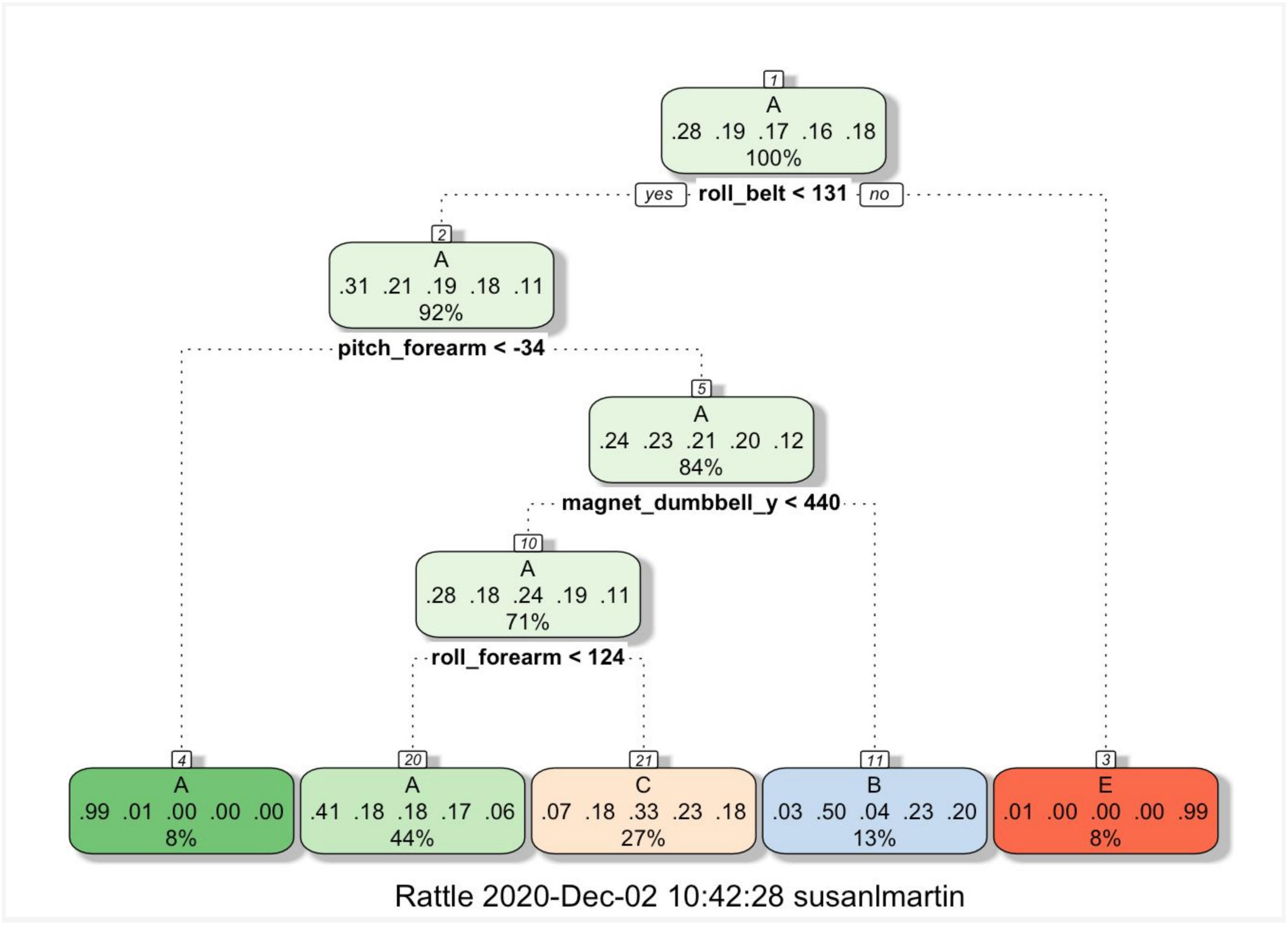
Train the model: for the Training data set, find accuracy for rpart with Model 1.

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 = train, method = "rpart",
                   trControl = con_trol)
print(model1_rpart, digits = 4)

## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10990, 10989, 10989
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03102  0.5260    0.38038
##   0.05954  0.3935    0.16982
##   0.11586  0.3168    0.04946
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03102.

fancyRpartPlot(model1_rpart$finalModel)
```

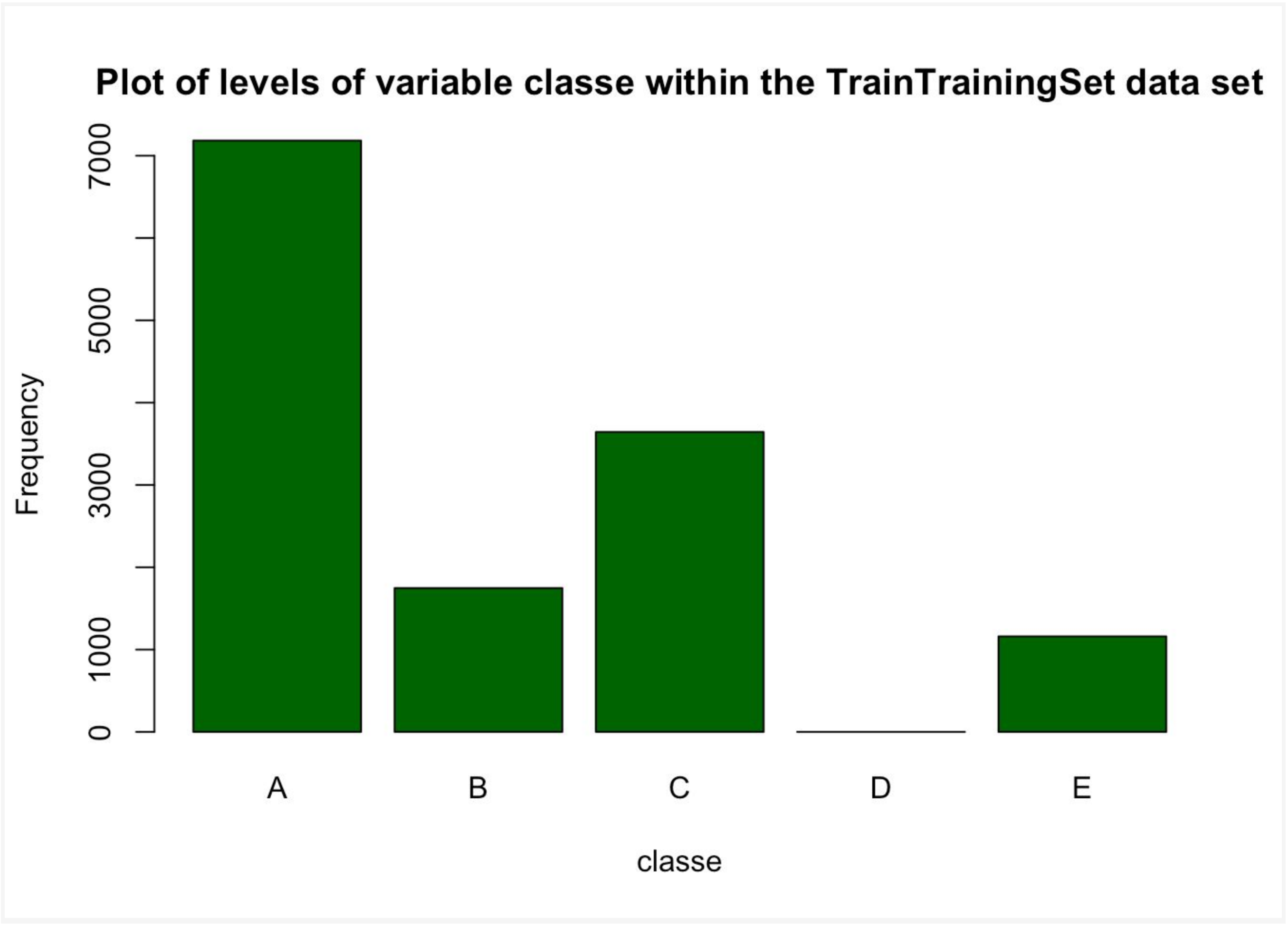


Support Model 1a: Levels of classe variable clearly rendered

The bar plot is a simple visual of the distribution of the training set

```
model1_predict_rpart <- predict(model1_rpart, train)

plot(model1_predict_rpart, ylim=c(0,7000), col="darkgreen", main="Plot of levels of variable classe within
the TrainTrainingSet data set",  xlab="classe", ylab="Frequency")
```



PREDICTION MODEL 2: Random Forest

0.991 accuracy/ 0.009 out-of-sample error.

Train the model: for the Training data set, find accuracy of randomForest model 2

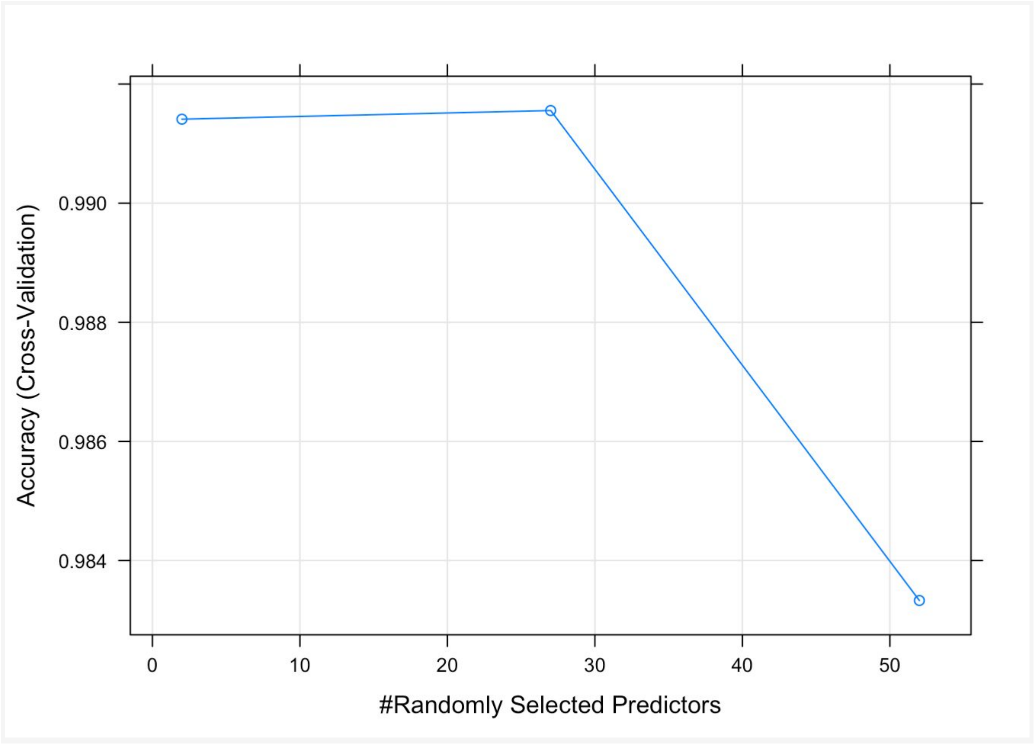
The resampling method ‘cv’ (cross validation) is used again here to train with the trainControl function. As a resampling procedure, cross validation is used to evaluate machine learning models on a limited data sample. It uses a limited sample in order to estimate how the model is expected to perform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

```
cont_rol <- trainControl(method="cv", 5)
model2_rf <- train(classe ~ ., data=train, method="rf",
          trControl=cont_rol, ntree=251)
```

```
model2_rf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10990, 10990, 10988, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9914101  0.9891329
##   27    0.9915555  0.9893175
##   52    0.9833292  0.9789102
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
plot(model2_rf, 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 0.991 compared to 0.526 of the rpart Model. The expected out-of-sample error for the RF model is estimated at 0.009, or 0.9%. In contrast, and the recursive rpart Model shows an out-of-sample error rate of 0.9470 or 95%. 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 <- testData

research_outcome <- predict(model2_rf, newdata=x)
research_outcome

##  [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.