Practical Machine Learning - Course Project

David Jensen

July 17, 2017

Introduction

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.

Data Preprocessing

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
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(corrplot)
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.
```

Prepare the Data:

```r
#set the working directory where the data reside and read in the data
setwd("C:/Practical Machine Learning")
testFile <- read.csv("pml-testing.csv", header = TRUE)
trainFile <- read.csv("pml-training.csv", header=TRUE)
```

By using the str function (not shown here for brevity) it is noticed there are a large number of variables that contain nothing missing values and can be deleted from the datasets.

```r
trainfinal <- trainFile[, colSums(is.na(trainFile)) == 0]
testfinal <- testFile[, colSums(is.na(testFile)) == 0]
```

```r
classe <- trainfinal$classe
remtrain <- grepl("^X|timestamp|window", names(trainfinal))
trainfinal <- trainfinal[, !remtrain]
trainfinal2 <- trainfinal[, sapply(trainfinal, is.numeric)]
trainfinal2$classe <- classe
remtest <- grepl("^X|timestamp|window", names(testfinal))
testfinal <- testfinal[, !remtest]
testfinal2 <- testfinal[, sapply(testfinal, is.numeric)]
```

Prepare Data for Modeling

Split the data into training and testing datasets

```r
seed <- as.numeric(as.Date("2017-07-17"))
set.seed(seed)
inTrain <- createDataPartition(trainfinal2$classe, p=0.7, list=F)
traindata <- trainfinal2[inTrain, ]
testdata <- trainfinal2[-inTrain, ]
```

Data Modeling

Fitting a predictive model using random forest. This should be particularly appropriate as it selects important explanatory variables. Use 5-fold cross validation.

```r
library(e1071)
controltrain <- trainControl(method="cv", 5)
rfmodel1 <- train(classe ~ ., data=traindata, method="rf", trControl=controltrain, ntree=250)
rfmodel1
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10990, 10989, 10990, 10991
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.9905364  0.9880283
##   27    0.9913376  0.9890424
##   52    0.9863142  0.9826867
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Test the model on the test set

```
predictmod <- predict(rfmodel1, testdata)
confusionMatrix(testdata$classe, predictmod)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    1    2    0    0
##          B   10 1128    1    0    0
##          C    0    2 1016    8    0
##          D    0    1   15  948    0
##          E    0    0    0    5 1077
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9898, 0.9944)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9965   0.9826   0.9865   1.0000
## Specificity            0.9993   0.9977   0.9979   0.9968   0.9990
## Pos Pred Value         0.9982   0.9903   0.9903   0.9834   0.9954
## Neg Pred Value         0.9976   0.9992   0.9963   0.9974   1.0000
## Prevalence             0.2856   0.1924   0.1757   0.1633   0.1830
## Detection Rate         0.2839   0.1917   0.1726   0.1611   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9967   0.9971   0.9903   0.9916   0.9995
```

Do calculations for accuracy and out of sample error

```
modelaccurate <- postResample(predictmod, testdata$classe)
modelaccurate
```

```
##  Accuracy     Kappa
## 0.9923534 0.9903268
```

Accuracy of the model is estimated to be 99.24%

```
outsampleerror <- 1 - as.numeric(confusionMatrix(testdata$classe, predictmod)$overall[1])
outsampleerror
```

```
## [1] 0.007646559
```

So, the out-of-sample error is estimated to be .76%

Test Data Set Predictions

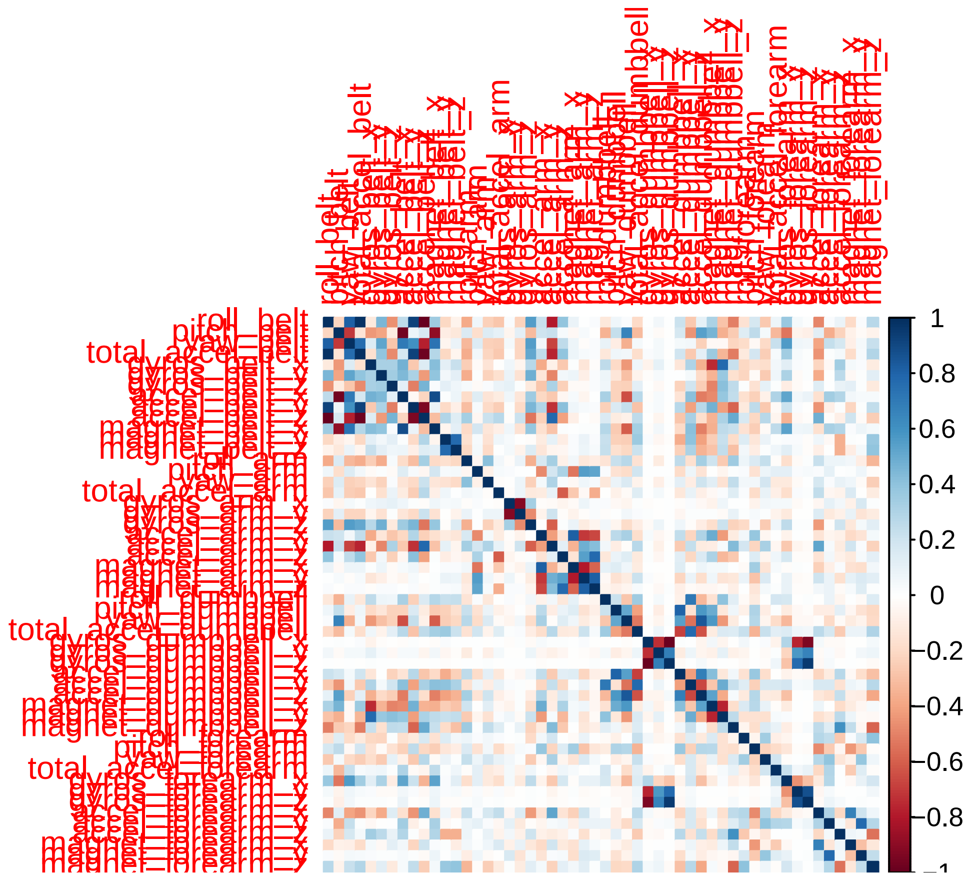
This section will make predictions as to what each of the observations in the test data set should be grouped into

```
result <- predict(rfmodel1, testfinal2[, -length(names(testfinal2))])
result
```

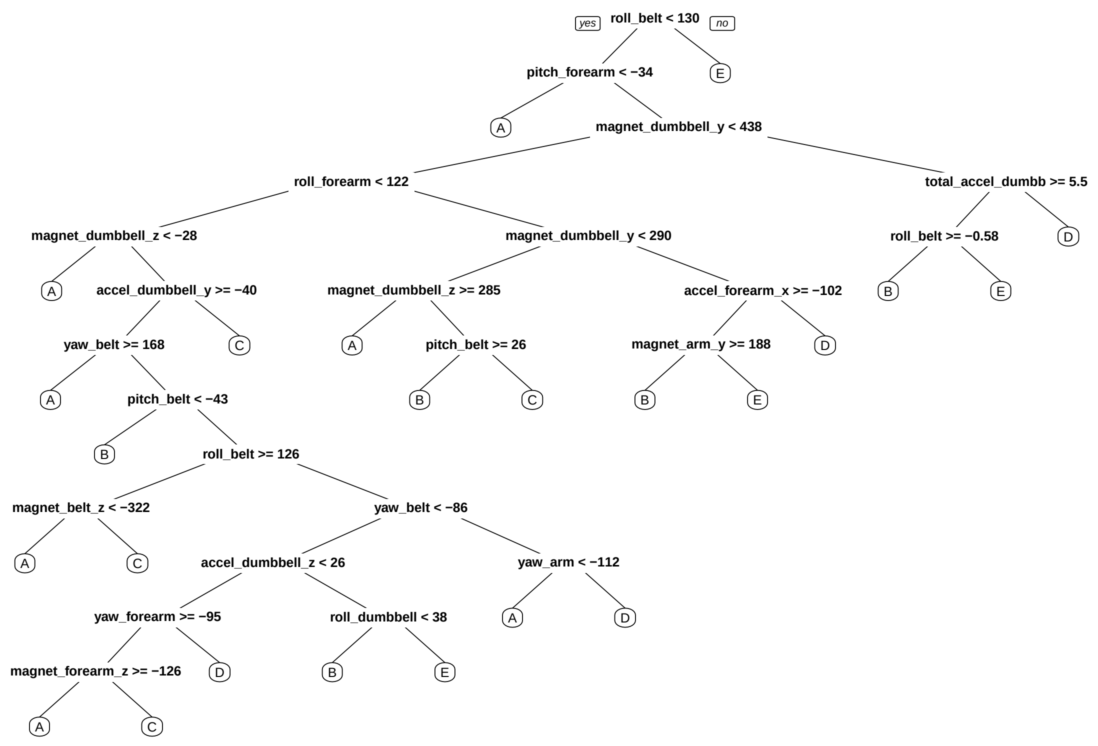
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Appendix (Figures):

```
cmatrix <- cor(traindata[, -length(names(traindata))])
corrplot(cmatrix, method="color")
```

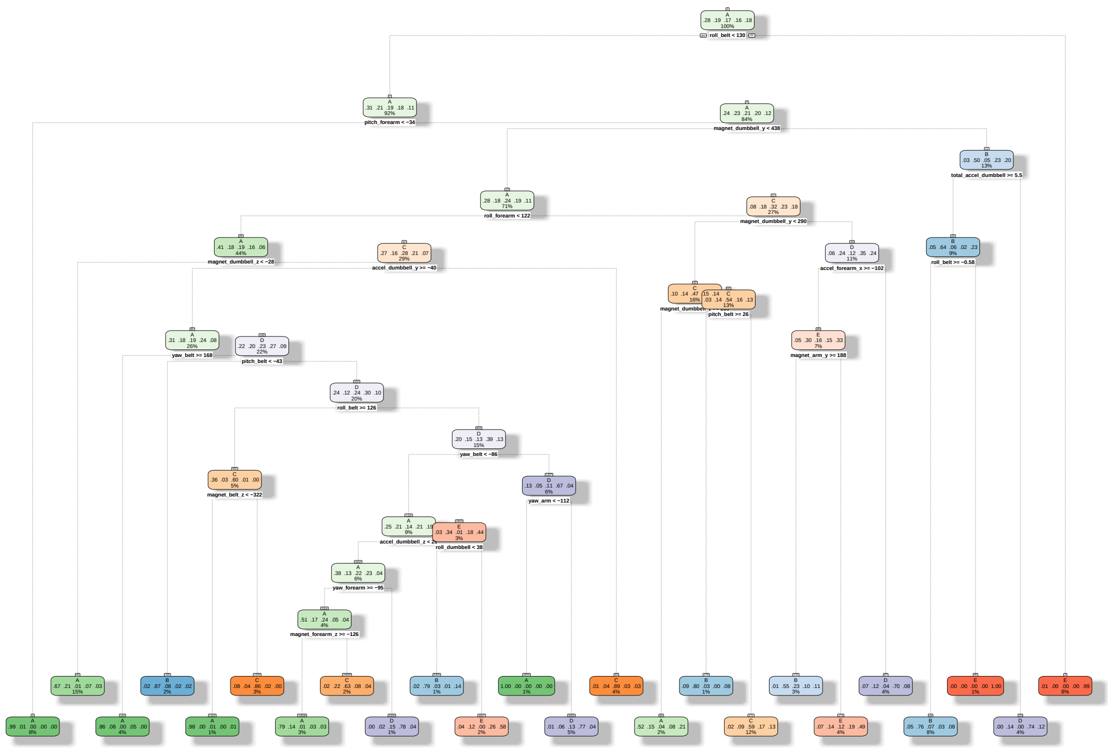


```
treediagram <- rpart(classe ~ ., data=traindata, method="class")
prp(treediagram)
```



```
fancyRpartPlot(treediagram)
```

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



Rattle 2017−Jul−17 23:42:33 David