Practical Machine Learning - Course Project

Victor Sousa

November 16, 2021

Executive Sumary

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

Goals

Based on dataset provide by HAR, we will train a predictive model to predict what exercice was accomplished. For this, we dispose of 159 features. We'll proceed with the following steps: * Obtain and Process the data * Data Exploration, focussing on top parameters we are interessed * Model Selection * Model Examination * Conclusions, answers questions based on the date * Predicting on test set and file creation for submission

Obtain Data

```
train.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test.url  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if (!file.exists ("pml-training.csv")) {
        download.file(train.url, destfile="pml-training.csv")
}
if (!file.exists("pml-testing.csv")) {
        download.file(test.url, destfile="pml-testing.csv")
}
```

Pre-processing and Analyses

```
training.raw <- read.csv("pml-training.csv")
testing.raw <- read.csv("pml-testing.csv")
```

```
dim(training.raw)
```

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

```
# All these outputs was hidden by excessive verbosity
head(training.raw)
str(training.raw)
summary(training.raw)
```

We need to handle the large amount of NA values.

```
maxNA_Percentage <- 15
maxNA_counter <-nrow (training.raw) / 100 * maxNA_Percentage
removeColumns <- which(colSums(is.na(training.raw) | training.raw =="")> maxNA_counter)
training.cleaned001 <- training.raw[,-removeColumns]
testing.cleaned001 <- testing.raw[,-removeColumns]
# Removing not relevant data
removeColumns <- grep("timestap", names(training.cleaned001))
training.cleaned002 <- training.cleaned001[,-c(1,removeColumns)]
testing.cleaned002 <- testing.cleaned001[,-c(1,removeColumns)]
# new dimensions
dim (training.cleaned002)
```

```
## [1] 19622    59
```

```
dim (testing.cleaned002)
```

```
## [1] 20 59
```

Factor to integers:

```
classeLevel <- levels(training.cleaned002$classe)
# Converting to Data Frame
training.cleaned003 <- data.frame(data.matrix(training.cleaned002))
training.cleaned003$classe <- factor(training.cleaned003$classe, labels = classeLevel)
testing.cleaned003 <- data.frame(data.matrix(testing.cleaned002))
# Final data set, appropriate to use.
ptrain <- training.cleaned003
ptest  <- testing.cleaned003
```

We want to be able to estimate the out-of-sample error, So we randomly split the full training data (ptrain) into a smaller training set (ptrain1) and a validation set (ptrain2):

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
set.seed(16112021)
inTrain<- createDataPartition(y=ptrain$classe, p=0.75, list = FALSE)
ptrain1 <- ptrain[inTrain,]
ptrain2 <- ptrain[-inTrain,]
```

Removing variables with nearly zero variance:

```
# remove variables with nearly zero variance
nzv <- nearZeroVar(ptrain1)
ptrain1 <- ptrain1[, -nzv]
ptrain2 <- ptrain2[, -nzv]

# remove variables that don't make intuitive sense for prediction (X, user_name, raw_timestamp_part_1,
ptrain1 <- ptrain1[, -(1:5)]
ptrain2 <- ptrain2[, -(1:5)]
```

Model Building

Randon Forest

```
# instruct train to use 3-fold CV to select optimal tuning parameters
fitControl <- trainControl(method="cv", number=3, verboseIter=F)

# fit model on ptrain1

start <- proc.time()
fit <- train(classe ~ ., data=ptrain1, method="rf", trControl=fitControl)
fit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.69%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4182    3    0    0    0 0.0007168459
## B   17 2826    5    0    0 0.0077247191
## C    0   18 2547    2    0 0.0077911959
## D    0    0   43 2366    3 0.0190713101
## E    0    0    1    9 2696 0.0036954915
```

```
proc.time() - start
```

```
##    user  system elapsed
## 236.968   1.899 239.335
```

This Random Forest model use 500 trees and 27 features at each split.

Model Evaluation

```
preds <- predict(fit, newdata=ptrain2)
confusionMatrix(ptrain2$classe, preds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    2    0    0    1
##          B    6  940    3    0    0
##          C    0   11  843    1    0
##          D    0    0   13  791    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9896, 0.9947)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9957   0.9864   0.9814   0.9987   0.9989
## Specificity            0.9991   0.9977   0.9970   0.9968   1.0000
## Pos Pred Value         0.9978   0.9905   0.9860   0.9838   1.0000
## Neg Pred Value         0.9983   0.9967   0.9960   0.9998   0.9998
## Prevalence             0.2851   0.1943   0.1752   0.1615   0.1839
## Detection Rate         0.2838   0.1917   0.1719   0.1613   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9974   0.9920   0.9892   0.9978   0.9994
```

This accuracy is 99.3 This is an good result, so we'll use Random Forests to predict on the test set.

Re-training the Selected Model

```r
# remove nearly zero variance
nzv <- nearZeroVar(ptrain)
ptrain <- ptrain[, -nzv]
ptest <- ptest[, -nzv]

# remove variables that don't make intuitive sense for prediction
ptrain <- ptrain[, -(1:5)]
ptest <- ptest[, -(1:5)]


# re-fit model using full training set (ptrain)
fitControl <- trainControl(method="cv", number=3, verboseIter=FALSE)
start <- proc.time()
fit <- train(classe ~ ., data=ptrain, method="rf", trControl=fitControl)
proc.time() - start
```

```
##    user  system elapsed
## 340.767   2.486 343.775
```

Test Set Predictions

```r
preds <- predict(fit, newdata=ptest)
preds <- as.character(preds)
preds
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```r
# create function to write predictions to files
pml_write_files <- function(x) {
    n <- length(x)
    for(i in 1:n) {
        filename <- paste0("problem_id_", i, ".txt")
        write.table(x[i], file=filename, quote=F, row.names=F, col.names=F)
    }
}

pml_write_files(preds)
```