Machine Learning Project

Binu Pillai

March 19, 2015

Executive Summary

The goal of this project is to predict how well 6 different people performed barbell lifts utilizing data collected from activity monitoring devices. Each of the 6 people were asked to perform the barbell lifts correctly and incorrect 5 different ways. Utilizing the activity monitor device data, a machine learning model is to be generated using a training set with class labels representing the 6 ways of performing the barbell lifts (supervised learning). Once the models are built, the generalization performance should be assessed, and then the training model is to be applied to a new set of testing data to make predictions.

Input Data

The input data consisted of various movement measurments including acceleration components of the arms and pitch and roll orientations of the dumbell collected using devices such as Jawbone Up, Nike FuelBand, and Fitbit .

The original data was taken from the originating study linked below. Please see the site and associated paper for more information. http://groupware.les.inf.puc-rio.br/har

Data Analysis

First of all load the R packages that makes the analysis easir.

```
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 mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(12345)
```

Getting the data

Assume that training data set and testing data set are already dowmloaded from the following URL to current working directory.

Testing Data Set URL :-> http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Training Data Set URL :-> http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

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

Partioning the training set into two

Partioning Training data set into two data sets (60% training and 40% testing).

```
new.train.data <- createDataPartition(y=training.data$classe, p=0.6, list=FALSE)
my.training <- training.data[new.train.data, ]
my.testing <- training.data[-new.train.data, ]

dim(my.training)
```

```
## [1] 11776   160
```

```
dim(my.testing)
```

```
## [1] 7846  160
```

Cleaning the data

Transformation 2:Cleaning data by eliminating NearZeroValidable

```
myDataNZV <- nearZeroVar(my.training, saveMetrics=TRUE)

NZV.vars <- names(my.training) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
                                      "kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1"
                                      "max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_a
                                      "var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_a
                                      "stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_p
                                      "kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "s
                                      "max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_
                                      "kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_ya
                                      "skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumb
                                      "amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_pict
                                      "skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_
                                      "max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitu
                                      "amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm
                                      "avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm",
```

```
                                        "stddev_yaw_forearm", "var_yaw_forearm")
my.training <- my.training[!NZV.vars]

dim(my.training)
```

```
## [1] 11776   100
```

Transformation 2: Removing first ID variable

```
my.training <- my.training[c(-1)]
```

Transformation 3: Cleaning Variables with too many NAs.

```
new.training <- my.training
for(i in 1:length(my.training)) {
  if( sum( is.na( my.training[, i] ) ) /nrow(my.training) >= .6 ) {
    for(j in 1:length(new.training)) {
      if( length( grep(names(my.training[i]), names(new.training)[j]) ) ==1)  {
        new.training <- new.training[ , -j]
      }
    }
  }
}

dim(new.training)
```

```
## [1] 11776    58
```

```
my.training <- new.training
rm(new.training)
```

Now let us do the exact same 3 transformations but for our myTesting and testing data sets.

```
clean.set1 <- colnames(my.training)
clean.set2 <- colnames(my.training[, -58])
my.testing <- my.testing[clean.set1]
testing.data <- testing.data[clean.set2]

dim(my.testing)
```

```
## [1] 7846   58
```

```
dim(testing.data)
```

```
## [1] 20 57
```

In order to ensure proper functioning of Decision Trees and especially RandomForest Algorithm with the Test data set (data set provided), we need to coerce the data into the same type.

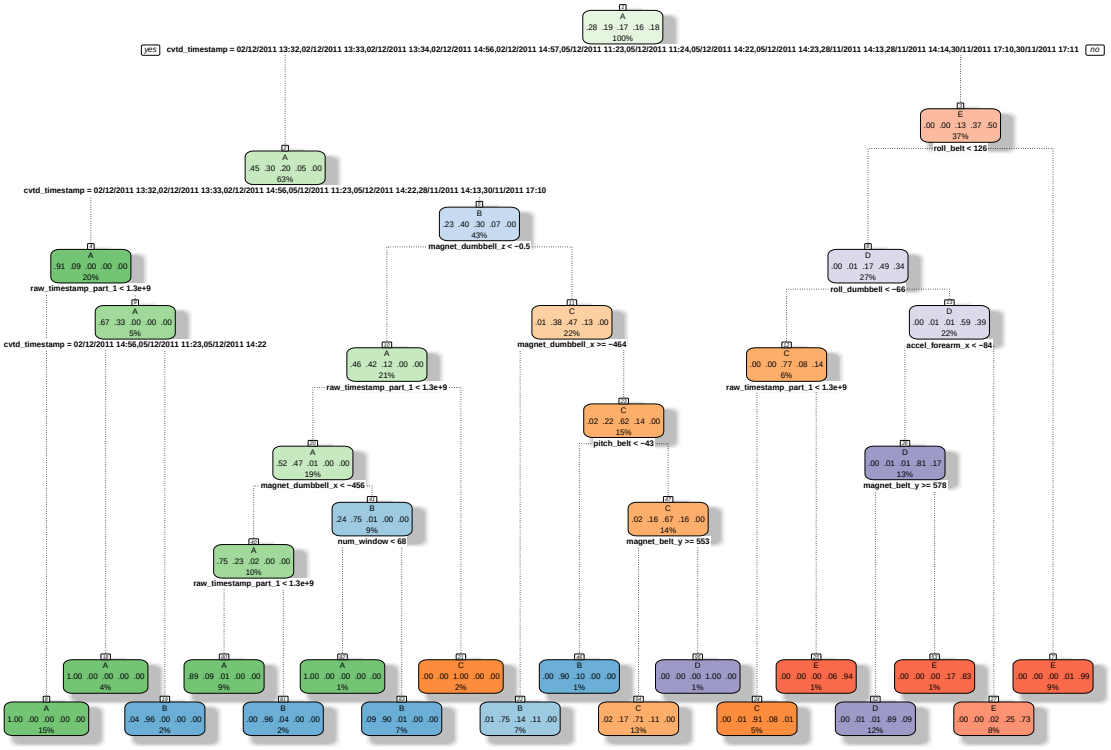
```
for (i in 1:length(testing.data) ) {
  for(j in 1:length(my.training)) {
    if( length( grep(names(my.training[i]), names(testing.data)[j]) ) ==1)  {
      class(testing.data[j]) <- class(my.training[i])
    }
  }
}

testing.data <- rbind(my.training[2, -58] , testing.data)
testing.data <- testing.data[-1,]
```

Using ML algorithms for prediction: Decision Tree

```
modFitA1 <- rpart(classe ~ ., data=my.training, method="class")

fancyRpartPlot(modFitA1)
```



Rattle 2015−Mar−19 14:52:14 binupillai

Predicting:

```
predictionsA1 <- predict(modFitA1, my.testing, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsA1, my.testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

Using ML algorithms for prediction: Random Forests

```
modFitB1 <- randomForest(classe ~. , data=my.training)
```

Predicting:

```
predictionsB1 <- predict(modFitB1, my.testing, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsB1, my.testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    2    0    0
##          C    0    0 1366    3    0
##          D    0    0    0 1282    2
##          E    0    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9969   0.9986
## Specificity            0.9996   0.9995   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9980   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9997   0.9994   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1634   0.1835
## Detection Prevalence   0.2846   0.1936   0.1745   0.1637   0.1837
## Balanced Accuracy      0.9996   0.9991   0.9990   0.9983   0.9992
```

Random Forests yielded better results.

Generating Files to submit as answers for the Assignment:

```
predictionsB2 <- predict(modFitB1, testing.data, type = "class")
```

Generating files with predictions to submit for assignment.

```
pml_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=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictionsB2)
```