Practical Machine Learning Course Project

Guojian Ou

27 March 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, our goal is 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. We will then predict whether the participants did the exercise correctly or incorrectly.

Download and prepare the data

```
library(caret)
```

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

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

```
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
```

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv", "pml-training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.
csv", "pml-testing.csv")
training<-read.csv("pml-training.csv")
testing<-read.csv("pml-testing.csv")
training<-training[-1]
testing<-testing[-1]
```

We now remove columns with almost 0 variance, as well as data that should not have any bearing on the outcome classe, such as timestamp and name of participants.

```
zerovar<-nearZeroVar(training)
training1<-training[,-zerovar]
training2<-training1[,6:99]
```

We also remove the colums which have at least 1 NA term. Hence, we only choose variables that are complete over all the observations.

```
colnoNA<-which(colSums(is.na(training2))==0)
training3<-training2[,colnoNA]
```

Creating the Model

We will use the Random Forest algorithm to create a model. However, we will first split the training data further into a training data (70%) and for validation (30%).

We first create a data parition and split the training data.

```
set.seed(2016)
inTrain <- createDataPartition(training3$classe, p=0.70, list=F)
trainData <- training3[inTrain, ]
testData <- training3[-inTrain, ]
```

We load the RandomForest package and train a Random Forest model. This will take a while due to the large number of variables and data points. We thus save the model so that we do not need to re-run the simulation.

```
model_RF<-train(classe~., data=trainData, method="rf")
saveRDS(model_RF, "RFModel.rds")
```

```
model_RF<-readRDS("RFModel.rds")
```

Validating the Data

We next test and validate the model on testData.

```
predictClasse<-predict(model_RF, testData)
confusionMatrix(testData$classe, predictClasse)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    4 1131    4    0    0
##          C    0   11 1015    0    0
##          D    0    0   16  948    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9895   0.9807   0.9989   1.0000
## Specificity            0.9998   0.9983   0.9977   0.9968   0.9998
## Pos Pred Value         0.9994   0.9930   0.9893   0.9834   0.9991
## Neg Pred Value         0.9991   0.9975   0.9959   0.9998   1.0000
## Prevalence             0.2850   0.1942   0.1759   0.1613   0.1837
## Detection Rate         0.2843   0.1922   0.1725   0.1611   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9987   0.9939   0.9892   0.9979   0.9999
```

From the table we can tell that the overall accuracy of the model is 99.37% on the testing data. This validates our model.