Practical Machine Learning Project

Dana Jacob

January 7, 2016

library(knitr)
library(markdown)
library(ggplot2) library(lattice) library(plyr)

Summary

This report is for the class project for Practical Machine Learning. The goal is to develop a model to predict the manner in which our subjects did a set of exercises. We'll use a training dataset and a testing dataset from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). The "classe" variable in the datasets is the outcome variable. We'll explore using other variables in the training dataset to predict classe. Once we have the prediction model built, we'll use it to predict 20 different test cases in the testing dataset.

Load 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.
```

```
library(rpart)
library(AppliedPredictiveModeling)
library(e1071)
##download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv", destfile = "pmlTraining.csv")
##download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv", destfile = "pmlTesting.csv")
##read datasets
trainingSet <- read.csv('pmlTraining.csv')
testingSet <- read.csv('pmlTesting.csv')
dim(trainingSet)
```

```
## [1] 19622   160
```

```
dim(testingSet)
```

```
## [1]  20 160
```

There are 160 variables in these datasets. We'll tidy up this dataset before we run modeling. We'll omit variables that are mostly empty, or with NA as values, and some of the user name, time stamp variables which will not be appropriate to use as predictors.

```
trainingSet <-trainingSet[,-c(1:7)]
zeroV <- nearZeroVar(trainingSet)
trainingSet <- trainingSet[-zeroV]
NAs <- apply(trainingSet,2,function(x) {sum(is.na(x))})
trainingSet <- trainingSet[,which(NAs == 0)]
dim(trainingSet)
```

```
## [1] 19622    53
```

```
names(trainingSet)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

We now have 53 'good' variables in our tidy dataset. With this many variables, Random Forest method would be appropriate.

First we'll split the training dataset into training and testing sets to build and cross validate the model for the writeup portion of the project. The original testing dataset will be used for the quiz portion of the project after we finalize the model.

```
set.seed(1)
inTrain <- createDataPartition(trainingSet$classe, p=0.7, list = FALSE)
training <- trainingSet[ inTrain,]
testing <- trainingSet[-inTrain,]
dim(training)
```

```
## [1] 13737    53
```

```
dim(testing)
```

```
## [1] 5885   53
```

Now we run Random Forest on the training portion from the original training dataset.

```
model <- randomForest(classe ~ ., data=training)
model
```

```
## 
## Call:
##  randomForest(formula = classe ~ ., data = training) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
## 
##         OOB estimate of  error rate: 0.58%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    5    0    0    1 0.001536098
## B   14 2637    7    0    0 0.007900677
## C    0   14 2378    4    0 0.007512521
## D    0    0   22 2229    1 0.010213144
## E    0    0    3    8 2514 0.004356436
```

We see the model is quite effective at 0.49% error rate.

We now apply the model to the test portion of our original training dataset to see how the model performs.

```
testPredict <- predict(model, testing)
testResults <- confusionMatrix(testPredict, testing$classe)
testResults
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    3    0    0    0
##          B    1 1135    6    0    0
##          C    0    1 1019    4    0
##          D    0    0    1  958    3
##          E    1    0    0    2 1079
## 
## Overall Statistics
## 
##                Accuracy : 0.9963
##                  95% CI : (0.9943, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9965   0.9932   0.9938   0.9972
## Specificity            0.9993   0.9985   0.9990   0.9992   0.9994
## Pos Pred Value         0.9982   0.9939   0.9951   0.9958   0.9972
## Neg Pred Value         0.9995   0.9992   0.9986   0.9988   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1929   0.1732   0.1628   0.1833
## Detection Prevalence   0.2846   0.1941   0.1740   0.1635   0.1839
## Balanced Accuracy      0.9990   0.9975   0.9961   0.9965   0.9983
```

We see the model performed really well, with an accuracy rate of 99.6%.

Now we are ready for the quiz portion of the project. We'll apply the model to the original test dataset we downloaded previously. In order to do this, we need to tidy up the test dataset the exact same way as we did with the training dataset:

```
testingSet <-testingSet[,-c(1:7)]

zeroV <- nearZeroVar(testingSet)
testingSet <- testingSet[-zeroV]

NAs <- apply(testingSet,2,function(x) {sum(is.na(x))})
testingSet <- testingSet[,which(NAs == 0)]

dim(testingSet)
```

```
## [1] 20 53
```

```
names(testingSet)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "problem_id"
```

We see that we have the same variables in our tidy test dataset as in our tidy train dataset. We then apply the model to this dataset.

```
quizPredict <- predict(model, testingSet)
quizPredict
```

```
##  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
```