Machine Learning - prediction Assignment

Sunday Edisi

June 18, 2017

Background

This data set was created by measuring several individuals performing a weight lifting exercise, in one of several ways: correctly, and while making one of several common mistakes.

Various physical measurements of their movement were made recording position, momentum, orientation, of several body parts. The goal was to be able to detect whether the exercise is performed correctly, or detect a specific mistake, based on these physical measurements.

To simplify the analysis, the raw digital signals were summarized by some summary statistics over various time-slices. Analysis

For this course assignment we are provided with some subset of this data, with labeled outcomes (the file pml-training.csv), as well as 20 cases (the file pml-testing.csv), whose outcomes are unlabeled and we have to guess.

We describe below each step of the analysis performed: Loading and cleaning the data

We find that the data contains mostly numerical features. However, many of them contain nonstandard coded missing values. In addition to the standard NA, there are also empty strings "", and error expressions "#DIV/0!".

Data description

The outcome variable is classe, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

```
exactly according to the specification (Class A)
throwing the elbows to the front (Class B)
lifting the dumbbell only halfway (Class C)
lowering the dumbbell only halfway (Class D)
throwing the hips to the front (Class E)
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```r
library(scales)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.3
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Attaching package: ‘dplyr’

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

```
## Warning: package 'lattice' was built under R version 3.3.3
```

Loading required package: lattice

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

randomForest 4.6-10

Type rfNews() to see new features/changes/bug fixes.

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.3.3
```

```
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
```

Data processing

In this section the data is processed. Some basic transformations and cleanup will be performed, so that NA values are omitted. Irrelevant columns such as user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, and num_window (columns 1 to 7) will be removed in the subset.

The pml-training.csv data is used to devise training and testing sets. The pml-test.csv data is used to predict and answer the 20 questions based on the trained model.

```
training <- read.csv("C:/Users/hp-pc/Desktop/data/pml-training.csv", row.names = 1, stringsAsFactors =
  mutate(cvtd_timestamp = mdy_hm(cvtd_timestamp),
         user_name      = as.factor(user_name),
         new_window     = as.factor(new_window),
         classe         = as.factor(classe))
```

```
## Warning in as.POSIXlt.POSIXct(x, tz): unable to identify current timezone 'H':
## please set environment variable 'TZ'
```

```
## Warning: 6472 failed to parse.
```

```
testing <- read.csv("C:/Users/hp-pc/Desktop/data/pml-testing.csv", row.names = 1, stringsAsFactors = FA
  mutate(cvtd_timestamp = mdy_hm(cvtd_timestamp),
         user_name      = as.factor(user_name),
         new_window     = as.factor(new_window))
```

```
## Warning: 12 failed to parse.
```

Subset data

```
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
```

Cross-validation

In this section cross-validation will be performed by splitting the training data in training (75%) and testing (25%) data.

```
subSamples <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subTraining <- training[subSamples, ]
subTesting <- training[-subSamples, ]
```

Expected out-of-sample error

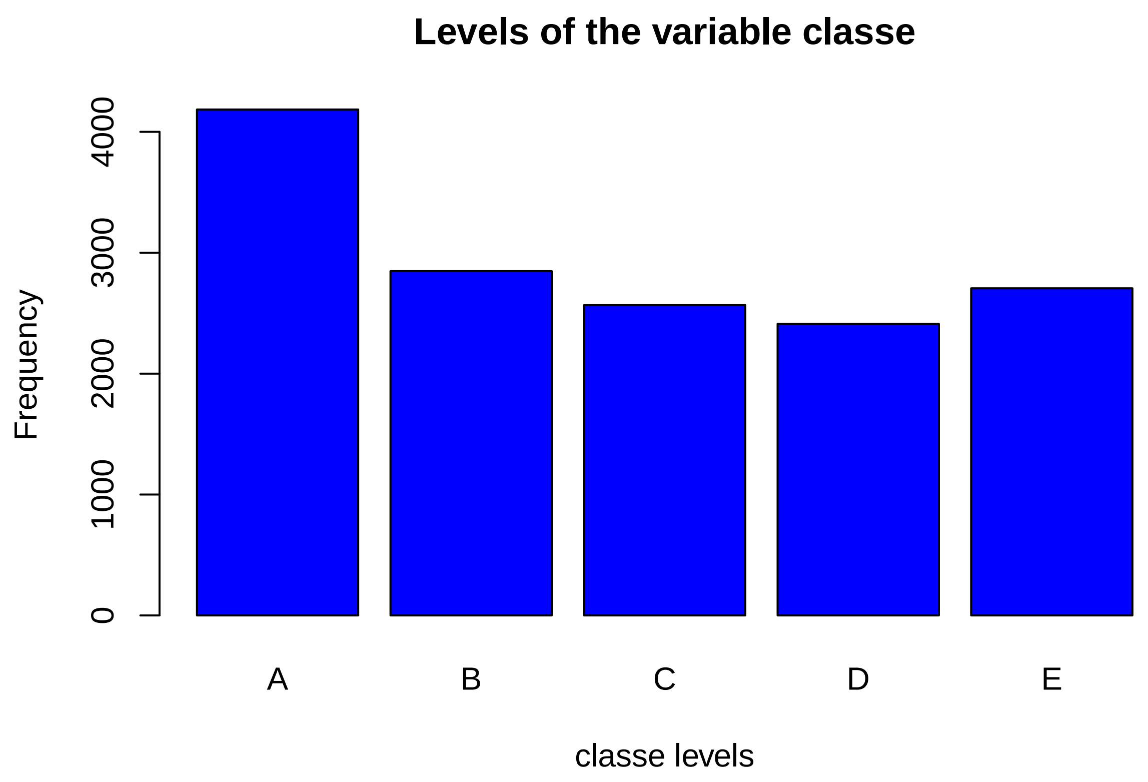
The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set).

Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Exploratory analysis

The variable classe contains 5 levels. The plot of the outcome variable shows the frequency of each levels in the subTraining data.

```
plot(subTraining$classe, col="blue", main="Levels of the variable classe", xlab="classe levels", ylab="F
```



The plot above shows that Level A is the most frequent classe. D appears to be the least frequent one.

Prediction models

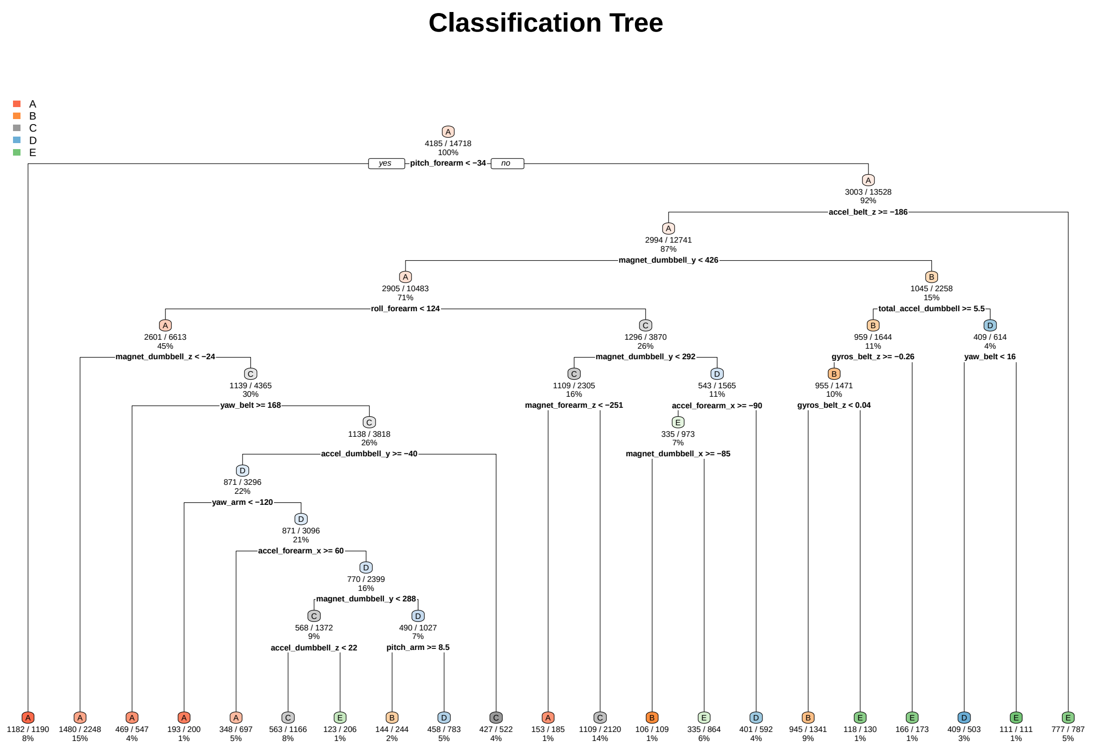
In this section a decision tree and random forest will be applied to the data.

Decision Tree

```
# Fit model
modFitDT <- rpart(classe ~ ., data=subTraining, method="class")
```

```r
# Perform prediction
predictDT <- predict(modFitDT, subTesting, type = "class")

# Plot result
rpart.plot(modFitDT, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Following confusion matrix shows the errors of the prediction algorithm.

```r
confusionMatrix(predictDT, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1246  200   47  105   74
##          B   37  348   56   29   57
##          C   39  198  684  205  169
##          D   45  111   17  418   57
##          E   28   92   51   47  544
##
## Overall Statistics
##
##                Accuracy : 0.6607
##                  95% CI : (0.6472, 0.6739)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                    Kappa : 0.5683
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8932  0.36670   0.8000  0.51990   0.6038
## Specificity            0.8786  0.95474   0.8491  0.94390   0.9455
## Pos Pred Value         0.7452  0.66034   0.5282  0.64506   0.7139
## Neg Pred Value         0.9539  0.86269   0.9526  0.90930   0.9138
## Prevalence             0.2845  0.19352   0.1743  0.16395   0.1837
## Detection Rate         0.2541  0.07096   0.1395  0.08524   0.1109
## Detection Prevalence   0.3409  0.10746   0.2641  0.13214   0.1554
## Balanced Accuracy      0.8859  0.66072   0.8245  0.73190   0.7747
```

Random forest

```
# Fit model
modFitRF <- randomForest(classe ~ ., data=subTraining, method="class")

# Perform prediction
predictRF <- predict(modFitRF, subTesting, type = "class")
```

Following confusion matrix shows the errors of the prediction algorithm.

```
confusionMatrix(predictRF, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    8    0    0    0
##          B    0  939    7    0    0
##          C    0    2  848   11    3
##          D    0    0    0  793    1
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9908, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9917
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9918   0.9863   0.9956
## Specificity            0.9977   0.9982   0.9960   0.9998   1.0000
## Pos Pred Value         0.9943   0.9926   0.9815   0.9987   1.0000
## Neg Pred Value         1.0000   0.9975   0.9983   0.9973   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1729   0.1617   0.1829
## Detection Prevalence   0.2861   0.1929   0.1762   0.1619   0.1829
## Balanced Accuracy      0.9989   0.9938   0.9939   0.9930   0.9978
```

Conclusion

Result

The confusion matrices show, that the Random Forest algorithm performens better than decision trees. The accuracy for the Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. The random Forest model is choosen.

Expected out-of-sample error

The expected out-of-sample error is estimated at 0.005, or 0.5%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified.

Submission

In this section the files for the project submission are generated using the random forest algorithm on the testing data.

```
# Perform prediction
predictSubmission <- predict(modFitRF, testing, type="class")
predictSubmission

# Write files for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("./data/submission/problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictSubmission)
```