Practical Machine Learning - Human Activity Recognition

Coursera Practical Machine Learning Project

Kevin Dyke

28 February 2016

Background

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).

Project Summary

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. The project may use any of the other variables to predict with. This report describes how I built the model, how I used cross validation, what I think the expected out of sample error is, and why I made the choices you did. The prediction model is also used to predict 20 different test cases.

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The first thing to do is to download the data if necessary

```
# set the working directory
setwd("C:/coursera/Data Science/machine learning/Project/Coursera-Data-Science-Machine-Learning-Project

# set up variables describing data locations
train_Url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_Url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- "./data/pml-training.csv"
testing  <- "./data/pml-testing.csv"
```

```
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(training)) {
  download.file(train_Url, destfile=training, mode = "w")
}
if (!file.exists(testing)) {
  download.file(test_Url, destfile=testing, mode = "w")
}

# read the data into memory and report the sizes
train <- read.csv(training)
test  <- read.csv(testing)
dim(train)
```

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

```
dim(test)
```

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

Explorary Data Analysis and Data Cleaning

The training set consists of 19622 observations of 160 variables. Whereas the test set consists of 20 observations of 160 variables. This is a very large number of variables which could cause problems for computational intense operations. Therefore we need to reduce the number of variables in the training set.

A summary of the first twenty observations shows that the data contains many unknown values. These columns can be removed.

```
str(train,list.len=20)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##   [list output truncated]
```

The first seven variables are used for logging purposes and so can be removed. Also any column with zero variability can be removed. This can be achieved using the nearZeroVar method in the Caret package.

```
train <- train[, -(1:7)] # remove the columns used for logging purposes
test <- test[,-(1:7)]
library(caret,quietly = TRUE)
train <- train[, colSums(is.na(train)) == 0] # remove any columns containg NAs
test  <- test[, colSums(is.na(test)) == 0]

nzvVar <- nearZeroVar(train) # obtain a list of columns with zero variability
train <- train[,-nzvVar]     # and remove them from the training set
```

After cleaning the training set consists of 19622 observations of 53 variables and the test set consists of 20 observations of 53 variables.

Create Training and Validation Sets

The training set needs to be divided into two subsets. The first set will be used for training purposes. The second will be used to validate the training. This allows the test set to be left alone for the final predictions. I decided to use a validation size of 25%.

```
# choose 75% of the training data
choosen <- createDataPartition(y=train$classe, p=0.75, list=FALSE)
# create the training set
training  <- train[choosen, ]
# create the validation by selecting the remainding 25%
validation <- train[-choosen, ]
```

There are 14718 rows in the training set and 4904 rows in the validation model.

Fitting A Prediction Model

A Random Forest model would seem to be a good prediction model for this project. The RF model will automatically select the most important variables. A list of the most important variables can be found in appendix A. The RF model has good robustness to the correlated covariates and outliers. A five fold cross validation process will be applied to the model. The number of trees used is limited to 250.

```
library(randomForest,quietly = TRUE)
```

```
## 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
```

```
# fit the prediction model
fitRf <- train(classe ~ .,data=training,method="rf",trControl=trainControl(method="cv", 5), ntree=250)
fitRf # display the results
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11774, 11774, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9900119  0.9873633  0.002374955  0.003006995
##   27    0.9912348  0.9889107  0.003183098  0.004029015
##   52    0.9827416  0.9781641  0.004749135  0.006012087
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

This model is plotted in the appendix B.

The next step is to apply the model to the validation data and analyse the results via a confusion Matrix.

```
validationRf <- predict(fitRf, validation)
confusionMatrix(validation$classe, validationRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    0    0    0    1
##          B    7  938    3    1    0
##          C    0    3  851    1    0
##          D    0    0    5  798    1
##          E    0    0    1    5  895
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9928
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9950   0.9968   0.9895   0.9913   0.9978
## Specificity            0.9997   0.9972   0.9990   0.9985   0.9985
## Pos Pred Value         0.9993   0.9884   0.9953   0.9925   0.9933
## Neg Pred Value         0.9980   0.9992   0.9978   0.9983   0.9995
## Prevalence             0.2857   0.1919   0.1754   0.1642   0.1829
## Detection Rate         0.2843   0.1913   0.1735   0.1627   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9974   0.9970   0.9943   0.9949   0.9981
```

Then the next step is to extract the accuracy figures.

```
accuracy <- postResample(validationRf, validation$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9942904 0.9927773
```

The accuracy is 99.43% and the Kappa is 99.28%.

Out-Of-Sample Error

The out of sample error is 1 - the testing accuracy

```
oose <- 1 - as.numeric(confusionMatrix(validation$classe, validationRf)$overall[1])
oose
```

```
## [1] 0.005709625
```

So the out of sample error is 0.57%.

Submitted Results

The training set prediction model can now be used to predict the answers to the twenty observations in the test set.

```
answers <- predict(fitRf, test) # get the answers to the twenty test questions
answers                         # display the answers
```

```
##  [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
```

These answers were submitted to the Coursera website and passed with a mark of 100%.

Conclusion

The training set was reduced from 160 observations to 53 allowing faster computation. A Random Forest model was a good predict model with an accuracy of 99.43% and an out of sample error of 0.57%.

This model was applied to the test set and the predicted answers passed the project quiz with a mark of 100%.

References

The author acknowledges the generosity of Groupware@LES in allowing the use of their data in this project.

The citation is: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Appendix A - Variable Important

Below is a list of the most important variables in the Random Forest Model

```
VarImp <- varImp(fitRf)
VarImp
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                       Overall
## roll_belt             100.000
## pitch_forearm          62.843
## yaw_belt               54.146
## pitch_belt             46.776
## magnet_dumbbell_z      42.613
## magnet_dumbbell_y      41.964
## roll_forearm           40.106
## accel_dumbbell_y       21.819
## roll_dumbbell          18.509
## magnet_dumbbell_x      18.373
## accel_forearm_x        16.394
## magnet_belt_z          15.798
## accel_dumbbell_z       14.936
## total_accel_dumbbell   14.675
## magnet_belt_y          14.055
## accel_belt_z           13.554
## magnet_forearm_z       12.238
## yaw_arm                11.239
## gyros_belt_z           10.939
## magnet_belt_x           9.237
```

Appendix B - Random Forest Model

```
library(rpart,quietly = TRUE)
library(rpart.plot, quietly = T)
treeModel <- rpart(classe ~ ., data=train, method="class")
prp(treeModel)
```