PML - Quality of Weightlifting Analysis

Sahil

4/11/2021

OVERVIEW:

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.

The goal of this project is to predict the manner in which they did the exercise. This is the `classe` variable in the training set.

Data set description

The outcome variable is `classe`, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- 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

Loading the Data and Required Packages

The initial configuration consists of loading some required packages and initializing some variables.

```
#Data Files
training.url    <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url  <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'



#Directories
if (!file.exists("data")){
  dir.create("data")
}
if (!file.exists("data/submission")){
  dir.create("data/submission")
```

```r
}


#R-Packages
InstallRpart<- require("rpart")
```

```
## Loading required package: rpart
```

```r
if(!InstallRpart){
    install.packages("rpart")
    library("rpart")
    }
InstallRpartPlot <- require("rpart.plot")
```

```
## Loading required package: rpart.plot
```

```r
if(!InstallRpartPlot){
    install.packages("rpart.plot")
    library("rpart.plot")
    }
InstallCaret <- require("caret")
```

```
## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
if(!InstallCaret){
    install.packages("caret")
    library("caret")
    }
InstallRF <- require("randomForest")
```

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

```r
if(!InstallRF){
    install.packages("randomForest")
    library("randomForest")
    }
Installggplot2 <- require("ggplot2")
if(!Installggplot2){
    install.packages("ggplot2")
    library("ggplot2")
}

InstallLattice <- require("lattice")
if(!InstallLattice){
    install.packages("lattice")
    library("lattice")
}

# Set seed for reproducability
set.seed(1234)
```

Download & Clean Data:

```r
#download
training.file   <- './data/pml-training.csv'
test.cases.file <- './data/pml-testing.csv'
download.file(training.url, training.file)
download.file(test.cases.url,test.cases.file )

#clean
training <-read.csv(training.file, na.strings=c("NA","#DIV/0!", ""))
testing <-read.csv(test.cases.file , na.strings=c("NA", "#DIV/0!", ""))
clean_training_set<-training[,colSums(is.na(training)) == 0]
clean_testing_set <-testing[,colSums(is.na(testing)) == 0]

# Remove unnecessary columns (first 7 cols)
clean_training_set<-clean_training_set[,-c(1:7)]
clean_testing_set  <-clean_testing_set [,-c(1:7)]
```

Brief Exploratory Analysis of Training Data set

```r
str(clean_training_set)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
```

```
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

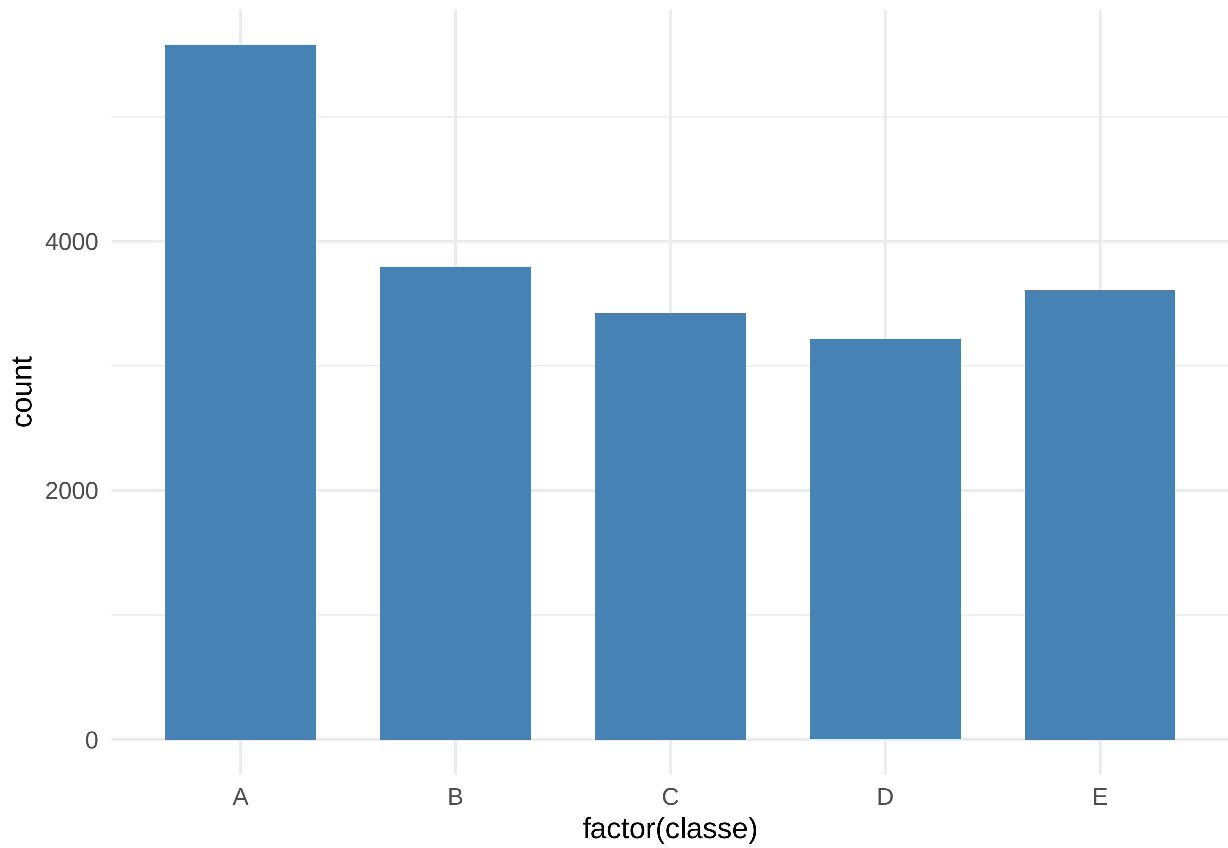
```
colSums(is.na(clean_training_set))
```

```
##            roll_belt           pitch_belt             yaw_belt
##                    0                    0                    0
##     total_accel_belt         gyros_belt_x         gyros_belt_y
##                    0                    0                    0
##         gyros_belt_z         accel_belt_x         accel_belt_y
```

```
##                    0                    0                    0
##          accel_belt_z        magnet_belt_x        magnet_belt_y
##                    0                    0                    0
##         magnet_belt_z             roll_arm            pitch_arm
##                    0                    0                    0
##               yaw_arm      total_accel_arm          gyros_arm_x
##                    0                    0                    0
##           gyros_arm_y          gyros_arm_z          accel_arm_x
##                    0                    0                    0
##           accel_arm_y          accel_arm_z         magnet_arm_x
##                    0                    0                    0
##          magnet_arm_y         magnet_arm_z        roll_dumbbell
##                    0                    0                    0
##        pitch_dumbbell         yaw_dumbbell total_accel_dumbbell
##                    0                    0                    0
##      gyros_dumbbell_x     gyros_dumbbell_y     gyros_dumbbell_z
##                    0                    0                    0
##      accel_dumbbell_x     accel_dumbbell_y     accel_dumbbell_z
##                    0                    0                    0
##     magnet_dumbbell_x    magnet_dumbbell_y    magnet_dumbbell_z
##                    0                    0                    0
##          roll_forearm        pitch_forearm          yaw_forearm
##                    0                    0                    0
##   total_accel_forearm      gyros_forearm_x      gyros_forearm_y
##                    0                    0                    0
##       gyros_forearm_z      accel_forearm_x      accel_forearm_y
##                    0                    0                    0
##       accel_forearm_z     magnet_forearm_x     magnet_forearm_y
##                    0                    0                    0
##      magnet_forearm_z               classe
##                    0                    0
```

```
# as.factor(clean_training_set$classe)

ggplot(clean_training_set, aes(x=factor(classe)))+
  geom_bar(stat="count", width=0.7, fill="steelblue")+
  theme_minimal()
```



The training data set has a total of 19622 obs.of 53 variables. We have also verified that there are no missing/NA/DIV0 values in the set

The plot indicates that class A (Proper repetitions) are the most frequent outcome in the data set. The least frequent outcome is D.

Cross-validation

We will use cross-validation by splitting the cleaned training data into a (sub)training (75%) and (sub)testing (25%) data sets.

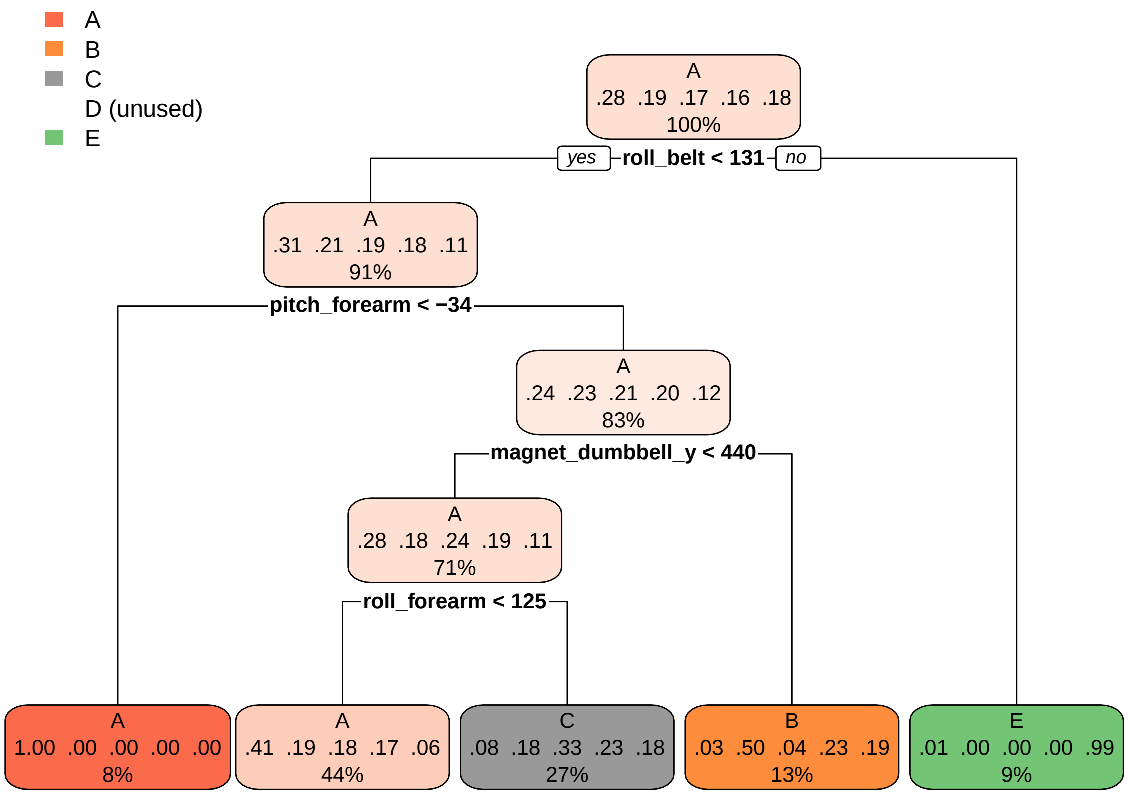
```
subSamples <- createDataPartition(y=clean_training_set$classe, p=0.75, list=FALSE)
subTraining <- clean_training_set[subSamples, ]
subTesting <- clean_training_set[-subSamples, ]
```

Applying models and prediction

Here we will apply two different models and compare the outcome of predictions - 1. Decision Tree 2. Random Forests

Decision tree

```r
# Fit model
decisionTreeMod <- train(classe ~., method='rpart', data=subTraining)
# Perform prediction
predictDT <- predict(decisionTreeMod, subTesting)
# Plot result
rpart.plot(decisionTreeMod$finalModel)
```



RESULTS: DECISION TREE Following confusion matrix shows the errors of the Decision tree prediction algorithm.

```r
subTest.factor <- as.factor(subTesting$classe)
confusionMatrix(subTest.factor, predictDT)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1274   18  100    0    3
##          B  390  344  215    0    0
##          C  401   36  418    0    0
##          D  386  131  287    0    0
##          E  141  127  244    0  389
##
## Overall Statistics
##
##                Accuracy : 0.4945
```

```
##                  95% CI : (0.4804, 0.5086)
##     No Information Rate : 0.5285
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3385
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4915  0.52439  0.33070       NA  0.99235
## Specificity            0.9477  0.85758  0.87995   0.8361  0.88652
## Pos Pred Value         0.9133  0.36249  0.48889       NA  0.43174
## Neg Pred Value         0.6244  0.92111  0.79106       NA  0.99925
## Prevalence             0.5285  0.13377  0.25775   0.0000  0.07993
## Detection Rate         0.2598  0.07015  0.08524   0.0000  0.07932
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7196  0.69099  0.60532       NA  0.93944
```

The overall accuracy of prediction on the testing (portion of training set used for cross validation - not the main "PLM-Testing" data set) was very poor - 0.5.

Let us try results of another model such as Random Forests

Random Forests

```
# Fit model
RandomForestMod <- train(classe ~., method='rf', data=subTraining, ntree = 64)
# Perform prediction
predictRF <- predict(RandomForestMod, subTesting)
```

```
#Evaluate
confusionMatrix(subTest.factor, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    0    1    0    0
##          B    9  938    1    1    0
##          C    0    5  848    2    0
##          D    0    0    5  799    0
##          E    0    1    0    2  898
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9936   0.9936   0.9918   0.9938   1.0000
## Specificity            0.9997   0.9972   0.9983   0.9988   0.9993
## Pos Pred Value         0.9993   0.9884   0.9918   0.9938   0.9967
## Neg Pred Value         0.9974   0.9985   0.9983   0.9988   1.0000
## Prevalence             0.2861   0.1925   0.1743   0.1639   0.1831
## Detection Rate         0.2843   0.1913   0.1729   0.1629   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9966   0.9954   0.9950   0.9963   0.9996
```

RESULTS: RANDOM FORESTS

The accuracy is dramatically improved - 0.99+ suggesting that this model is much better to predict the desired outcome from the vairables given.

Out of Sample Error Rate

The expected out-of-sample error is estimated at 0.0075, or 0.75%. The expected OoS error is calculated as 1 - accuracy for predictions made against the cross-validation set.

Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified.

Conclusion

We trained two different models - Decision Tree and Random Forests - on the given training data set (PML-Training) and utilized cross validation with a partition of p = 0.75.

The results of both models indicate that the accuracy Random Forest based prediction far exceeds that of Decision Tree, achieving 99%+ and 50% respectively.