Practical Machine Learning

Don Resnik

August 12, 2016

Background

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).

Data

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Load Data

Load the trainging and test data from the URLs listed above. Do some preliminary cleaning to normalize the null and empty values.

```r
set.seed(1234)
trainingUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
trainingData <- fread(trainingUrl, na.strings=c("", "NA", "#DIV/0!"))

testingUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testingData <- fread(testingUrl, na.strings=c("", "NA", "#DIV/0!"))
trainingData <- as.data.frame(trainingData)
testingData <- as.data.frame(testingData)

trainingData$classe <- as.factor(trainingData$classe)
```

Data Cleaning

The training and test data is now cleaned using the following steps: 1. Remove all columns with no values 2. Remove the columns with non-scored values (columns 1-7)

The training data is then split 75/25 into training and validation data sets

```r
dim(trainingData)
```

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

```
dim(testingData)
```

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

```
# Remove columns with all missing values
trainingData<-trainingData[,colSums(is.na(trainingData)) == 0]
testingData<-testingData[,colSums(is.na(testingData)) == 0]
dim(trainingData)
```

```
## [1] 19622    60
```

```
dim(testingData)
```

```
## [1] 20 60
```

```
# Remove columns with labels or other non-comparative data
trainingData <-trainingData[,-c(1:7)]
testingData <-testingData[,-c(1:7)]

dim(trainingData)
```

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

```
dim(testingData)
```

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

```
#create a validation set from the training set
trainingPartition <- createDataPartition(y=trainingData$classe, p = 0.75, list = FALSE)
trainingDataSplit <- trainingData[trainingPartition, ]
validationDataSplit <- trainingData[-trainingPartition, ]
```

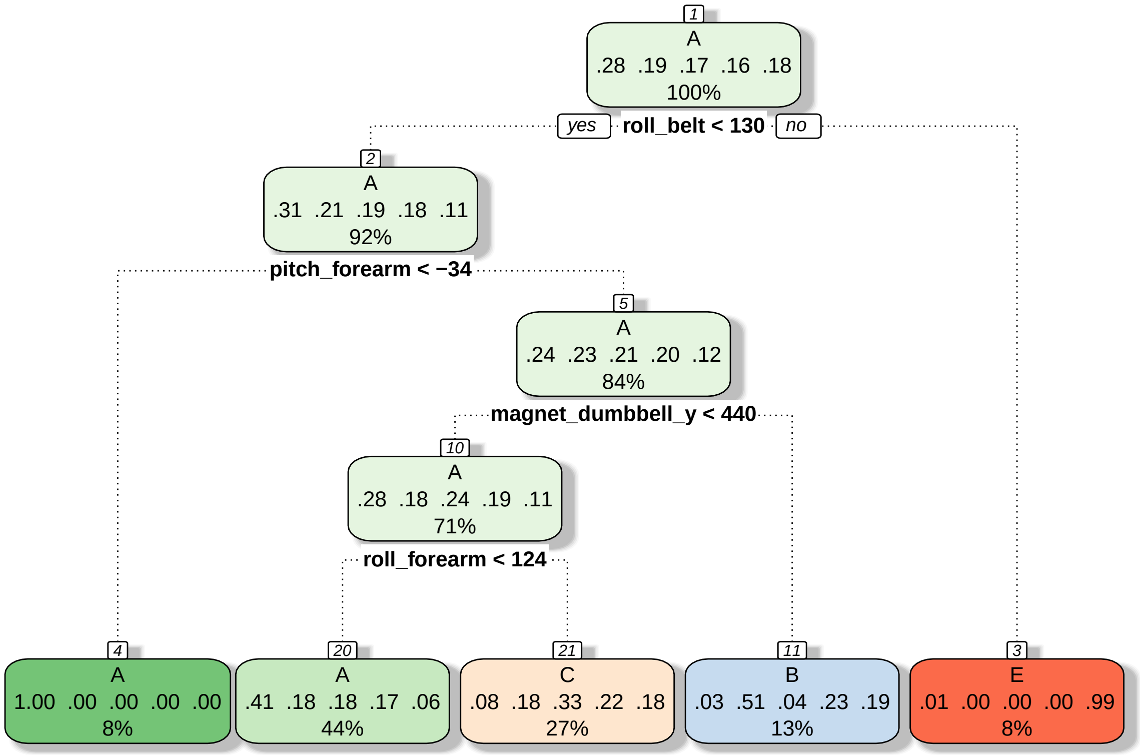
Model Creation

RPart

```
modFit <- train(classe~., data=trainingDataSplit,method="rpart")
print(modFit)
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03446312  0.5208194  0.38176529
##   0.05987531  0.4004313  0.18424306
##   0.11535175  0.3349994  0.07855886
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03446312.
```

```
fancyRpartPlot(modFit$finalModel)
```



Rattle 2016−Aug−12 21:22:05 VandelayMacBookAir

Random Forest

```
randomForestModel <- randomForest(classe~., data=trainingDataSplit)
predictionTrainingDataSplit <- predict(randomForestModel, validationDataSplit)
print(confusionMatrix(predictionTrainingDataSplit, validationDataSplit$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  946   11    0    0
##          C    0    1  843    6    0
##          D    0    0    1  798    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9860   0.9925   1.0000
## Specificity            0.9994   0.9972   0.9983   0.9998   1.0000
## Pos Pred Value         0.9986   0.9885   0.9918   0.9987   1.0000
## Neg Pred Value         1.0000   0.9992   0.9970   0.9985   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1719   0.1627   0.1837
## Detection Prevalence   0.2849   0.1951   0.1733   0.1629   0.1837
## Balanced Accuracy      0.9997   0.9970   0.9921   0.9961   1.0000
```

Run prediction with Test Data

```
predictionTestData <- predict(randomForestModel, testingData)
table(predictionTestData,testingData$problem_id)
```

```
##
## predictionTestData 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##                  A 0 1 0 1 1 0 0 0 1  1  0  0  0  1  0  0  1  0  0  0
##                  B 1 0 1 0 0 0 0 1 0  0  1  0  1  0  0  0  0  1  1  1
##                  C 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  0  0  0  0  0
##                  D 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0
##                  E 0 0 0 0 0 1 0 0 0  0  0  0  0  0  1  1  0  0  0  0
```

```
predictionTestData
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```