PML Predict

Mihail Petkov

7/9/2020

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

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Data Importing and Cleaning

```
library(ggplot2)
library(dplyr)
library(caret)
library(e1071)
library(randomForest)
library(tictoc)

set.seed(60)
# Download and unzip the data
fileurl1 = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
fileurl2 = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

if (!file.exists('./PML/trainSample')){
```

```
  download.file(fileurl1,'./PML/trainSample.csv')
  dateDownloaded1 <- date()
}

if (!file.exists('./PML/testSample')){
  download.file(fileurl2,'./PML/testSample.csv')
  dateDownloaded2 <- date()
}



trainSample <- read.csv('./PML/trainSample.csv', na.strings=c("NA", "#DIV/0!", ""))
testSample <- read.csv('./PML/testSample.csv', na.strings=c("NA", "#DIV/0!", ""))

# Checking the dimensions of the data
head(colnames(trainSample),10)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"
```

```
head(colnames(testSample), 10)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"
```

```
dim(trainSample)
```

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

```
dim(testSample)
```

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

```
trainNULLS <- sapply(trainSample, function(y) sum(length(which(is.na(y)))))
testNULLS <- sapply(testSample, function(y) sum(length(which(is.na(y)))))

# There are many columns with all of the values being NAs, so deleting these would be wise
# I took a threshold of below 40% of the samples can be NAs
threshold_40_per <- dim(trainSample)[1]*0.4

trainNULLS <- as.data.frame(trainNULLS)
indexes <- which(trainNULLS<=threshold_40_per)

cleanTrain <- trainSample[, indexes]
cleanTest <- testSample[,indexes]
```

```r
# Next we need to take away all the variables that have
# near-zero variance, one can do this with the function from caret
# We only need to take care of the variance in the train sample
# and we are to assume the testing sample corresponds to this
nzv <- nearZeroVar(cleanTrain)
filteredTrain <- na.omit(cleanTrain[,-nzv])
filteredTest <- na.omit(cleanTest[,-nzv])

summary(filteredTrain)
```

```
##        X            user_name    raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   :    1   adelmo  :3892   Min.   :1.322e+09    Min.   :   294
##  1st Qu.: 4906   carlitos:3112   1st Qu.:1.323e+09    1st Qu.:252912
##  Median : 9812   charles :3536   Median :1.323e+09    Median :496380
##  Mean   : 9812   eurico  :3070   Mean   :1.323e+09    Mean   :500656
##  3rd Qu.:14717   jeremy  :3402   3rd Qu.:1.323e+09    3rd Qu.:751891
##  Max.   :19622   pedro   :2610   Max.   :1.323e+09    Max.   :998801
##
##           cvtd_timestamp    num_window      roll_belt
##  28/11/2011 14:14: 1498   Min.   :  1.0   Min.   :-28.90
##  05/12/2011 11:24: 1497   1st Qu.:222.0   1st Qu.:  1.10
##  30/11/2011 17:11: 1440   Median :424.0   Median :113.00
##  05/12/2011 11:25: 1425   Mean   :430.6   Mean   : 64.41
##  02/12/2011 14:57: 1380   3rd Qu.:644.0   3rd Qu.:123.00
##  02/12/2011 13:34: 1375   Max.   :864.0   Max.   :162.00
##  (Other)         :11007
##    pitch_belt          yaw_belt       total_accel_belt  gyros_belt_x
##  Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00    Min.   :-1.040000
##  1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00    1st Qu.:-0.030000
##  Median :  5.2800   Median : -13.00   Median :17.00    Median : 0.030000
##  Mean   :  0.3053   Mean   : -11.21   Mean   :11.31    Mean   :-0.005592
##  3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00    3rd Qu.: 0.110000
##  Max.   : 60.3000   Max.   : 179.00   Max.   :29.00    Max.   : 2.220000
##
##   gyros_belt_y       gyros_belt_z      accel_belt_x       accel_belt_y
##  Min.   :-0.64000   Min.   :-1.4600   Min.   :-120.000   Min.   :-69.00
##  1st Qu.: 0.00000   1st Qu.:-0.2000   1st Qu.: -21.000   1st Qu.:  3.00
##  Median : 0.02000   Median :-0.1000   Median : -15.000   Median : 35.00
##  Mean   : 0.03959   Mean   :-0.1305   Mean   :  -5.595   Mean   : 30.15
##  3rd Qu.: 0.11000   3rd Qu.:-0.0200   3rd Qu.:  -5.000   3rd Qu.: 61.00
##  Max.   : 0.64000   Max.   : 1.6200   Max.   :  85.000   Max.   :164.00
##
##   accel_belt_z     magnet_belt_x   magnet_belt_y   magnet_belt_z
##  Min.   :-275.00   Min.   :-52.0   Min.   :354.0   Min.   :-623.0
##  1st Qu.:-162.00   1st Qu.:  9.0   1st Qu.:581.0   1st Qu.:-375.0
##  Median :-152.00   Median : 35.0   Median :601.0   Median :-320.0
##  Mean   : -72.59   Mean   : 55.6   Mean   :593.7   Mean   :-345.5
##  3rd Qu.:  27.00   3rd Qu.: 59.0   3rd Qu.:610.0   3rd Qu.:-306.0
##  Max.   : 105.00   Max.   :485.0   Max.   :673.0   Max.   : 293.0
##
##     roll_arm          pitch_arm          yaw_arm          total_accel_arm
##  Min.   :-180.00   Min.   :-88.800   Min.   :-180.0000   Min.   : 1.00
##  1st Qu.: -31.77   1st Qu.:-25.900   1st Qu.: -43.1000   1st Qu.:17.00
```

```
##  Median :   0.00   Median :  0.000   Median :   0.0000   Median :27.00
##  Mean   :  17.83   Mean   : -4.612   Mean   :  -0.6188   Mean   :25.51
##  3rd Qu.:  77.30   3rd Qu.: 11.200   3rd Qu.:  45.8750   3rd Qu.:33.00
##  Max.   : 180.00   Max.   : 88.500   Max.   : 180.0000   Max.   :66.00
##
##   gyros_arm_x        gyros_arm_y       gyros_arm_z       accel_arm_x
##  Min.   :-6.37000   Min.   :-3.4400   Min.   :-2.3300   Min.   :-404.00
##  1st Qu.:-1.33000   1st Qu.:-0.8000   1st Qu.:-0.0700   1st Qu.:-242.00
##  Median : 0.08000   Median :-0.2400   Median : 0.2300   Median : -44.00
##  Mean   : 0.04277   Mean   :-0.2571   Mean   : 0.2695   Mean   : -60.24
##  3rd Qu.: 1.57000   3rd Qu.: 0.1400   3rd Qu.: 0.7200   3rd Qu.:  84.00
##  Max.   : 4.87000   Max.   : 2.8400   Max.   : 3.0200   Max.   : 437.00
##
##   accel_arm_y      accel_arm_z       magnet_arm_x     magnet_arm_y
##  Min.   :-318.0   Min.   :-636.00   Min.   :-584.0   Min.   :-392.0
##  1st Qu.: -54.0   1st Qu.:-143.00   1st Qu.:-300.0   1st Qu.:  -9.0
##  Median :  14.0   Median : -47.00   Median : 289.0   Median : 202.0
##  Mean   :  32.6   Mean   : -71.25   Mean   : 191.7   Mean   : 156.6
##  3rd Qu.: 139.0   3rd Qu.:  23.00   3rd Qu.: 637.0   3rd Qu.: 323.0
##  Max.   : 308.0   Max.   : 292.00   Max.   : 782.0   Max.   : 583.0
##
##   magnet_arm_z    roll_dumbbell     pitch_dumbbell     yaw_dumbbell
##  Min.   :-597.0   Min.   :-153.71   Min.   :-149.59   Min.   :-150.871
##  1st Qu.: 131.2   1st Qu.: -18.49   1st Qu.: -40.89   1st Qu.: -77.644
##  Median : 444.0   Median :  48.17   Median : -20.96   Median :  -3.324
##  Mean   : 306.5   Mean   :  23.84   Mean   : -10.78   Mean   :   1.674
##  3rd Qu.: 545.0   3rd Qu.:  67.61   3rd Qu.:  17.50   3rd Qu.:  79.643
##  Max.   : 694.0   Max.   : 153.55   Max.   : 149.40   Max.   : 154.952
##
##  total_accel_dumbbell gyros_dumbbell_x    gyros_dumbbell_y
##  Min.   : 0.00        Min.   :-204.0000   Min.   :-2.10000
##  1st Qu.: 4.00        1st Qu.:  -0.0300   1st Qu.:-0.14000
##  Median :10.00        Median :   0.1300   Median : 0.03000
##  Mean   :13.72        Mean   :   0.1611   Mean   : 0.04606
##  3rd Qu.:19.00        3rd Qu.:   0.3500   3rd Qu.: 0.21000
##  Max.   :58.00        Max.   :   2.2200   Max.   :52.00000
##
##  gyros_dumbbell_z  accel_dumbbell_x  accel_dumbbell_y  accel_dumbbell_z
##  Min.   : -2.380   Min.   :-419.00   Min.   :-189.00   Min.   :-334.00
##  1st Qu.: -0.310   1st Qu.: -50.00   1st Qu.:  -8.00   1st Qu.:-142.00
##  Median : -0.130   Median :  -8.00   Median :  41.50   Median :  -1.00
##  Mean   : -0.129   Mean   : -28.62   Mean   :  52.63   Mean   : -38.32
##  3rd Qu.:  0.030   3rd Qu.:  11.00   3rd Qu.: 111.00   3rd Qu.:  38.00
##  Max.   :317.000   Max.   : 235.00   Max.   : 315.00   Max.   : 318.00
##
##  magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z  roll_forearm
##  Min.   :-643.0    Min.   :-3600     Min.   :-262.00   Min.   :-180.0000
##  1st Qu.:-535.0    1st Qu.:  231     1st Qu.: -45.00   1st Qu.:  -0.7375
##  Median :-479.0    Median :  311     Median :  13.00   Median :  21.7000
##  Mean   :-328.5    Mean   :  221     Mean   :  46.05   Mean   :  33.8265
##  3rd Qu.:-304.0    3rd Qu.:  390     3rd Qu.:  95.00   3rd Qu.: 140.0000
##  Max.   : 592.0    Max.   :  633     Max.   : 452.00   Max.   : 180.0000
##
##  pitch_forearm     yaw_forearm      total_accel_forearm gyros_forearm_x
```

```
##  Min.   :-72.50   Min.   :-180.00   Min.   :  0.00      Min.   :-22.000
##  1st Qu.:  0.00   1st Qu.: -68.60   1st Qu.: 29.00      1st Qu.: -0.220
##  Median :  9.24   Median :   0.00   Median : 36.00      Median :  0.050
##  Mean   : 10.71   Mean   :  19.21   Mean   : 34.72      Mean   :  0.158
##  3rd Qu.: 28.40   3rd Qu.: 110.00   3rd Qu.: 41.00      3rd Qu.:  0.560
##  Max.   : 89.80   Max.   : 180.00   Max.   :108.00      Max.   :  3.970
##
##  gyros_forearm_y     gyros_forearm_z    accel_forearm_x   accel_forearm_y
##  Min.   : -7.02000   Min.   : -8.0900   Min.   :-498.00   Min.   :-632.0
##  1st Qu.: -1.46000   1st Qu.: -0.1800   1st Qu.:-178.00   1st Qu.:  57.0
##  Median :  0.03000   Median :  0.0800   Median : -57.00   Median : 201.0
##  Mean   :  0.07517   Mean   :  0.1512   Mean   : -61.65   Mean   : 163.7
##  3rd Qu.:  1.62000   3rd Qu.:  0.4900   3rd Qu.:  76.00   3rd Qu.: 312.0
##  Max.   :311.00000   Max.   :231.0000   Max.   : 477.00   Max.   : 923.0
##
##  accel_forearm_z   magnet_forearm_x  magnet_forearm_y magnet_forearm_z
##  Min.   :-446.00   Min.   :-1280.0   Min.   :-896.0   Min.   :-973.0
##  1st Qu.:-182.00   1st Qu.: -616.0   1st Qu.:   2.0   1st Qu.: 191.0
##  Median : -39.00   Median : -378.0   Median : 591.0   Median : 511.0
##  Mean   : -55.29   Mean   : -312.6   Mean   : 380.1   Mean   : 393.6
##  3rd Qu.:  26.00   3rd Qu.:  -73.0   3rd Qu.: 737.0   3rd Qu.: 653.0
##  Max.   : 291.00   Max.   :  672.0   Max.   :1480.0   Max.   :1090.0
##
##  classe
##  A:5580
##  B:3797
##  C:3422
##  D:3216
##  E:3607
##
##
```

```r
# Next, normalizing and centering the variables would
# be good, since there seems to be large differences between the minimums
# and maximums of the variables
# The variable index is excluded since it's only a sequence along row numbers

preProcTrain <- preProcess(filteredTrain[,6:58], method=c("center", "scale"))
normTrain <- predict(preProcTrain, filteredTrain[,6:58])

preProcTest <- preProcess(filteredTest[,6:58], method=c("center", "scale"))
normTest <- predict(preProcTest, filteredTest[,6:58])



# See the correlation between the numeric predictors
highlyCor <- findCorrelation(cor(normTrain), cutoff=.75, verbose=FALSE)
# And we want to remove these to not clutter the analysis
normTrain <- normTrain[,-highlyCor]
normTest <- normTest[, -highlyCor]

# We add back the classe (outcome) predictor
normTrain$classe <- as.factor(filteredTrain$classe)
normTest$classe <- as.factor(filteredTest$problem_id)
```

```r
# Create a validation test partition / Data Slicing
CV <- createDataPartition(normTrain$classe, p=0.70, list=FALSE)

normTrainT <- normTrain[CV,]
normValid <- normTrain[-CV,]
```

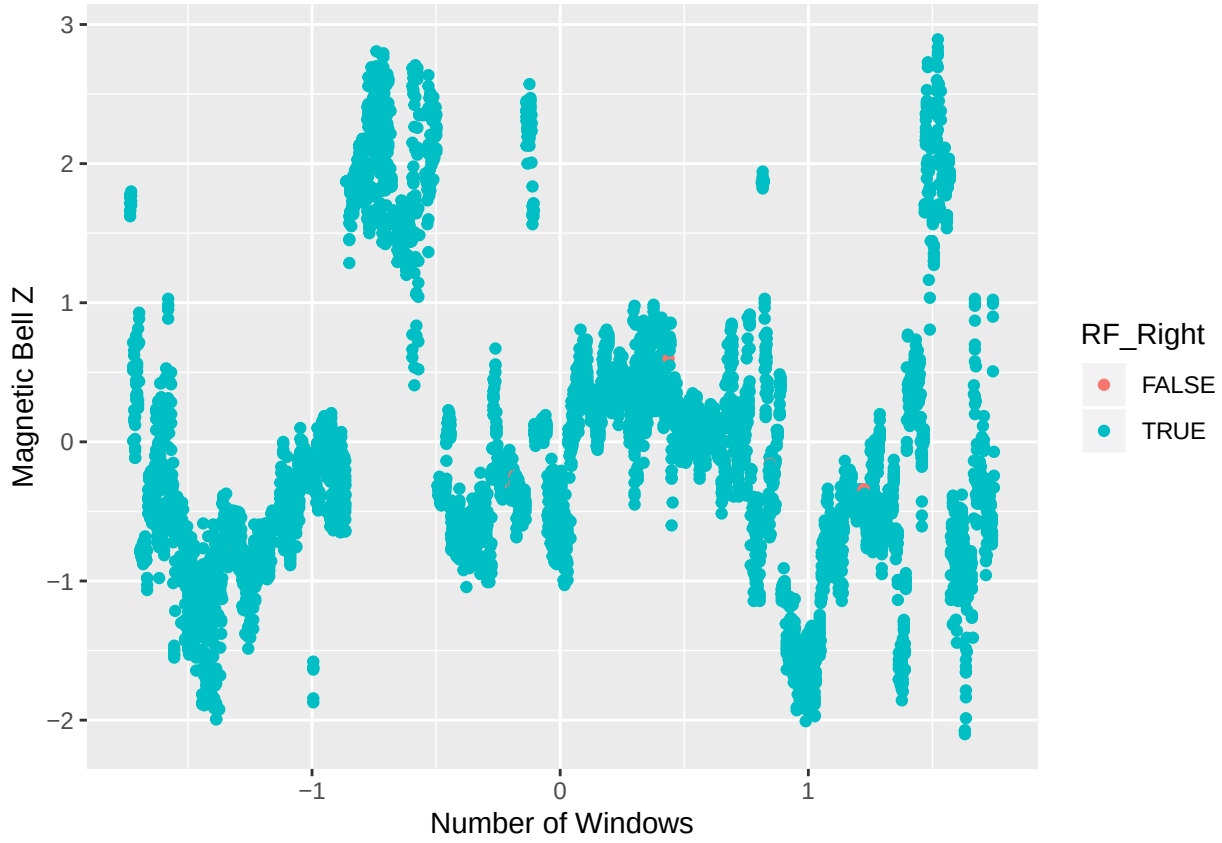
Applying predictors:: Random Forest

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    1    0    0
##          C    0    0 1024    7    0
##          D    0    0    1  957    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9927   1.0000
## Specificity            0.9998   0.9998   0.9986   0.9998   1.0000
## Pos Pred Value         0.9994   0.9991   0.9932   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9986   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1626   0.1839
## Detection Prevalence   0.2846   0.1935   0.1752   0.1628   0.1839
## Balanced Accuracy      0.9999   0.9995   0.9983   0.9963   1.0000
```

```
## Random Forest: 27.91 sec elapsed
```

```
##                        Overall
## num_window           1353.70891
## yaw_belt              896.00691
## gyros_belt_x          157.43027
## gyros_belt_y          161.12174
## gyros_belt_z          388.74167
## magnet_belt_x         289.80813
```

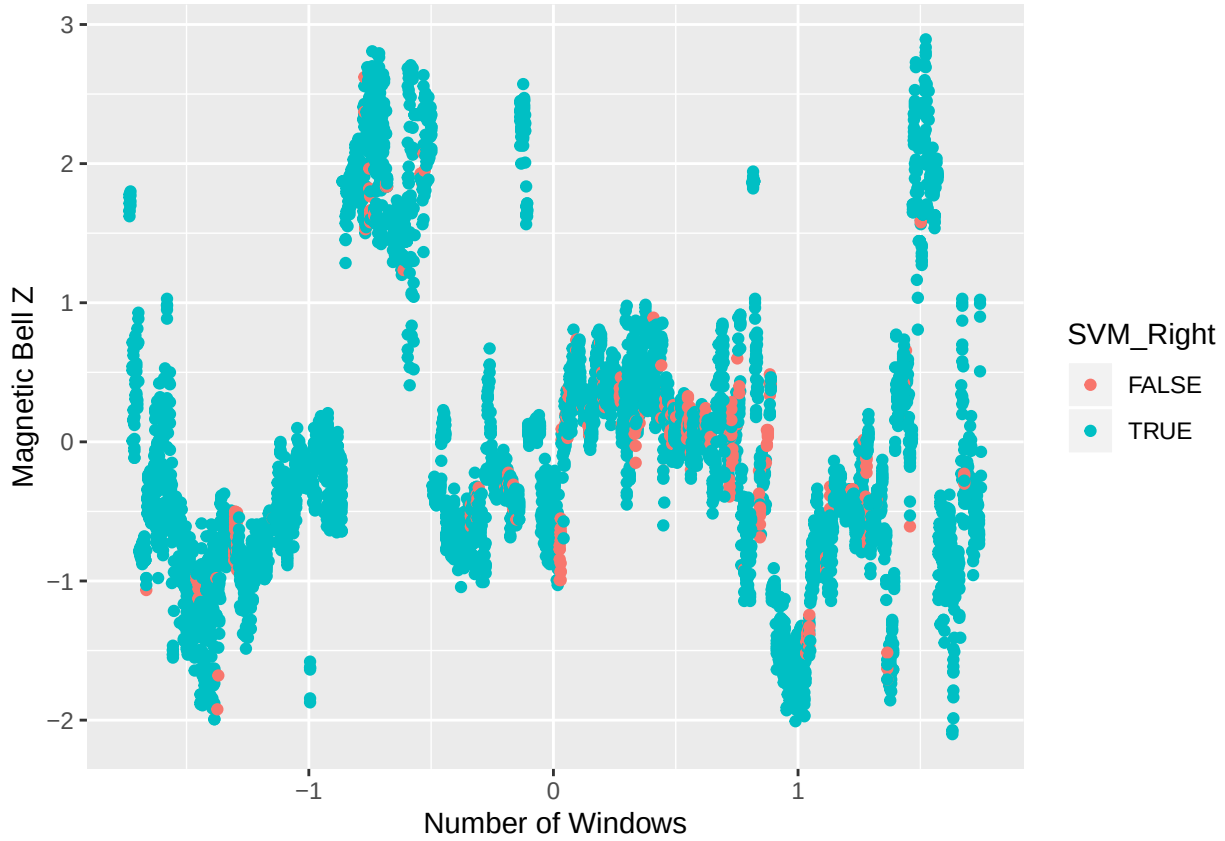
```
## magnet_belt_y          623.49218
## roll_arm               340.58774
## pitch_arm              186.04558
## yaw_arm                234.26490
## total_accel_arm        118.98744
## gyros_arm_y            154.16641
## gyros_arm_z             71.98039
## magnet_arm_x           268.17667
## magnet_arm_z           177.66930
## roll_dumbbell          522.52612
## pitch_dumbbell         275.98720
## yaw_dumbbell           344.03079
## total_accel_dumbbell   352.62439
## gyros_dumbbell_y       321.10667
## magnet_dumbbell_z      688.08889
## roll_forearm           495.54174
## pitch_forearm          652.67108
## yaw_forearm            173.52054
## total_accel_forearm    117.65067
## gyros_forearm_x         95.97449
## gyros_forearm_z         96.74258
## accel_forearm_x        318.53257
## accel_forearm_z        279.34368
## magnet_forearm_x       205.99494
## magnet_forearm_y       219.67639
## magnet_forearm_z       277.83335
```



Applying predictors::SVM

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1658   89    4    2    1
##          B    4 1018   21    0    7
##          C   11   31  963  106   13
##          D    0    0   36  855   29
##          E    1    1    2    1 1032
##
## Overall Statistics
##
##                Accuracy : 0.939
##                  95% CI : (0.9326, 0.945)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9227
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9904   0.8938   0.9386   0.8869   0.9538
## Specificity            0.9772   0.9933   0.9669   0.9868   0.9990
## Pos Pred Value         0.9453   0.9695   0.8568   0.9293   0.9952
## Neg Pred Value         0.9961   0.9750   0.9868   0.9780   0.9897
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2817   0.1730   0.1636   0.1453   0.1754
## Detection Prevalence   0.2980   0.1784   0.1910   0.1563   0.1762
## Balanced Accuracy      0.9838   0.9435   0.9527   0.9369   0.9764

## SVM: 39.74 sec elapsed
```



We can see that Random Forest creates a very accurate prediction of the output class with 0.998 with an out of sample error 0.0003960396. Nevertheless, both algorithms perform very well in predicting the outcome at 0.939 and above (SVM is 0.9397) The running time of the algorithms is much worse with SVM taking 43 seconds, and the Random Forest algorithm taking 24 seconds Last thing is to apply the model to produce the results of the test data with the test data When I applied the RF model to the test data set, the actual results were much worse at the testing dataset

RESULTS

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  E  A  A  A  D  B  D  B  A  A  B  C  D  A  E  E  E  B  B  D
## Levels: A B C D E
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  A  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

The validation test got good results But if the algorithm performs better on the test set (that we know the results for) It is better to use the SVM model