Prediction Assignment

Graeme Jamieson

April 26, 2016

Backgroud

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).

Analysis

```
library(caret)
library(e1071)
library(dplyr)
library(knitr)

#Download the files if you do not already have them
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv","training.csv")
#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv","testing.csv")

#Load the training and testing sets
testing_set <- read.csv("testing.csv", stringsAsFactors = F)
training_set <- read.csv("training.csv",stringsAsFactors = F)
```

Pre-processing the data

Now that we have the datasets we will take a look at the columns of the testing set to get an idea of which fields might be the best for our predictive model. First we should look at the testing set to see which columns have sufficient data to use in our model.

```
t <- testing_set %>%
      summarise_each(funs(sum(!is.na(.))))

col_names <- sapply((data.frame(name = colnames(t[1, t[1,] != 0])) %>%
arrange(name)), as.character)

cbind(col_names[1:20], col_names[21:40], col_names[41:60])
```

```
##       [,1]               [,2]                [,3]
##  [1,] "accel_arm_x"      "gyros_dumbbell_y"  "pitch_belt"
##  [2,] "accel_arm_y"      "gyros_dumbbell_z"  "pitch_dumbbell"
##  [3,] "accel_arm_z"      "gyros_forearm_x"   "pitch_forearm"
##  [4,] "accel_belt_x"     "gyros_forearm_y"   "problem_id"
##  [5,] "accel_belt_y"     "gyros_forearm_z"   "raw_timestamp_part_1"
##  [6,] "accel_belt_z"     "magnet_arm_x"      "raw_timestamp_part_2"
##  [7,] "accel_dumbbell_x" "magnet_arm_y"      "roll_arm"
##  [8,] "accel_dumbbell_y" "magnet_arm_z"      "roll_belt"
##  [9,] "accel_dumbbell_z" "magnet_belt_x"     "roll_dumbbell"
## [10,] "accel_forearm_x"  "magnet_belt_y"     "roll_forearm"
## [11,] "accel_forearm_y"  "magnet_belt_z"     "total_accel_arm"
## [12,] "accel_forearm_z"  "magnet_dumbbell_x" "total_accel_belt"
## [13,] "cvtd_timestamp"   "magnet_dumbbell_y" "total_accel_dumbbell"
## [14,] "gyros_arm_x"      "magnet_dumbbell_z" "total_accel_forearm"
## [15,] "gyros_arm_y"      "magnet_forearm_x"  "user_name"
## [16,] "gyros_arm_z"      "magnet_forearm_y"  "X"
## [17,] "gyros_belt_x"     "magnet_forearm_z"  "yaw_arm"
## [18,] "gyros_belt_y"     "new_window"        "yaw_belt"
## [19,] "gyros_belt_z"     "num_window"        "yaw_dumbbell"
## [20,] "gyros_dumbbell_x" "pitch_arm"         "yaw_forearm"
```

After we remove the columns that have no records in the testing set a couple column types jump out. The columns that start with accel, magnet, and gyros seem to have complete data and have measurement for all 3 dimensions and so these could be good columns to build our model on. So we will use all the columns that start with these three names to build our predictive model. We can look at adding or removing columns depending on the results of our predictive model.

```
# Make sure the variable we are predicting is a factor variable
training_set$classe <- factor(training_set$classe)

# Create a reduced training set with only our selected columns
train_reduced <- training_set %>%
                        select(
                            starts_with('gyros')
                            , starts_with('accel')
                            , starts_with('magnet')
                            , classe
                        )
```

Building our prediction model

Now since the testing set provided does not contain the classe variable, as it is the test for which our models are graded, we will split our training set into a training set and test set.

```
set.seed(123)
inTrain = createDataPartition(train_reduced$classe, p = 3/4)[[1]]
training = train_reduced[ inTrain,]
testing = train_reduced[-inTrain,]
```

The first model we will try to fit is a random forest. We will add the train control method repeatedcv to perform a 5- fold cross validation on the training set and we will repeat this 5 times.

```
# fit a random forest
tr<-trainControl(method="repeatedcv", number=5, repeats = 5)
fit1 <- train(classe ~., data = training, method = 'rf', trControl = tr)

max(fit1$results$Accuracy)

## [1] 0.9814974
```

We can see that the max accuracy of repeated CV is listed at around 98%. We expect our out of sample error to be 1-Accuracy of our cross validation data set. So we estimate that our out of sample error is very small as our model accuracy is very good.

Testing the accuracy of the model

```
pred1 <- predict(fit1, testing)

confusionMatrix(pred1, testing$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    4    0    3    0
##          B    2  939   13    0    0
##          C    0    6  838   28    0
##          D    1    0    4  772    1
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9872
##                  95% CI : (0.9836, 0.9901)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9837
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9895   0.9801   0.9602   0.9989
## Specificity            0.9980   0.9962   0.9916   0.9985   0.9998
## Pos Pred Value         0.9950   0.9843   0.9610   0.9923   0.9989
## Neg Pred Value         0.9991   0.9975   0.9958   0.9922   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1915   0.1709   0.1574   0.1835
## Detection Prevalence   0.2853   0.1945   0.1778   0.1586   0.1837
## Balanced Accuracy      0.9979   0.9928   0.9859   0.9794   0.9993
```

We can see that after running our model on the test set we have an accuracy around 98.7%. Looking at the sensitivity and specificity within each classe we can also see that it is around 96-99% for each classe.

I tried some other models such as svm, rda and lda as well as an ensemble of some of these models but since our original model was about 99% accurate this was mainly for fun.

The model was then run on the test set for evaluation and it achieved a score of 20/20.