Prediction Assignment

Vadim K

July 21, 2016

Introduction

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.

Objective

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, the 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Data Processing

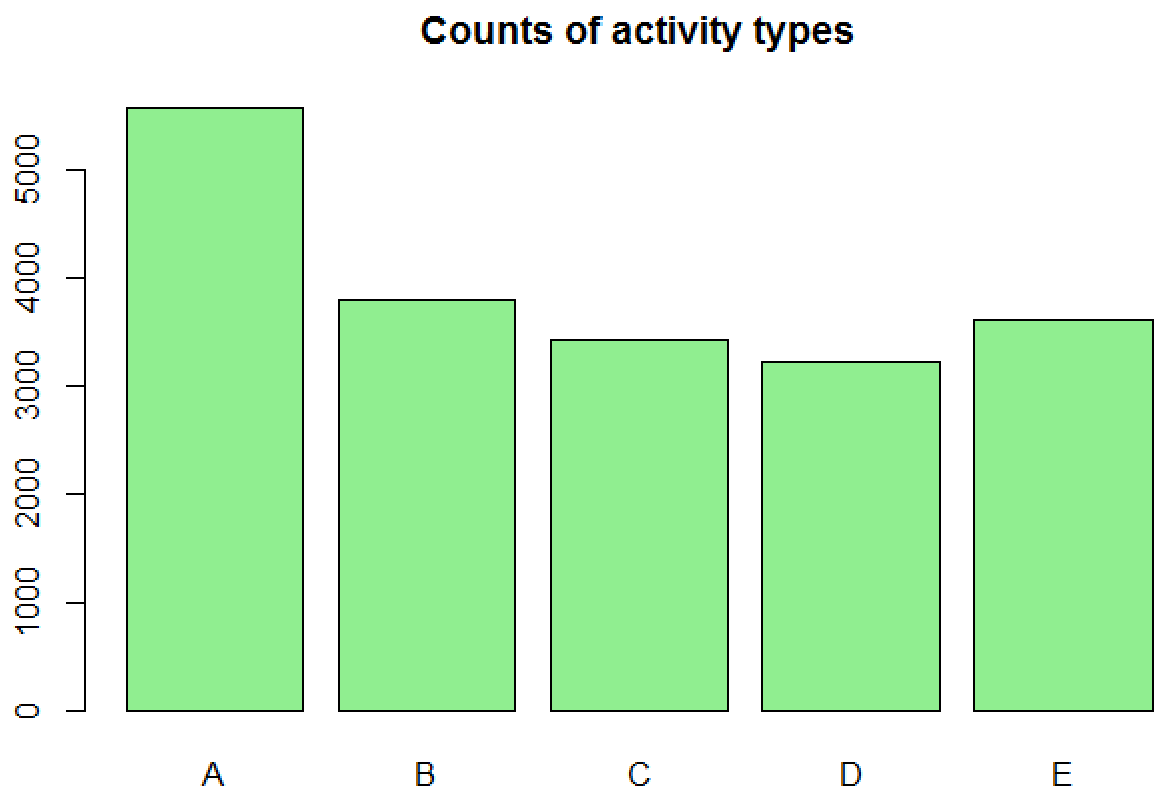
Start by downloading the data and reading it into the R as a dataframe

```
knitr::opts_chunk$set(echo = TRUE)

set.seed(1234)
library(caret)
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "trainin
g.csv")
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "testing.c
sv")

training <- read.csv("training.csv")
testing <- read.csv("testing.csv")

#Quick look at the training results
barplot(summary(training$classe), col = "lightgreen", main = "Counts of activity types")
```



Inspection of the data reveals that in many place the data has NA values, so at the 1st step let's inspect what the magnitude of the issue is.

```
a <- as.data.frame(names(training))
a$`names(training)` <- as.character(a$`names(training)`)

for (i in 1:nrow(a)) {
      a[i,2] <- 100*(sum(is.na(training[,i]))/length(training[,i]))
}
sum(a[,2]!=0)
```

```
## [1] 67
```

Furthermore, there are data where values are missing. In oder to roceed forward with the modeling, we need to clean the data first. I will do it for both the training set and the testing set.

```
testing <- testing[,apply(training, 2, function(x) any(is.na(x) | x == "") == FALSE)]
training <- training[,apply(training, 2, function(x) any(is.na(x) | x == "") == FALSE)]


b <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_
window", "num_window")

training <- training[, !(names(training) %in% b)]
testing <- testing[, !(names(testing) %in% b)]
```

Analysis

Now, we are ready to build the prediction model. Since the desired prediction is a qualatative variable, it would be appropriate to apply the Random Forest method. To do the model let's split the Training dataset into 2 subsets: one for model building and the second for cross validation. We will train the model on the training set (70% of the sample data) and validate with the validation set (30% of the sample data) to understand the accuracy of the model and the out of sample error.

```
library(caret)
inBuild <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
validation <- training[-inBuild,]; train <- training[inBuild,]

fitRF <- train(classe ~ ., method = "rf", data = train, trControl = trainControl(method="cv" ,nu
mber=3))

predRF.in <- predict(fitRF, train)
confusionMatrix(predRF.in, train$classe)$overall['Accuracy']
```

```
## Accuracy
##        1
```

```
insampleError <- 1 - sum(predRF.in == train$classe)/nrow(train)
print(insampleError)
```

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

```
predRF <- predict(fitRF, validation)
confusionMatrix(predRF, validation$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.9921835
```

```
outsampleError <- 1 - sum(predRF == validation$classe)/nrow(validation)
print(outsampleError)
```

```
## [1] 0.007816483
```

It looks as if the model provides an extremely good accuracy on cross validation and low out of sample error. Let's take a look at the confusion matrix in full

```
confusionMatrix(predRF, validation$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    0 1126    6    0    0
##          C    1    5 1017   12    3
##          D    0    0    3  951    7
##          E    0    0    0    1 1072
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9922          
##                  95% CI : (0.9896, 0.9943)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9901          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9886   0.9912   0.9865   0.9908
## Specificity            0.9981   0.9987   0.9957   0.9980   0.9998
## Pos Pred Value         0.9952   0.9947   0.9798   0.9896   0.9991
## Neg Pred Value         0.9998   0.9973   0.9981   0.9974   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1913   0.1728   0.1616   0.1822
## Detection Prevalence   0.2856   0.1924   0.1764   0.1633   0.1823
## Balanced Accuracy      0.9988   0.9937   0.9935   0.9922   0.9953
```

Lastly, let's use the model to predict the results on the 20 test cases

```
predRF.new <- predict(fitRF, testing)
print(predRF.new)
```

```
##  [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
```