Practical Machine Learning Assignment

Inder Chettri

Synopsis

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).

The data for this assignment has been obtained from - Training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv - Test: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

This has been downloaded to ./MachineLearning folder on local machine.

Exploring the data

Loading required libraries

```
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
set.seed(12345)
```

We will load the files into memory removing NA values.

```
#setwd("./machine_learning")
Trn <- read.csv("./MachineLearning/pml-training.csv", header=T,
na.strings=c("NA", "#DIV/0!"))
Test <- read.csv("./MachineLearning/pml-testing.csv", header=T,
na.string=c("NA", "#DIV/0!"))
```

Evaluating the training set, we see a number of columns with NA values.

```
dim(Trn)
## [1] 19622   160
str(Trn)
## 'data.frame':    19622 obs. of  160 variables:
```

```
##  $ X                        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name                : Factor w/ 6 levels "adelmo","carlitos",..: 2
2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1     : int  1323084231 1323084231 1323084231
1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232
...
##  $ raw_timestamp_part_2     : int  788290 808298 820366 120339 196328
304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp           : Factor w/ 20 levels "02/12/2011 13:32",..: 9
9 9 9 9 9 9 9 9 9 ...
##  $ new_window               : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1
1 1 1 ...
##  $ num_window               : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt                : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42
1.43 1.45 ...
##  $ pitch_belt               : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13
8.16 8.17 ...
##  $ yaw_belt                 : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -
94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt        : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt        : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x             : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02
0.03 ...
##  $ gyros_belt_y             : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z             : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -
0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x             : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21
...
##  $ accel_belt_y             : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z             : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x            : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y            : int  599 608 600 604 600 603 599 603 602 609
...
```

```
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313
-312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128
-128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8
21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161
-161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02
...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -
0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -
0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289
-288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110
...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124
-122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372
-369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334
...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516
...
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Preprocessing

There are a number of columns with NA values. These have to be removed. Also removing the first 7 columns as they do not contribute to predicting the outcome. Carrying out the same operations on the test set.

```
RemCol <- which(colSums(is.na(Trn))!=0)
Trn <- Trn[, -RemCol]
Trn <- Trn[,-(1:7)]
Test <- Test[, -RemCol]
Test <- Test[,-(1:7)]
dim(Trn)
## [1] 19622    53
```

Our new Training dataset now has 53 columns.

We will partition this data to create a new training and test sets.

```
NT <- createDataPartition(y=Trn$classe, p=0.75,list=FALSE)
NTrn <- Trn[NT,]
NTst <- Trn[-NT,]
dim(NTrn)
## [1] 14718    53
dim(NTst)
## [1] 4904   53
```

Training the Model

We will use 2 methods to to train the model

Boosting with trees using PCA for preprocessing

```
model1 <- train(classe~., data=NTrn, method='gbm', preProcess='pca', verbose
= FALSE)
```

Predicting the outcome for the partitioned test set.

```
print(model1, digits = 3)
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction, scaled, centered
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa  Accuracy SD  Kappa SD
##   1                   50      0.553     0.424  0.00864      0.0111
##   1                  100      0.612     0.504  0.00898      0.0113
##   1                  150      0.642     0.543  0.00978      0.0121
##   2                   50      0.651     0.554  0.01000      0.0126
##   2                  100      0.718     0.641  0.00842      0.0106
##   2                  150      0.756     0.690  0.00951      0.0120
##   3                   50      0.707     0.627  0.00941      0.0120
##   3                  100      0.771     0.710  0.00891      0.0113
##   3                  150      0.808     0.756  0.00757      0.0096
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3 and shrinkage = 0.1.
pr1 <- predict(model1, newdata = NTst)
## Loading required package: 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
## Loading required package: plyr
confusionMatrix(pr1,NTst$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1257  106   53   25   27
##          B   37  710   61   25   70
##          C   39   99  710  105   62
##          D   50   15   13  628   38
##          E   12   19   18   21  704
##
## Overall Statistics
##
##                Accuracy : 0.8175
```

```
##                  95% CI : (0.8064, 0.8282)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7688
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9011   0.7482   0.8304   0.7811   0.7814
## Specificity            0.9399   0.9512   0.9247   0.9717   0.9825
## Pos Pred Value         0.8563   0.7863   0.6995   0.8441   0.9096
## Neg Pred Value         0.9598   0.9403   0.9627   0.9577   0.9523
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2563   0.1448   0.1448   0.1281   0.1436
## Detection Prevalence   0.2993   0.1841   0.2070   0.1517   0.1578
## Balanced Accuracy      0.9205   0.8497   0.8775   0.8764   0.8819
```

Ramdom forest

```
FCtl <- trainControl(method="cv", number=5, allowParallel=TRUE,
verbose=FALSE)
model2<-train(classe~.,data=NTrn, method="rf", trControl=FCtl, verbose=FALSE)
print(model2, digits = 3)
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 11774, 11775, 11774, 11773, 11776
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.992     0.990  0.001771     0.002240
##   27    0.992     0.990  0.000705     0.000891
##   52    0.990     0.987  0.001549     0.001959
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
pr2 <- predict(model2,newdata = NTst)
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
confusionMatrix(pr2,NTst$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
```

```
##          B    0  937    3    0    0
##          C    0    3  852   11    1
##          D    0    0    0  793    3
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9913, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9874   0.9965   0.9863   0.9956
## Specificity            0.9974   0.9992   0.9963   0.9993   1.0000
## Pos Pred Value         0.9936   0.9968   0.9827   0.9962   1.0000
## Neg Pred Value         1.0000   0.9970   0.9993   0.9973   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1911   0.1737   0.1617   0.1829
## Detection Prevalence   0.2863   0.1917   0.1768   0.1623   0.1829
## Balanced Accuracy      0.9987   0.9933   0.9964   0.9928   0.9978
```

Out of sample error

Boosting with trees gives us an out of sample accuracy of 81.75% and random forest gives us out of sample accuracy of 99.39%. Therefore the out of sample error for boosting was 18.25% and for random forest was 0.61%

We will go with random forest to predict values in the test set.

Applying selected model to the provided test set

Prediction with the provided test set yields the following result.

```
prT <- predict(model2,newdata = Test[,-53])
prT
##  [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
```

Writing out the output

```
pml_write_files = function(x){
  n = length(x)
```

```
  for(i in 1:n){
    filename = paste0("./MachineLearning/problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(prT)
```