Machine Learning Course Project

Jackie O'Dwyer

Monday, July 25, 2016

Practical Machine Learning Course Project

Executive Summary

In the course project for Practical Machine Learning we will investigate the data set of accelerometers for particular activities. We will focus on predicting the manner in which exercise was performed.

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

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 Load and Package Load

```
Train <- read.csv('pml-training.csv', na.strings=c("NA","#DIV/0!",""))
Test <- read.csv('pml-testing.csv', na.strings=c("NA","#DIV/0!",""))

library(caret)
```

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

```
## Loading required package: ggplot2
```

```
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Split Up Training and Testing Data

```
inTrain <- createDataPartition(y=Train$classe, p=0.6, list=FALSE)
myTrain <- Train[inTrain, ]
myTest <- Train[-inTrain, ]
dim(myTrain)
```

```
## [1] 11776   160
```

```
#Remove first column of IDs to avoid use in models
myTrain <- myTrain[c(-1)]

# Remove Variables with too many NAs. With a threshold of 70%
trainingV1 <- myTrain #creating another subset to iterate in loop
for(i in 1:length(myTrain)) { #for every column in the training dataset
        if( sum( is.na( myTrain[, i] ) ) /nrow(myTrain) >= .7 ) { #if n?? NAs > 70% of total observatio
        for(j in 1:length(trainingV1)) {
            if( length( grep(names(myTrain[i]), names(trainingV1)[j]) ) ==1)  { #if the columns are the
                trainingV1 <- trainingV1[ , -j] #Remove that column
            }
        }
    }
}
#To check the new # of observations
dim(trainingV1)
```

```
## [1] 11776    59
```

```
# Run same for Testing data
myTest <- myTest[c(-1)]
dim(myTest)
```

```
## [1] 7846  159
```

```
# Remove Variables with too many NAs. With a threshold of 70%
testingV1 <- myTest #creating another subset to iterate in loop
for(i in 1:length(myTest)) { #for every column in the training dataset
        if( sum( is.na( myTest[, i] ) ) /nrow(myTest) >= .7 ) { #if n?? NAs > 70% of total observations
        for(j in 1:length(testingV1)) {
            if( length( grep(names(myTest[i]), names(testingV1)[j]) ) ==1)  { #if the columns are the s
                testingV1 <- testingV1[ , -j] #Remove that column
            }
        }
    }
}
#To check the new # of observations
dim(testingV1)
```

```
## [1] 7846   59
```

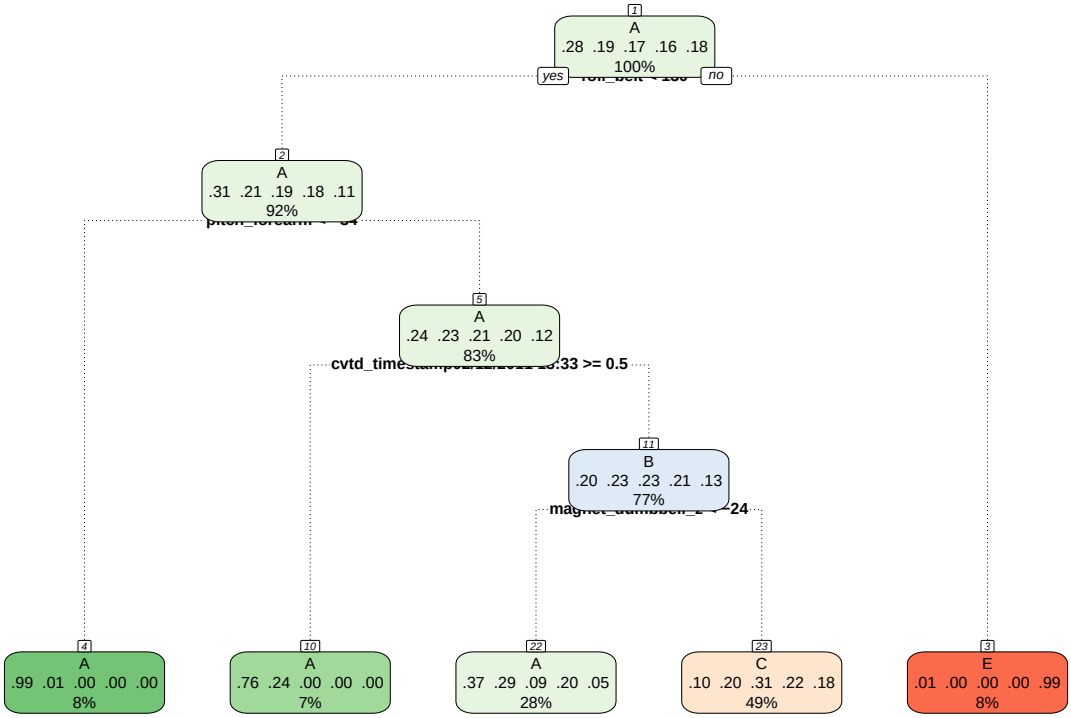
Model Building and Selection

```
# rpart model testing
set.seed(500)
rpartModFit<-train(classe~.,method="rpart", data=trainingV1)

print(rpartModFit$finalModel)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 10788 7449 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.15 956    7 A (0.99 0.0073 0 0 0) *
##      5) pitch_forearm>=-34.15 9832 7442 A (0.24 0.23 0.21 0.2 0.12)
##       10) cvtd_timestamp02/12/2011 13:33>=0.5 773  189 A (0.76 0.24 0 0 0) *
##       11) cvtd_timestamp02/12/2011 13:33< 0.5 9059 6976 B (0.2 0.23 0.23 0.21 0.13)
##         22) magnet_dumbbell_z< -24.5 3307 2078 A (0.37 0.29 0.09 0.2 0.054) *
##         23) magnet_dumbbell_z>=-24.5 5752 3994 C (0.1 0.2 0.31 0.22 0.18) *
##    3) roll_belt>=130.5 988    9 E (0.0091 0 0 0 0.99) *
```

```
fancyRpartPlot(rpartModFit$finalModel,cex=.5,under.cex=1,shadow.offset=0)
```



Rattle 2016−Aug−03 13:36:05 jaodwyer

```
classepredict <- predict(rpartModFit,testingV1)
confusionMatrix(testingV1$classe,classepredict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1823    0  404    0    5
##          B  775    0  743    0    0
##          C  223    0 1145    0    0
##          D  470    0  816    0    0
##          E  109    0  681    0  652
##
## Overall Statistics
##
##                Accuracy : 0.4614
##                  95% CI : (0.4503, 0.4725)
##     No Information Rate : 0.4829
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.3069
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5362       NA   0.3022       NA  0.99239
## Specificity            0.9080   0.8065   0.9450   0.8361  0.89011
## Pos Pred Value         0.8168       NA   0.8370       NA  0.45215
## Neg Pred Value         0.7191       NA   0.5918       NA  0.99922
## Prevalence             0.4333   0.0000   0.4829   0.0000  0.08374
## Detection Rate         0.2323   0.0000   0.1459   0.0000  0.08310
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639  0.18379
## Balanced Accuracy      0.7221       NA   0.6236       NA  0.94125
```

```
# This model is not very strong, as it has only a 55.4% accuracy.

# Random Forest Model Testing
set.seed(500)

RFmodFit <- randomForest(classe~., data = trainingV1)
print(RFmodFit)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingV1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.16%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    5 2274    0    0    0 0.0021939447
## C    0    2 2049    3    0 0.0024342746
## D    0    0    4 1924    2 0.0031088083
## E    0    0    0    2 2163 0.0009237875
```

```
RFclassepredict <- predict(RFmodFit,testingV1)
confusionMatrix(testingV1$classe,RFclassepredict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    1    0    0    0
##          B    1 1517    0    0    0
##          C    0    3 1362    3    0
##          D    0    0   10 1276    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9977
##                  95% CI : (0.9964, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9971
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9974   0.9927   0.9977   1.0000
## Specificity            0.9998   0.9998   0.9991   0.9985   1.0000
## Pos Pred Value         0.9996   0.9993   0.9956   0.9922   1.0000
## Neg Pred Value         0.9998   0.9994   0.9985   0.9995   1.0000
## Prevalence             0.2845   0.1939   0.1749   0.1630   0.1838
## Detection Rate         0.2843   0.1933   0.1736   0.1626   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9997   0.9986   0.9959   0.9981   1.0000
```

```
# The Random Forest Model is a much better predictor of the classe field. This model has a 99.9% accura
```