Activity Recognition of Weight Lifting

Xray Wang

March 14, 2016

Synopsis

This report is about creating qualitative models to recognize different classes of activities of correctly and incorrectly performed weight lifting exercises. We created different models on training data, evaluated them on the validation data, choosing the best performing model as our final model to be used on test data.

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

The training data for this project are available here.

The test data are available here.

The data for this project come from this source. 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.

Prepare Data

The report requires the following R libraries.

```
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
```

We first load the training and testing datasets using a funcion.

```
load = function(url, file) {
        if(!file.exists(file)){
                download.file(url, file)
        }
        read.csv(file, na.strings=c("NA","#DIV/0!",""))
```

```
}

training = load("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                "training.csv")
testing = load("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
               "testing.csv")
```

Then we dropped the first 7 columns as they are not useful for our task.

```
trainClean = training[,-(1:7)]
testClean = testing[,-(1:7)]
```

Drop all the columns with NA values.

```
dropC = c()
for(i in 1:ncol(trainClean)){
        m = mean(is.na(trainClean[,i]))
        if(m >0) {
                dropC = c(dropC, i)
        }
}

trainClean = trainClean[,-dropC]
testClean = testClean[,-dropC]
```

Next, we try to drop the near zero columns, but there are none left.

```
inNZR = nearZeroVar(trainClean)
length(inNZR)
```

```
## [1] 0
```

Then we split the training set into a training set with 60% of the training set data and the rest 40% as a validation set.

```
set.seed(20160314)
inTrain <- createDataPartition(trainClean$classe, p=0.6, list=FALSE)
train <- trainClean[inTrain, ]
validation <- trainClean[-inTrain, ]
```

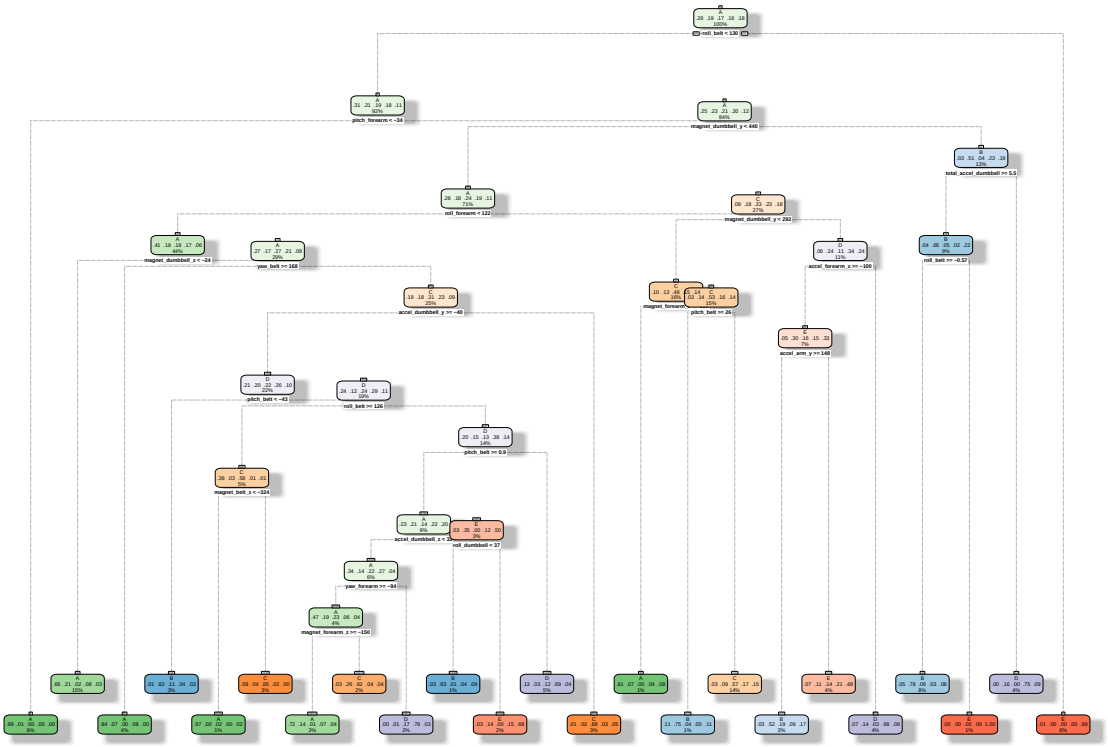
Models

Now we build models base on the training set we have now.

Decision Tree Model

```
set.seed(20160314)

dt = rpart(classe~., data=train, method='class')
fancyRpartPlot(dt)
```



Then we test the model with the validation data set.

```
dtPredict = predict(dt, newdata=validation, type='class')
confusionMatrix(dtPredict, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2014  314   39  139   57
##          B   49  885  121   39  106
##          C   54  160 1099  202  185
##          D   78   89   78  815   79
##          E   37   70   31   91 1015
##
## Overall Statistics
##
##                Accuracy : 0.7428
##                  95% CI : (0.733, 0.7524)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6731
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9023   0.5830   0.8034   0.6337   0.7039
## Specificity            0.9022   0.9502   0.9072   0.9506   0.9642
## Pos Pred Value         0.7858   0.7375   0.6465   0.7155   0.8159
## Neg Pred Value         0.9587   0.9048   0.9562   0.9298   0.9353
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2567   0.1128   0.1401   0.1039   0.1294
## Detection Prevalence   0.3267   0.1529   0.2167   0.1452   0.1586
## Balanced Accuracy      0.9023   0.7666   0.8553   0.7922   0.8341
```

Random Forests

Build a random forest model and test with validation data.

```
set.seed(20160314)

rfmodel = randomForest(classe ~ ., data=train)
rfPredict = predict(rfmodel, newdata=validation)

confusionMatrix(rfPredict, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   12    0    0    0
##          B    2 1497    4    0    0
##          C    0    9 1361   22    4
##          D    0    0    3 1263    1
##          E    1    0    0    1 1437
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9903, 0.9943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9862   0.9949   0.9821   0.9965
## Specificity            0.9979   0.9991   0.9946   0.9994   0.9997
## Pos Pred Value         0.9946   0.9960   0.9749   0.9968   0.9986
## Neg Pred Value         0.9995   0.9967   0.9989   0.9965   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1908   0.1735   0.1610   0.1832
## Detection Prevalence   0.2856   0.1916   0.1779   0.1615   0.1834
## Balanced Accuracy      0.9983   0.9926   0.9947   0.9908   0.9981
```

We can see the accuracy is with pretty high accuracy, which is a good estimation of our out of sample accuracy (and therefore error). So we decided to go with this model for test set.

Apply to Test Set

```
predict(rfmodel, newdata=testClean)
```

```
##  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
```