Practical Machine Learning Assignment

Yvette Winton

October 8, 2016

```
knitr::opts_chunk$set(echo = TRUE)
```

Objective

The dataset in this study is from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. The final model will be used to predict 20 different test cases.

Load required packages for the prediction

```
library(AppliedPredictiveModeling)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

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

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
library(rattle)
```

```
## R session is headless; GTK+ not initialized.
```

```
## 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.
```

```
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(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
library(plyr)
```

Load datasets for training and final dataset for 20 test cases

```
fileUrl_train<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(fileUrl_train,destfile="./train4.csv",method="curl")
data_train = read.csv("~/train4.csv", na.strings=c("NA","") , header=TRUE)

fileUrl_test<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrl_test,destfile="./test4.csv",method="curl")
data_test = read.csv("~/test4.csv", na.strings=c("NA","") , header=TRUE)
```

60% of the original train set is being used as test set and the rest is used as validation set

```
# create a partition with the training dataset
inTrain  <- createDataPartition(data_train$classe, p=0.6, list=FALSE)
TrainSet <- data_train[inTrain, ]
TestSet  <- data_train[-inTrain, ]
dim(TrainSet)
```

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

```
dim(TestSet)
```

```
## [1] 7846  160
```

Clean up data set by omitting all the columns with mainly zero values, NAs or not for predication use.

```
NZero <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZero]
TestSet  <- TestSet[, -NZero]
dim(TrainSet)
```

```
## [1] 11776   123
```

```
dim(TestSet)
```

```
## [1] 7846  123
```

```
NAOBS   <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.9
TrainSet <- TrainSet[, NAOBS==FALSE]
TestSet  <- TestSet[, NAOBS==FALSE]
TrainSet <- TrainSet[, -(1:6)]
TestSet <- TestSet[, -(1:6)]
dim(TrainSet)
```

```
## [1] 11776    53
```

```
dim(TestSet)
```

```
## [1] 7846   53
```

```
#Check that the columns matches for the training and validation sets
colnames(TrainSet)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
colnames(TestSet)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

I plan on building the prediction model with trees , random forest and boosting with trees. I will pick the final model with the highest accuracy after predicting each model with the validation set. I originally used the entire dataset to run, but it took a long time, I decided to use a subset of the traing set to come up with model. Since in the end, the best model accuracy turned out to be high, I did not increase my training sample size.

The accuracy of random forest prediction is 0.9936

```
set.seed(2345)
fitRF <- train(classe ~ ., data=TrainSet,method="rf",trControl=trainControl(method="cv",
 number=3))
predRF<- predict(fitRF, TestSet)
print(confusionMatrix(predRF, TestSet$classe))
```

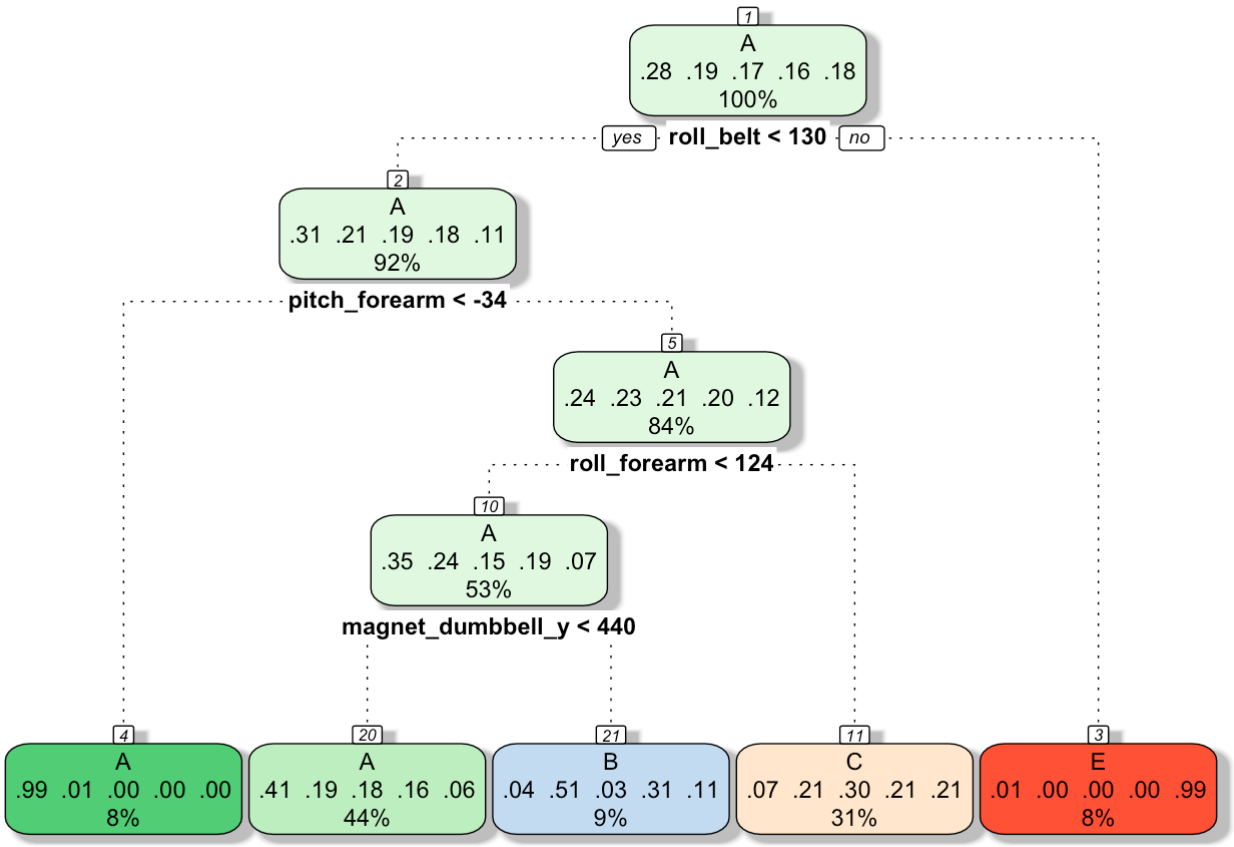
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   15    0    0    0
##          B    3 1497    7    0    1
##          C    0    6 1355   18    0
##          D    0    0    6 1266   10
##          E    1    0    0    2 1431
##
## Overall Statistics
##
##                Accuracy : 0.9912
##                  95% CI : (0.9889, 0.9932)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9889
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9862   0.9905   0.9844   0.9924
## Specificity            0.9973   0.9983   0.9963   0.9976   0.9995
## Pos Pred Value         0.9933   0.9927   0.9826   0.9875   0.9979
## Neg Pred Value         0.9993   0.9967   0.9980   0.9970   0.9983
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1908   0.1727   0.1614   0.1824
## Detection Prevalence   0.2859   0.1922   0.1758   0.1634   0.1828
## Balanced Accuracy      0.9978   0.9922   0.9934   0.9910   0.9960
```

The accuracy of tree prediction is 0.494. As expected, this is a poorer prediction than random forest.

```
set.seed(2345)
Fitrpart <- train(classe ~ ., data = TrainSet, method="rpart")
print(Fitrpart$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 10812 7472 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 945    7 A (0.99 0.0074 0 0 0) *
##      5) pitch_forearm>=-33.95 9867 7465 A (0.24 0.23 0.21 0.2 0.12)
##       10) roll_forearm< 123.5 6255 4093 A (0.35 0.24 0.15 0.19 0.072)
##         20) magnet_dumbbell_y< 439.5 5192 3073 A (0.41 0.19 0.18 0.16 0.064) *
##         21) magnet_dumbbell_y>=439.5 1063  516 B (0.04 0.51 0.025 0.31 0.11) *
##       11) roll_forearm>=123.5 3612 2516 C (0.066 0.21 0.3 0.21 0.21) *
##    3) roll_belt>=130.5 964    8 E (0.0083 0 0 0 0.99) *
```

```
fancyRpartPlot(Fitrpart$finalModel)
```



Rattle 2016-Oct-12 22:05:56 yvette

```
predRpart <- predict(Fitrpart, TestSet)
print(confusionMatrix(predRpart , TestSet$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2023  605  656  606  194
##          B   37  404   26  217   88
##          C  166  509  686  463  485
##          D    0    0    0    0    0
##          E    6    0    0    0  675
##
## Overall Statistics
##
##                Accuracy : 0.4828
##                  95% CI : (0.4717, 0.4939)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3245
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9064  0.26614  0.50146   0.0000  0.46810
## Specificity            0.6329  0.94185  0.74946   1.0000  0.99906
## Pos Pred Value         0.4953  0.52332  0.29710      NaN  0.99119
## Neg Pred Value         0.9444  0.84252  0.87683   0.8361  0.89295
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2578  0.05149  0.08743   0.0000  0.08603
## Detection Prevalence   0.5205  0.09839  0.29429   0.0000  0.08680
## Balanced Accuracy      0.7696  0.60399  0.62546   0.5000  0.73358
```

The accuracy of boosting with trees prediction is 0.9625 which is an improvement from classification trees but not as good as random forest prediction.

```
set.seed(2345)
fitgbm <- train(classe~ ., data=TrainSet,method="gbm",trControl=trainControl(method="rep
eatedcv",number=5,repeats=1),verbose=FALSE)
predgbm <- predict(fitgbm , TestSet)
print(confusionMatrix(predgbm, TestSet$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2190   56    0    1    6
##          B   32 1422   57    3   19
##          C    6   34 1294   39    8
##          D    4    6   16 1235   23
##          E    0    0    1    8 1386
##
## Overall Statistics
##
##                Accuracy : 0.9593
##                  95% CI : (0.9547, 0.9636)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9486
##  Mcnemar's Test P-Value : 8.709e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9812   0.9368   0.9459   0.9603   0.9612
## Specificity            0.9888   0.9825   0.9866   0.9925   0.9986
## Pos Pred Value         0.9720   0.9276   0.9370   0.9618   0.9935
## Neg Pred Value         0.9925   0.9848   0.9886   0.9922   0.9913
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2791   0.1812   0.1649   0.1574   0.1767
## Detection Prevalence   0.2872   0.1954   0.1760   0.1637   0.1778
## Balanced Accuracy      0.9850   0.9596   0.9662   0.9764   0.9799
```

Since random forest prediction has the highest accuracy of 0.9936, it is used to predict the 20 test case and here is the result.

```
print(predict(fitRF,data_test))
```

```
##  [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
```