# Practical Machine Learning Assignment

**Edmond Low**

**5/14/2019**

# 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, we will be using 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).

The Training and Test data are available for download at the following links:

- Training (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Test (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Our task in this project is to attempt to build a prediction model to classify each barbell lift into the 5 possible categories: 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) and throwing the hips to the front (Class E). This is stored as outcome variable *classe* in the dataset.

# Reading the data

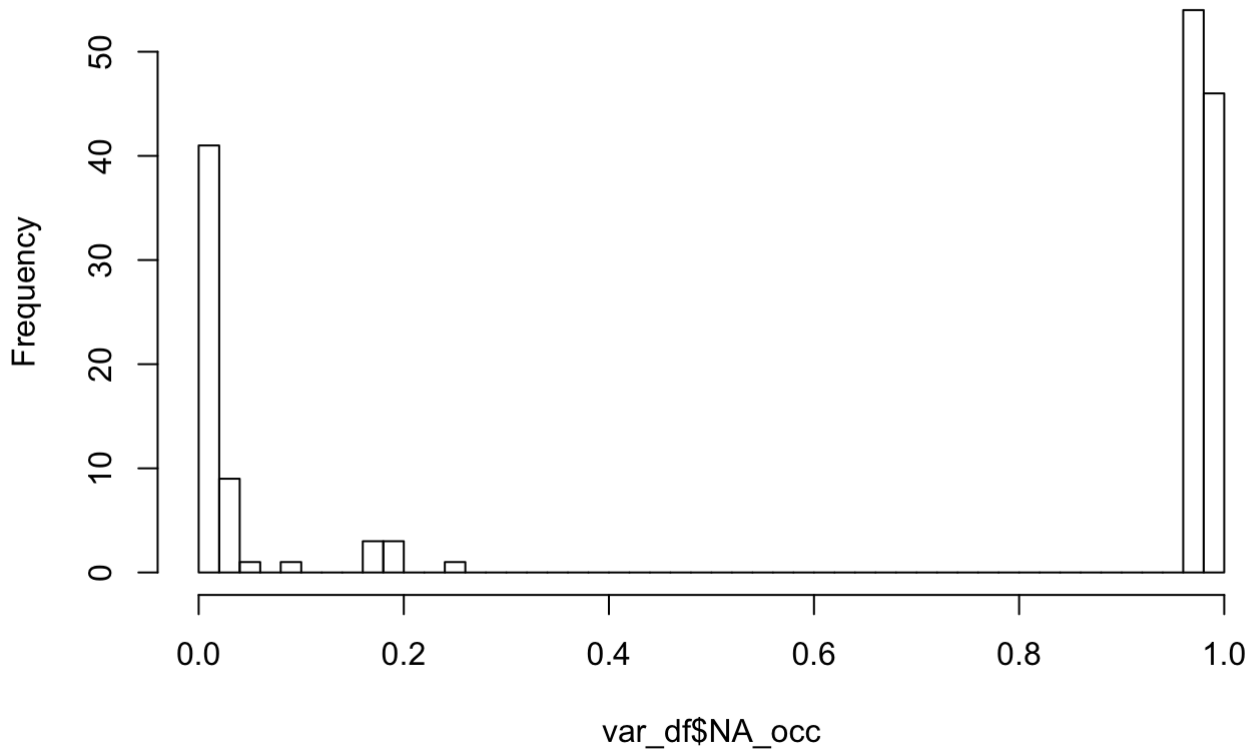
We start off by reading the training data into our R environment.

```
train_data<- read.csv('/Users/edmondlowjy/datasciencecoursera/Practical-Machine-Learn
ing/Week\ 4/Project/pml-training.csv',stringsAsFactors = FALSE); train_data<- train_d
ata[,(-1)]
dim(train_data)
```

```
## [1] 19622   159
```

We notice that there are a large number of predictors (158) in the dataset. A closer look at the data tells us that there are a significant number of predictors with a large proportion of non-meaningful values such as empty strings `''` or empty values `NA`. We try to quantify this for each variable.

```
var_df<- data.frame(var=names(train_data),NA_occ=rep(0,ncol(train_data)),stringsAsFac
tors = FALSE)
total_rows<- nrow(train_data)
for(i in 1:ncol(train_data)){
  var_df[i,2]=(sum(train_data[,i] %in% c(NA,0,''))/total_rows)
}
hist(var_df$NA_occ,breaks=50)
```




From the histogram, we observe that close to 100 variables being >90% comprised of non-meaningful values. Additionally, we also omit variables that we expect to have no meaningful relation to the outcome *classe*, such as the participants' names and timestamp during which the exerise is performed. Here we also transform the outcome variable into a factor.

```
var_remain_manual<- var_df$var[(var_df$NA_occ<=0.95)] #exclude variables with largely
empty or identical readings
var_remain_manual<- var_remain_manual[-(1:6)] #exclude variables that seem to have no
relation to exercise performance e.g. user_name
train_data_small<- train_data[,var_remain_manual]
train_data_small$classe<- as.factor(train_data_small$classe); train_data_small<- na.o
mit(train_data_small)
str(train_data_small)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt            : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45
...
##  $ pitch_belt           : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17
...
##  $ yaw_belt             : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.
4 -94.4 ...
##  $ total_accel_belt     : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x         : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y         : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z         : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.0
2 0 ...
##  $ accel_belt_x         : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y         : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z         : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x        : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y        : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z        : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308
...
##  $ roll_arm             : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128
...
##  $ pitch_arm            : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm              : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161
...
##  $ total_accel_arm      : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x          : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y          : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -
0.03 ...
##  $ gyros_arm_z          : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x          : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288
...
##  $ accel_arm_y          : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z          : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124
...
##  $ magnet_arm_x         : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376
...
##  $ magnet_arm_y         : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z         : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell        : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell       : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell         : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y     : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.0
2 -0.02 ...
##  $ gyros_dumbbell_z     : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x     : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235
...
##  $ accel_dumbbell_y     : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z     : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270
...
##  $ magnet_dumbbell_x    : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558
...
##  $ magnet_dumbbell_y    : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z    : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm         : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm        : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.
```

```
8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152
...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02
...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215
...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1
1 ...
```

Finally in this section we split the Training data into a sub-training dataset `training` and a validation set `validation`.

```
set.seed(123)
intrain<- createDataPartition(train_data_small$classe,p=0.6,list=FALSE)
training<- train_data_small[intrain,]
validation<- train_data_small[-intrain,]
nrow(training); nrow(validation)
```

```
## [1] 11776
```

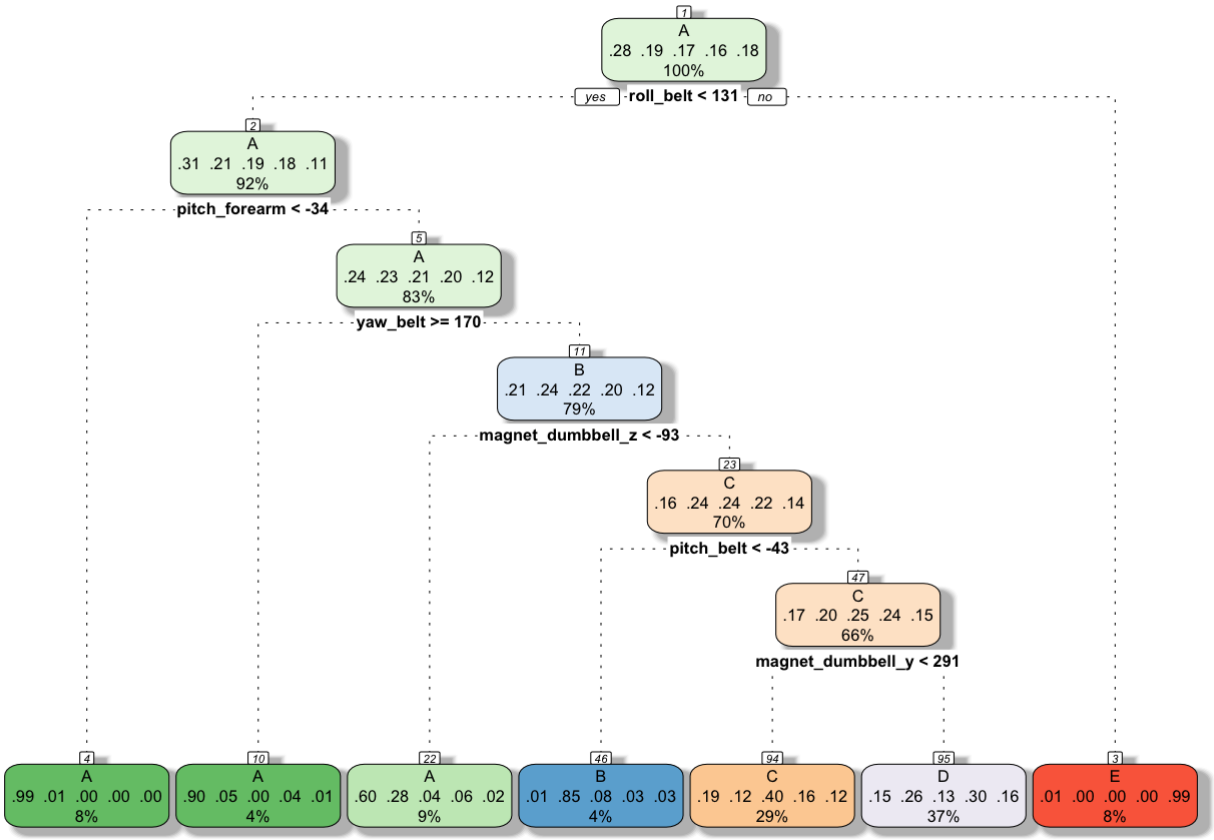
```
## [1] 7846
```

# Training Algorithms

We proceed to use the **caret** package to perform a number of machine learning algorithms on our `training` dataset. For each of the algorithm, we establish the accuracy of the model by cross-validating against our `validation` dataset.

## Predicting with Trees

```
set.seed(234)
modelRPart<- train(classe~.,method='rpart',data=training)
fancyRpartPlot(modelRPart$finalModel)
```




Rattle 2019-May-14 23:51:39 edmondlowjy

```
predictRPart<- predict(modelRPart,newdata=validation)
sum(predictRPart==validation$classe)/nrow(validation)
```

```
## [1] 0.5068825
```

We observe that a prediction model trained using this *rpart* tree approach provides an out-of-sample accuracy of around 50%.

# Predicting with Model-based approach

```
set.seed(345)
modelLDA<- train(classe~.,method='lda',data=training)
predictLDA<- predict(modelLDA,newdata=validation)
sum(predictLDA==validation$classe)/nrow(validation)
```

```
## [1] 0.7055825
```

We observe that a model-based approach for this prediction problem gives us an out-of-sample accuracy of approximately 70%.

# Predicting with Bootstrap Aggregating

```
set.seed(456)
modelBAG<- train(classe~., method='treebag',data=training)
predictBAG<- predict(modelBAG,newdata=validation)
sum(predictBAG==validation$classe)/nrow(validation)
```

```
## [1] 0.980882
```

```
table(pred=predictBAG,actual=validation$classe)
```

```
##     actual
## pred    A    B    C    D    E
##    A 2214   14    4    4    0
##    B   11 1479   18    3    2
##    C    3   19 1331   22   10
##    D    1    3   14 1249    7
##    E    3    3    1    8 1423
```

Finally with a prediction model built using bootstrap aggregating, we observe an astonishing out-of-sample accuracy >90%. Thus, we decide upon this prediction model `modelBAG` for use in predicting the outcome in the Test dataset.

# Prediction on Test Dataset

We first read the Test dataset into our R environment and compare its variables with the original (all variables included) Training dataset.

```
test_data<- read.csv('/Users/edmondlowjy/datasciencecoursera/Practical-Machine-Learning/Week\ 4/Project/pml-testing.csv',stringsAsFactors = FALSE); test_data<- test_data[,(-1)]
setdiff(union(names(train_data),names(test_data)),intersect(names(train_data),names(test_data)))
```

```
## [1] "classe"     "problem_id"
```

We note that the outcome variable *classe* is missing from the Test dataset. Additionally there is an additional variable called *problem_id*. We will need to apply the same predictors removal step to the Test dataset as we have done for the training data.

```
test_data_small<- test_data[,(var_remain_manual[var_remain_manual!='classe'])]
test_data_small<- na.omit(test_data_small)
```

With the reduced dimension in the Test dataset, we proceed to perform outcome prediction using the Bootstrap Aggregating prediction model `modelBAG` previously created.

```
predictOUT<- predict(modelBAG,newdata=test_data_small)
test_data$classe<- predictOUT
data.frame(problem_id=test_data$problem_id,prediction=test_data$classe)
```

```
##    problem_id prediction
## 1           1          B
## 2           2          A
## 3           3          B
## 4           4          A
## 5           5          A
## 6           6          E
## 7           7          D
## 8           8          B
## 9           9          A
## 10         10          A
## 11         11          B
## 12         12          C
## 13         13          B
## 14         14          A
## 15         15          E
## 16         16          E
## 17         17          A
## 18         18          B
## 19         19          B
## 20         20          B
```

```
table(test_data$classe)
```

```
##
## A B C D E
## 7 8 1 1 3
```

The above results represents our final prediction outcomes on the Test dataset.