# Practical Machine Learning Project

*Cody Coleman*

*9/16/2019*

Libraries

```r
suppressWarnings(suppressMessages(library(ggplot2)))
suppressWarnings(suppressMessages(library(caret)))
suppressWarnings(suppressMessages(library(randomForest)))
suppressWarnings(suppressMessages(library(e1071)))
```

## Summary

### 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

### Tasks

- Build a model.
- Use Cross Validation.
- What is the Out of Sample Error?
- Explain why you made your choices.

## Data

### Download

```r
trainingdata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.stri
testdata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

### Cleaning & Viewing

```r
names(trainingdata)
names(testdata)

trainingdata <- trainingdata[, colSums(is.na(trainingdata)) == 0]
testdata <- testdata[, colSums(is.na(testdata)) == 0]
```

```
dim(trainingdata)
dim(testdata)

head(trainingdata)
```

First, we must download the data. Then, clean the data if necessary. In this section we just took out any NA's, empty spaces, or '#DIV/0!'. These values could be seen by looking at the first few observations of each variable. After seeing these things, I used the na.strings function to make all of these values NAs, and then cleaned up the NAs as necessary.

In the next section, we will clean the data up to get it ready to be used in the prediction model. We rid ourselves of the first 7 variables since they have nothing to do with the prediction of the classe variable. d

Get Data Ready for Training and Testing

```
trainingdata <- trainingdata[, -c(1:7)]
testdata <- testdata[, -c(1:7)]

head(trainingdata)
```

## Model

```
trainset <- createDataPartition(trainingdata$classe, p = .6, list = FALSE, times = 1)
Trainset <- trainingdata[trainset, ]
Testset <- trainingdata[-trainset, ]
dim(trainset)
```

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

```
dim(Testset)
```

```
## [1] 7846   53
```

First we must partition the set into a training set and a test set. Let's not get Testset and testdata confused here. The Testset variable is one made specifically for trying out this model. The following is the model. It takes quite a bit of time to run, so it might be better to it in parallel. This is not something that I am comfortable with, so for now, I'll allow it to take its time. The model is built using Cross Validation with Random Forests on 4 folds.

```
trainingModel <- train(classe ~.,
                       data = trainingdata,
                       method = 'rf',
                       metric = 'Accuracy',
                       trControl = trainControl(method = 'cv',
                                                number = 4,
                                                p = .6))

print(trainingModel)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 14717, 14716, 14716, 14717
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9949037  0.9935533
##   27    0.9938335  0.9921992
##   52    0.9880746  0.9849131
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

We find that using the trainingdata, the traningModel is very accurate.

```
PredictionTest <- predict(trainingModel, newdata = Testset)

confusionMatrix(PredictionTest, Testset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    0 1368    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We see here from the confustion matrix that the training model is 100% accurate based on the prediction of the Testset. This implies that there is no Out of Sample Error since the accuracy is 1. This does, at least in my opinion, throw up some red flags. How is this model 100% accurate? Have I made a mistake? Based on a littl digging, I can't seem to find any mistakes, so I can only assume that the model is correct until proven otherwise.

The following is the final model that has a OOB estimate of error rate of .43%. This is extremely low (as it should be based on the accuracy results from above). The number of variables tried at each split is 2, and the top 20 most important variables used in the data are listed below as well.

```
trainingModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.42%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5578    2    0    0    0 0.0003584229
## B   11 3784    2    0    0 0.0034237556
## C    0   19 3401    2    0 0.0061367621
## D    0    0   39 3175    2 0.0127487562
## E    0    0    0    5 3602 0.0013861935
```

```
varImp(trainingModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               91.19
## magnet_dumbbell_z      77.36
## magnet_dumbbell_y      71.04
## pitch_belt             69.86
## pitch_forearm          62.79
## magnet_dumbbell_x      57.12
## roll_forearm           56.76
## accel_belt_z           50.15
## magnet_belt_z          47.86
## accel_dumbbell_y       47.47
## roll_dumbbell          46.84
## magnet_belt_y          43.11
## accel_dumbbell_z       41.15
## roll_arm               39.93
## accel_forearm_x        37.36
## gyros_belt_z           34.84
## accel_dumbbell_x       34.07
## total_accel_dumbbell   32.64
## yaw_dumbbell           32.22
```

This will be the final test using the testdata. It will give an extremely accurate prediction of what movements are made in the classe variable.

```
FinalTest <- predict(trainingModel, newdata = testdata)

print(FinalTest)
```

```
##  [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
```