$MachineLearningProject - ExerciseBand$

Introduction

This report presents an analysis of a *FitBit-like* device used to monitor a specific exercise routine. Data is stored from the on-device accelerometer, when placed at three positions on the participants' body: belt, forearm and arm. An extra placement is recorded on the equipment-type used.

The motivation for the project is to predict how the exercises are performed. Questions answered concern: 1. how the software model was built 2. the use of cross-validation 3. expected out-of-sample error 4. why these choices

Data

The data for this assignment is from this location (http://groupware.les.inf.puc-rio.br/har), and contains information from the placement accelerometers. The data is split into a training and testing groups.

Method

Set the libraries and read the training test file. Split input training data into training and testing at 90%

```
library(lattice)
library(ggplot2)
library(caret)
library(gbm)
```

```
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
```

```
set.seed(9175)
fileLocation = "K:/COURSES/JHU_DataScience/PracticalMachineLearning/project/data/pml-training.csv"
pml.training <- read.csv(fileLocation)
```

Load data to memory

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

Split the training data at the 90% point

```
inTrain <- createDataPartition(y=pml.training$classe, p=0.9, list=FALSE)
newTraining <- pml.training[inTrain,]
newTesting <- pml.training[-inTrain,]
```

Cleaning the data

Clean the data of NAs The following process was used to clean the data:

Mod 1: Cleaning NearZeroVariance Variables

(from Help - *nearZeroVar* diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large. checkConditionalX looks at the distribution of the columns of x conditioned on the levels of y and identifies columns of x that are sparse within groups of y.)

Inspect possible Non-Zero Variables:

```
myDataNZV <  nearZeroVar(newTraining, saveMetrics=TRUE)
```

Run this code to create another training subset without NZV

```
myNZVvars <- names(newTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbel
l",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
newTraining <- newTraining[!myNZVvars]
#To check the new N?? of observations
dim(newTraining)
```

```
## [1] 17662   100
```

Mod 2: remove first column of training (ID) Removing the variable (ID) so that it doesn’t interfer with ML

```
newTraining <  newTraining[c( 1)]
```

Mod 3: Cleaning variables of too many NAs. For Variables that have >60% threshold of NA’s

```
trainingMx <- newTraining                           #make another subset to iterate over
for(i in 1:length(newTraining))
{                                                   #do next column in the training dataset
    if(sum( is.na( newTraining[, i] )) /nrow(newTraining) >= .6 )
    {                                               #if number NAs >60% of total obs
        for(j in 1:length(trainingMx))
        {
            if(length( grep(names(newTraining[i]), names(trainingMx)[j])) ==1)
            {                                       #if same columns
                trainingMx <- trainingMx[ , -j] #remove column
            }
        }
    }
}
#check the new observations
dim(trainingMx)
```

```
## [1] 17662    58
```

```
#point back to our original training
newTraining <- trainingMx
rm(trainingMx)
```

adjust newTesting and testing data as above

```
cleanUp1 <- colnames(newTraining)
cleanUp2 <- colnames(newTraining[, -58]) #already with classe column removed
newTesting <- newTesting[cleanUp1]
testing <- testing[cleanUp2]

#To check the new observations
dim(newTesting)
```

```
## [1] 1960   58
```

```
#To check the new observations
dim(testing)
```

```
## [1] 19622    57
```

```
#The last column (problem_id) which is not equal to training sets, was removed
```

To ensure proper functioning of Decision Trees and RandomForest Algorithm with the test data (original data provided), we need to force to the same type.

```
for (i in 1:length(newTesting))
{
    for(j in 1:length(newTraining))
    {
        if(length(grep(names(newTraining[i]), names(newTesting)[j]))==1)
        {
            class(newTesting[j]) <- class(newTraining[i])
        }
    }
}
#make sure coertion really worked
testing <- rbind(newTraining[2, -58], testing) #row 1,2 are useless and were be removed
testing <- testing[-1,]
```

```
names(newTraining)
```

```
##  [1] "user_name"            "raw_timestamp_part_1" "raw_timestamp_part_2"
##  [4] "cvtd_timestamp"       "num_window"           "roll_belt"
##  [7] "pitch_belt"           "yaw_belt"             "total_accel_belt"
## [10] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
## [13] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
## [16] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [19] "roll_arm"             "pitch_arm"            "yaw_arm"
## [22] "total_accel_arm"      "gyros_arm_x"          "gyros_arm_y"
## [25] "gyros_arm_z"          "accel_arm_x"          "accel_arm_y"
## [28] "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
## [31] "magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
## [34] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"
## [37] "gyros_dumbbell_y"     "gyros_dumbbell_z"     "accel_dumbbell_x"
## [40] "accel_dumbbell_y"     "accel_dumbbell_z"     "magnet_dumbbell_x"
## [43] "magnet_dumbbell_y"    "magnet_dumbbell_z"    "roll_forearm"
## [46] "pitch_forearm"        "yaw_forearm"          "total_accel_forearm"
## [49] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [52] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [55] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [58] "classe"
```

```
smt <- summary(newTraining)
write.table(smt, file="K:/COURSES/JHU_DataScience/PracticalMachineLearning/project/data/summary.txt"
, sep=',')
```

Reasons for process: 1. 90 percent subsample is used to train the module 2. 10 percent sample is used for cross-validation.
3. used this simple cross-validation rather than using K-fold with the `cv.folds` option to decrease run time, which was already rather long
4. implement a Stochastic Gradient Boosting algorithm via the `gbm` package.

```
ptm <- proc.time()
modFitA1 <- train(classe ~ user_name + pitch_arm + yaw_arm + roll_arm + roll_belt + pitch_belt + yaw
_belt + gyros_belt_x + gyros_belt_y + gyros_belt_z + accel_belt_x + accel_belt_y + accel_belt_z + ma
gnet_belt_x + magnet_belt_y + magnet_belt_z + gyros_arm_x + gyros_arm_y + gyros_arm_z + accel_arm_x
+ accel_arm_y + accel_arm_z + magnet_arm_x + magnet_arm_y + magnet_arm_z + roll_dumbbell + pitch_dum
bbell + yaw_dumbbell, method="gbm", data=newTraining, verbose=FALSE)
```

```
## Loading required package: plyr
```

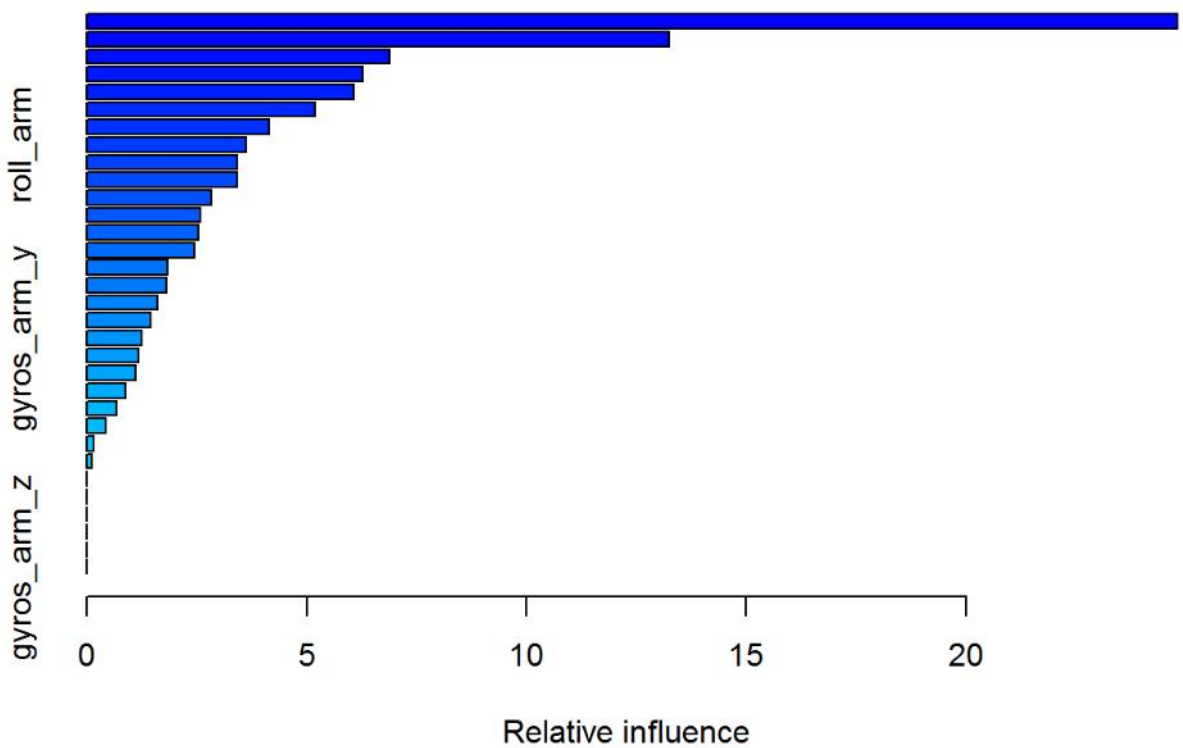
```
tm <- proc.time() - ptm
tm
```

```
##    user  system elapsed
## 1265.75    2.16 1273.98
```

```
ptm <- proc.time()
modFitA2 <- train(classe ~ user_name + pitch_arm + yaw_arm + roll_arm + roll_belt + pitch_belt + yaw
_belt + gyros_belt_x + gyros_belt_y + gyros_belt_z + accel_belt_x + accel_belt_y + accel_belt_z + ma
gnet_belt_x + magnet_belt_y + magnet_belt_z + gyros_arm_x + gyros_arm_y + gyros_arm_z + accel_arm_x
+ accel_arm_y + accel_arm_z + magnet_arm_x + magnet_arm_y + magnet_arm_z + roll_dumbbell + pitch_dum
bbell + yaw_dumbbell, method="gbm", data=newTesting, verbose=FALSE)
tm <- proc.time() - ptm
tm
```

```
##    user  system elapsed
##  126.97    0.42  136.05
```

```
summary(modFitA1)
```


roll_arm
gyros_arm_y
gyros_arm_z
0
5
10
15
20
Relative influence

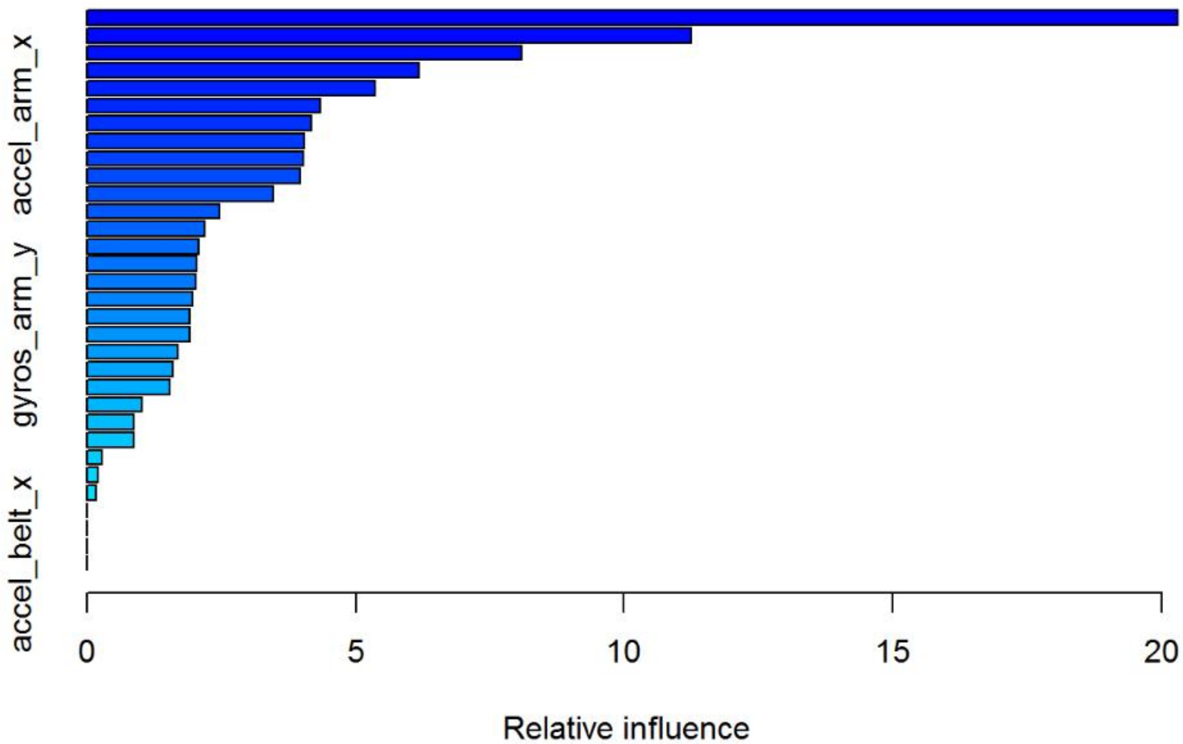
```
##                                       var    rel.inf
## roll_belt                       roll_belt 24.8004549
## yaw_belt                         yaw_belt 13.2581575
## magnet_belt_z                magnet_belt_z  6.8879286
## roll_dumbbell                roll_dumbbell  6.2670735
## pitch_belt                      pitch_belt  6.0828479
## magnet_arm_x                  magnet_arm_x  5.1978168
## magnet_arm_z                  magnet_arm_z  4.1531520
## roll_arm                          roll_arm  3.6326962
## gyros_belt_z                  gyros_belt_z  3.4184291
## accel_arm_x                    accel_arm_x  3.4133532
## user_nameeurico            user_nameeurico  2.8309509
## yaw_dumbbell                  yaw_dumbbell  2.5753052
## pitch_dumbbell              pitch_dumbbell  2.5310924
## accel_arm_z                    accel_arm_z  2.4527768
## yaw_arm                            yaw_arm  1.8361944
## accel_belt_z                  accel_belt_z  1.8160015
## magnet_belt_x                magnet_belt_x  1.6058697
## pitch_arm                        pitch_arm  1.4446081
## gyros_arm_y                    gyros_arm_y  1.2518722
## magnet_belt_y                magnet_belt_y  1.1788608
## magnet_arm_y                  magnet_arm_y  1.1150757
## user_namecharles          user_namecharles  0.8853653
## gyros_belt_y                  gyros_belt_y  0.6747000
## gyros_arm_x                    gyros_arm_x  0.4252966
## accel_arm_y                    accel_arm_y  0.1545834
## gyros_belt_x                  gyros_belt_x  0.1095373
## user_namecarlitos user_namecarlitos  0.0000000
## user_namejeremy            user_namejeremy  0.0000000
## user_namepedro              user_namepedro  0.0000000
## accel_belt_x                  accel_belt_x  0.0000000
## accel_belt_y                  accel_belt_y  0.0000000
## gyros_arm_z                    gyros_arm_z  0.0000000
```

```
predictTraining <- predict(modFitA1, newTraining)
table(predictTraining, newTraining$classe)
```

```
##
## predictTraining    A    B    C    D    E
##               A 4725  151   56   47   36
##               B   87 3128  125   17   33
##               C   57  117 2830  129   23
##               D  113   10   62 2679   18
##               E   40   12    7   23 3137
```

The model correctly classifies 93.6% of the observations in the training data with 150 trees. The *roll_belt* and *yaw_belt* features were by far the most important

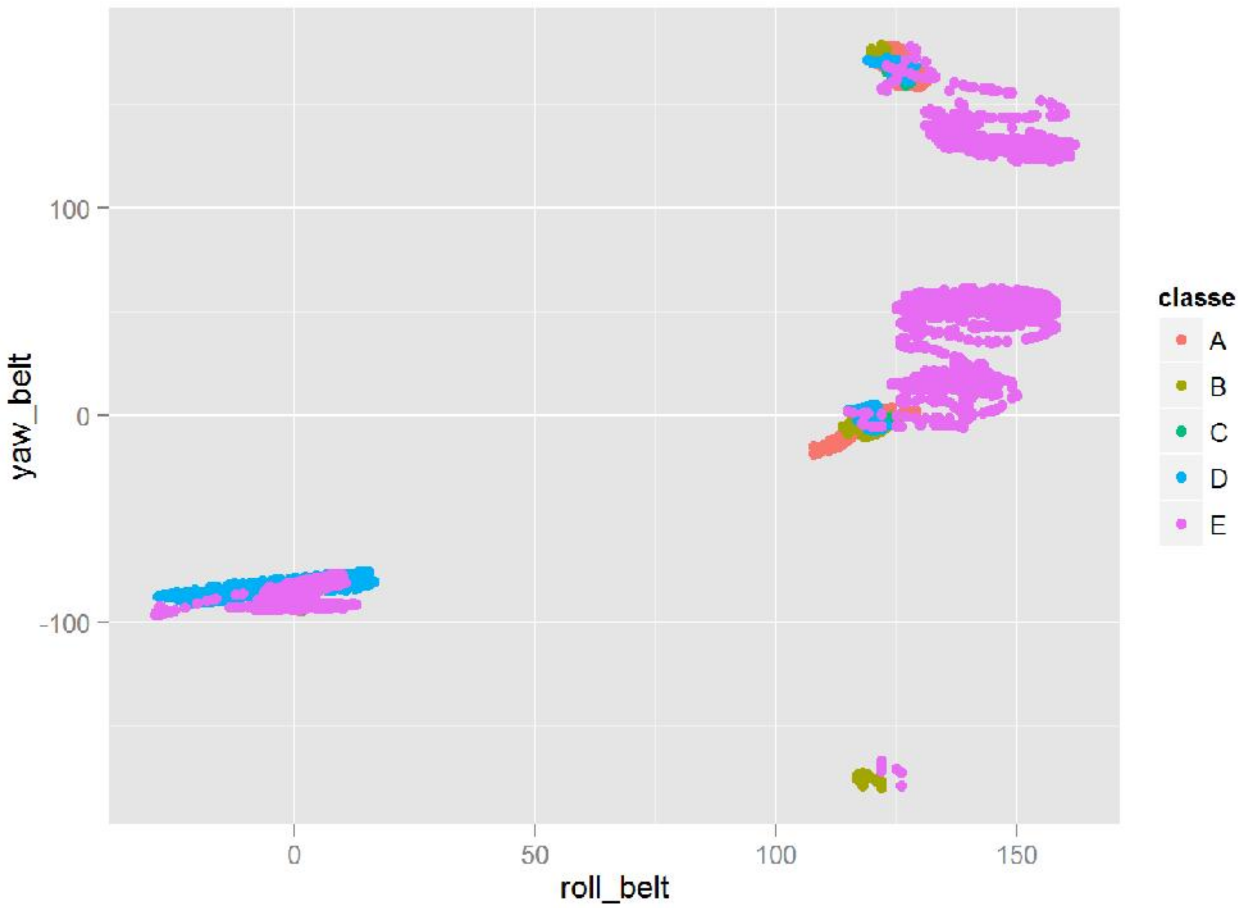
```
summary(modFitA2,n.trees=150)
```



```
##                                   var    rel.inf
## roll_belt                   roll_belt 20.2945582
## yaw_belt                     yaw_belt 11.2603480
## roll_dumbbell           roll_dumbbell  8.0878723
## pitch_belt                 pitch_belt  6.1718943
## magnet_belt_z           magnet_belt_z  5.3493030
## yaw_arm                       yaw_arm  4.3351289
## accel_arm_x               accel_arm_x  4.1733592
## magnet_arm_z             magnet_arm_z  4.0399129
## pitch_dumbbell         pitch_dumbbell  4.0193920
## magnet_arm_x             magnet_arm_x  3.9627574
## gyros_belt_z             gyros_belt_z  3.4699891
## magnet_arm_y             magnet_arm_y  2.4762284
## roll_arm                     roll_arm  2.1975738
## pitch_arm                   pitch_arm  2.0843119
## accel_arm_z               accel_arm_z  2.0346517
## magnet_belt_y           magnet_belt_y  2.0290223
## yaw_dumbbell             yaw_dumbbell  1.9648800
## user_nameeurico       user_nameeurico  1.9048732
## gyros_arm_y               gyros_arm_y  1.9023023
## magnet_belt_x           magnet_belt_x  1.6857479
## gyros_arm_x               gyros_arm_x  1.6000038
## accel_arm_y               accel_arm_y  1.5463860
## accel_belt_z             accel_belt_z  1.0177495
## gyros_belt_x             gyros_belt_x  0.8740361
## gyros_belt_y             gyros_belt_y  0.8646887
## user_namepedro         user_namepedro  0.2764315
## accel_belt_y             accel_belt_y  0.2126767
## gyros_arm_z               gyros_arm_z  0.1639205
## user_namecarlitos user_namecarlitos  0.0000000
## user_namecharles   user_namecharles  0.0000000
## user_namejeremy     user_namejeremy  0.0000000
## accel_belt_x             accel_belt_x  0.0000000
```

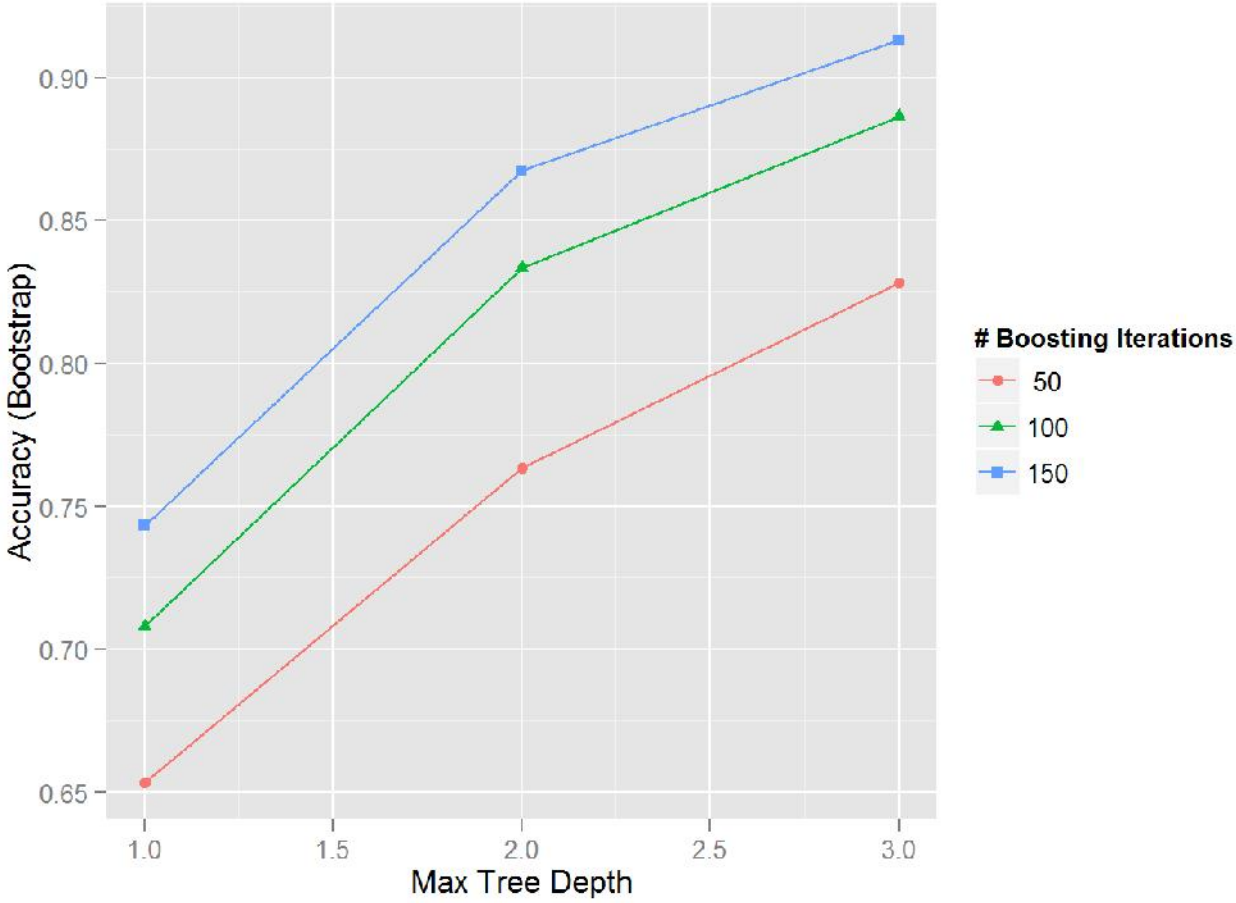
A plot of these top two features colored by outcome demonstrates their relative importance.

```
qplot(roll_belt, yaw_belt,colour=classe,data=newTraining)
```



Although these are the top observations, however, not great predictors. Obviously, some bunching can be seen in this plot. The choice of a boosting algorithm is a good choice given the large number of weak predictors. The next plot shows the improved performance using boosting iterations.

```
ggplot(modFitA1)
```



Next, I check the performance on the 10 percent subsample to get an estimate of the algorithm's out-of-sample performance.

```
predictTesting <- predict(modFitA2, newTesting) #newTesting chaged to testing
table(predictTesting, newTesting$classe)
```

```
##
## predictTesting   A   B   C   D   E
##              A 547   8   1   1   1
##              B   2 365   6   0   1
##              C   1   6 332   5   0
##              D   6   0   3 314   2
##              E   2   0   0   1 356
```

The algorithm actually peforms only does slightly worse on the testing subset than it did on the full training set, correctly classifying 93.4 percent of the observations.

Prediction Phase

Finally, predict using the original testing set. The results go to the `pml_write_files()` function and stored.

```
pml.testing <- read.csv("K:/COURSES/JHU_DataScience/PracticalMachineLearning/project/data/pml-testin
g.csv")
answers <- as.character(predict(modFitA1, pml.testing))
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(answers)
```

The algorithm correctly predicted the outcome for 20/20 observations, in agreement with its strong out-of-sample classification accuracy.