Predicting Exercise Quality

Sarthak Dasadia

July 30, 2016

Introduction

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.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.

Goal

The goal of this project is to predict the manner in which the participants did the exercise. This is the “classe” variable of the training set, which classifies the correct and incorrect outcomes into A, B, C, D, and E categories. This report describes how the model for the project was built, its cross validation, expected out of sample error calculation, and the choices made. It was used successfully to accurately predict all 20 different test cases on the Coursera website.

More information is available from the website http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Getting and Cleaning Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

```
## Load Training and Test Data set

training <- read.csv("training.csv",na.strings = c("NA", "#DIV/0!", ""))
test <- read.csv("test.csv",na.strings = c("NA", "#DIV/0!", ""))


dim(training) # Dimension of the training data set
```

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

```
table(training$classe)  # Total no cases for each classe
```

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

```
table(training$user_name)  # Total no of users
```

```
##
##   adelmo carlitos  charles   eurico   jeremy    pedro
##     3892     3112     3536     3070     3402     2610
```

```
# In the data sets, 1-6 colums are just information. Let's exclude that from training an
d test data sets.

training <- training[, 7:160]
test <- test[, 7:160]

# Next step is to remove non a number values from both data sets.

removeNA <- apply(!is.na(training), 2, sum) > 19621 # apply sum on each column, pick tho
se columns for which sum > 19621 (no of rows)
training <- training[,removeNA]  # subset those columns
test <- test[,removeNA]
```

Building Prediction Model

1. Correlation Analysis

```
# Load required libraries

library(caret)
library(randomForest)
library(rattle)

# Let's build validation data set out of training data

inValid <- createDataPartition(y = training$classe, p = 0.7, list = F)
training <- training[inValid,]
valid <- training[-inValid,]
```

We filtered essential data and now ready to apply machine learning algorithms.

```
# PCA, lets check if any variable are correlated.

M <- abs(cor(training[,-54]))
diag(M) <- 0
nrow(which(M > 0.8, arr.ind=T))
```

```
## [1] 30
```

This suggests 38 varibles are strongly correlated. We can reduce the dimension (and noise) by performing Principal Component Analysis (PCA).

2. Principal Component Analysis (PCA) with Radom Forest Model

```
# The proceduce is following :

#preProc <- preProcess(training[,-54], method ="pca", pcaComp = 5)
#trainPC <- predict(preProc,training[,-54]) # Apply preProc to training data
#modelFit <- train(training$classe ~ ., method = "rf", data = trainPC) # Fit a model

#validPC <- predict(preProc,valid[,-54])  # Apply model to validation data set
#confusionMatrix(valid$classe, predict(modelFit, validPC)) ## 100% accurate in valid set


## Alternatively, we can use tain command in caret library,

modelFit1 <- train(training$classe ~ ., method = "rf", preProcess = "pca", data = traini
ng)
confusionMatrix(valid$classe, predict(modelFit1, valid))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1141    0    0    0    0
##          B    0  839    0    0    0
##          C    0    0  706    0    0
##          D    0    0    0  678    0
##          E    0    0    0    0  769
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2761
##     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    1.000   1.0000    1.000   1.0000
## Specificity            1.0000    1.000   1.0000    1.000   1.0000
## Pos Pred Value         1.0000    1.000   1.0000    1.000   1.0000
## Neg Pred Value         1.0000    1.000   1.0000    1.000   1.0000
## Prevalence             0.2761    0.203   0.1708    0.164   0.1861
## Detection Rate         0.2761    0.203   0.1708    0.164   0.1861
## Detection Prevalence   0.2761    0.203   0.1708    0.164   0.1861
## Balanced Accuracy      1.0000    1.000   1.0000    1.000   1.0000
```

The PCA is 100% accurate on training and validation data sets. Notice the 95% confidence interval!

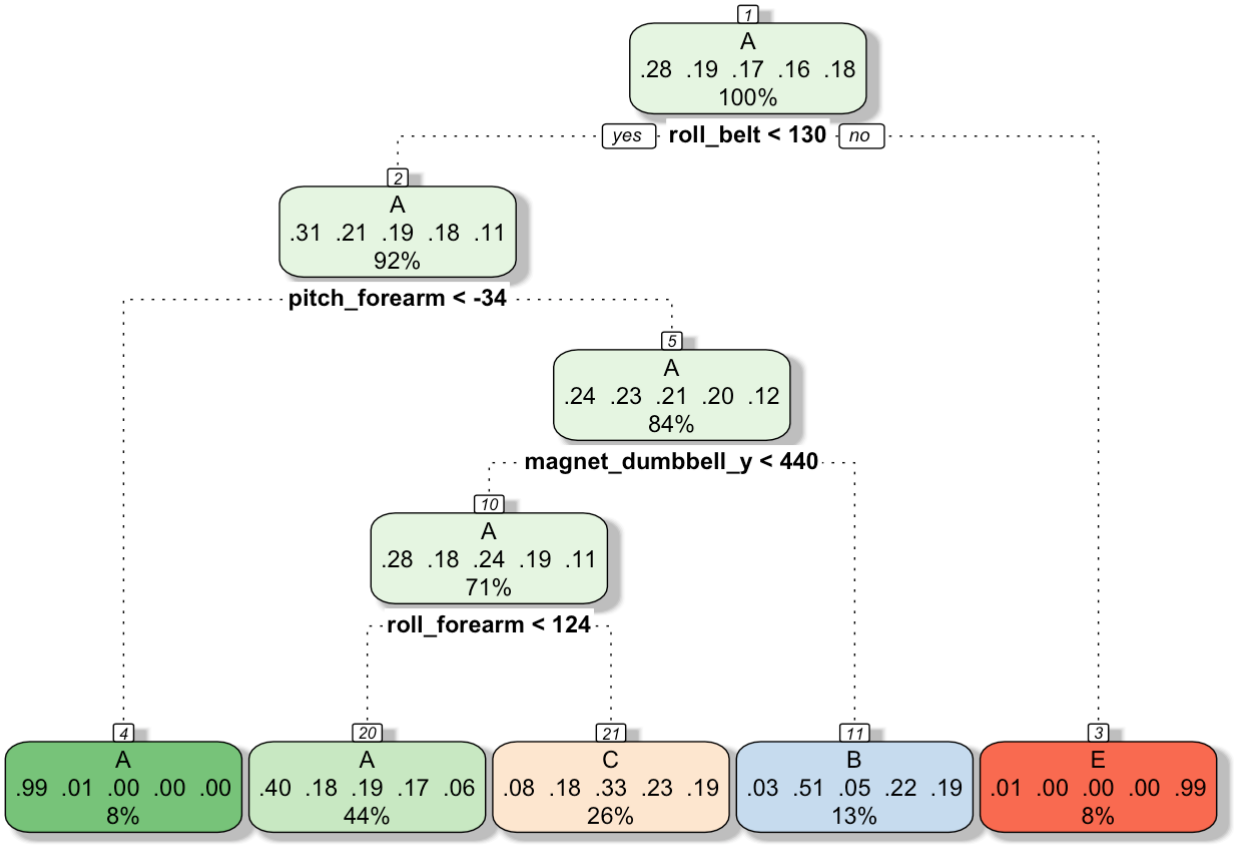
3. DeDecision Trees

Let's try decision tree method.

```
modelFit2 <- train(classe ~ ., method = "rpart", data = training)
confusionMatrix(valid$classe, predict(modelFit2, valid))  ## Not a reliable model
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1044   24   71    0    2
##          B  356  271  212    0    0
##          C  342   25  339    0    0
##          D  326  111  241    0    0
##          E  107  105  217    0  340
##
## Overall Statistics
##
##                Accuracy : 0.4825
##                  95% CI : (0.4671, 0.4978)
##     No Information Rate : 0.5263
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3264
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4800  0.50560  0.31389       NA  0.99415
## Specificity            0.9505  0.84209  0.87979    0.836  0.88684
## Pos Pred Value         0.9150  0.32300  0.48017       NA  0.44213
## Neg Pred Value         0.6220  0.91955  0.78378       NA  0.99941
## Prevalence             0.5263  0.12969  0.26131    0.000  0.08275
## Detection Rate         0.2526  0.06557  0.08202    0.000  0.08226
## Detection Prevalence   0.2761  0.20300  0.17082    0.164  0.18606
## Balanced Accuracy      0.7152  0.67384  0.59684       NA  0.94049
```

```
fancyRpartPlot(modelFit2$finalModel) # Plot decision tree
```



Rattle 2016-Jul-30 16:39:37 sarthakdasadia

As shown by the confussion matrix, this is not a reliable method. Thus, let's predict test data set with Radom Forest method described above.

Applying the Selected Model to the Test Data

In this case, the Random Forest model will be applied to predict the 20 test data sets.

```
pred_PCA <- predict(modelFit1, newdata = test)
pred_PCA
```

```
##  [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
```

The model **correctly** predicted all 20 data points from the test data set (verified by coursera).