Practical Machine Learning Project Writeup - Predicting Exercise Performance

emeko

January 29, 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, the 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).

Loading libraries

```
library(caret)
library(dplyr)
library(randomForest)
```

Get, Load and Clean Data

The following code chunk will download the source files from the URLs provided, read the CSV files and clean the data as we read it in by excluding NAs and zero values.

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

```
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url_train, "./train.csv", method = "curl")
download.file(url_test, "./test.csv", method = "curl")
train <- read.csv("./train.csv", na.strings = c("", "NA", "#DIV/0!"))
test <- read.csv("./test.csv", na.strings = c("", "NA", "#DIV/0!"))
```

Building our model

Let's partition the source training data into a training and test dataset. We are partitioning the data as 70% training and 30% test.

```
## create the training set
table(train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
set.seed(999)
inTrain <- createDataPartition(train$classe, p = 0.7, list = FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

Refine the training data

Initial analysis of the data showed that there are features which are not likely to contribute to the accuracy of the model. Using the nearZeroVar() function, remove these features from our training set.

```
## remove features that are not likely to contribute to the model
nsv <- nearZeroVar(training)
training <- training[, -nsv]
training <- select(training, -(X:num_window))
training<-training[,colSums(is.na(training))/nrow(training) < 0.50]
```

Training the model

Let's use randomForest to train the model. We choose randomForest for the follwing reasons.

- Accuracy
- Runs efficiently on large data bases
- Handles thousands of input variables without variable deletion
- Gives estimates of what variables are important in the classification
- Generates an internal unbiased estimate of the generalization error as the forest building progresses
- Provides effective methods for estimating missing data
- Maintains accuracy when a large proportion of the data are missing
- Provides methods for balancing error in class population unbalanced data sets
- Generated forests can be saved for future use on other data
- Prototypes are computed that give information about the relation between the variables and the classification.
- Computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data
- Capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection
- Offers an experimental method for detecting variable interactions

```
## Training the model
RFmodel <- randomForest(classe~., data = training, method = "class")
```

Use cross validation on our trained model to determine the accuracy.

```
## Cross validate the model
myPredict <- predict(RFmodel, testing, type = "class")
confusionMatrix(myPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    7    0    0    0
##          B    0 1129    5    0    0
##          C    0    3 1021    9    1
##          D    0    0    0  954    1
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9912   0.9951   0.9896   0.9982
## Specificity            0.9983   0.9989   0.9973   0.9998   0.9998
## Pos Pred Value         0.9958   0.9956   0.9874   0.9990   0.9991
## Neg Pred Value         1.0000   0.9979   0.9990   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1918   0.1735   0.1621   0.1835
## Detection Prevalence   0.2856   0.1927   0.1757   0.1623   0.1837
## Balanced Accuracy      0.9992   0.9951   0.9962   0.9947   0.9990
```

Based on the cross validation performed, the model has an accuracy of 99.54% with confidence intervals (0.9933, 0.997). This is a very high accuracy rate and based on that accuracy, one would expect an out of sample error rate of 0.46% (calulated as 1 - .9954 = 0.0046).

Run predictions against the source test set

Using our model against the source test set, let's generate the results for 20 tests.

```
## Run our prediction against the test set
pResults <- predict(RFmodel, test, type = "class")
pResults
```

```
##  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
```

Submit results

Submitting the result above to the project quiz yields a 20/20 result.