Practical Machine Learning Project: Prediction Assignment

Andrés Miniguano Trujillo

7/30/2020

Introduction

This document aims to predict how 6 participants performed their exercise routines This is stored in the `classe` variable. The machine learning algorithm described here is applied to the 20 test cases available in the test data.

Reading data

```
theURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
download.file(theURL, 'pml-training.csv', mode = 'wb')

theURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
download.file(theURL, 'pml-testing.csv', mode = 'wb')

pml.training <- read.csv('pml-training.csv')
pml.testing  <- read.csv('pml-testing.csv')

summary(factor(pml.training$classe))
```

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

According to [1], the datasets store the following information:

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class `A`), throwing the elbows to the front (Class `B`), lifting the dumbbell only halfway (Class `C`), lowering the dumbbell only halfway (Class `D`) and throwing the hips to the front (Class `E`).

Class `A` corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Exploratory data analysis

If we ran `str(pml.training)`, we would find that there are several variables that are empty or constant. As a result, we will take these out.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

```r
summary(nearZeroVar(pml.training, saveMetrics = T))
```

```
##    freqRatio        percentUnique       zeroVar           nzv
##  Min.   :   1.000   Min.   :  0.01019   Mode :logical   Mode :logical
##  1st Qu.:   1.067   1st Qu.:  0.78101   FALSE:160       FALSE:100
##  Median :   3.003   Median :  1.65121                   TRUE :60
##  Mean   : 279.422   Mean   :  4.93767
##  3rd Qu.:  77.000   3rd Qu.:  2.03598
##  Max.   :9608.000   Max.   :100.00000
```

First, let's remove these nearly constant items:

```r
inBuild      <- nearZeroVar(pml.training)
pml.training <- pml.training[, -inBuild]
pml.testing  <- pml.testing[ , -inBuild]
```

Now let's take care of the series of `NA`'s:

```r
inBuild      <- which((sapply(pml.training, function(x) mean(is.na(x))) > 0.9) == F)
pml.training <- pml.training[, inBuild]
pml.testing  <- pml.testing[ , inBuild]
```

This operations have drastically reduced the datasets, narrowing the number of variables to 59. That is almost three parts of the dataset wouldn't contribute to the overall prediction.

Now, let's remove the digital footprint of the participants. These are the first 5 series, which count the number of observations, identify the user, record a timestamp, etc.

```r
inBuild      <- c(1:5)
pml.training <- pml.training[, -inBuild]
pml.testing  <- pml.testing[ , -inBuild]
```

At last, if we ran `summary(pml.training)`, it would be seen that the only variable left that stores character data is `classe`. We proceed to store this variables as a factor.

```r
pml.training$classe <- factor(pml.training$classe)
```

Partitioning data

After tidying the datasets, we divide training data into two subsets, one of 60% for training and the last 40% for testing. This will be useful for cross validation.

```r
set.seed(4234)
inTrain    <- createDataPartition( y = pml.training$classe, p = .6, list = F )

training   <- pml.training[ inTrain,]
testing    <- pml.training[-inTrain,]
```

There are 11776 observations in `training` and 7846 measurements in `testing`, respectively.

Prediction models

We will use two models to predict the `classe` variable: (1) a decision tree and (2) a random forest.

Decision tree

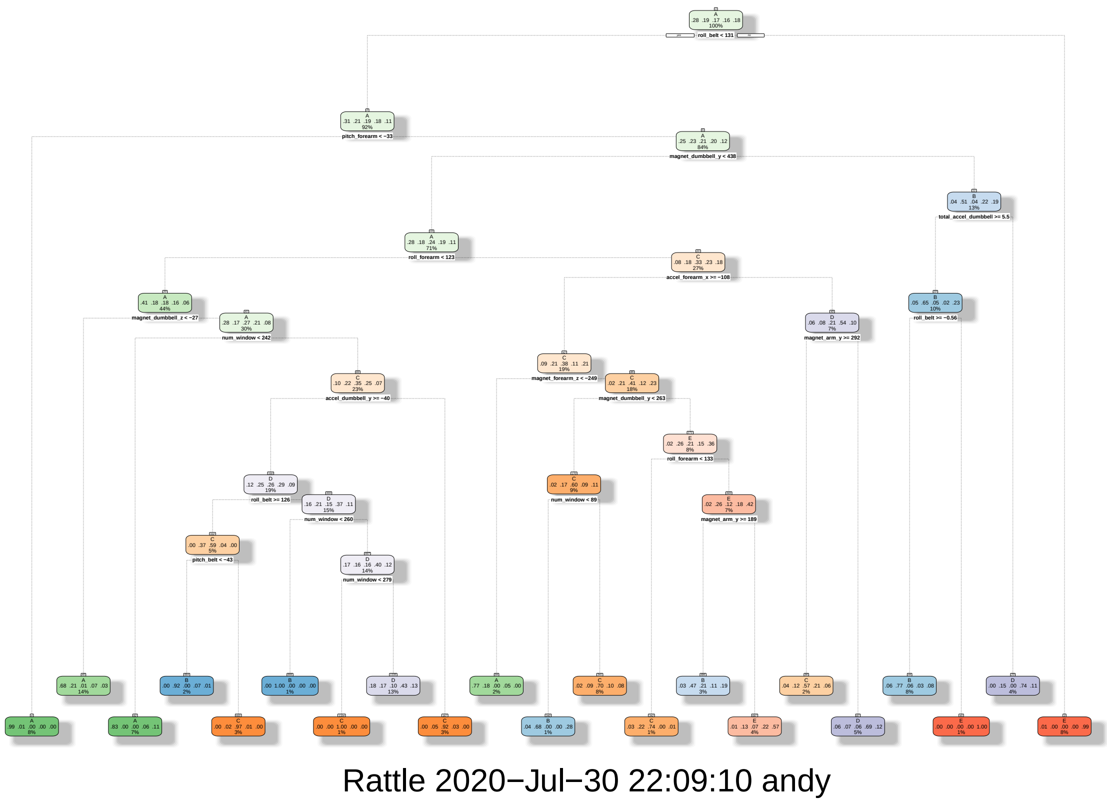
Let's build the first model and take a look at the generated tree:

```r
library(rpart); library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.0.2
```

```r
modFit_1 <- rpart(classe ~. , method = 'class', data = training)
fancyRpartPlot( modFit_1 )
```

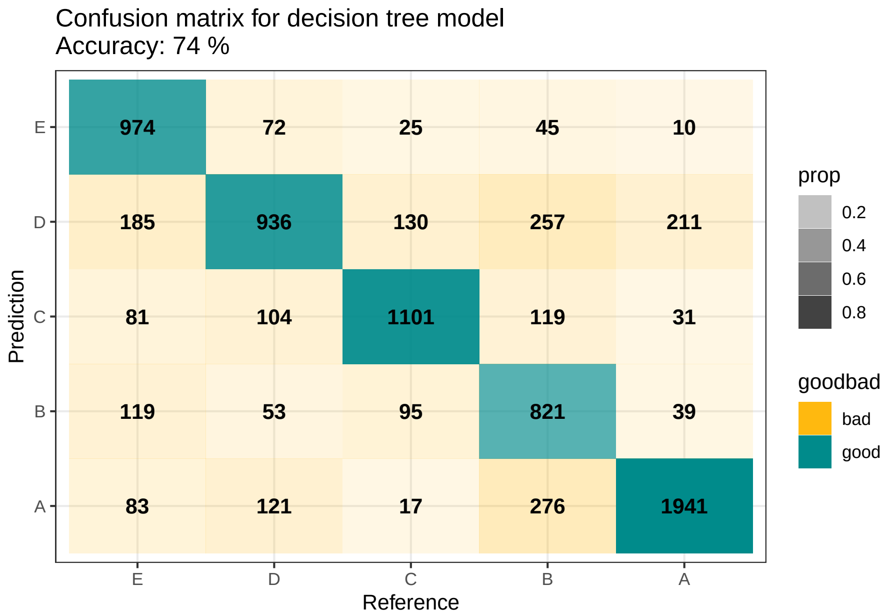
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



After this, we can check the confusion matrix and the accuracy of this model. For this, we will use the code suggested in [2].

```r
pred_1 <- predict(modFit_1, newdata = testing, type = 'class')
conf_1 <- confusionMatrix(pred_1, testing$classe)
titl_1 <- paste('Confusion matrix for decision tree model \nAccuracy:',
                as.character( round(100*conf_1$overall[['Accuracy']]) ), '%' )

library(ggplot2)
library(dplyr)

table <- data.frame(conf_1$table)

plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, 'good', 'bad')) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))

# fill alpha relative to sensitivity/specificity by proportional outcomes within reference groups
ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad, alpha = prop)) +
```

```r
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = 'bold', alpha = 1) +
  scale_fill_manual(values = c(good = 'cyan4', bad = 'darkgoldenrod1')) +
  theme_bw() + xlim(rev(levels(table$Reference))) + ggtitle(titl_1)
```



As we can see, the accuracy of the model is low.

Random forest

Now we switch to the next model. Here, using the routines explained during the course would take too much time to process. As a result, we will use a parallel approach.

```r
library(randomForest);  library(doParallel)
```

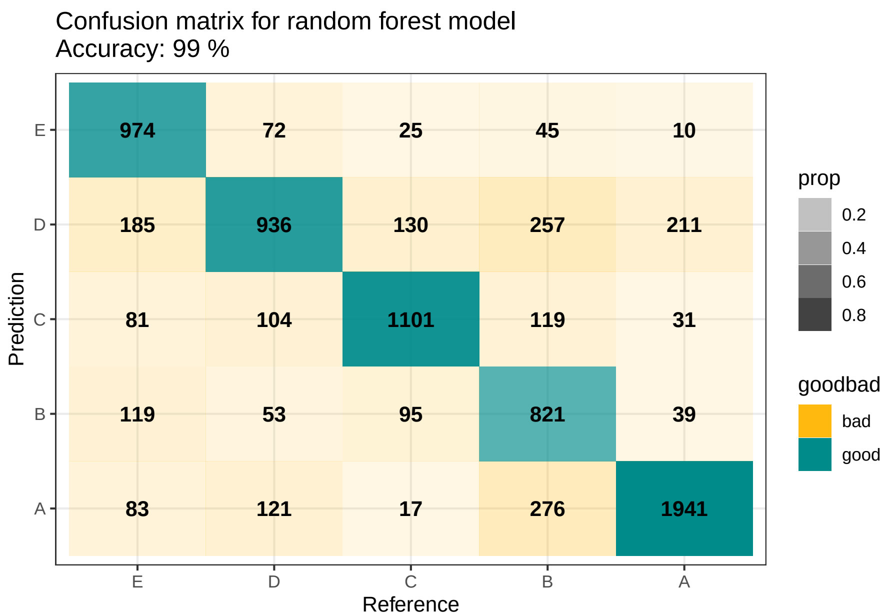
```
## Warning: package 'randomForest' was built under R version 4.0.2

## Warning: package 'doParallel' was built under R version 4.0.2

## Warning: package 'foreach' was built under R version 4.0.2

## Warning: package 'iterators' was built under R version 4.0.2
```

```r
registerDoParallel()
modFit_2 <- train(classe ~ ., data = training, method = 'parRF',
             tuneGrid   = data.frame(mtry=3),
             trControl  = trainControl(method="none") )
```

Let's check the accuracy of this model:

```r
pred_2 <- predict(modFit_2, newdata = testing)
conf_2 <- confusionMatrix(pred_2, testing$classe)
titl_2 <- paste('Confusion matrix for random forest model \nAccuracy:',
               as.character( round(100*conf_2$overall[['Accuracy']]) ), '%' )


table  <- data.frame(conf_1$table)

plotTable <- table %>%
```

```
  mutate(goodbad = ifelse(table$Prediction == table$Reference, 'good', 'bad')) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))

ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad, alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = 'bold', alpha = 1) +
  scale_fill_manual(values = c(good = 'cyan4', bad = 'darkgoldenrod1')) +
  theme_bw() + xlim(rev(levels(table$Reference))) + ggtitle(titl_2)
```



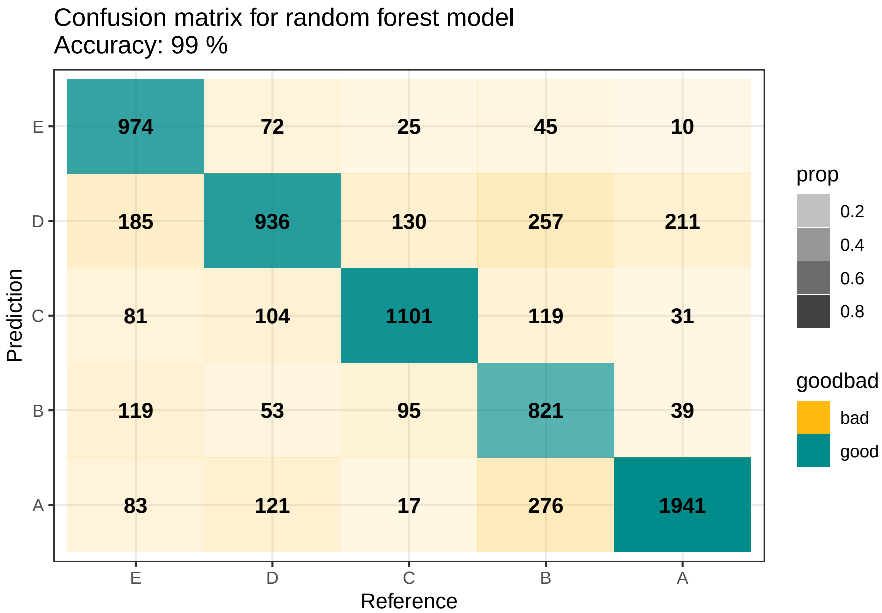
We see a clear improvement in accuracy.

Combining models

Now let's fit a model that combines the two approaches.

```
predDF     <- data.frame(pred_1, pred_2, classe = testing$classe)

registerDoParallel()
combModFit <- train(classe ~ ., data = predDF, method = 'parRF',
                  tuneGrid   = data.frame(mtry=3),
                  trControl  = trainControl(method="none") )

combPred   <- predict(combModFit, newdata = testing )

conf_3 <- confusionMatrix(combPred, testing$classe)
titl_3 <- paste('Confusion matrix for random forest model \nAccuracy:',
              as.character( round(100*conf_3$overall[['Accuracy']]) ), '%' )


table  <- data.frame(conf_1$table)

plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, 'good', 'bad')) %>%
  group_by(Reference) %>%
```

```
  mutate(prop = Freq/sum(Freq))

ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad, alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = 'bold', alpha = 1) +
  scale_fill_manual(values = c(good = 'cyan4', bad = 'darkgoldenrod1')) +
  theme_bw() + xlim(rev(levels(table$Reference))) + ggtitle(titl_3)
```



We see that the combined model captures the accuracy of the former model.

Final prediction

We will use the last combined model to get a final prediction.

```
pred1V    <- predict(modFit_1, pml.testing, type = 'class')
pred2V    <- predict(modFit_2, pml.testing)
predVDF   <- data.frame(pred_1 = pred1V, pred_2 = pred2V)
combPredV <- predict(combModFit, predVDF)
combPredV
```

```
##  [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
```

References

[1] Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. *Qualitative Activity Recognition of Weight Lifting Exercises.* Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013. URL: http://groupware.les.inf.puc-rio.br/har#ixzz6TjFWJeXO

[2] davedgd. *Answer to: Plot confusion matrix in R using ggplot by Haroon Rashid.* URL: https://stackoverflow.com/questions/37897252/plot-confusion-matrix-in-r-using-ggplot