ExerciseAnalysis

Krupa

Wednesday, March 30, 2016

Practical Machine Learning - Course Project Report

Project 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 (see the section on the Weight Lifting Exercise Dataset).

Project Goal

The goal of the project is to predict the manner in which people exercise.

Project Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: 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.

```r
# Include required libraries
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
```

Data loading and cleanup

Download datafiles and read into dataframes

```r
setwd("D:/Data specialist course/Practical Machine Learning/Course Project"
)
```

```
if (!file.exists("pml-training.csv"))
{
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-tra
ining.csv", destfile = "pml-training.csv")
}
if (!file.exists("pml-testing.csv"))
{
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv", destfile = "pml-testing.csv")
}

training <- read.csv("pml-training.csv", sep = ",", na.strings = c("", "NA"
))
testing <- read.csv("pml-testing.csv", sep = ",", na.strings = c("", "NA"))
```

Exploring Data

```
dim(training)
```

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

```
dim(testing)
```

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

```
str(training)
```

```
## '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 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 13230
84232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304
277 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 1.4
2 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.1
3 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 -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/ 396 levels "-0.016850","-0.021024
",..: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_picth_belt     : Factor w/ 316 levels "-0.021887","-0.060755
",..: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA
NA NA NA NA NA ...

##  $ skewness_roll_belt      : Factor w/ 394 levels "-0.003095","-0.010002
",..: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_roll_belt.1    : Factor w/ 337 levels "-0.005928","-0.005960
",..: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA 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            : Factor w/ 67 levels "-0.1","-0.2",..: 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            : Factor w/ 67 levels "-0.1","-0.2",..: 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      : Factor w/ 3 levels "#DIV/0!","0.00",..: 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 -2
1 ...
```

```
##  $ 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 60
9 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -31
3 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -12
8 -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 -16
1 -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 -28
9 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 11
0 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -12
4 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -37
2 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 33
4 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 51
6 ...
```

```
##  $ kurtosis_roll_arm       : Factor w/ 329 levels "-0.02438","-0.04190",
..: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_picth_arm      : Factor w/ 327 levels "-0.00484","-0.01311",
..: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_yaw_arm        : Factor w/ 394 levels "-0.01548","-0.01749",
..: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_roll_arm       : Factor w/ 330 levels "-0.00051","-0.00696",
..: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_pitch_arm      : Factor w/ 327 levels "-0.00184","-0.01185",
..: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_yaw_arm        : Factor w/ 394 levels "-0.00311","-0.00562",
..: 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  : Factor w/ 397 levels "-0.0035","-0.0073",..
: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_picth_dumbbell : Factor w/ 400 levels "-0.0163","-0.0233",..
: NA NA NA NA NA NA NA NA NA NA ...

##  $ kurtosis_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA
NA NA NA NA NA ...

##  $ skewness_roll_dumbbell  : Factor w/ 400 levels "-0.0082","-0.0096",..
: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_pitch_dumbbell : Factor w/ 401 levels "-0.0053","-0.0084",..
: NA NA NA NA NA NA NA NA NA NA ...

##  $ skewness_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA 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        : Factor w/ 72 levels "-0.1","-0.2",..: 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        : Factor w/ 72 levels "-0.1","-0.2",..: 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]
```

Preprocess Data

Step 1: Eliminate column 1 as it is not relevant

```
training <- training[,-1]
```

Step 2: Eliminate Zero and Near Zero variance columns

```
myDataNZV <- nearZeroVar(training, saveMetrics=TRUE)
NZVList <- c()
for (i in 1:159)
{
    if((myDataNZV[i,3] == TRUE) | (myDataNZV[i,4] == TRUE)) NZVList <- c(NZ
VList,i)
}
training <- training[,-NZVList]
```

Step 3: Eliminate columns having more than 80% values as NA

```
NaList <- c()
MaxEntries <- nrow(training)
for (i in 1:ncol(training))
{
  if(sum(is.na(training[ ,i])) > MaxEntries*0.8)NaList <- c(NaList,i)


}
cleanedTraining <- training[,-NaList]
```

Final dimension of cleaned data

```
dim(cleanedTraining)
```

```
## [1] 19622    58
```

Data Partition

Partition 70% of Data as Training Data and remaining 30% as Testing Data;

These wil be used for building the prediction model

```
inTrain <- createDataPartition(y=cleanedTraining$classe, p=0.7, list=FALSE)
myTraining <- cleanedTraining[inTrain, ]
myTesting <- cleanedTraining[-inTrain, ]
```

Final Testing data should have the same columns as the training data to test the model

```
cleanCols <- colnames(myTraining[, -58])
testing <- testing[cleanCols]
```

Dimensions of the Three data frames

```
dim(myTraining)
## [1] 13737    58
dim(myTesting)
## [1] 5885   58
dim(testing)
## [1] 20 57
```

Ensure that predictors in myTraining and Testing dataframes have the same class

```
for (i in 1:length(testing) )
{
    for(j in 1:length(myTraining))
    {
        if( names(myTraining[j]) == names(testing[i]))
       {
            class(testing[i]) <- class(myTraining[j])
        }
    }
}
```

Ensure coertion of dataframes by adding and deleting a row from myTraining dataframe to testing dataframe

```
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Model Building

Build Model with myTraining data using rpart(Recursive partitioning)function

classe is the outcome variable and all other columns are predictors

```
modelFit1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modelFit1)
```

Use the model on myTesting data and predict classe

```
predictions1 <- predict(modelFit1, myTesting, type = "class")
confusionMatrix(predictions1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1605   50    6    3    0
##          B   53  965   68   52    0
##          C   16  117  936  143    2
##          D    0    7    9  612   58
##          E    0    0    7  154 1022
## 
## Overall Statistics
## 
##                Accuracy : 0.8734
##                  95% CI : (0.8646, 0.8818)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.8398
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9588   0.8472   0.9123   0.6349   0.9445
## Specificity            0.9860   0.9635   0.9428   0.9850   0.9665
## Pos Pred Value         0.9645   0.8480   0.7710   0.8921   0.8639
## Neg Pred Value         0.9837   0.9633   0.9807   0.9323   0.9872
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2727   0.1640   0.1590   0.1040   0.1737
## Detection Prevalence   0.2828   0.1934   0.2063   0.1166   0.2010
## Balanced Accuracy      0.9724   0.9054   0.9275   0.8099   0.9555
```

The accuracy of the prediction model is low (87.34%)

Build Model with myTraining data using randomForest function to improve predictive accuracy

```
modelFit2 <- randomForest(classe ~. , data=myTraining)
predictions2 <- predict(modelFit2, myTesting, type = "class")
confusionMatrix(predictions2, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    0
##          B    1 1139    3    0    0
##          C    0    0 1022    1    0
##          D    0    0    1  960    0
##          E    0    0    0    3 1082
## 
## Overall Statistics
## 
##                Accuracy : 0.9985
##                  95% CI : (0.9971, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9981
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   1.0000   0.9961   0.9959   1.0000
## Specificity            1.0000   0.9992   0.9998   0.9998   0.9994
## Pos Pred Value         1.0000   0.9965   0.9990   0.9990   0.9972
## Neg Pred Value          0.9998   1.0000   0.9992   0.9992   1.0000
## Prevalence              0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate          0.2843   0.1935   0.1737   0.1631   0.1839
## Detection Prevalence    0.2843   0.1942   0.1738   0.1633   0.1844
## Balanced Accuracy       0.9997   0.9996   0.9979   0.9978   0.9997
```

Accurancy of the Prediction Model is now 99.85%

Validate prediction model on testing data

```
predictionsFinal <- predict(modelFit2, testing, type = "class")
```

Print output

```
predictionsFinal
```

```
##  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
```