Practical Machine Learning Course

O. Zamora

18/9/2021

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

Exploratory Analysis

```
# environment
library(caret)
library(rpart.plot)
library(rattle)
```

Data preparation Load training and test data source. The training dataset is splited in 70% of the data training and 30% for testing

```
trainData <- read.csv('./pml-training.csv', header=T)
testData <- read.csv('./pml-testing.csv', header=T)

set.seed(3433)

inTrain <-createDataPartition(trainData$classe, p=0.7, list=FALSE)
training <- trainData[inTrain,]
testing <- trainData[-inTrain,]
dim(training)
```

```
## [1] 13737   160
```

```
dim(testing)
```

```
## [1] 5885  160
```

Cleaning data sets (remove NA values) and the Near Zero Variance variables are removed. With this reduce the number of variables

```
# near zero variance
NZV <- nearZeroVar(training)
training <- training[, -NZV]
testing  <- testing[, -NZV]
dim(training)
```

```
## [1] 13737   106
```

```
dim(testing)
```

```
## [1] 5885  106
```

```
# remove NA
withNA <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, withNA==FALSE]
testing <- testing[, withNA==FALSE]

# remove identification
training <- training[, -(1:5)]
testing  <- testing[, -(1:5)]

dim(training)
```

```
## [1] 13737    54
```

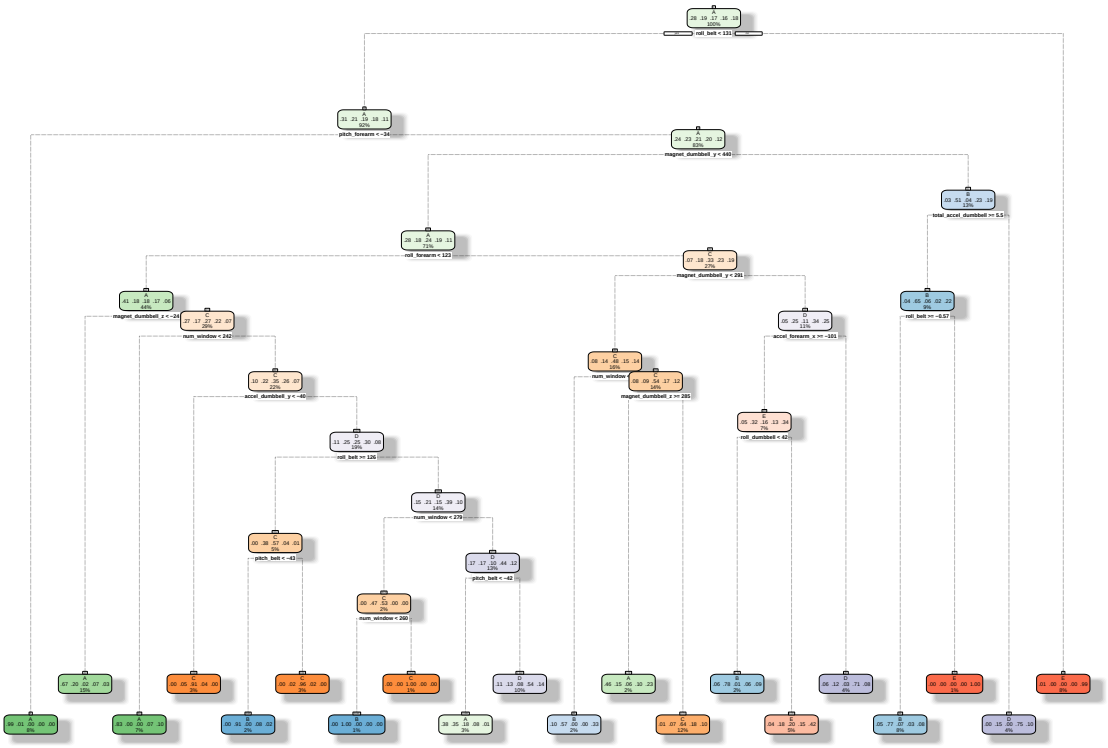
```
dim(testing)
```

```
## [1] 5885   54
```

Decision Tree Model

```
# model fit
set.seed(3433)
modFitTree <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(modFitTree)
```

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



Rattle 2021−sep.−21 08:27:13 ozamora

```
# predicting with decision tree model
predictTree <- predict(modFitTree, newdata=testing, type="class")
confusionMatrix(table(predictTree, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predictTree    A    B    C    D    E
##           A 1504  273   49  121   94
##           B   43  600   36   22  103
##           C   13   54  822  138   58
##           D   87  142   54  616  134
##           E   27   70   65   67  693
##
## Overall Statistics
##
##                Accuracy : 0.7196
##                  95% CI : (0.708, 0.7311)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6429
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8984   0.5268   0.8012   0.6390   0.6405
## Specificity            0.8725   0.9570   0.9459   0.9153   0.9523
## Pos Pred Value         0.7369   0.7463   0.7576   0.5963   0.7516
## Neg Pred Value         0.9558   0.8939   0.9575   0.9283   0.9216
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2556   0.1020   0.1397   0.1047   0.1178
## Detection Prevalence   0.3468   0.1366   0.1844   0.1755   0.1567
## Balanced Accuracy      0.8855   0.7419   0.8735   0.7771   0.7964
```

Random Forest Model

```
# model fit
set.seed(3433)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitForest <- train(classe ~ ., data=training, method="rf", trControl=controlRF)
modFitForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.26%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    8 2648    2    0    0 0.0037622272
## C    0    5 2390    1    0 0.0025041736
## D    0    0   14 2238    0 0.0062166963
## E    0    0    0    4 2521 0.0015841584
```

```
# predicting with random forest model
predictForest <- predict(modFitForest, newdata=testing)
confusionMatrix(table(predictForest, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predictForest    A    B    C    D    E
##             A 1674    1    0    0    0
##             B    0 1133    1    0    0
##             C    0    4 1025    5    0
##             D    0    1    0  958    2
##             E    0    0    0    1 1080
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9975
##                  95% CI : (0.9958, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9990   0.9938   0.9982
## Specificity            0.9998   0.9998   0.9981   0.9994   0.9998
## Pos Pred Value         0.9994   0.9991   0.9913   0.9969   0.9991
## Neg Pred Value         1.0000   0.9987   0.9998   0.9988   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1742   0.1628   0.1835
## Detection Prevalence   0.2846   0.1927   0.1757   0.1633   0.1837
## Balanced Accuracy      0.9999   0.9973   0.9986   0.9966   0.9990
```

Applying models to testing data

From de confusion matrix we see that Random Forest Model es very accurate, about 99%, while Decision Tree Model 72%.