Machine Learning Course Project

Bijesh Rajamohanan

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 (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
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(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 (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.

Exploratory Analysis

```
cat("\014")
```

```
rm(list=ls())

set.seed(345)
training = read.csv("C:\\datascience\\ML\\week4\\assignment\\pml-
training.csv",na.strings=c("NA","#DIV/0!", ""))

testing = read.csv("C:\\datascience\\ML\\week4\\assignment\\pml-
testing.csv",na.strings=c("NA","#DIV/0!", ""))


library(caret)
```

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

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

```
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(e1071)
library(gbm)
```

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

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

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

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

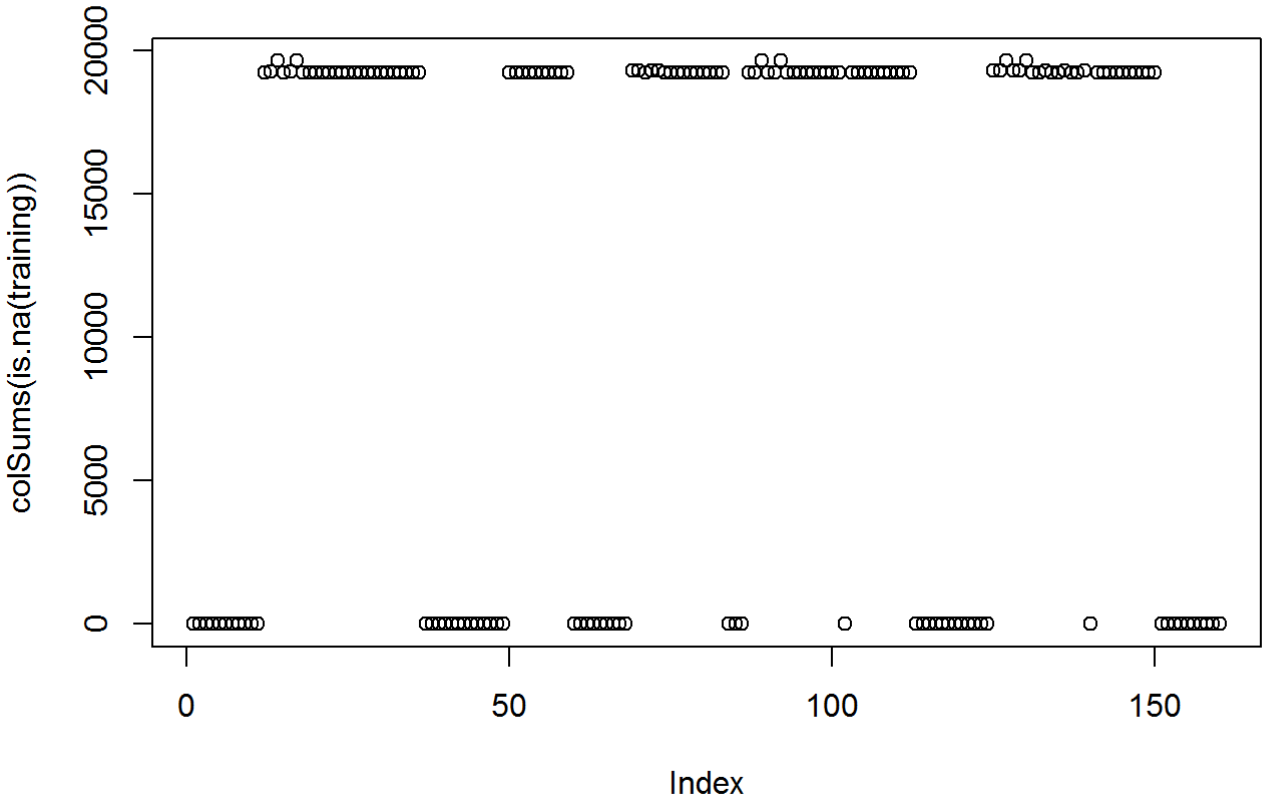
```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
dim(training)
```

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

```
plot(colSums(is.na(training)))
```



Clearly, there are variables with most of the values being invalid. We fix a threshold from the plot to get rid of the useless variables.

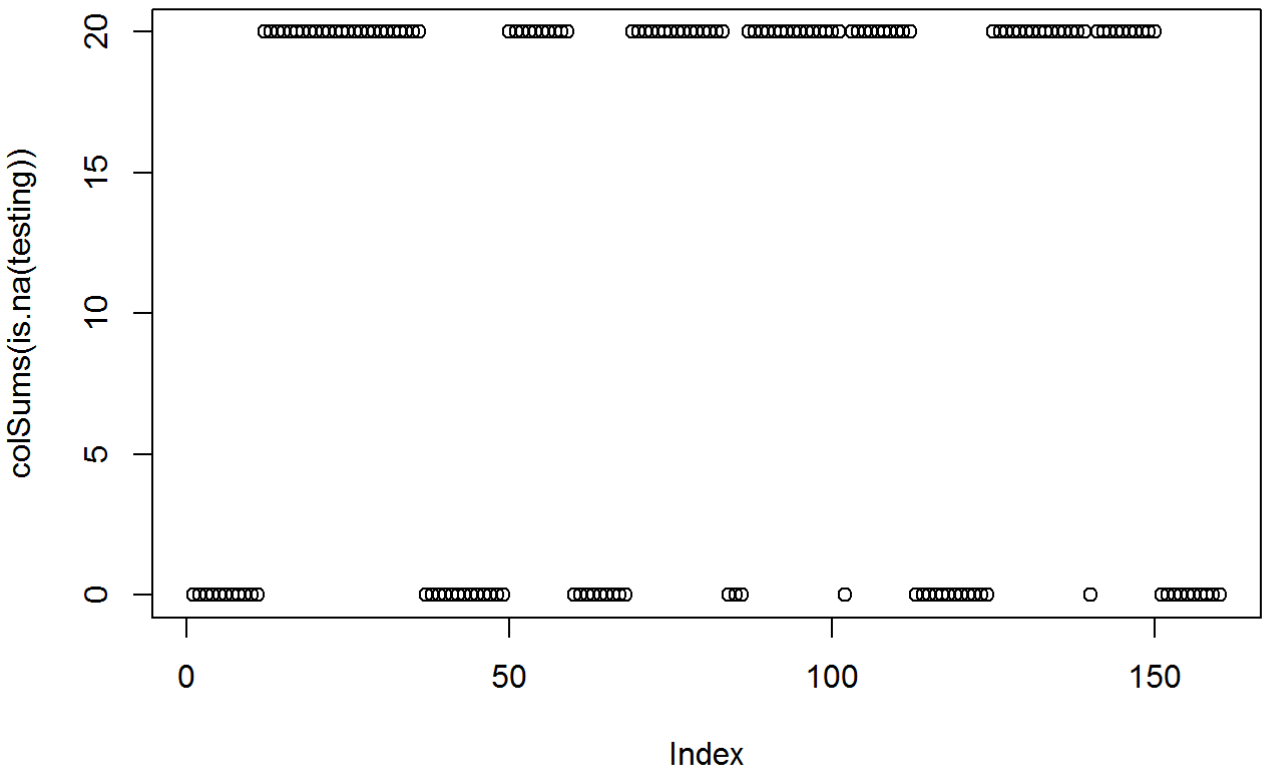
```
training<-training[,colSums(is.na(training))<19000]

dim(training)
```

```
## [1] 19622    60
```

Similary we clean up the testing data set as well

```
plot(colSums(is.na(testing)))
```




```
testing<-testing[,colSums(is.na(testing))!=20]
dim(testing)
```

```
## [1] 20 60
```

We further take a look at the variable names in both testing and training set, and we get rid of other irrelevant variables as shown below

```
colnames(training)
```

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

```
colnames(testing)
```

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

```
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
```

Cross Validation Data Partioning

For cross validation, we divide the training set to another training set and testing set as shown below

```
cvsamples <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
cvtraining <- training[cvsamples, ]
cvtesting <- training[-cvsamples, ]
```

Prediction Models

Three models were tried: random forest and support vector machine as shown below

```
mod_rf<- randomForest(classe ~. , data=cvtraining, method="class")
mod_svm <- svm(classe ~. , data=cvtraining)
pred_rf <- predict(mod_rf, cvtesting, type = "class")
confusionMatrix(pred_rf, cvtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    2    0    0    0
##          B    0  943    3    0    0
##          C    0    4  852    6    0
##          D    0    0    0  798    1
##          E    1    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9937   0.9965   0.9925   0.9989
## Specificity            0.9994   0.9992   0.9975   0.9998   0.9998
## Pos Pred Value         0.9986   0.9968   0.9884   0.9987   0.9989
## Neg Pred Value         0.9997   0.9985   0.9993   0.9985   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1923   0.1737   0.1627   0.1835
## Detection Prevalence   0.2847   0.1929   0.1758   0.1629   0.1837
## Balanced Accuracy      0.9994   0.9965   0.9970   0.9961   0.9993
```

```
pred_svm <- predict(mod_svm, cvtesting, type = "class")
confusionMatrix(pred_svm, cvtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1388   71    3    4    0
##          B    1  859   14    0    5
##          C    3   18  829   80   25
##          D    0    0    7  717   20
##          E    3    1    2    3  851
##
## Overall Statistics
##
##                Accuracy : 0.947
##                  95% CI : (0.9403, 0.9531)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9328
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9950   0.9052   0.9696   0.8918   0.9445
## Specificity            0.9778   0.9949   0.9689   0.9934   0.9978
## Pos Pred Value         0.9468   0.9772   0.8681   0.9637   0.9895
## Neg Pred Value         0.9980   0.9776   0.9934   0.9791   0.9876
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2830   0.1752   0.1690   0.1462   0.1735
## Detection Prevalence   0.2989   0.1792   0.1947   0.1517   0.1754
## Balanced Accuracy      0.9864   0.9501   0.9692   0.9426   0.9711
```

Clearly, random forest method gives the best accuracy, although the other method also gave comparable accuracy levels. Hence we use the random forest model and apply on the test set. ### Final Prediction

```
predict(mod_rf, testing)
```

```
##  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  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```