Final Project: Machine Learning

Gilvan C. Souza

3/8/2017

Introduction

In this project, we apply machine learning techniques to predict how well people perform physical activities. Accelerometer data is collected from six participants, from various locations in their bodies: Belt, forearm, arm, and dumbell. They were asked to perform barbell lifts correctly and incorrectly in six different ways. The outcome variable is "classe", which has five different levels: A, B, C, D, and E, which indicate how well a subject performs the activity. Two datasets were made available: a training set, with 19622 observations for 160 variables, as well as a testing set, with 20 observations.

Methodology

The datasets were downloaded, as ".csv" files. The testing dataset is rather small (only 20 observations), and is therefore not used in the actual model building / validation / testing stage. Instead, it is only used for providing the requested predictions using the final model, and also answering the quiz questions at the end. Upon first inspection, it was clear that some variables are irrelevant, as they contain time stamps, dates, whether it is a new window, and window number. So, as a first step, these variables (column numbers 1,3,4,5,6,7) were removed from the datasets. For cross-validation, the training dataset was then split into training, validation, and testing datasets. The training dataset is used to build the model, the validation dataset is used to assess its performance (and then for model refinement), and the testing dataset is used to assess the accuracy of predictions.

```r
library(caret)
library(ggplot2)
library(gbm)
library(plyr)
```

```
testingf <- read.csv("pml-testing.csv", stringsAsFactors = FALSE)
testingf <- testingf[, -c(1,3:7)]
testingf[,1] <- as.factor(testingf[,1])
trainingo <- read.csv("pml-training.csv", stringsAsFactors = FALSE,
                      na.strings = c("NA", "", " "))
trainingo <- trainingo[, -c(1,3:7)]
trainingo[,1] <- as.factor(trainingo[,1])
inTrain <- createDataPartition(trainingo$classe, p = 0.7, list = FALSE)
validation <- trainingo[-inTrain,]
restData <- trainingo[inTrain,]
inTrain2 <- createDataPartition(restData$classe, p = 0.7, list = FALSE)
training <- restData[inTrain2,]
testing <- restData[-inTrain2,]
```

The next step was to clean the data. There are several variables with a large number of “NA” or blank values; these can’t be possibly used for prediction. As a result, I removed from all datasets any variable that had more than 30% of missing (“NA”) or blank values. The end result is that we have 53 variables with integer or numerical values, plus a factor variable (with the person’s name); all of which can be used in a model to predict the outcome variable “classe”.

```
indexcolumrem <- function (x) {
    a <- rep(0.0,dim(x)[2])
    for (j in 1:dim(x)[2]) {
        a[j] <- sum(is.na(x[,j]))/dim(x)[1]
    }
    a <- a < 0.3
}
tr <- indexcolumrem(training)
training <- training[,tr]
testing <- testing[,tr]
testingf <- testingf[,tr]
validation <- validation[,tr]
dim(training)
```

```
## [1] 9619   54
```

With the choice of variables set, we fit various machine learning algorithms available in the caret package: **(rpart, lda, gbm)**, using the training dataset. We assess their accuracy (% of correct predictions) using the *validation* dataset, which should be a good indication of the out of sample error. We display the code only for the “rpart”, method, as the others are similar and can be seen in the .Rmd file.

```
set.seed(150)
mod <- train(classe ~ ., method = "rpart", data = training)
fit <- predict(mod,newdata = validation)
accur <- sum(1*(fit == validation$classe))/length(fit)
cat("Accuracy for rpart (Trees) model (%) = ", accur, "\n")
```

```
## Accuracy for rpart (Trees) model (%) =  0.4943076
```

```
table(fit,validation$classe)
```

```
## 
## fit    A    B    C    D    E
##   A 1516  478  468  430  169
##   B   33  384   49  177  142
##   C  121  277  509  357  271
##   D    0    0    0    0    0
##   E    4    0    0    0  500
```

```
## Accuracy for LDA model (%) =  0.7349193
```

```
## 
## fit1    A    B    C    D    E
##    A 1415  188   78   68   38
##    B   39  704  104   43  124
##    C  117  146  725  117   78
##    D  103   41  108  721   82
##    E    0   60   11   15  760
```

```
## Accuracy for tree boosting (gbm) model (%) =  0.9590484
```

```
## 
## fit2    A    B    C    D    E
##    A 1654   52    0    0    4
##    B    7 1045   38    2   15
##    C    6   40  972   23   14
##    D    4    1   14  934   10
##    E    3    1    2    5 1039
```

The Final Model, Accuracy, and Predictions for 20 cases

Considering that the gbm method (Stochastic Gradient Boosting) had by far the best accuracy (using the validation datset), we select it as the final model. We then compute the accuracy (out of sample error) in the **testing** dataset, which yields a value similar to the one obtained in the *validation* dataset:

```
## Accuracy for Final model (%) =  0.9562895
```

```
## Table of Predictions (Final Model) vs. Actual
```

```
## 
## fit3    A    B    C    D    E
##    A 1146   35    0    0    1
##    B   18  729   26    3    6
##    C    4   30  682   23    5
##    D    2    1    7  643    7
##    E    1    2    3    6  738
```

Finally, we now predict the 20 cases in the small **testingf** dataset. We print the predictions for each of the cases.

```
## 20 Predictions (Final Model) vs. Actual
```

```
##  [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
```