Practical Machine Learning Human Activity Recognition

E. Manton

July 08, 2017

Executive Summary

The goal of this project is to *predict the manner* in which six study participants performed dumbell lifts. The participants were fitted with accelerometers on their belts, forearms, and upper arms. The dumbells used by each participant were also fitted with accelerometers.

The data, provided by Groupware@LES, was broken into TEST and TRAINING sets. The development of the prediction model is described herein. The prediction model was then applied Course Project Prediction Quiz for automated grading. Results indicate that, with the Kappa statistic and the out-of-sample error, that our chosen Random Forest model will work well in predicting outcomes in the **TEST** data set.

The classe Variable

The variable *classe* is the *dependent* variable in the **TRAINING** data set. It's importance is that, with the development of a prediction model for the given data, the *classe* variable will be able to predict the outcomes of the cases provided in the **TEST** data set.

Prediction Variables

The *Prediction Variables* will be any variables within the **TRAINING** data set that contain values. Variables containing 'N/A' won't be used as part of the prediction model. Also, identifying columns (i.e. name, timestamp, etc.) will also be removed from the **TRAINING** data set because they are not relevant in predicting outcomes in the **TEST** data set.

Building The Model

Load and Clean Data

First we will load and clean both the **TEST** and **TRAINING** data sets, and set the seed.

```
rm(list = ls())
setwd('/Users/Mommy/OneDrive/Coursera/Data Science Specialization/Practical Machine Learning/Assignment

library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
training_data <- read.csv(file="my-pml-training.csv",na.strings=c('NA','','#DIV/0!'))
test_data <- read.csv(file="my-pml-testing.csv",na.strings=c('NA','','#DIV/0!'))
set.seed(23232)
```

Remove NA Variables

Remove variables from the **TEST** data set that contain only NA. We do not want NA variables in the **TRAINING** set as part of the prediction model.

```
cols_with_NA<-colnames(test_data)[colSums(is.na(test_data)) > 0]
training_data<-training_data[,!(names(training_data) %in% cols_with_NA)]
```

Remove NZV Variables

NZV (near zero variance) variables have a low fraction of unique values within the sample, usually less than 10%. These do not make good prediction variables, so we will remove them from the **TRAINING** data set using the *nearZeroVar()* function.

```
nzv <-nearZeroVar(training_data, saveMetrics=TRUE)
training_data <- training_data[,nzv$nzv==FALSE]
```

Remove Identifying Columns

The first six variables of the **TRAINING** set are:

1. X
2. user_name
3. raw_timestamp_part_1
4. raw_timestamp_part_2
5. cvtd_timestamp
6. num_window

and as showing here from the **TRAINING** data set itself:

```
training_data[1,c(1,2,3,4,5,6)]
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2  cvtd_timestamp
## 1 1  carlitos           1323084231               788290 5/12/2011 11:23
##   num_window
## 1         11
```

These columns contain information identifying the participant, and do not factor in creating the prediction model. We will remove them:

```
training_data <- training_data[,-(1:6)]
```

These columns are left in the **TRAINING** data set, and will be used to develop the prediction model:

```
names(training_data)
```

```
##  [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"
```

Cross Validation

Train the Data

For Cross Validation of the **TRAINING** data set, we will use the *trainControl()* function within the *caret* package, where paremeters:

1. cv = cross validation
2. number = numbers of folds (we will use two)
3. verboseIter = create a log (set to FALSE since we do not need a log)

```
traincon <- trainControl(method = "cv", number=2, verboseIter=FALSE)
```

Random Forest

We will use the *Random Forest* model with the prediction variables that we now have in the **TRAINING** data set, along with the *traincontrol* arguments that we have established in the prior step, to get the value of the *classe* variable:

```
random_forest <- train(classe ~ ., data=training_data, method="rf",trControl=traincon)
```

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

Final Model

Now we fit the final model so that we may make predictions with the **TEST** data set:

```
random_forest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.47%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5576    2    1    0    1 0.0007168459
## B   21 3770    5    1    0 0.0071108770
## C    0   11 3400   11    0 0.0064289889
## D    0    0   25 3188    3 0.0087064677
## E    0    1    4    6 3596 0.0030496257
```

```r
random_forest
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 9810, 9812
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9897564  0.9870401
##   27    0.9902151  0.9876217
##   52    0.9840489  0.9798200
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Accuracy

At this point we should check our Random Forest model for accuracy by finding the Kappa statistic, which is a metric that compares an observed accuracy with a random choice, or expected accuracy. We will use the *max()* function for this purpose, and with these values we can determine if our model will be a good predictor:

```r
max(random_forest$results$Accuracy)
```

```
## [1] 0.9902151
```

```r
max(random_forest$results$Kappa)
```

```
## [1] 0.9876217
```

Expected Sample Error

To find the out-of-sample error that our model has, we again apply the *max()* function prepended with "1-".

```r
1-max(random_forest$results$Accuracy)
```

```
## [1] 0.009784852
```

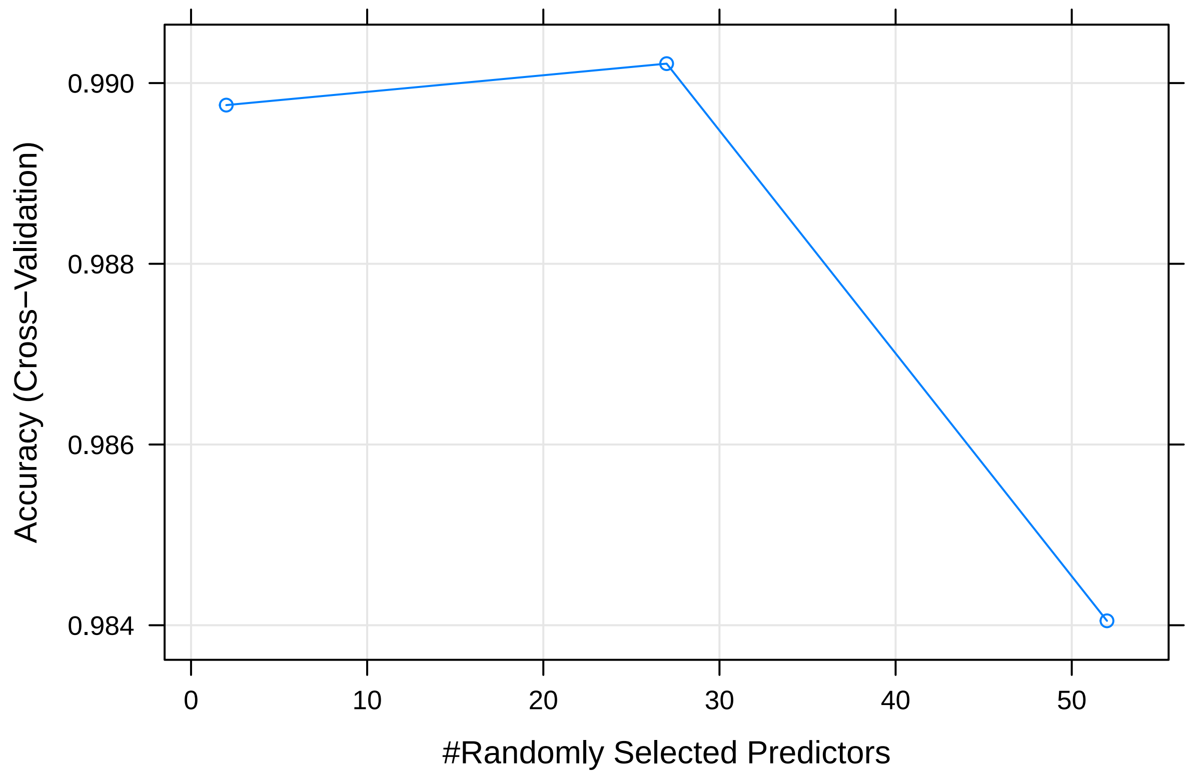
Choices Made

We made these choices in developing our model:

1. exclude NA variables from the **TRAINING** data set
2. exclude NZA variables from the **TRAINING** data set
3. exclude the first six columns of identifying data from the **TRAINING** data set
4. use the Random Forest model to predict

Plots

```
plot(random_forest)
```


Accuracy (Cross−Validation)
#Randomly Selected Predictors