Prediction Assignment Writeup

cansheng wei

September 2, 2016

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

1.1 Project Data Source

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

1.2 Project Object

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.

2. Preparing Data

2.1 Load Library

Load the necessary libraies.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(data.table)
library(ggplot2)
```

2.2 Load Data

Read the data from URL:

```r
urlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Load the data into memory from the files, and check the data variables of train and test data set.

```r
fileTrain <- "pml-training.csv"
if (file.exists(fileTrain)) {
        traindata <- read.csv(fileTrain, na.strings=c("NA","#DIV/0!",""))
} else {
        download.file(urlTrain,fileTrain)
        traindata <- read.csv(fileTrain, na.strings=c("NA","#DIV/0!",""))
        }

fileTest <- "pml-testing.csv"

if (file.exists(fileTest)) {
        testdata <- read.csv(fileTest, na.strings=c("NA","#DIV/0!",""))
} else {
        download.file(urlTest,fileTest)
        testdata <- read.csv(fileTest, na.strings=c("NA","#DIV/0!",""))
}

summary(traindata$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```r
summary(testdata$classe)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

Here, we perform the basic data exporatory analysis.

```r
#summary(traindata)
#summary(traindata$classe)
#summary(testdata)
#summary(testdata$classe)
#dim(traindata)
#dim(testdata)
#str(traindata)
#str(testdata)
#head(traindata)
#str(testdata)
```

We can see that both data sets contains lots of NAs. we choose to delete these columns.For the first 7 columns, we also delet them, which are unnecessary for prediction.

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

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

traindata <- traindata[,-c(1:7)]

testdata <- testdata[,-c(1:7)]
```

Now, set the seed for reproduceability.

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

2.3 Partitioning Traindata

Partioning Training data set into two data sets, 70% for regressiong Training, 30% for regression Testing.The data is partitioned by the "classe" variable, which is the varible we will predict in the model.

```r
inTrain <- createDataPartition(y=traindata$classe, p = 0.70, list=FALSE)
innertraining <- traindata[inTrain,]
innertesting <- traindata[-inTrain,]

dim(innertraining)
```

```
## [1] 13737    53
```

```r
dim(innertesting)
```

```
## [1] 5885   53
```

```r
innertraining$classe <- as.factor(innertraining$classe)

innertesting$classe <- as.factor(innertesting$classe)
```

3. Prediction Analysis Model

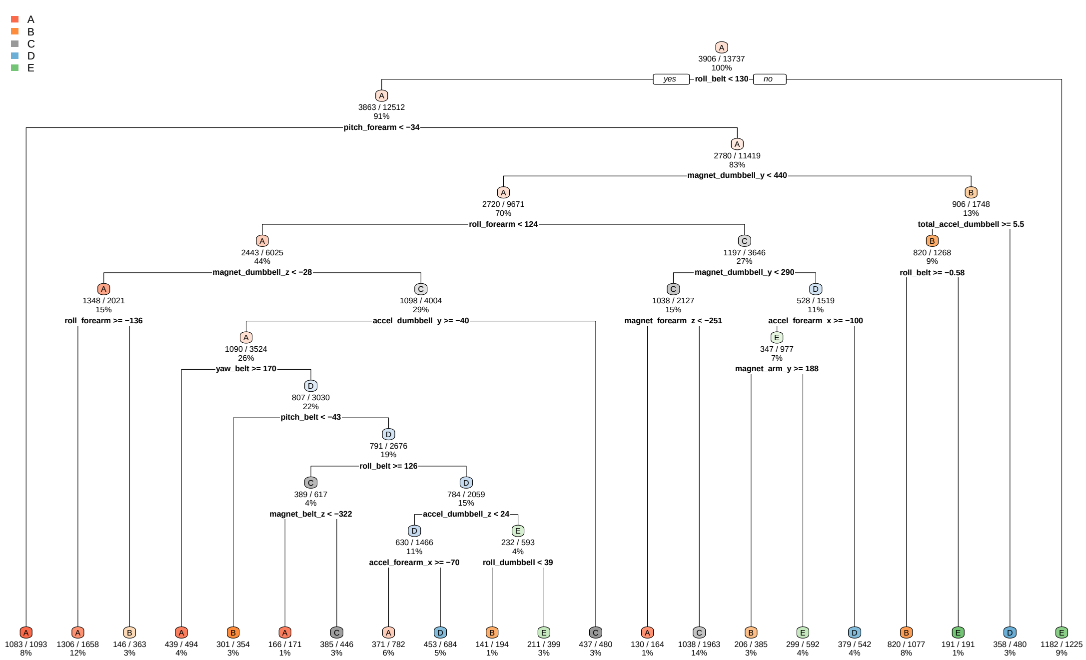
In the analysis prediction model,the dependent variable is classe, which is a factor variable. For this data set, "participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions: - exactly according to the specification (Class A) - throwing the elbows to the front (Class B) - lifting the dumbbell only halfway (Class C) - lowering the dumbbell only halfway (Class D) - throwing the hips to the front (Class E)

We will run two models to test decision tree and random forest. The model with the highest accuracy will be chosen as our final model.

3.1 Model 1: Decision Tree

```r
set.seed(12345)
treefitmodel <- rpart(classe ~ ., data=innertraining, method="class")

treeprediction <- predict(treefitmodel, innertesting, type = "class")

rpart.plot(treefitmodel, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```

Classification Tree



For the decision tree model, we give the test result as follows:

```r
confusionMatrix(treeprediction, innertesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1498  196   69  106   25
##          B   42  669   85   86   92
##          C   43  136  739  129  131
##          D   33   85   98  553   44
##          E   58   53   35   90  790
##
## Overall Statistics
##
##                Accuracy : 0.722
##                  95% CI : (0.7104, 0.7334)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6467
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8949   0.5874   0.7203  0.57365   0.7301
## Specificity            0.9060   0.9357   0.9097  0.94717   0.9509
## Pos Pred Value         0.7909   0.6869   0.6273  0.68020   0.7700
## Neg Pred Value         0.9559   0.9043   0.9390  0.91897   0.9399
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2545   0.1137   0.1256  0.09397   0.1342
## Detection Prevalence   0.3218   0.1655   0.2002  0.13815   0.1743
## Balanced Accuracy      0.9004   0.7615   0.8150  0.76041   0.8405
```

```r
accuracyTreemodel <- postResample(treeprediction, innertesting$classe)
```

Here, the accuracy of the decision tree model is:

```r
accuracyTreemodel <- postResample(treeprediction, innertesting$classe)

accuracyTreemodel
```

```
##  Accuracy     Kappa
## 0.7220051 0.6466943
```

So, see that the decision tree model correctly classified 72% of the movements.

Then, the out-of-sample estimated accuracy is given by:

```
oostreemodle <- (1-as.numeric(confusionMatrix(innertesting$classe, treeprediction)$overall[1]))
oostreemodle
```

```
## [1] 0.2779949
```

we can say that about 26% of movements will be misclassified by the tree model.

3.2 Model 2: Random Froest

The reason for this is that Random Forest is very accurate among other algorithms and it runs very efficiently on large data sets. We will run the set on 5-fold cross validation.

```
set.seed(12345)
controlfolds <- trainControl(method = "cv", 5)
RFmodel <- train(classe ~ ., data = innertraining, method = "rf", trControl = controlfolds, ntree = 250)
RFmodel
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10989, 10990, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9904638  0.9879357
##   27    0.9896631  0.9869235
##   52    0.9821643  0.9774367
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Now, we run the random forest model for the testing data:

```
RFprediction <- predict(RFmodel, innertesting)
```

Here, the accuracy of the random forest model is:

```
confusionMatrix(innertesting$classe, RFprediction)
```

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

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   13 1122    4    0    0
##          C    0   16 1006    4    0
##          D    0    0   25  939    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9888
##                  95% CI : (0.9858, 0.9913)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9858
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9923   0.9851   0.9720   0.9926   1.0000
## Specificity            0.9998   0.9964   0.9959   0.9949   0.9994
## Pos Pred Value         0.9994   0.9851   0.9805   0.9741   0.9972
## Neg Pred Value         0.9969   0.9964   0.9940   0.9986   1.0000
## Prevalence             0.2865   0.1935   0.1759   0.1607   0.1833
## Detection Rate         0.2843   0.1907   0.1709   0.1596   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9960   0.9907   0.9839   0.9938   0.9997
```

```
accuracyRFmodel <- postResample(RFprediction, innertesting$classe)

accuracyRFmodel
```

```
##  Accuracy     Kappa
## 0.9887850 0.9858101
```

From the confusion matrix and accurancy result, we can already tell the random forest model is better than the decison tree model. For each class of movement, the sensitivity and specificity are more than. And the random forest model achieves an accuracy of 98%, much higher than the accuracy of the decison tree model.

Furthermore, the out-of-sample accuracy is:

```
oosrandommodel <- (1-as.numeric(confusionMatrix(innertesting$classe, RFprediction)$overall[1]))

oosrandommodel
```

```
## [1] 0.01121495
```

For random forest, we can see that only about 1% of movements to be misclassified.

4. Model Test

4.1 Decison Tree Model Test

```r
resultTree <- predict(treefitmodel, testdata, type = "class")

print(resultTree)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  C  A  B  D  A  C  D  D  A  A  C  B  C  A  E  E  A  B  B  B
## Levels: A B C D E
```

4.2 Random Forest Model Test

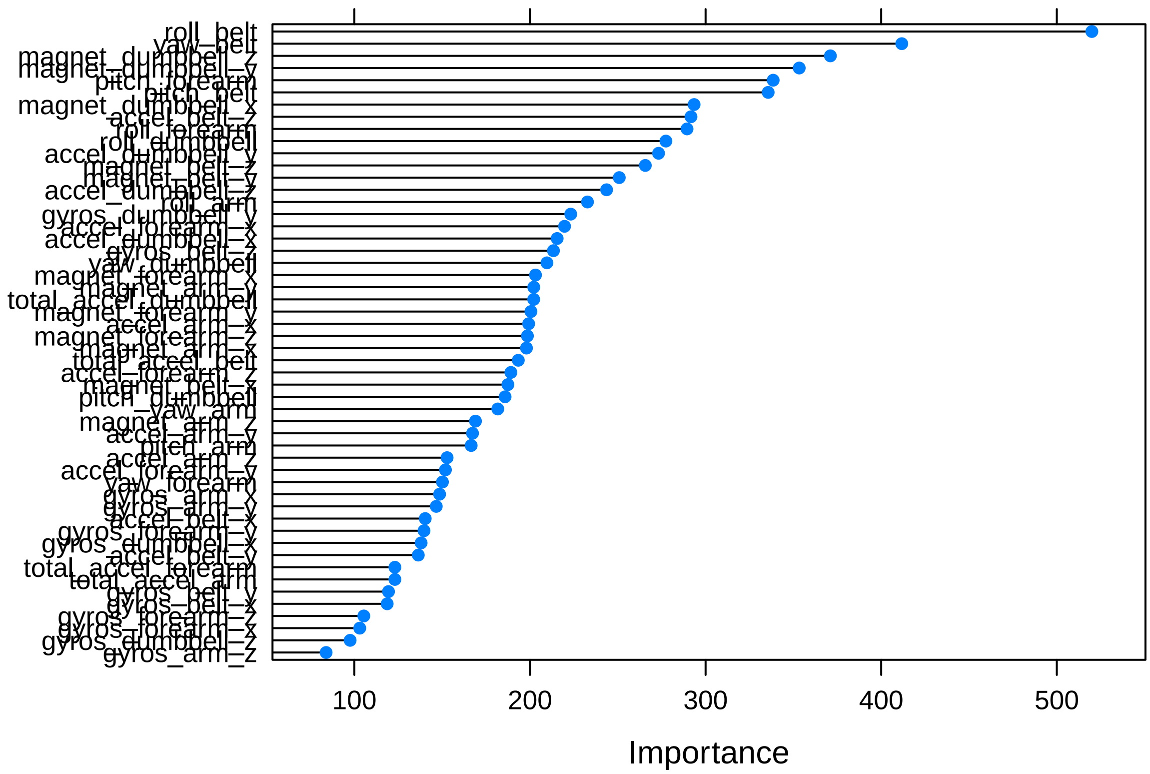
```r
resultRF <- predict(RFmodel,testdata)

print(resultRF)
```

```
##  [1] 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
```

5. Conclusion

In this machine learning, we choose decision tree model and random forest model for testing. And we can see that the random forest model is obviously better than decision tree. Meanwhile, we then find the importance of features under the random model fit.

```r
importance <- varImp(RFmodel, scale=FALSE)
plot(importance)
```



6. Reference

[1]Data from the website:http://groupware.les.inf.puc-rio.br/har

[2]Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.