Practical Machine Learning

Saeed Nusri

4/08/2017

Executive Summary

This reports presents Practical Machine Learning Project that involves applying various machine learning algorithms to Human Activity Recognition dataset to predict the quality of excercise based on the acquired raw data by activity tracker. More precisely, the Weight Lifting Exercises Dataset has been used to predict the movement of athlete and how likely they mistake during repetition of certain barbell weight lifting movement.

Introduction

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).

Data Acquisition & R packages

The project uses the following packages for datacleaning, exploration and machine learning.

```r
library(dplyr)
library(caret)
library(rpart)
library(rattle)
library(randomForest)
```

The Weight Lifting Exercises datasets is acquired from the groupware website and stored as the training and testing dataset in the working directory under its respective names

```r
#Training dataset
if(!file.exists("training")){
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "training", method = "curl")
}

training <- read.csv("training", header = TRUE)

#Testing dataset
if(!file.exists("testing")){
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              destfile = "testing", method = "curl")
}
```

```r
testing <- read.csv("testing", header = TRUE)
```

Once the data is obtained the training dataset is used to do exploratory data analysis

Data Cleaning and Exploratory Data Analysis

It is important to clean the data and find the covariates that would provide us with the prediction model. For simplicity, same tranformations are carried out on both, training and testing datasets.

```r
#Name of all the variables
colnames(training)
```

```
##   [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"
```

As seen many of variables are unncessary for the prediction algorithm. These include, for instance, the average values, standard deviation and variance - values with least amount of variance are removed. Also, zero value columns are not required for the algorithm. The following RScript removes redundant values

```r
near_zero_variance <- nearZeroVar(training)

training <- training[,-near_zero_variance]

training <- training[, colSums(is.na(training))==0]
```

Even the first 6 columns are removed from the dataset

```r
training <- training[,-c(1,2,3,4,5,6)]



dim(training)
```

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

With this, the training dataset is ready to train the model. At first, the raw dataset contained 19622 observation, with 160 variables. Many variables contained largely missing data (usually with only one row of data), so these were removed from the dataset. In addition, variables not concerning the movement sensors were also removed. This resulted in a dataset of variables

Training

The following section covers various models used to fit the dataset. But first the training dataset is further partitioned into training and testing datasets because actual testing dataset has only 20 observations.

Partioning Data - Cross Validation (Holdout Method)

The training dataset is partitioned into traning and testing sets for cross validation of the various models. Other cross validation methods like K fold CV and leave one out CV but hold out method retains simplicity for the purpose of this project while preventing overfitting.

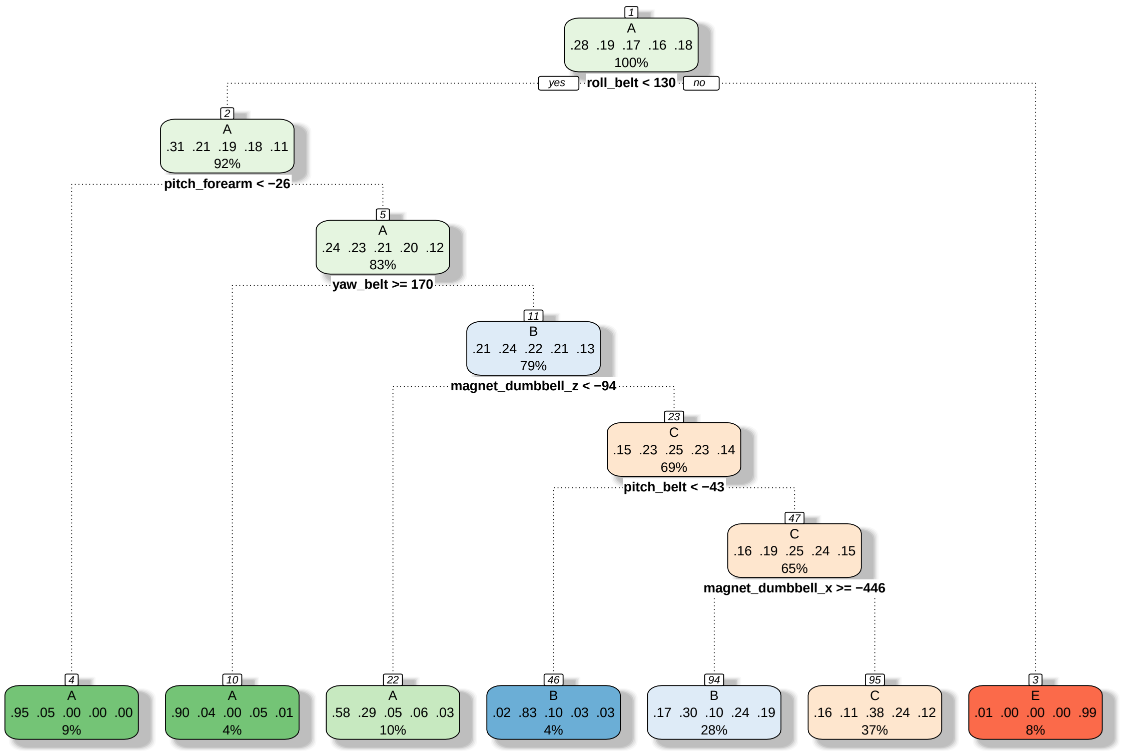
We split the cleaned training set trainData into a training set (train, 70%) for prediction and a validation set (valid 30%) to compute the out-of-sample errors.

```
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)

trainData <- training[inTrain,]

testData <- training[-inTrain,]
```

We use classification trees and random forests to predict the outcome.

Decision Tree

```
set.seed(1234)
modelFitDT <- train(classe~., data = trainData, method = "rpart")
fancyRpartPlot(modelFitDT$finalModel)
```



Rattle 2017−Apr−19 17:39:30 Nusri

```
predictDT <- predict(modelFitDT, testData)

confusionMatrix(table(predictDT, testData$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predictDT    A    B    C    D    E
##         A 1061  196   22   48   12
##         B  295  698  183  401  305
##         C  312  245  821  515  254
##         D    0    0    0    0    0
##         E    6    0    0    0  511
##
## Overall Statistics
##
##                Accuracy : 0.5252
##                  95% CI : (0.5124, 0.5381)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4018
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6338   0.6128   0.8002   0.0000  0.47227
## Specificity            0.9340   0.7505   0.7271   1.0000  0.99875
## Pos Pred Value         0.7924   0.3709   0.3824      NaN  0.98839
## Neg Pred Value         0.8652   0.8898   0.9452   0.8362  0.89363
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.1803   0.1186   0.1395   0.0000  0.08683
## Detection Prevalence   0.2275   0.3198   0.3648   0.0000  0.08785
## Balanced Accuracy      0.7839   0.6817   0.7636   0.5000  0.73551
```

From the confusion matrix, the accuracy rate is 0.5, and so the out-of-sample error rate is 0.5. Using classification tree does not predict the outcome classe very well hence cannot be used as a prediction model.

Random Forest

```r
modelFitRF <- randomForest(classe~., data = trainData)

predictRF <- predict(modelFitRF, testData, type = "class")

confusionMatrix(predictRF, testData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   13    0    0    0
##          B    0 1125    6    0    0
##          C    0    1 1019   28    0
##          D    0    0    1  936    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9888, 0.9937)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9877   0.9932   0.9710   0.9991
## Specificity            0.9969   0.9987   0.9940   0.9996   1.0000
## Pos Pred Value         0.9923   0.9947   0.9723   0.9979   1.0000
## Neg Pred Value         1.0000   0.9971   0.9986   0.9943   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1912   0.1732   0.1590   0.1837
## Detection Prevalence   0.2867   0.1922   0.1781   0.1594   0.1837
## Balanced Accuracy      0.9985   0.9932   0.9936   0.9853   0.9995
```

Random Forest (RF) method yielded a very high accuracy and out of sample error of (100-99.51) 0.49%. This may be due to the fact that many predictors are highly correlated. Random forests chooses a subset of predictors at each split and decorrelate the trees. This leads to high accuracy, although this algorithm is sometimes difficult to interpret and computationally inefficient.

Testing Prediction Model on Testing Dataset

```r
pred <- predict(modelFitRF, testing)

print(pred)
```

```
##  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
```

With high accuracy, the model accurately predicted all of the 20 test subjects.

Conclusion

This reports includes the machine learning algorithm implemented on activity tracker dataset to predict the dumbell movements of athelte. The best predictive model was found out to be through Random Forest with 99.51% accuracy in predicting the results.