Pratiacal-Machine_Learning-Project

Enrique Estrada

September 13, 2018

Practical Machine Learning Course Project

Introduction

Data for this project came from the Human Activity Recognition project from Groupware@LES. http://groupware.les.inf.puc-rio.br/har.

The background for the research is that a number of human subjects performed weightlifting exercises while wearing a number of sensors to track movement. The objective of the research project was to predict how well a particular exercise would be performed, given the multitude of sensor inputs. The possible outcomes were:

Class A: correct, done according to the specification Class B: incorrect: elbows thrown to the front **Class C: incorrect: dumbbell lifted only halfway** Class D: incorrect: dumbbell lowered only halfway **Class E: incorrect: hips thrown to the front.

**There are 160 variables in the data set

Load libraries & Data

Data Download

```
training_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
# download training dataset if it doesn't exist locally
if(!file.exists("./pml-training.csv")) {
  download.file(training_url,destfile = "./pml-training.csv", method = "curl")
}

# download testing dataset if it doesn't exist locally
if(!file.exists("./pml-testing.csv")) {
  download.file(testing_url,destfile = "./pml-testing.csv", method = "curl")
}

# During the data import to R, strings matching "NA" and "#DIV/0!" were coerced to NA values in R.
trainingDataSet <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))

testingDataSet<- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
```

Explorattory Analysis, Data Cleaning, Remove NA's, and Additional Columns

```
#Exploratory Analysis
dim(trainingDataSet)
```

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

```
#[1] 19622   160
dim(testingDataSet)
```

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

```
# [1]  20 160


colnames(trainingDataSet)[1] <- "observationId"
names(trainingDataSet)
```

```
##   [1] "observationId"            "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"
```

```
names(testingDataSet)
```

```
##   [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"         "problem_id"
```

** Machine learning algorithms work on data without missing values, thus variables with missing values were eliminated.

```
# create a T/F vector identify variables with at least one NA
      missingcols <- sapply(trainingDataSet, function(x) { any(is.na(x)) })

# replace data by keeping only those variables that don't have missing data
      trainingDataSet<- trainingDataSet[ , !missingcols]
      testingDataSet<- testingDataSet[ , !missingcols]
```

Cross Validation

In this method, we randomly divide the available data into two parts, a training set, and a validation (test) set. The model is fit on the training set, then the fitted model is used to predict the responses for the observations in the validation set.

The original dataset pml-training.csv dataset will be randomly sliced into two parts: **a training set (70%) and a test set (30%).

```
# set seed
      set.seed(123)

# Create training set indexes with 70% of data
      inTrain <- caret::createDataPartition(y = trainingDataSet$classe, p = 0.7, list = FALSE)

# subsets
      training <- trainingDataSet[inTrain, ]
      testing <- trainingDataSet[-inTrain, ]
      dim(training) ; dim(testing)
```

```
## [1] 13737    60
```

```
## [1] 5885   60
```

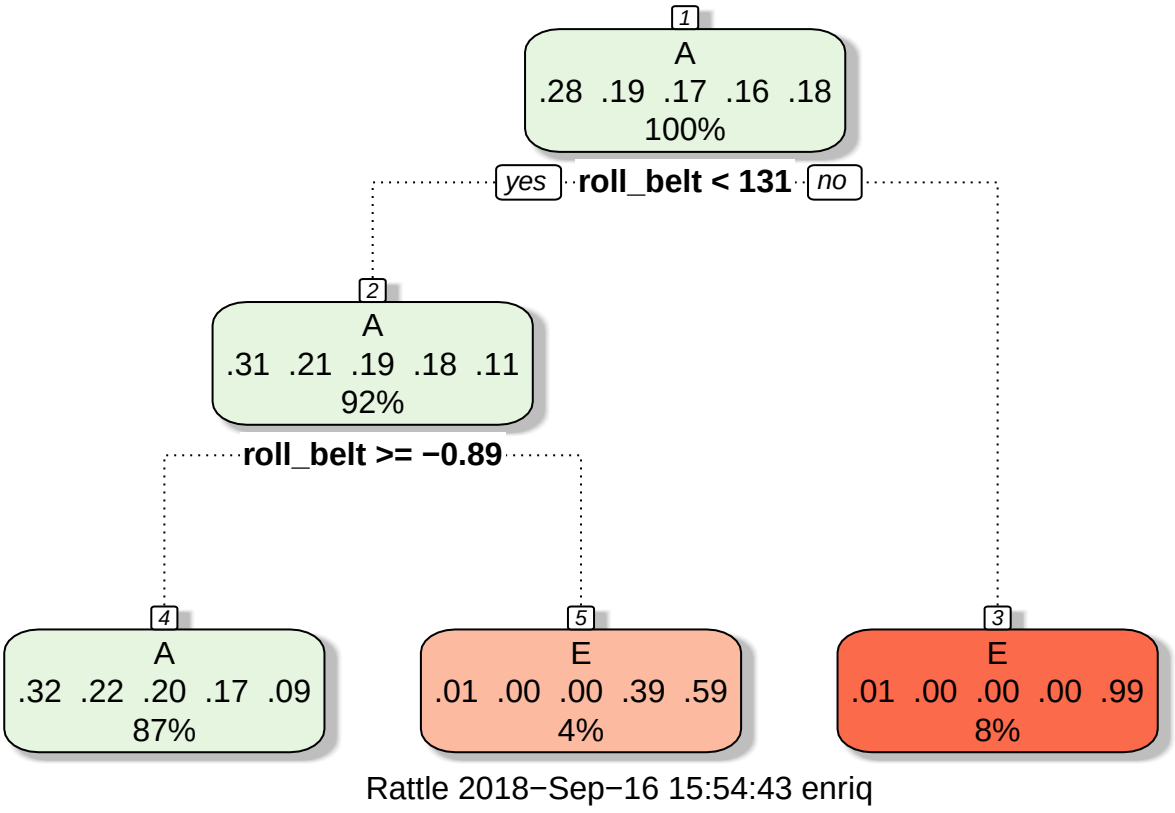
Prediction Model Building (Test of prediction Models)

The Two algorithm are applied: ** Decision Tree ** Random Forest

Decision Tree

```
# simple decision tree model, isolating the class (60) as the outcome and  variables 8 - 11 as predicto
      modFit <- caret::train(classe ~ ., method = "rpart", data = training[,c(8:11,60)])

      rattle::fancyRpartPlot(modFit$finalModel)
```




```
# Perform prediction against the test portion of the training data
      predictions <- predict(modFit, newdata = testing[,c(8:11,60)])

# Evaluate prediction against known classification
      confusionMatrix(predictions, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1664 1139 1026  874  439
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E   10    0    0   90  643
##
## Overall Statistics
##
##                Accuracy : 0.392
```

```
##                  95% CI : (0.3795, 0.4046)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1651
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9940   0.0000   0.0000   0.0000   0.5943
## Specificity            0.1741   1.0000   1.0000   1.0000   0.9792
## Pos Pred Value         0.3236      NaN      NaN      NaN   0.8654
## Neg Pred Value         0.9865   0.8065   0.8257   0.8362   0.9146
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2828   0.0000   0.0000   0.0000   0.1093
## Detection Prevalence   0.8737   0.0000   0.0000   0.0000   0.1263
## Balanced Accuracy      0.5840   0.5000   0.5000   0.5000   0.7867
```

**The model preforms poorly with a overall accuracy of 40%

Random Forest

```
# random forest using all predictors using
      modFit.rf <- randomForest::randomForest(classe ~ ., data = training[,c(8:60)])
      tr <- trainControl(method = "repeatedcv", number = 5 )
      modFit.rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training[, c(8:60)])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.56%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    2    0    0    2 0.001024066
## B   12 2639    7    0    0 0.007148232
## C    1   18 2375    2    0 0.008764608
## D    0    0   24 2226    2 0.011545293
## E    0    0    3    4 2518 0.002772277
```

```
# Perform prediction against the test portion of the training data
      predictions.rf <- predict(modFit.rf, newdata = testing[,c(8:60)])

# Evaluate prediction against known classification
      confusionMatrix(predictions.rf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1673    5    0    0    0
##          B    1 1134   13    0    0
##          C    0    0 1013   13    0
##          D    0    0    0  950    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9956   0.9873   0.9855   1.0000
## Specificity            0.9988   0.9971   0.9973   1.0000   0.9998
## Pos Pred Value         0.9970   0.9878   0.9873   1.0000   0.9991
## Neg Pred Value         0.9998   0.9989   0.9973   0.9972   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1927   0.1721   0.1614   0.1839
## Detection Prevalence   0.2851   0.1951   0.1743   0.1614   0.1840
## Balanced Accuracy      0.9991   0.9963   0.9923   0.9927   0.9999
```

** The accuracy for Random Forest was significantly better at 99.3%
** Based on this approach, we choose to use the Random Forest model to predict against the test data s

Applying the Selected Model on the 20 test cases provided

In this section, we use random forest model we built in last section to predict the test data and output the result into text files.

```
# Perform prediction against the Random Forest model
final_prediction <- predict(modFit.rf,testingDataSet)

# Final prediction
final_prediction
```

```
##  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
```