Machine Learning - Exercise Analysis

Jose Diaz

October 15, 2015

Machine Learning - Exercise Analysis

Jose Diaz October 15 2015

Introduction

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. The goal of this project is 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. Participants were asked to perform one set of 10 repetitionsof the Unilateral Dumbbell Biceps Curl invedierent fashions: exactly according to the specication (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specied execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate.

The Approach - Data processing and Exploratory Analysis

We begin by doing some data processing and exploratory analysis to get familiar with our data set. Below is a brief summary of the findings.
All 6 participants performed comparable amount of total exercises
All 5 classes were performed a relatively similar amount of times per user (No Rare Classes)
Raw Training data and raw Test data have the same number of columns but different column atomic data types. This was resolved by converting logical atomic types to factor atomic types

```r
#Loading Necessary Packages
library(readr)
library(ggplot2)
library(caret)
```

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

```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
```

```r
library(magrittr)
library(randomForest)
```

```
## 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
```

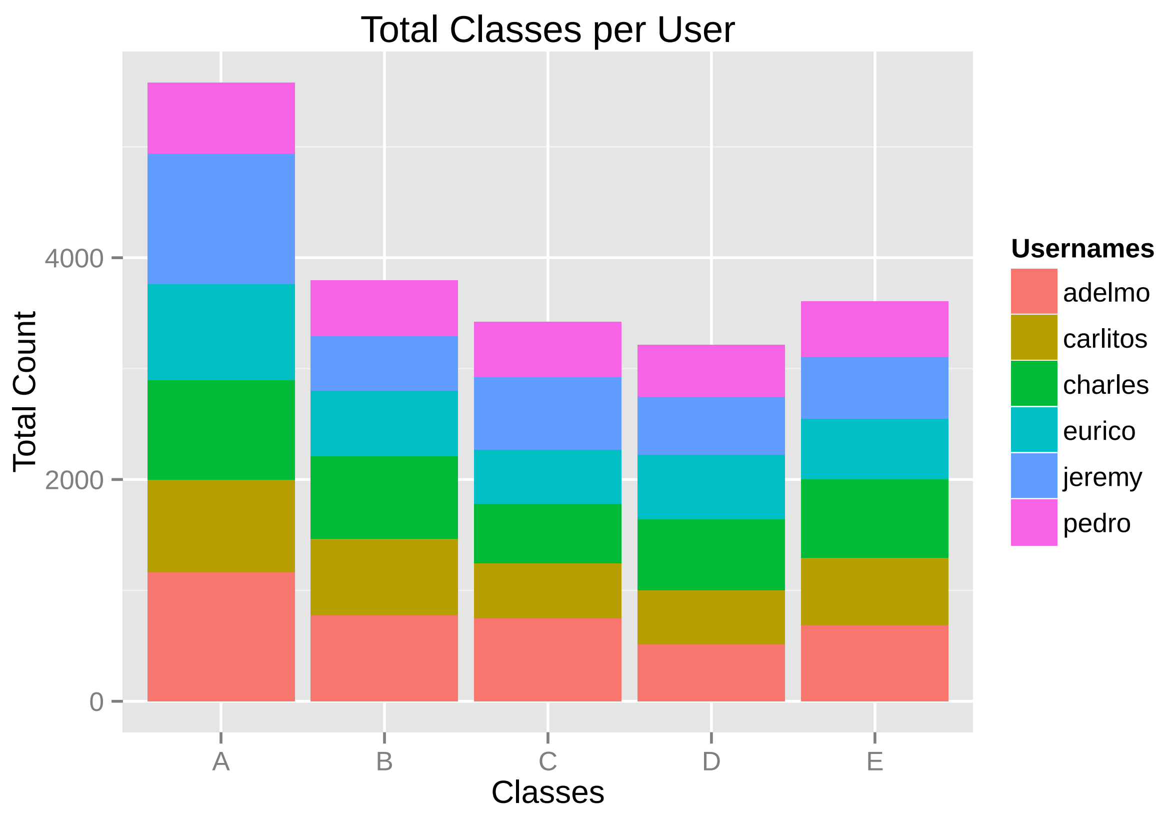
```r
library(RWeka)

#Load our data
training.raw <- read.csv("pml-training.csv")
test.raw <- read.csv("pml-testing.csv")

# Exploratory Analysis
table(training.raw$user_name)
```

```
##
##   adelmo carlitos  charles   eurico   jeremy    pedro
##     3892     3112     3536     3070     3402     2610
```

```r
table(training.raw$classe)
```

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

```r
ggplot(training.raw, aes(x = classe)) + geom_histogram(aes(fill = factor(user_name))) + xlab("Classes")
```



```
test.raw_append <- test.raw[,1:159]
test.raw_append$classe <- NA
test.raw_append$classe <- factor(test.raw_append$classe)

#Resolving different column atomic types - converted logical atomic types to factors and appended it to
#raw training data set.
compare_atomic_types <- unlist(lapply(test.raw_append, class)) != unlist(lapply(training.raw, class))
different_atomic_types <-which(as.logical(compare_atomic_types))


test.raw_append[sapply(test.raw_append, is.logical)] <- lapply(test.raw_append[sapply(test.raw_append, i

training.raw <- rbind(training.raw, test.raw_append)
rm(test.raw_append)
```

The Approach - Feature Selection and NAs

We notice there are 160 variables in our data set, many of which have NAs. Many of these variables are not direct sensor measurements and thus I used regular expressions to extract only the features that are relevant to the sensor data we need. Below is a summary of the findings. * A Total of 50 Columns are left containing: roll, pitch, yaw, classe, and username
Most NAs have been removed because they were associated with the derived features

```
regular_expression <- "(^roll|^pitch|^yaw|classe|user_name|.*_x$|.*_y$|.*_z$).*"
selected_columns <-grep(regular_expression, names(training.raw), perl=TRUE, value=TRUE)
```

```r
training.raw <- training.raw[,selected_columns]
```

The Approach - Machine Learning Algorithm

The machine learning algorithm chosen is the C4.5 algorithm developed by Ross Quinlan. It was ranked #1 in Top 10 Algorithims in Data Mining by Springer LNCS in 2008. C4.5 builds decision trees from a set of training data. The training data is a set S = {s1, s2, ...} of already classified samples. Each sample si consists of a p-dimensional vector (x{1,i}, x{2,i}, ...,x{p,i}) , where the xj represent attribute values or features of the sample, as well as the class in which si falls. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

Below are key findings. I expect the out of sample errors to be quite low because of the quality of the training data. Most NA's have been removed and only relevant features have been kept. Additionally, there are no rare classes - I expect the tree based algorithm to do quite well.

- The Out of sample error is .5249%
- Estimated Accuracy of the model is 99.4751%!
- The submission to Coursera showed 19/20 predictions were accurate

```r
test.data <- tail(training.raw,20)
train.data <- training.raw[-20,]

fit <- J48(classe ~ ., data = train.data)
summary(fit)
```

```
##
## === Summary ===
##
## Correctly Classified Instances       19518               99.4751 %
## Incorrectly Classified Instances       103                0.5249 %
## Kappa statistic                          0.9934
## Mean absolute error                      0.0038
## Root mean squared error                  0.0433
## Relative absolute error                  1.1874 %
## Root relative squared error             10.8968 %
## Coverage of cases (0.95 level)          99.7044 %
## Mean rel. region size (0.95 level)      20.3099 %
## Total Number of Instances            19621
##
## === Confusion Matrix ===
##
##     a    b    c    d    e   <-- classified as
##  5568    4    2    2    3 |    a = A
##    10 3776    6    3    2 |    b = B
##     2   13 3394    8    5 |    c = C
##     3    6   15 3186    6 |    d = D
##     2    4    3    4 3594 |    e = E
```

```r
predictions <- predict(fit, newdata = test.data)
tail(predictions, 20)
```

```
##  [1] A 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
```

Generating the files for submission

```r
submission_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

submission_files(predictions)
```