Peer Graded Assignment: Prediction Assignment Writeup

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).

Project goal

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Getting the data

```
train_URL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv"
test_URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv"

download.file(train_URL, destfile="train.csv", method="curl")
download.file(test_URL, destfile="test.csv", method="curl")

train_dataset <- read.csv("train.csv")
test_dataset <- read.csv("test.csv")
```

Presenting the basic characteristics of the datasets

```
dim(train_dataset)
```

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

```
dim(test_dataset)
```

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

```
names(train_dataset)
```

```
##   [1] "X"                       "user_name"
##   [3] "raw_timestamp_part_1"    "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"          "new_window"
##   [7] "num_window"              "roll_belt"
##   [9] "pitch_belt"              "yaw_belt"
##  [11] "total_accel_belt"        "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"     "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"      "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"       "max_roll_belt"
##  [19] "max_picth_belt"          "max_yaw_belt"
##  [21] "min_roll_belt"           "min_pitch_belt"
##  [23] "min_yaw_belt"            "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"    "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"    "avg_roll_belt"
##  [29] "stddev_roll_belt"        "var_roll_belt"
##  [31] "avg_pitch_belt"          "stddev_pitch_belt"
##  [33] "var_pitch_belt"          "avg_yaw_belt"
##  [35] "stddev_yaw_belt"         "var_yaw_belt"
##  [37] "gyros_belt_x"            "gyros_belt_y"
##  [39] "gyros_belt_z"            "accel_belt_x"
##  [41] "accel_belt_y"            "accel_belt_z"
##  [43] "magnet_belt_x"           "magnet_belt_y"
##  [45] "magnet_belt_z"           "roll_arm"
##  [47] "pitch_arm"               "yaw_arm"
##  [49] "total_accel_arm"         "var_accel_arm"
##  [51] "avg_roll_arm"            "stddev_roll_arm"
##  [53] "var_roll_arm"            "avg_pitch_arm"
##  [55] "stddev_pitch_arm"        "var_pitch_arm"
##  [57] "avg_yaw_arm"             "stddev_yaw_arm"
##  [59] "var_yaw_arm"             "gyros_arm_x"
##  [61] "gyros_arm_y"             "gyros_arm_z"
##  [63] "accel_arm_x"             "accel_arm_y"
##  [65] "accel_arm_z"             "magnet_arm_x"
##  [67] "magnet_arm_y"            "magnet_arm_z"
##  [69] "kurtosis_roll_arm"       "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"        "skewness_roll_arm"
##  [73] "skewness_pitch_arm"      "skewness_yaw_arm"
##  [75] "max_roll_arm"            "max_picth_arm"
##  [77] "max_yaw_arm"             "min_roll_arm"
##  [79] "min_pitch_arm"           "min_yaw_arm"
##  [81] "amplitude_roll_arm"      "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"       "roll_dumbbell"
##  [85] "pitch_dumbbell"          "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"  "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"   "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell" "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"       "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"        "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"      "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell" "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"  "total_accel_dumbbell"
## [103] "var_accel_dumbbell"      "avg_roll_dumbbell"
```

```
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

removing the first 7 unrelated colums

```
train_dataset <- train_dataset[ , -c(1:7)]
test_dataset <- test_dataset[ , -c(1:7)]
```

removing NA values

```
v_NA <- sapply(train_dataset, function (x) any(is.na(x) | x == ""))
train_dataset <- train_dataset[, names(v_NA[!v_NA])]
```

staying with 53 features left

```
dim(train_dataset)
```

```
## [1] 19622    53
```

building a model

let's try not to be too fancy, I choose the Random Forest model

loading necessary libraries

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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)
```

```
## Warning: package 'e1071' was built under R version 3.2.5
```

dividing the training dataset 60/40

```
set.seed(730108)
ts <- createDataPartition(train_dataset$classe, p = 0.60, list = FALSE)
train_set <- train_dataset[ts, ]
test_set <- train_dataset[-ts, ]
```

actual training

```
ctr <- trainControl(method = "cv", 5)
fit <- train(classe ~ ., data = train_set, method = "rf", trControl = ctr, n
tree = 250)
```

presenting the training results & applying them to the training dataset

```
fit
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9421, 9421, 9421, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9876867  0.9844216
##   27    0.9888758  0.9859269
##   52    0.9802989  0.9750789
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
p <- predict(fit, test_set)
confusionMatrix(test_set$classe, p)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    3    0    0    1
##          B   16 1497    5    0    0
##          C    0    5 1357    6    0
##          D    1    1   11 1272    1
##          E    0    0    3    4 1435
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9906, 0.9945)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9924   0.9940   0.9862   0.9922   0.9986
## Specificity            0.9993   0.9967   0.9983   0.9979   0.9989
## Pos Pred Value         0.9982   0.9862   0.9920   0.9891   0.9951
## Neg Pred Value         0.9970   0.9986   0.9971   0.9985   0.9997
## Prevalence             0.2861   0.1919   0.1754   0.1634   0.1832
## Detection Rate         0.2840   0.1908   0.1730   0.1621   0.1829
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9959   0.9954   0.9922   0.9950   0.9988
```

Accurancy over 99% looks too optimistic but well…

Applying the trained model to the test dataset of 20 records

```
test_results <- predict(fit, test_dataset)
test_results
```

```
##  [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
```

Suprisingly the Assignment Quiz gives 20/20 for 100% result.