Practical Machine Learning Project

Cindy

Sunday, February 28, 2016

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).

Data Details

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Data Analysis

Load package:

```
## Loading required package: lattice
## Loading required package: ggplot2
## 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.
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Read data from URL provided and change missing data to the same format:

```
urlTrain = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
train = read.csv(url(urlTrain), na.strings=c("NA","#DIV/0!",""))
test = read.csv(url(urlTest), na.strings=c("NA","#DIV/0!",""))
dim(train)
```

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

```
dim(test)
```

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

Clean data in training dataset:

1. remove the first 7 columns since not used for model, 153 column left.

```
train=train[,-c(1:7)]
```

2. Remove variables with 50% more missing value, 60 columns left.

```
NAPerc=sapply(colnames(train),
       function(x) if(sum(is.na(train[, x])) > 0.50*nrow(train)) {return(TRUE)}else{return(FALSE)}
       )
train=train[, !NAPerc]
```

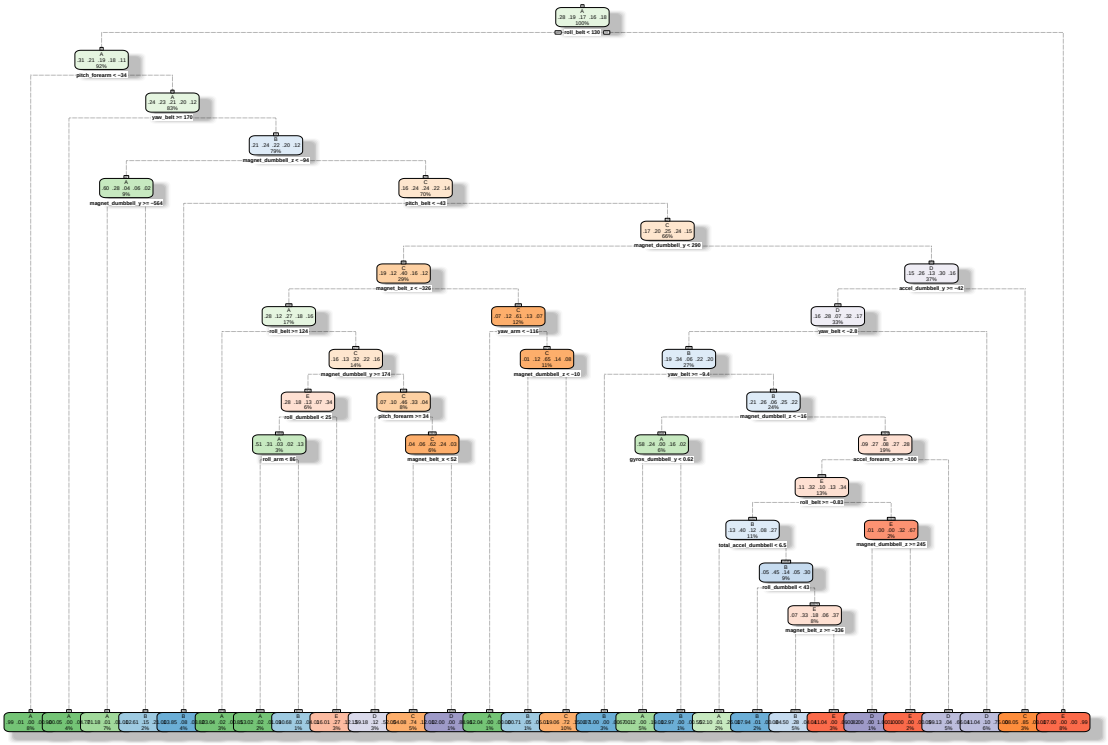
Partition the training data into a new training and testing data by 0.6 and 0.4. Cross-validation will be performed by subsampling our training data set randomly without replacement into: NewTrain data (60% of the original Training data set) and NewTest data (40% of the original Training data set). Our models will be fitted on the NewTraining data set, and tested on the NewTesting data. Once the most accurate model is chosen, it will be tested on the original Testing data set. The set.seed function is for the reproducibility. Same records will be assigned to the training and testing datasets if the code is to be run again.

```
set.seed(123)
PartTrain=createDataPartition(y=train$classe, p=0.6, list=FALSE)
NewTrain = train[PartTrain, ]
NewTest = train[-PartTrain, ]
```

Decision Tree and Random Forest is built and plotted:

```
dtModel=rpart(classe ~. , data=NewTrain, method="class")
fancyRpartPlot(dtModel)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−Feb−28 12:14:07 xinxin

```
rfModel = randomForest(classe ~. , data=NewTrain, method="class")
```

Test Decision Tree and Random Forest on the "New" testing data:

```
dtpredNewTest= predict(dtModel, NewTest, type = "class")
confusionMatrix(dtpredNewTest, NewTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2059  203   18  128   79
##          B   60 1086  211   70   93
##          C   20   83  998  184   83
##          D   71  138   81  863  120
##          E   22    8   60   41 1067
##
## Overall Statistics
##
##                Accuracy : 0.774
##                  95% CI : (0.7646, 0.7832)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.713
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9225   0.7154   0.7295   0.6711   0.7399
## Specificity            0.9238   0.9314   0.9429   0.9375   0.9795
## Pos Pred Value         0.8279   0.7145   0.7295   0.6779   0.8907
## Neg Pred Value         0.9677   0.9317   0.9429   0.9356   0.9436
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2624   0.1384   0.1272   0.1100   0.1360
## Detection Prevalence   0.3170   0.1937   0.1744   0.1622   0.1527
## Balanced Accuracy      0.9231   0.8234   0.8362   0.8043   0.8597
```

```
rfpredNewTest= predict(rfModel, NewTest, type = "class")
confusionMatrix(rfpredNewTest, NewTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    7    0    0    0
##          B    2 1508   11    0    0
##          C    0    3 1354   11    4
##          D    0    0    3 1275    3
##          E    1    0    0    0 1435
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9923, 0.9958)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9934   0.9898   0.9914   0.9951
## Specificity            0.9988   0.9979   0.9972   0.9991   0.9998
## Pos Pred Value         0.9969   0.9915   0.9869   0.9953   0.9993
## Neg Pred Value         0.9995   0.9984   0.9978   0.9983   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1922   0.1726   0.1625   0.1829
## Detection Prevalence   0.2850   0.1939   0.1749   0.1633   0.1830
## Balanced Accuracy      0.9987   0.9957   0.9935   0.9953   0.9975
```

Conclusion and Prediction

Random Forest performs better with 0.9941 accuracy. Make predictions for test data using Random Forest model:

```r
predRF = predict(rfModel, test, type = "class")
predRF
```

```
##  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
```