Machine Learning on Weight Lifting Exercise Dataset

kimnewzealand

27 June 2017

Setup

Load packages

```
library(ggplot2)
library(dplyr)
library(knitr)
library(caret)
library(rpart)
library(randomForest)
library(parallel)
library(doParallel)
```

Load data

```
# Load datasets, first checking if the file exists in the working directory.
setwd("~/MachLearning")
if (!file.exists("pml-training.csv")) { download.file("https://d396qusza40orc.cloudfront.net/predmachlea
}
if (!file.exists("pml-testing.csv")) {
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",destfile="./pmltest
}
pmltrain <- read.csv("pmltraining.csv")
pmltest <- read.csv("pmltesting.csv")
```

Summary

The goal of this project is to predict the manner in which a participant does the Weight Lifting Exercise; using sensors on the belt, forearm, arm, and dumbbell.

This would be useful to assess and provide feedback to an athlete on how to correct mistakes in the weight lifting exercises. This is called qualitative activity recognition.

Part 1: Data

We will first take a look at the data structure of the Weight Lifting Exercise Dataset. The data is sourced from this website: http://groupware.les.inf.puc-rio.br/har, and further information from this paper:

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.

As per the dataset documentation, 6 male participants aged between 20-28 years were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five ways, the first doing it well, and the other 4 as mistakes:
- 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)

There were four sensors in each of the users' glove, armband, lumbar belt and dumbbell. Four each of these in each sliding window the following were measured:
- Euler angles (roll, pitch and yaw)
- For each Euler angle: mean, variance, standard deviation, max, min, amplitude, kurtosis and skewness
- Raw accelerometer, gyroscope and magnetometer readings

Part 2: Exploratory data analysis

There are 19622 observations and 160 variables in the **pmltrain** dataset.

There are 20 observations and 160 variables in the **pmltest** dataset.

The data has already been divided into training and testing sets, however we would expect a higher ratio instead the given test set of 20 observations. Additionally this pmltest does not have the outcome variable. This set will be used in the final 'test'.

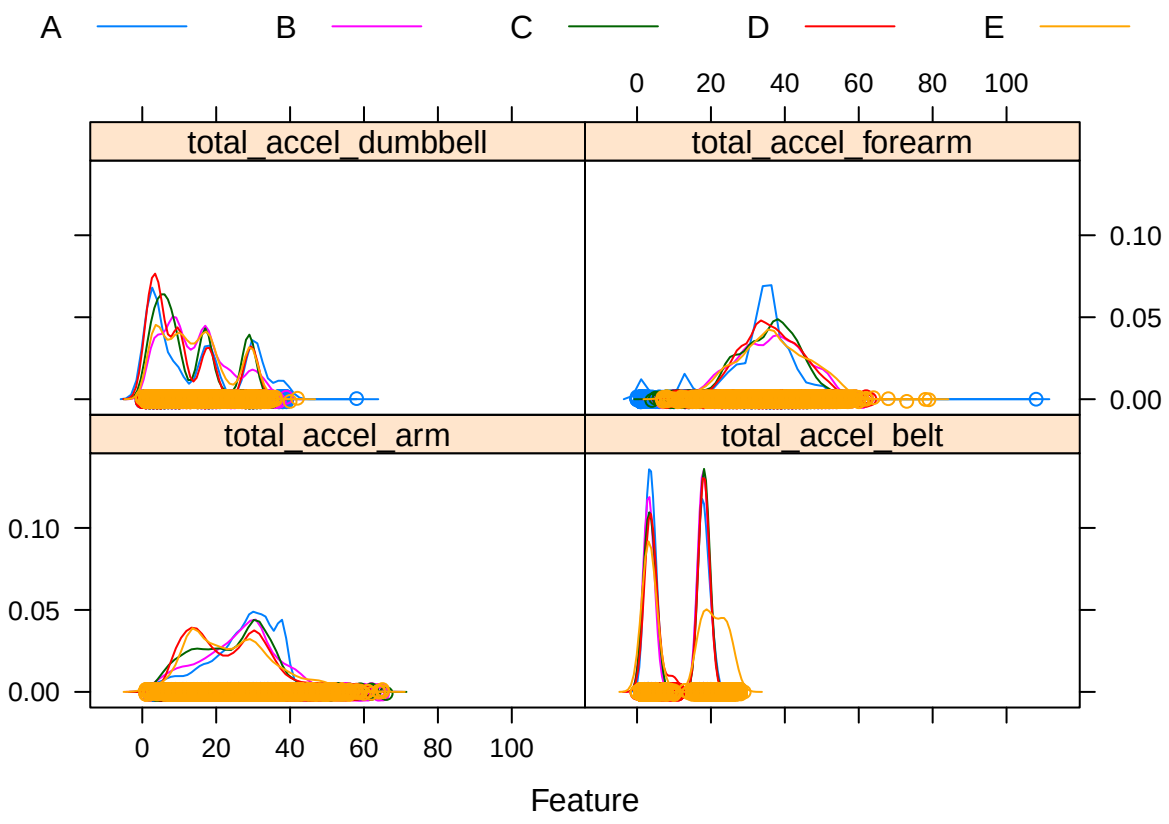
We will divide the pmltrain up into testing and training sets after the EDA, for our modelling.

```
# Summary statistics of the categorical outcome variable ie "classe" variable in the training set.
kable(summary(pmltrain$classe),format="pandoc")
```

A	5580
B	3797
C	3422
D	3216
E	3607

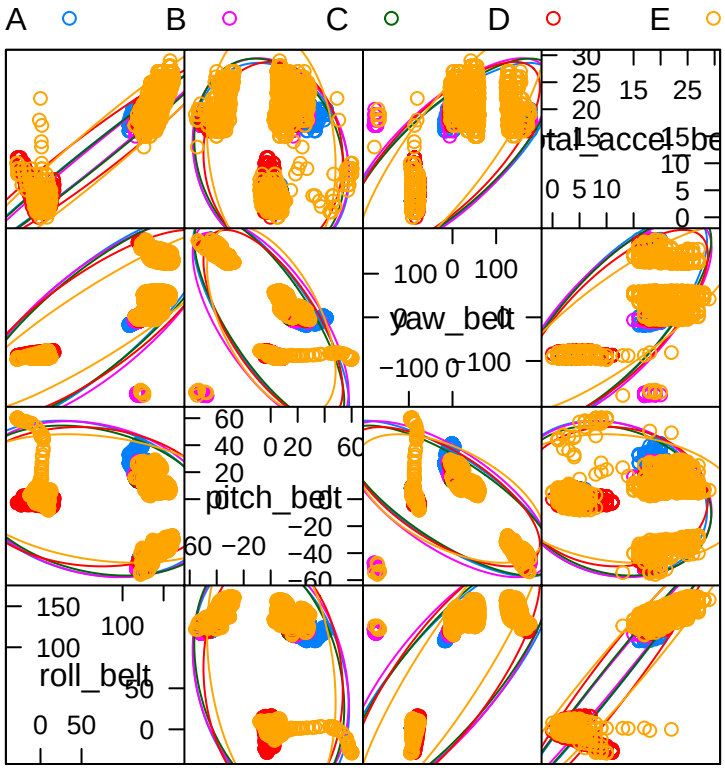
The classe A appears to be the most frequent of the 5 classes.

```
# Take a look at the density plot for the total accel for each measure using the featurePlot function i
featurePlot(x = pmltrain[,c("total_accel_belt","total_accel_arm","total_accel_dumbbell","total_accel_fo
            y = pmltrain$classe,
            plot = "density",
            type = c("p", "smooth"),
            auto.key = list(columns = 5))
```



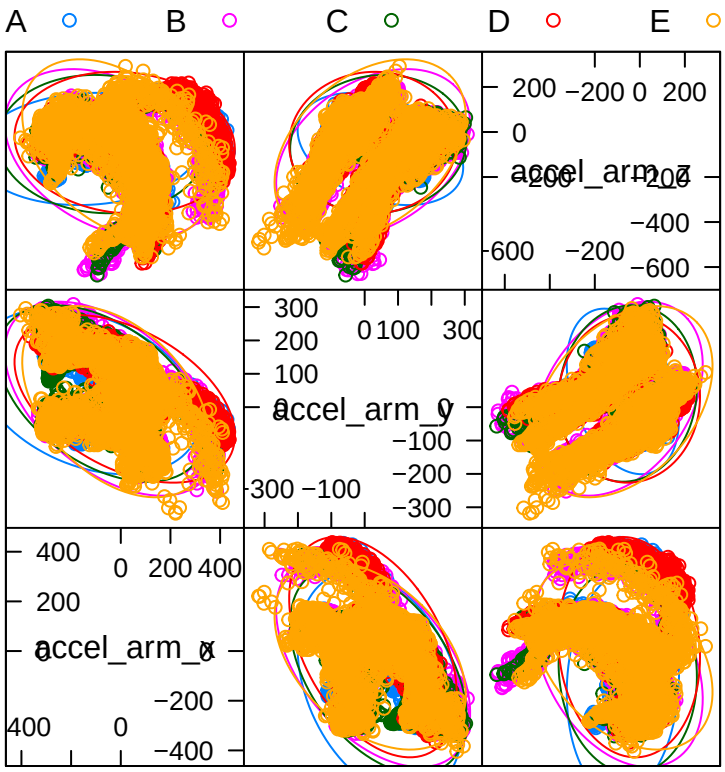
The total acceleration arm, belt, dumbbell and forearm curves appear to be bimodal, high peaks bimodal, multimodal and almost normal respectively, where the belt and forearm sensors seem to be noticeable for classe A, but there is not much differences in the classes for the arm and dumbbell sensors.

```
# Take a further look at the pairs plot for the belt measurement
featurePlot(x = pmltrain[, 8:11],
            y = pmltrain$classe,
            plot = "ellipse",
            auto.key = list(columns = 5))
```



Scatter Plot Matrix

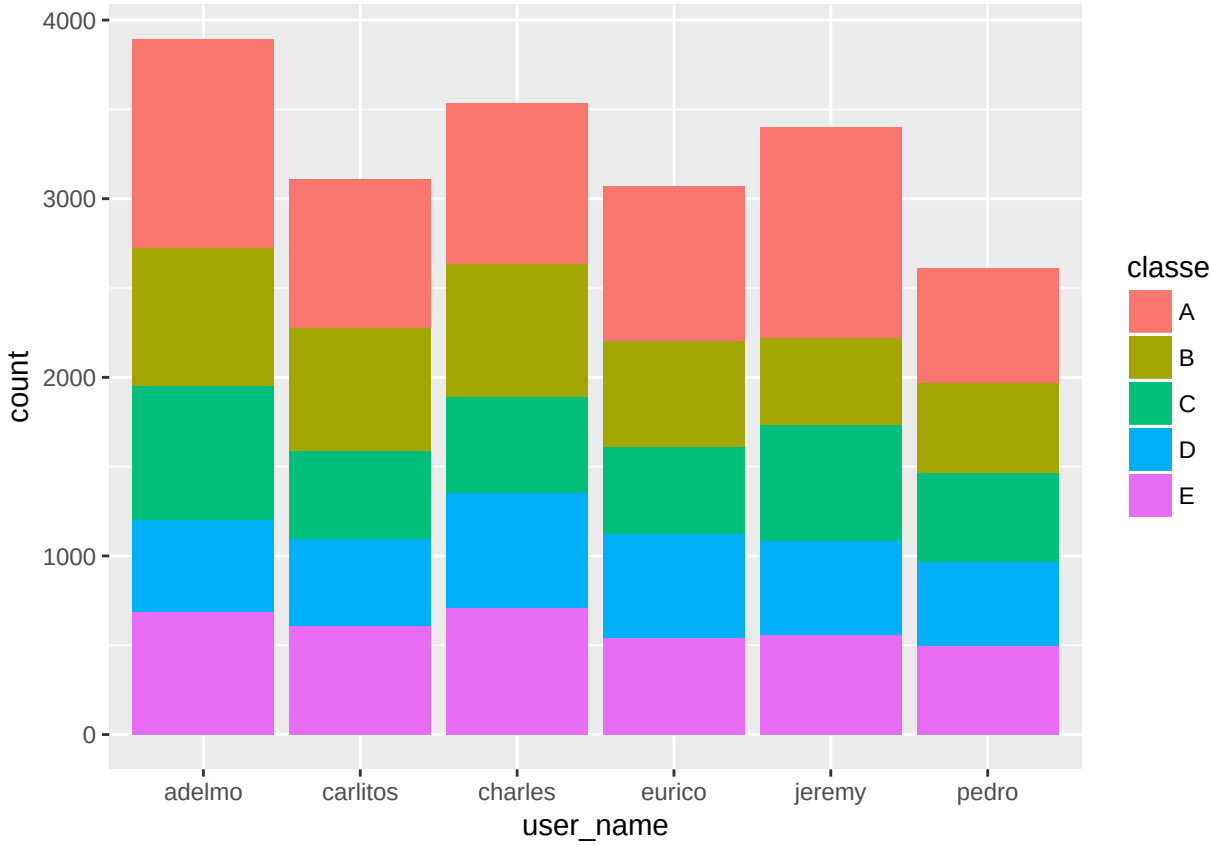
```
# Take a further look at the pairs plot for the forearm measurement
featurePlot(x = pmltrain[, 63:65],
            y = pmltrain$classe,
            plot = "ellipse",
            auto.key = list(columns = 5))
```



Scatter Plot Matrix

From this plot, there are distinct groupings or clusters by classe for belt measurements, and the yaw and the roll may influence the total acceleration and classe. The clustering does not seem as apparent for the forearm.

```
# Create a segmented barplot to view the participants frequency against the classe outcomes
g <- ggplot(pmltrain,aes(user_name))
g + geom_bar(aes(fill=classe))
```



The proportions by username and classe appear similar, therefore the participant does not seem to be a factor.

Part 4: Modeling

For our modeling we will be using the R package caret.

DATA SPLITTING

```
# Use the caret package to split the pmltrain2 into training and testing sets
inTrain <- createDataPartition(pmltrain$classe,p=0.6,list=FALSE)
training <- pmltrain[inTrain,]
testing <- pmltrain[-inTrain,]
```

PREPROCESSING

1. First we will filter out the predictor variables that are not useful.

```
# Since we are only predicting accelerometer readings, remove the gyroscope and magnetometer readings,
# Remove the user_name, raw dates and X as these are identifier variables and do not have predictive va
# we will also remove the mean, variance, standard deviation, max, min, amplitude, kurtosis and skewnes

training <- training %>%
  dplyr::select(-starts_with("user_name")) %>%
  dplyr::select(-starts_with("X")) %>%
  dplyr::select(-starts_with("raw"))  %>%
```

```
  dplyr::select(-starts_with("cvtd")) %>%
  dplyr::select(-starts_with("gyros")) %>%
  dplyr::select(-starts_with("magnet")) %>%
  dplyr::select(-starts_with("mean")) %>%
  dplyr::select(-starts_with("min")) %>%
  dplyr::select(-starts_with("max")) %>%
  dplyr::select(-starts_with("skewness")) %>%
  dplyr::select(-starts_with("kurtosis")) %>%
  dplyr::select(-starts_with("amplitude")) %>%
  dplyr::select(-starts_with("var")) %>%
  dplyr::select(-starts_with("stddev")) %>%
  dplyr::select(-starts_with("avg"))

testing <- testing %>%
  dplyr::select(-starts_with("user_name")) %>%
  dplyr::select(-starts_with("X")) %>%
  dplyr::select(-starts_with("raw"))  %>%
  dplyr::select(-starts_with("cvtd")) %>%
  dplyr::select(-starts_with("gyros")) %>%
  dplyr::select(-starts_with("magnet")) %>%
  dplyr::select(-starts_with("mean")) %>%
  dplyr::select(-starts_with("min")) %>%
  dplyr::select(-starts_with("max")) %>%
  dplyr::select(-starts_with("skewness")) %>%
  dplyr::select(-starts_with("kurtosis")) %>%
  dplyr::select(-starts_with("amplitude")) %>%
  dplyr::select(-starts_with("var")) %>%
  dplyr::select(-starts_with("stddev")) %>%
  dplyr::select(-starts_with("avg"))
```

2. Remove near zero variances

```
# Look for near zero variance covariates to eliminate from dataset
nsv <- nearZeroVar(training,saveMetrics = TRUE)
nsv[nsv$nzv==TRUE,]
```

```
##            freqRatio percentUnique zeroVar  nzv
## new_window  44.46718     0.0169837   FALSE TRUE
```

```
training <- training %>%
  dplyr::select(-starts_with("new_window"))

# Apply to the test set too
testing <- testing %>%
  dplyr::select(-starts_with("new_window"))
```

We have also removed the following variables from the testing: new_window

2. Convert integer variables to numeric, to work in the modeling functions

```
# Convert integer variables to numeric in train set
training$num_window <- as.numeric(training$num_window)
training$total_accel_belt <- as.numeric(training$total_accel_arm)
training$total_accel_arm <- as.numeric(training$total_accel_arm)
training$total_accel_dumbbell <- as.numeric(training$total_accel_arm)
training$total_accel_forearm <- as.numeric(training$total_accel_arm)
```

```
# Convert integer variables to numeric in test set
testing$num_window <- as.numeric(testing$num_window)
testing$total_accel_belt <- as.numeric(testing$total_accel_arm)
testing$total_accel_arm <- as.numeric(testing$total_accel_arm)
testing$total_accel_dumbbell <- as.numeric(testing$total_accel_arm)
testing$total_accel_forearm <- as.numeric(testing$total_accel_arm)
```

3. Find the highly correlated variables

```
# ID highly correlated predictors with cutoff 0.75,leaving out the outcome variable
descrCor <-  abs(cor(training[ ,2:18]))
highlyCorDescr <- findCorrelation(descrCor, cutoff = .75)
training[0,highlyCorDescr]
```

```
## [1] pitch_belt       num_window       accel_belt_y     total_accel_belt
## [5] yaw_belt
## <0 rows> (or 0-length row.names)
```

We may want to consider reducing these highly correlated predictors.

4. PreProcess with caret

```
# preProcess center and scaled, and ignored.
preProc <- preProcess(training,method=c('center','scale'))
# Print out the preprocessed dataframe results
preProc
```

```
## Created from 11776 samples and 30 variables
##
## Pre-processing:
##   - centered (29)
##   - ignored (1)
##   - scaled (29)
```

We may want to consider this preprocessing in our modeling.

Since the remaining factors are numerical other than the outcome variable, classe, we do not need to create dummy variables. Additionally no imputation is required as there are no remaining missing values.

MODEL TRAINING

Since we know the outcome categorical variable, we will use supervised machine learning classification algorithms. It also appears from our EDA that we will need a non-linear model. The algorithms that we will use are classification trees, and random forest which uses bootstrapping.

After filtering out sparse variables there are still 52 input variables to work with. Random forests are particularly well suited to handle a large number of inputs, especially when the interactions between variables are unknown.

Reasons for Random forest: A random forest has a built in cross-validation component that gives an unbiased estimate of the out-of-sample (OOB) error rate. A random forest can be used to estimate variable importance. A random forest can handle unscaled variables and categorical variables, which reduces the need for cleaning and transforming variables which are steps that can be subject to overfitting and noise. Individual trees can be pulled out of the random forest and examined. The random forest classification output can be expressed as a probability which can be used as a confidence estimate for each classification.

We have reduced the number of variables from 159 to 29 through preprocessing. We will also look into further dimension reduction in our modeling.

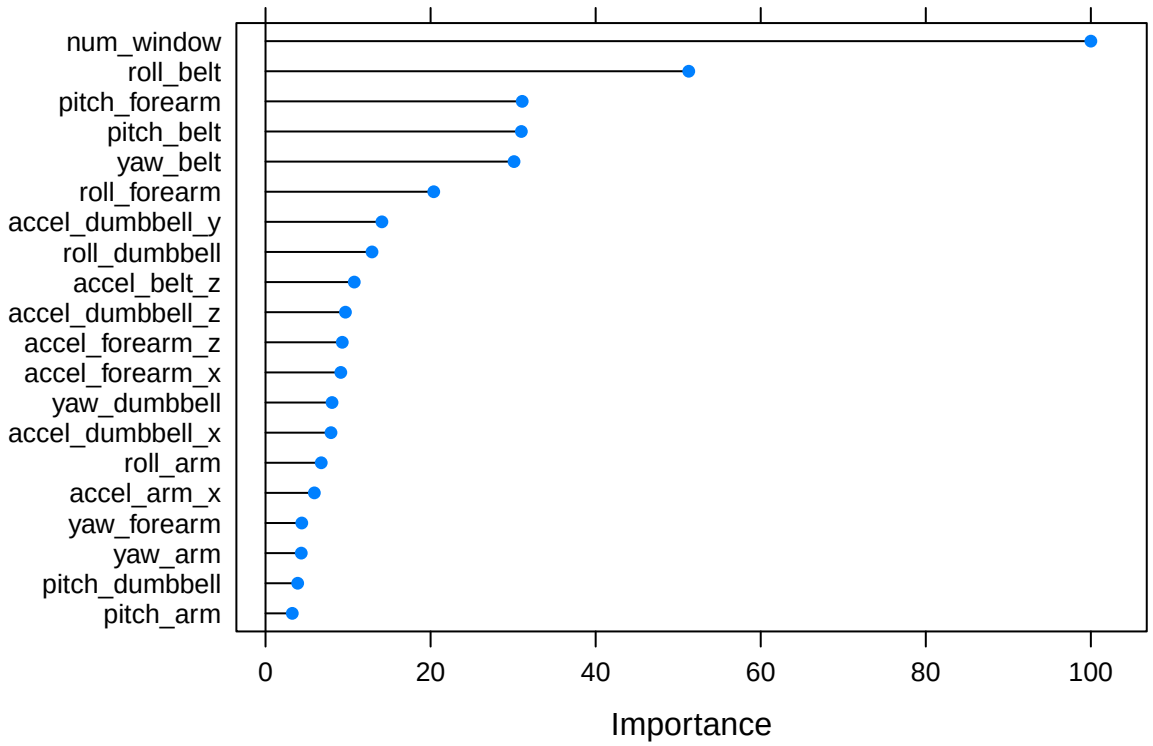
We will select the final model based on the highest accuracy measure.

```
# create a model using tree classification in caret
modtree<- suppressMessages(train(classe~.,training,method="rpart"))
```

The accuracy for the tree classification is 0.3691053

```
# Create a model using random forest in caret
set.seed(123)
# Configure parallel processings as per https://github.com/lgreski/datasciencectacontent/blob/master/ma
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
# create trainControl object for k fold cross validation
control <- trainControl(method = "cv",
                            number = 10,
                            allowParallel = TRUE)
modforest<- train(classe~.,training,method="rf",trControl=control)
# Deregister parallel processing
stopCluster(cluster)
registerDoSEQ()
```

```
varimp <- varImp(modforest)
plot(varimp, main = "Variable Importance of Top 20 variables", top = 20)
```



Part 5: Prediction

Predict the classe variable in the pmltest dataset:

```
predict(modforest,pmltest)
```

```
##  [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
```

Part 6: Cross-Validation and Model Evaluation

To evaluate the model, we can review the accuracy and the out of sample error.

The accuracy for the random forest is 0.9982157

Out of sample error or out of bag (OOB) error is 1-accuracy ie 0.0017843

Part 7: Conclusion

We developed a random forest training model on the created training set from pmltest and ran a prediction on the created testing set using k fold cross validation.

The exploratory data analysis indicated that the belt yaw and roll could be indicators in the model, which we saw. . . .

Based on the calculated accuracy we can conclude that very few test samples will be miss-classified using the random forest model.