# Machine Learning Project

*Angus Macdonald*

*12/10/2019*

This project serves as the final project for the Coursera Practical Machine Learning Course.

### Background

This project seeks to predict the manner in which they users of the fitness device did the exercise. Devices such as *Jawbone Up*, *Nike FuelBand*, and *FitBit* are able to collect vast amounts of data about the way in which people perform exercise. Using this data, enthusiasts are trying to predict (originally) how much exercise they did, while now we can predict how well we did the exercise.

## Intro

Using machine learning techniques, this project builds a machine learning model, cross validation and the choices that led to the final prediction. This prediction was done by training a large dataset and then testing these methods on the 20 different test cases.

### The Data

The data used in this proejct comes from Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements"

This includes two datasets, a training set and a testing set.

### Data Processing

The data is loaded and processed to remove any fields that contain no/missing data as well as any other data that is not useful in predicting our model.

```
training_set <- read.csv("Data/pml-training.csv")
validation_set <- read.csv("Data/pml-testing.csv")
```

The following data contains fields that are irrelevant to the model so they are removed.

```
KeepThrow <- colSums(is.na(training_set) |training_set=="")
Observ <- which(KeepThrow == 0)
training_set <- training_set[, Observ]
validation_set <- validation_set[, Observ]
training_set <- training_set[, -c(1:7)]
validation_set <- validation_set[, -c(1:7)]
```

The dataset is almost ready. The datasets now need to be split into training and testing datasets before checking for any fields with zero variance.

Because we want to save our validation set for the very last test we do, we shall split the training set into two separate datasets.

```
set.seed(1903)
inTrain <- createDataPartition(training_set$classe,
                               p = 0.7, list = FALSE)
training <- training_set[inTrain, ]
testing <- training_set[-inTrain, ]
```

The data are now ready to begin using different models.

## Modelling

This section provides the methodology, preforms the different modelling techniques that are used and lastly discusses the results and the final model that best predicts the data. Three different techniques are used that were shown throughout the course, namely: *Classification Trees*, *Random Forests* and *Boosting*.

*Classification Trees* or simply *predicting with trees* is a process where the data are split iteratively into groups where the homogeneity is evaluated within each group. Each observation is then placed into its most commonly occuring class of training observations.

*Ranond Forests* are widely accepted as one of the two top performing algorithms in prediction contests along with boosting (our third model!). *Random Forests* provides an improvement over *Bagged Trees* by decorrelating the trees. We build a number of decision trees on bootstrapped training samples and each time a split occurs we choose *a random sample of m predictors* as split candidates from the full set of predictors.

Normally $m$ is approximately the square root of $p$ predictors which we can also verify.

## Model 1: Classification Trees

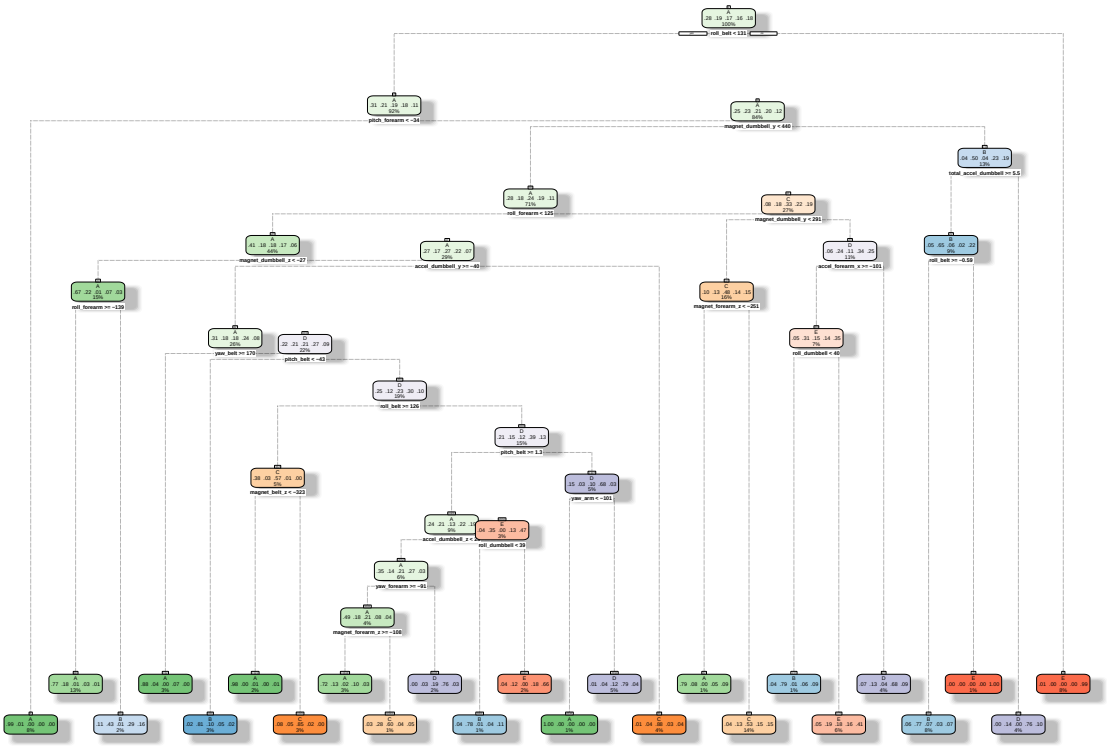
The first model is fit on the training data using the rpart command.

```
modelFit_1 <- rpart(classe ~ ., data = training, method = "class")
modelFit_1
```

The data can be seen, along with the associated splits:

```
fancyRpartPlot(modelFit_1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2019−Oct−22 07:26:13 AngusMacDonald

The model must now be validated against the testing set. This is stored in the variable *pred_model1*. Thereafter the data is inserted into a decision matrix.

```
pred_model1 <- predict(modelFit_1, testing, type = "class")
confmatrix_model1 <- confusionMatrix(pred_model1, testing$classe)
confmatrix_model1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1552  171   18   56   15
##          B   38  648   56   74  100
##          C   46  153  825  140  130
##          D   17   91   62  617   60
##          E   21   76   65   77  777
##
## Overall Statistics
##
##                Accuracy : 0.7509
##                  95% CI : (0.7396, 0.7619)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6841
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9271   0.5689   0.8041   0.6400   0.7181
## Specificity            0.9383   0.9435   0.9035   0.9533   0.9502
## Pos Pred Value         0.8565   0.7074   0.6376   0.7285   0.7648
## Neg Pred Value         0.9700   0.9012   0.9562   0.9311   0.9374
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2637   0.1101   0.1402   0.1048   0.1320
## Detection Prevalence   0.3079   0.1556   0.2199   0.1439   0.1726
## Balanced Accuracy      0.9327   0.7562   0.8538   0.7967   0.8342
```

## Model 2: Random Forests

To predict using *Random Forests* the model is fit using the train function from the caret package.

Normally when performing prediction it is best to set the k folds for cross validation equal to 10. The reason why cross validation with 10 folds is used is that there is evidence that suggests that there is little added benefit form using more than 10 folds. However, with *Random Forests* there is no need to perform cross validation to get unbiased estimates of the test set.

That being said we can run the model.

```
modelFit_2 <- randomForest(classe~., data = training)
```

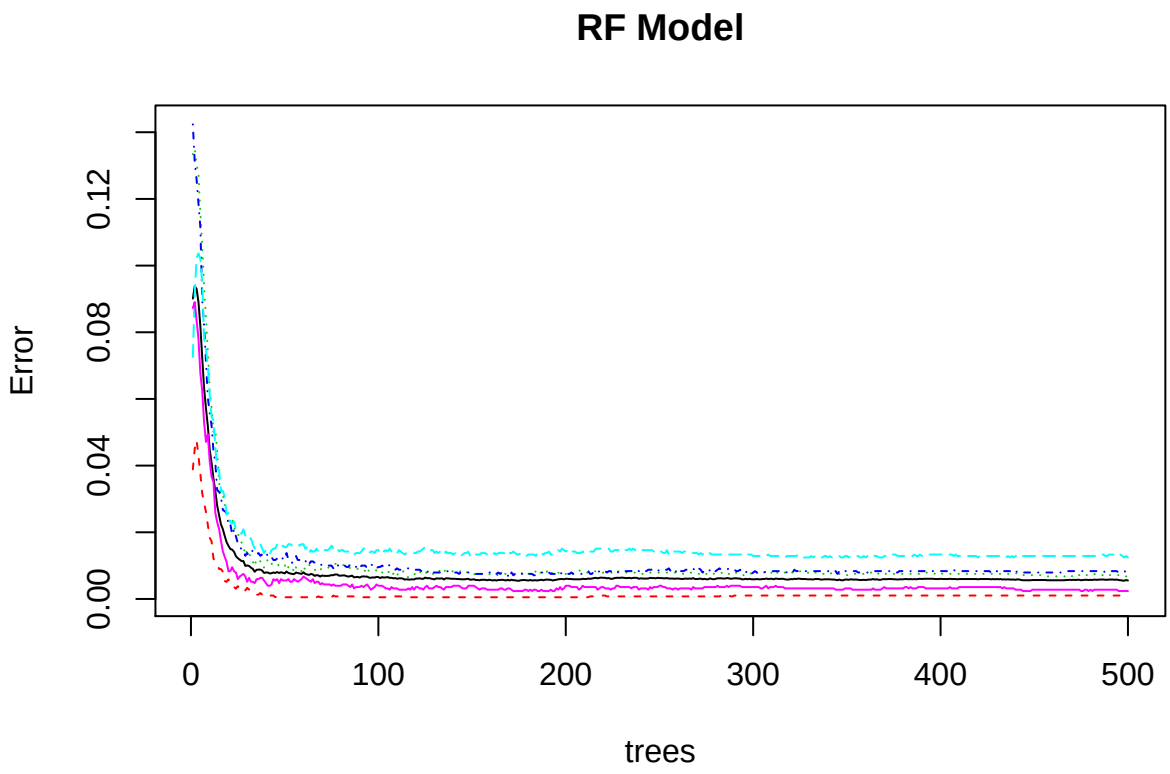
We can then have a look at the results as well as the confusion matrix for the trained model.

```
print(modelFit_2)
confusionMatrix(modelFit_2$y, training$classe)
```

The model is trained on the training data so there is no surprise that the accuracy is 1. Further this gives an in-sample error of 0%.

We can also have a look at the effect that the number of trees have on the error rate.

```
plot(modelFit_2, main = "RF Model")
```




This model selects a final value of *mtry* = 2, and has an accuracy of 0.9917. We now run the prediciton on the test data set and look at the results.

```
predict2 <- predict(modelFit_2, testing)
confusionMatrix(predict2, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1131    6    0    0
##          C    0    2 1019    4    0
##          D    0    0    1  959    2
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9943, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9932   0.9948   0.9982
## Specificity            0.9986   0.9987   0.9988   0.9994   0.9998
## Pos Pred Value         0.9964   0.9947   0.9941   0.9969   0.9991
## Neg Pred Value         1.0000   0.9983   0.9986   0.9990   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1732   0.1630   0.1835
## Detection Prevalence   0.2855   0.1932   0.1742   0.1635   0.1837
## Balanced Accuracy      0.9993   0.9959   0.9960   0.9971   0.9990
```

# Results

This section takes the results of all three models and compares them so that a final model can be selected.

## Accuracy

The first model that was trained was the *Classification Trees* model. This model had an accuracy of aproximately 75%.

The second model, *Random Forests*, performed much better than the first model in terms of accuracy with a value of 99.41%. This is a vast improvement on the *Classification Trees* model.

## Errors

The *Classification Trees* model performed moderately in terms of Sensitivity, Specificity, and Positive and Negative predictive values. However, not all classes of our predictor were consistently predicted. This is in contrast to the *Random Forests* model where all 5 classes were predicted with a high degree of accuracy among all error types.

## In and Out of Sample Errors

The in-sample errors for the *Classification Trees* are #check in sample error

## Model Selection

Unsurprisingly, the *Random Forests* model performs much better than that of the *Classification Trees* model. Thus we will use the *Random Forests* model to validate our model on the validation dataset. As we expected, the out-of-sample error is lower than that of the training data.

```
predict_fin <- predict(modelFit_2, validation_set, type = "class")
print(predict_fin)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```