Practical Machine Learning Course Project

Kim Paulo D. Magboo

December 12, 2018

Overview

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, the goal is 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 Sources

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).

Datasets

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(knitr)

set.seed(12345)
url1<-
train <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),he
ader=TRUE)
dim(train)
```

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

```
test <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),head
er=TRUE)
dim(test)
```

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

Data Partition

```
inTrain <- createDataPartition(train$classe, p=0.7, list=FALSE)
myTraining <- train[inTrain, ]
myTesting <- train[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 13737   160
```

```
## [1] 5885  160
```

```
str(myTraining)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  2 3 4 5 6 7 8 12 13 14 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2
...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084232 1323084232 1323084232 1323
084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  808298 820366 120339 196328 304277 368296 440390 528316 560
359 576390 ...
##  $ 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 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.42 1.42 ...
##  $ pitch_belt              : num  8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.18 8.2 8.21 ...
##  $ 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      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1
...
##  $ 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  0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_belt_y            : num  0 0 0 0.02 0 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 ...
##  $ accel_belt_x            : int  -22 -20 -22 -21 -21 -22 -22 -22 -22 -22 ...
##  $ accel_belt_y            : int  4 5 3 2 4 3 4 2 4 4 ...
##  $ accel_belt_z            : int  22 23 21 24 21 21 21 23 21 21 ...
##  $ magnet_belt_x           : int  -7 -2 -6 -6 0 -4 -2 -2 -3 -8 ...
##  $ magnet_belt_y           : int  608 600 604 600 603 599 603 602 606 598 ...
##  $ magnet_belt_z           : int  -311 -305 -310 -302 -312 -311 -313 -319 -309 -310 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
```

```
##  $ pitch_arm               : num  22.5 22.5 22.1 22.1 22 21.9 21.8 21.5 21.4 21.4 ...
##  $ 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  0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.02 0 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 0.02 0 0 0 0 0 -0.02 -0.03 ...
##  $ accel_arm_x             : int  -290 -289 -289 -289 -289 -289 -289 -288 -287 -288 ...
##  $ accel_arm_y             : int  110 110 111 111 111 111 111 111 111 111 ...
##  $ accel_arm_z             : int  -125 -126 -123 -123 -122 -125 -124 -123 -124 -124 ...
##  $ magnet_arm_x            : int  -369 -368 -372 -374 -369 -373 -372 -363 -372 -371 ...
##  $ magnet_arm_y            : int  337 344 344 337 342 336 338 343 338 331 ...
##  $ magnet_arm_z            : int  513 513 512 506 513 509 510 520 509 523 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 12.9 13.4 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.6 -70.3 -70.4 -70.4 -70.8 ...
##  $ yaw_dumbbell            : num  -84.7 -85.1 -84.9 -84.9 -84.5 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Clean-up

```
#Remove Variables with more than 90% NA using the Training dataset
indColToRemove <- which(colSums(is.na(myTraining)|myTraining=="")>0.9*dim(myTraining)[1])
Train_Clean <- myTraining[,-indColToRemove]
Train_Clean <- Train_Clean[,-c(1:7)]
dim(Train_Clean)
```

```
## [1] 13737    53
```

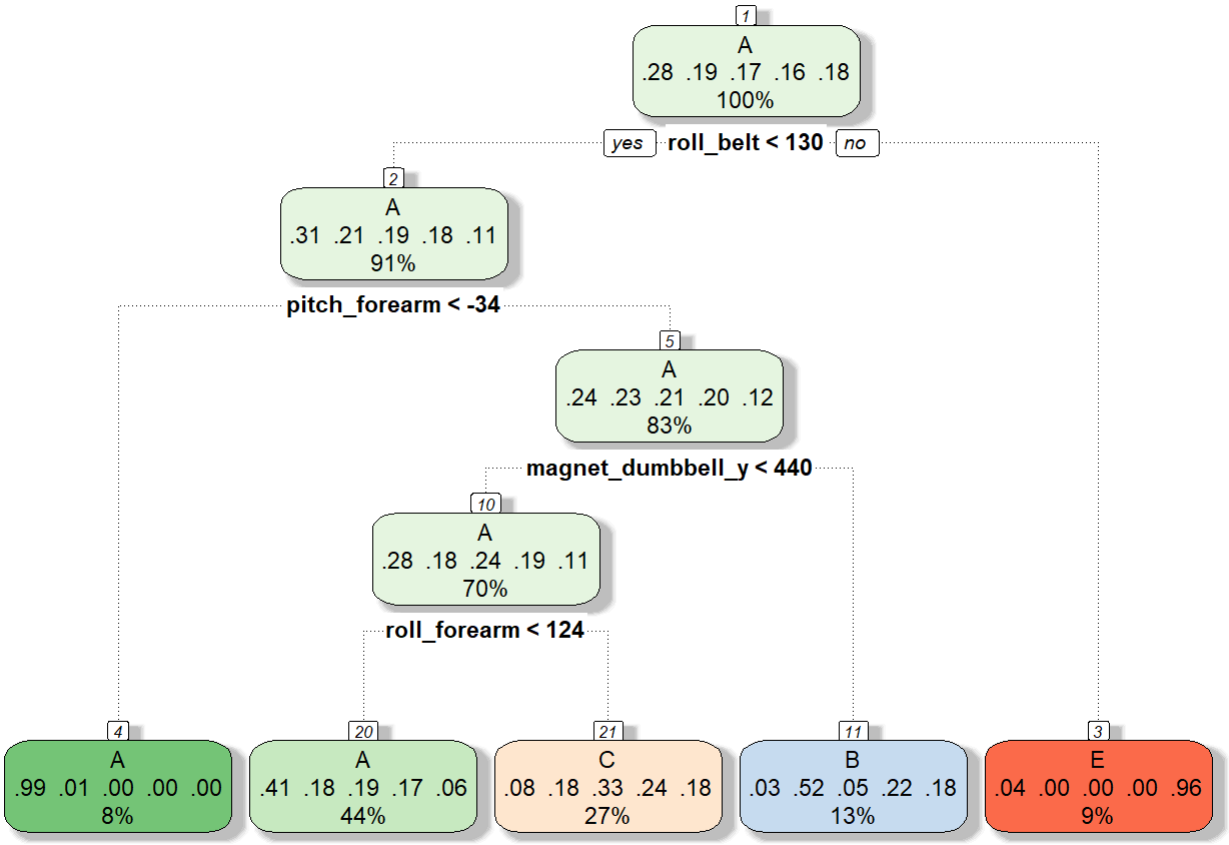
```
#Remove Variables with more than 90% NA using the Test dataset
indColToRemove <- which(colSums(is.na(myTesting)|myTesting=="")>0.9*dim(myTesting)[1])
Test_Clean <- myTesting[,-indColToRemove]
Test_Clean <- Test_Clean[,-1]
dim(Test_Clean)
```

```
## [1] 5885   59
```

Model Estimation

In this section, we will develop 2 models: decision tree and random forest.Cross-validation with 3 folds will be utilized to limit the effects of overfitting and improve the accuracy of the models.

```
#Model Development using Decision Tree
trControl <- trainControl(method="cv", number=3)
dt <- train(classe~., data=Train_Clean, method="rpart", trControl=trControl)
fancyRpartPlot(dt$finalModel)
```



Rattle 2018-Dec-13 03:47:44 dabidoo's

```
trainpred <- predict(dt,newdata=Test_Clean)
confMatdt <- confusionMatrix(Test_Clean$classe,trainpred)
confMatdt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1494   21  128    0   31
##          B  470  380  289    0    0
##          C  467   29  530    0    0
##          D  438  184  342    0    0
##          E  141  147  277    0  517
##
## Overall Statistics
##
##                Accuracy : 0.4963
##                  95% CI : (0.4835, 0.5092)
##     No Information Rate : 0.5115
##     P-Value [Acc > NIR] : 0.9902
##
##                   Kappa : 0.3425
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4963  0.49934  0.33844       NA  0.94343
## Specificity            0.9374  0.85187  0.88516   0.8362  0.89414
## Pos Pred Value         0.8925  0.33363  0.51657       NA  0.47782
## Neg Pred Value         0.6400  0.91972  0.78679       NA  0.99355
## Prevalence             0.5115  0.12931  0.26610   0.0000  0.09312
## Detection Rate         0.2539  0.06457  0.09006   0.0000  0.08785
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7169  0.67561  0.61180       NA  0.91878
```

```
#Model Development using Random Forest
rf <- train(classe~., data=Train_Clean, method="rf", trControl=trControl, verbose=FALSE)
print(rf)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9157, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9892262  0.9863686
##   27    0.9886440  0.9856329
##   52    0.9820924  0.9773428
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
trainpred <- predict(rf,newdata=Test_Clean)
confMatRF <- confusionMatrix(Test_Clean$classe,trainpred)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   11 1122    6    0    0
##          C    0   13 1009    4    0
##          D    0    0   24  940    0
##          E    0    0    1    3 1078
##
## Overall Statistics
##
##                Accuracy : 0.9893
##                  95% CI : (0.9863, 0.9918)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9865
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.9877   0.9702   0.9926   1.0000
## Specificity            0.9998   0.9964   0.9965   0.9951   0.9992
## Pos Pred Value         0.9994   0.9851   0.9834   0.9751   0.9963
## Neg Pred Value         0.9974   0.9971   0.9936   0.9986   1.0000
## Prevalence             0.2862   0.1930   0.1767   0.1609   0.1832
## Detection Rate         0.2843   0.1907   0.1715   0.1597   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9966   0.9920   0.9833   0.9939   0.9996
```

Based on the results, the model using random forest obtained higher accuracy rate of 99.15% as compared to that of the model with an accuract rate of 49.24%. Using the random forest model, the estimated out-of sample error is about 0.85%.

Prediction

```
predValidation <- predict(rf, newdata=test)
Results <- data.frame(
  problem_id=test$problem_id,
  predicted=predValidation
)
print(Results)
```

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

```
Results<- as.character (Results)

#Create a text file for the Prediction
write_files <- function(x) {
    n <- length(x)
    for(i in 1:n) {
        filename <- paste0("problem_id_", i, ".txt")
        write.table(x[i], file=filename, quote=F, row.names=F, col.names=F)
    }
}

write_files(Results)
```