Machine Learning Course Project

January 27, 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

For ease of processing files and simplification of code, download the csv files and save to the set working directory.

```
setwd("~/Desktop/coursera/MachineLearning")
```

Load the libraries we'll be using:

```
## Load the preferred libraries
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

```
library(rpart)
library(plotmo)
```

```
## Loading required package: plotrix
```

```
## Warning: package 'plotrix' was built under R version 3.2.3
```

```
## Loading required package: TeachingDemos
```

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(AppliedPredictiveModeling)
```

With the following code, we'll read the data, and take a quick look at the properties. I also took a quick look at the .csv file by opening the test version in Excel.

```
pmlTrain <- read.csv("pml-training.csv", header=TRUE, na.strings=c("NA","#Div/0!"))  ## The training set
pmlTest <- read.csv("pml-testing.csv", header=TRUE, na.string=c("NA", "#Div/0!")) ## The test set -set a
dim(pmlTrain)
```

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

```
dim(pmlTest)
```

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

```
summary(pmlTrain$classe)
```

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

In the training set, there are 19622 records with 159 variables (the first column is just a numeric count of observations). The "classe" variable we are solving for is divided among 5 classes.

The test set that was provided contains the exercise readings for 20 participants without the "classe" variable provided. We'll attempt to determine this "classe" with the use of predictive modelling, built using the training set. We'll set the test set to the side until the models are completed. At the end, we'll apply the same cleaning and transformations to that data, then apply our model.

Cleaning the data

We'll clean out the NearZeroVariance variables and remove them and the first column (a count of the observations) from our data as these will not contribute to the predictive model.

```
nzv <- nearZeroVar(pmlTrain, saveMetrics=TRUE)  ## remove nearZeroVariances
pmlTrain <- pmlTrain[,nzv$nzv==FALSE]
pmlTrain <- pmlTrain[c(-1)]  ## remove first column (count)
```

Remove observations with 75% NA:

```
noNAs<- pmlTrain ## find and remove 75% of NAs
for(i in 1:length(pmlTrain)) {
    if( sum( is.na( pmlTrain[, i] ) ) /nrow(pmlTrain) >= 0.75) {
        for(j in 1:length(noNAs)) {
            if( length( grep(names(pmlTrain[i]), names(noNAs)[j]) ) == 1)  {
```

```
                noNAs <- noNAs[ , -j]
            }
        }
    }
}
pmlTrain <- noNAs ## set back to name
rm(noNAs) ## remove excess data
dim(pmlTrain)
```

```
## [1] 19622    58
```

This brings us down to 58 columns.

Split data

Now we'll split the data into a 60/40 training/test set to train the model then test the model before using for our prediction on the 20 observations in the final set.

```
inTrain <- createDataPartition(y=pmlTrain$classe,p=.60,list=FALSE)
train <-pmlTrain[inTrain,]
test <- pmlTrain[-inTrain,]
```

```
dim(train)
```

```
## [1] 11776    58
```

```
dim(test)
```

```
## [1] 7846   58
```

Prediction with Random Forests

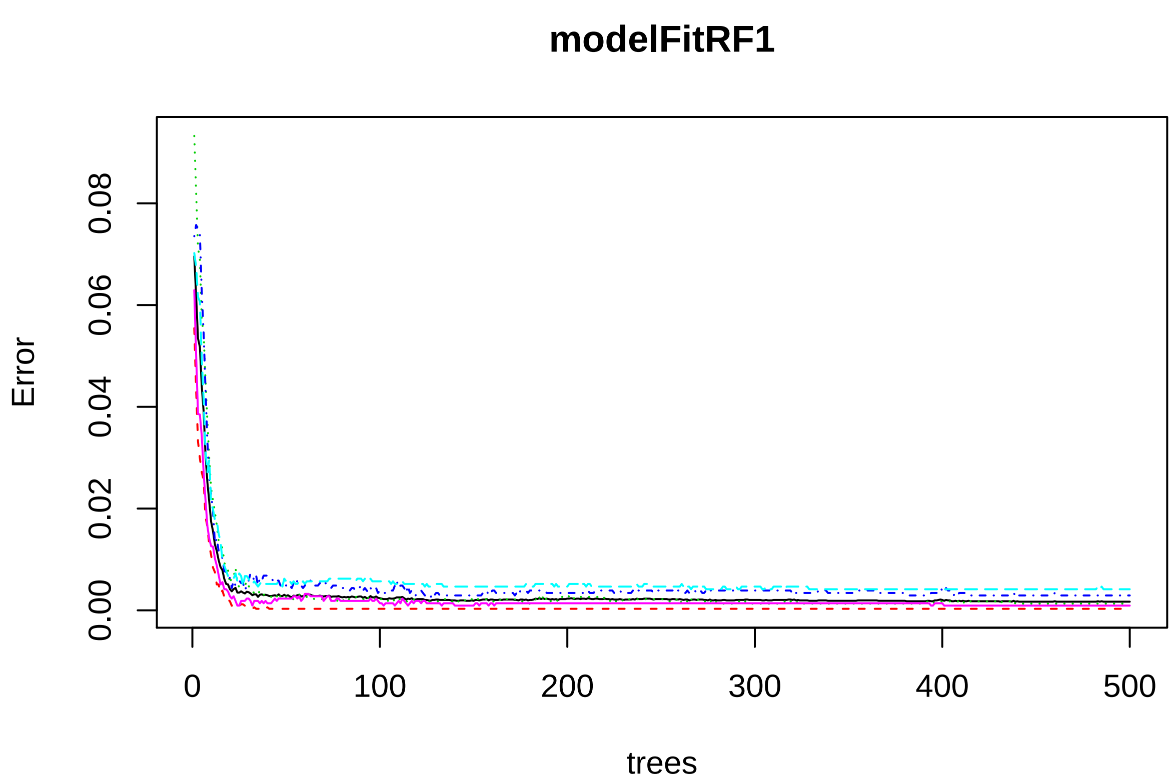
For Random Forest information

```
set.seed(4726)
modelFitRF1 <- randomForest(classe ~ ., data=train)
predictionRF1 <- predict(modelFitRF1, test, type = "class")
modelRF <- confusionMatrix(predictionRF1, test$classe)
modelRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    0    0    0
##          C    0    0 1365    2    0
##          D    0    0    3 1283    0
##          E    0    0    0    1 1442
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9991
##                  95% CI : (0.9982, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9978   0.9977   1.0000
## Specificity            0.9998   1.0000   0.9997   0.9995   0.9998
## Pos Pred Value         0.9996   1.0000   0.9985   0.9977   0.9993
## Neg Pred Value         1.0000   0.9998   0.9995   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1740   0.1635   0.1838
## Detection Prevalence   0.2846   0.1933   0.1742   0.1639   0.1839
## Balanced Accuracy      0.9999   0.9997   0.9987   0.9986   0.9999
```

```r
plot(modelFitRF1)
```




```r
importance(modelFitRF1)
```

```
##                      MeanDecreaseGini
```

```
## user_name                  83.44910
## raw_timestamp_part_1      955.04706
## raw_timestamp_part_2       10.27330
## cvtd_timestamp           1398.27820
## num_window                593.69649
## roll_belt                 541.82288
## pitch_belt                299.54879
## yaw_belt                  351.81616
## total_accel_belt          112.56807
## gyros_belt_x               36.44526
## gyros_belt_y               53.04001
## gyros_belt_z              119.58138
## accel_belt_x               64.96234
## accel_belt_y               66.89162
## accel_belt_z              192.83989
## magnet_belt_x             105.66752
## magnet_belt_y             190.87321
## magnet_belt_z             166.96206
## roll_arm                  121.17241
## pitch_arm                  56.48048
## yaw_arm                    73.75083
## total_accel_arm            31.03000
## gyros_arm_x                40.37999
## gyros_arm_y                45.21023
## gyros_arm_z                18.55010
## accel_arm_x               102.64520
## accel_arm_y                53.23422
## accel_arm_z                40.54220
## magnet_arm_x               89.34380
## magnet_arm_y               74.23590
## magnet_arm_z               58.02497
## roll_dumbbell             194.92801
## pitch_dumbbell             85.76263
## yaw_dumbbell              116.03882
## total_accel_dumbbell      124.97924
## gyros_dumbbell_x           39.60897
## gyros_dumbbell_y           90.25696
## gyros_dumbbell_z           24.61543
## accel_dumbbell_x          125.77462
## accel_dumbbell_y          170.73895
## accel_dumbbell_z          140.49083
## magnet_dumbbell_x         238.78228
## magnet_dumbbell_y         309.69705
## magnet_dumbbell_z         304.78809
## roll_forearm              219.52523
## pitch_forearm             305.94126
## yaw_forearm                50.67718
## total_accel_forearm        28.68905
## gyros_forearm_x            22.67795
## gyros_forearm_y            39.42047
## gyros_forearm_z            24.13554
## accel_forearm_x           133.13730
## accel_forearm_y            43.98957
## accel_forearm_z            91.79566
```

```
## magnet_forearm_x             81.03219
## magnet_forearm_y             66.71090
## magnet_forearm_z             88.31883
```

```r
print(modelRF)
```

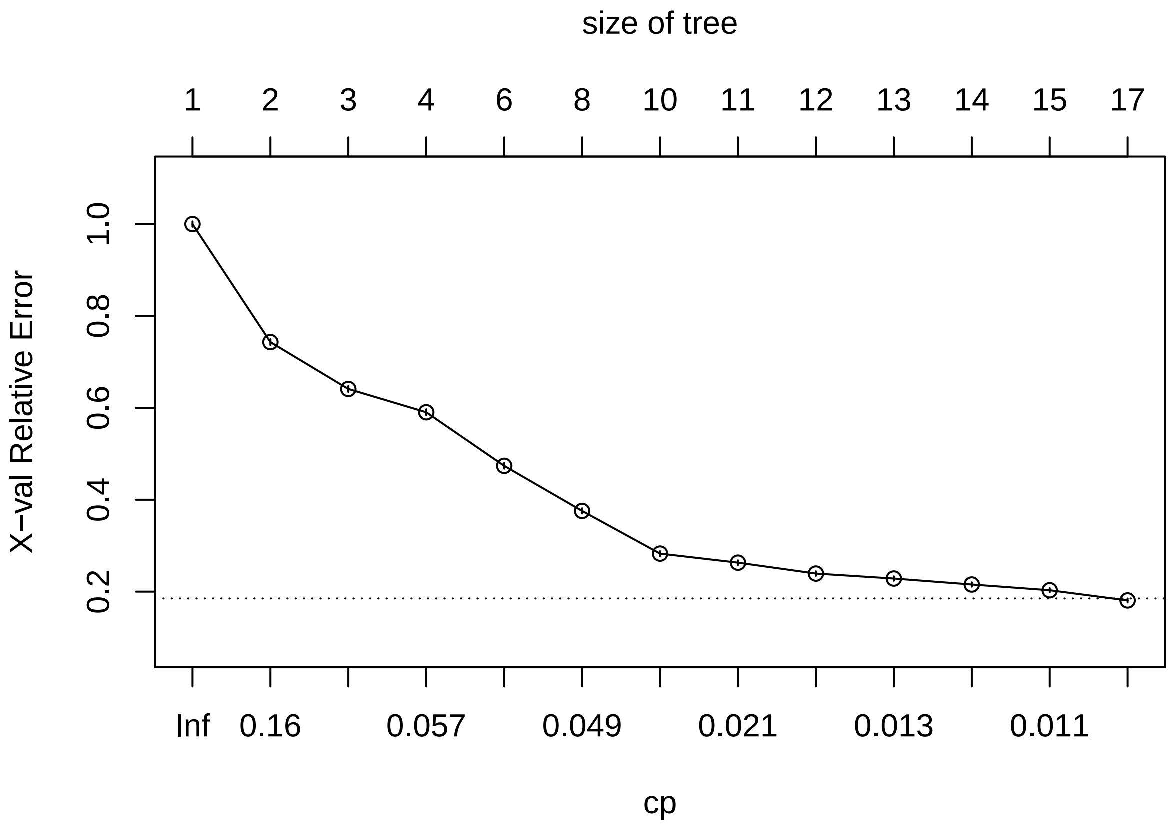
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    0    0    0
##          C    0    0 1365    2    0
##          D    0    0    3 1283    0
##          E    0    0    0    1 1442
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9991          
##                  95% CI : (0.9982, 0.9996)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9989          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9978   0.9977   1.0000
## Specificity            0.9998   1.0000   0.9997   0.9995   0.9998
## Pos Pred Value         0.9996   1.0000   0.9985   0.9977   0.9993
## Neg Pred Value         1.0000   0.9998   0.9995   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1740   0.1635   0.1838
## Detection Prevalence   0.2846   0.1933   0.1742   0.1639   0.1839
## Balanced Accuracy      0.9999   0.9997   0.9987   0.9986   0.9999
```

Prediction with Decision Trees

Create decision tree

Instead of plotting a decision tree, we can quickly look at a graph of the cross-validation results, and review the confusion matrix resutls and see that the error rate is higher than the random forest method.

```r
set.seed(4726)
modelFitDT1 <- rpart(classe ~.,method="class", data=train)
predictionDT1 <- predict(modelFitDT1, test, type = "class")
modelDT <- confusionMatrix(predictionDT1,test$classe)
plotcp(modelFitDT1)
```




```
modelDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2151   53   10    1    0
##          B   54 1254   60   67    0
##          C   27  202 1272  206   63
##          D    0    9   26  959  181
##          E    0    0    0   53 1198
##
## Overall Statistics
##
##                Accuracy : 0.871
##                  95% CI : (0.8634, 0.8784)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.837
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9637   0.8261   0.9298   0.7457   0.8308
## Specificity            0.9886   0.9714   0.9231   0.9671   0.9917
```

```
## Pos Pred Value         0.9711   0.8739   0.7186   0.8162   0.9576
## Neg Pred Value         0.9856   0.9588   0.9842   0.9510   0.9630
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2742   0.1598   0.1621   0.1222   0.1527
## Detection Prevalence   0.2823   0.1829   0.2256   0.1498   0.1594
## Balanced Accuracy      0.9762   0.8987   0.9265   0.8564   0.9113
```

A quick look at the results on the Decision Tree method and we see a lower accuracy rate, 88.73, than the Random Forest accuracy rate of 99.83% witha .17% for our out-of-sample error rate, so we'll progress with the Random Forest for our prediction set.

Predicting our results

First, we'll use the same cleaning methods as above:

```
cleanFormat <- colnames(train[,-58]) # classe column removal
pmlTest <-pmlTest[cleanFormat]
dim(pmlTest)
```

```
## [1] 20 57
```

And we'll coerce the data into the same format:

```
for (i in 1:length(pmlTest) ) {
    for(j in 1:length(train)) {
        if( length( grep(names(train[i]), names(pmlTest)[j]) ) == 1)  {
            class(pmlTest[j]) <- class(train[i])
        }
    }
}

# To get the same class between pmlTest and train
pmlTest <- rbind(train[2,-58], pmlTest) ## remove excess rows
pmlTest <- pmlTest[-1,]
```

Then we apply the prediction model to the data:

```
predictionFinal <-predict(modelFitRF1, pmlTest, type="class")
```

And our final results for our 20 test cases.

```
predictionFinal
```

```
## 22  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```