Practical Machine Learning Assignment

- Predictions on Weight Lifting Exercise using a random forest algorithm -

Executive summary:

Our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner (i.e. correctly or incorrectly) in which they perform barbell lifts in 5 different ways. The training and testing datasets are taken from the Human Activity Recognition data set from Groupware. Here, we use a random forest algorithm to predict the outcome based on all the relevant predictors.

Data Analysis

First, we load the nescessary library and the data files (training and testing). We also set the seed for random number generation.

```r
library(caret); library(randomForest);
```

```
## Warning: package 'caret' was built under R version 3.0.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.0.3

## Loading required package: ggplot2

## Warning: package 'randomForest' was built under R version 3.0.3

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
inTrain <- read.csv("pml-training.csv",na.strings=c("NA",""))
inTest <- read.csv("pml-testing.csv",na.strings=c("NA",""))
set.seed(123)
```

We then remove the columns that consist mainly of NAs, and we also remove unnecessary factor columns. We then remove the exact same columns for the test data set.

```r
colNACounts <- colSums(is.na(inTrain))
NAcol <- colNACounts >= 19000
sTrain <- inTrain[!NAcol]
sTrain <- sTrain[,-c(1,2,5,6)]
sTest <- inTest[!NAcol]
sTest <- sTest[,-c(1,2,5,6)]
```

We can look at how many examples are including in each of the “classe” that we are trying to predict. Figure displays an histogram for the number of examples for each classe outcomes.

We then ensure that we do not have “near-zezo-variance predictors” using:

```
NZVP <- nearZeroVar(sTrain[,-56], saveMetrics = TRUE)
```

The results indicate that all the remaining predictors should be included in the analysis.

We then partition the training dataset into a training and testing data set.

```
partition <- createDataPartition(y = sTrain$classe, p = 0.6, list = FALSE)
trainingdata <- sTrain[partition, ]
testdata <- sTrain[-partition, ]
```

We then train a random forest algorithm on the training dataset. First, we tune the optimum "mtry"" parameter using a buildin function:

```
tuneRF(trainingdata[,-56], trainingdata$classe, mtryStart = 3)
```

The results indicate that mtry = 12 is optimal. The model is then obtained with

```
model <- train(trainingdata[,-56],trainingdata$classe, method = "rf", tuneGrid = data.frame(mtry=12), t
```

```
## Warning: package 'e1071' was built under R version 3.0.3
```

```
model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##         OOB estimate of  error rate: 0.15%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3348    0    0    0    0   0.0000000
## B    2 2277    0    0    0   0.0008776
## C    0    4 2048    2    0   0.0029211
## D    0    0    9 1920    1   0.0051813
## E    0    0    0    0 2165   0.0000000
```

As can be seen on the Confusion matrix, the accuracy for the training set is high and the out-of-bag (oob) error is low. The importance of each variables is plotted on figure 2. Finally, to measure the out-of-sample error properly, we compare the prediction to the actual value on the testing data set.

```
confusionMatrix(predict(model,newdata=testdata[-56]), testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    3    0    0
##          C    0    0 1360    0    0
##          D    0    0    5 1286    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 0.999)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    0.994    1.000    0.999
## Specificity             1.000    1.000    1.000    0.999    1.000
## Pos Pred Value          1.000    0.998    1.000    0.995    1.000
## Neg Pred Value          1.000    1.000    0.999    1.000    1.000
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.173    0.164    0.184
## Detection Prevalence    0.284    0.194    0.173    0.165    0.184
## Balanced Accuracy       1.000    1.000    0.997    1.000    1.000
```

The accuracy kappa values are high, indicating that the model is accurate. And confusion matrix shows that very few of the examples were mislabelled.

We then apply the trained algorithm on the testing dataset for the submission section, and 20 our of 20 of the predictions were accurate.

In summary, our random forest algorithm is capable of accurately predicting the manner in which the exercise were preformed.

Figures

Figure 1

```
plot(sTrain$classe)
```

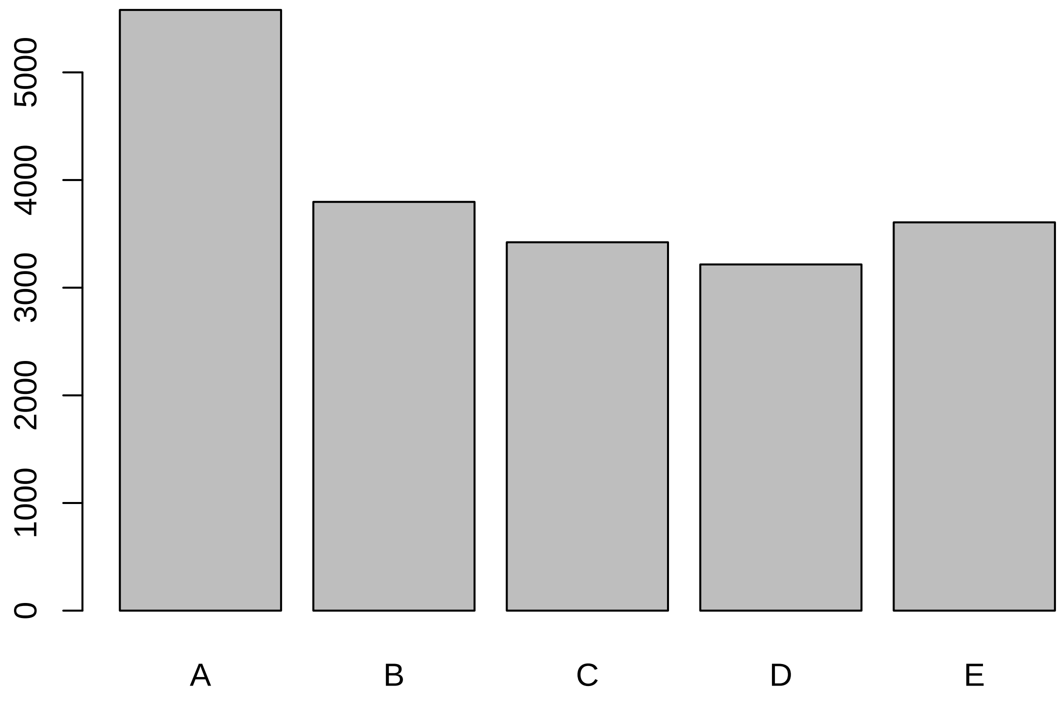

FIGURE 1

Figure 2

```
plot( varImp(model) )
```

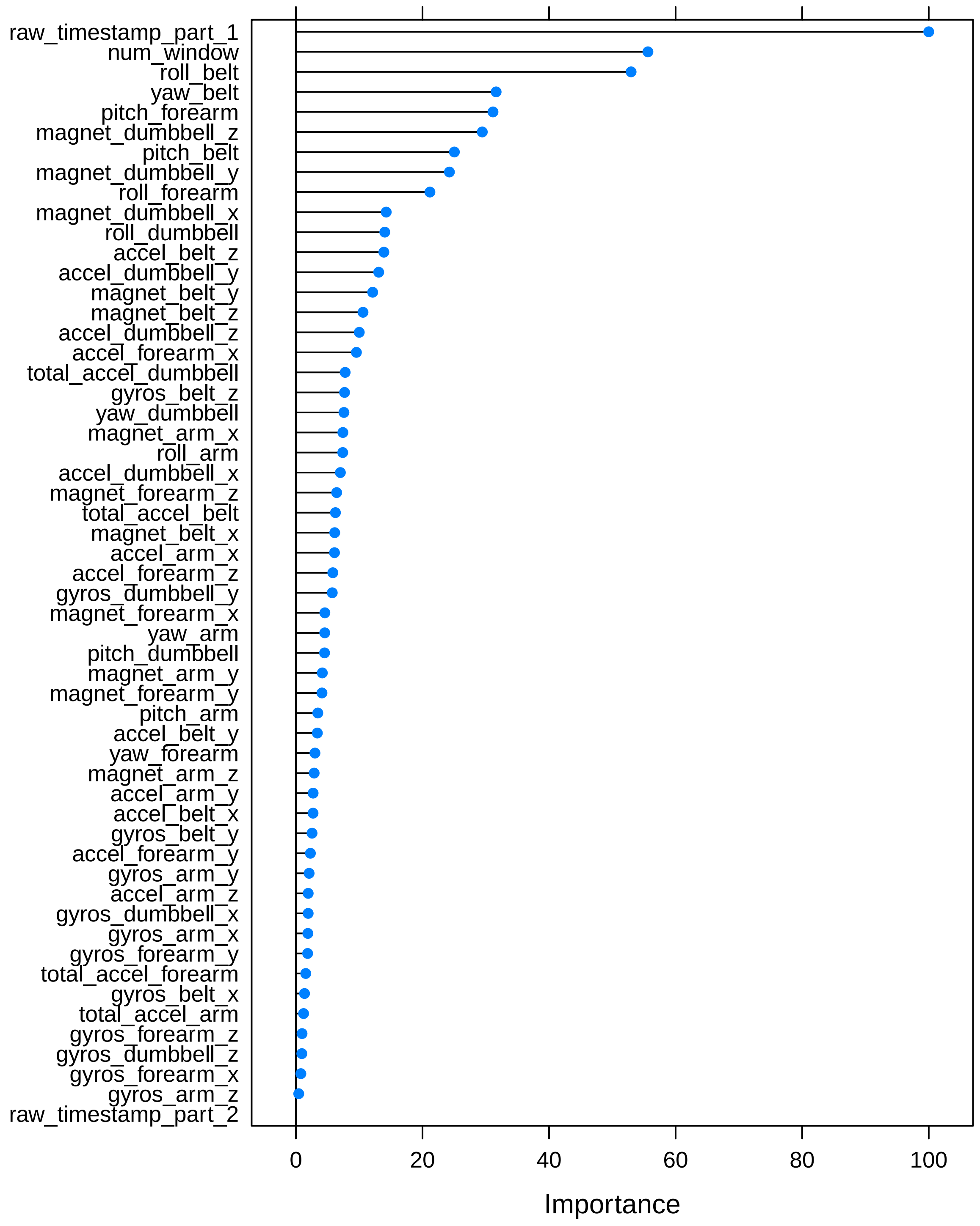
raw_timestamp_part_1
num_window
roll_belt
yaw_belt
pitch_forearm
magnet_dumbbell_z
pitch_belt
magnet_dumbbell_y
roll_forearm
magnet_dumbbell_x
roll_dumbbell
accel_belt_z
accel_dumbbell_y
magnet_belt_y
magnet_belt_z
accel_dumbbell_z
accel_forearm_x
total_accel_dumbbell
gyros_belt_z
yaw_dumbbell
magnet_arm_x
roll_arm
accel_dumbbell_x
magnet_forearm_z
total_accel_belt
magnet_belt_x
accel_arm_x
accel_forearm_z
gyros_dumbbell_y
magnet_forearm_x
yaw_arm
pitch_dumbbell
magnet_arm_y
magnet_forearm_y
pitch_arm
accel_belt_y
yaw_forearm
magnet_arm_z
accel_arm_y
accel_belt_x
gyros_belt_y
accel_forearm_y
gyros_arm_y
accel_arm_z
gyros_dumbbell_x
gyros_arm_x
gyros_forearm_y
total_accel_forearm
gyros_belt_x
total_accel_arm
gyros_forearm_z
gyros_dumbbell_z
gyros_forearm_x
gyros_arm_z
raw_timestamp_part_2
0
20
40
60
80
100
Importance

FIGURE 2