# Practical Machine Learning

```
Course Project / Peer Assessment
Johns Hopkins Bloomberg School of Public Health (Coursera)
24 July 2015

Michael Fortune
```

## Background

For this project, the requirement is, to analyse data collected by a group of individuals in relation to their personal activity as they attempt to improve their health etc. One issue that arises here, is that while it is easy to quantify how much exercise these individiuals do, but not how well it is done. For this purpose, we will take data from accelerometers on the belt, forearm, arm and dumbbell of 6 participants, who were then asked to perform barbell lifts correctly and incorrectly in 5 different ways.

## Project goal

The goal of the project is therefore to predict the manner in which the exercises were done i.e the "classe" variable in the training set. This report describes how I built my model, how I used cross-validation and what I believe the out of sample error is and why I made the choices that I did. I also used my prediction model to predict 20 test cases that I have submitted separately.

## Packages

A number of packages need to be included for this project:

```
install.packages("randomForest")
install.packages("rpart.plot")

library(caret)
library(randomForest)
library(corrplot)
library(rpart)
library(rpart.plot)
```

## Data loading and preparation

The training and test data was provided as two csv files that were downloaded i.e. pml-training.csv and pml-testing.csv. The first tasks are to load the data and replace all the NULL values in the data with NA.

```
training_data<-read.csv("pml-training.csv",na.strings=c("NA","NaN","#DIV/0!", ""))
testing_data<-read.csv("pml-testing.csv",na.strings=c("NA","NaN","#DIV/0!", ""))

summary(training_data)
summary(testing_data)
```

## Initial processing

At this point, the training data set (training_data) contains 19622 observations and 160 variables. The testing data set (testing_data) contains 20 observations and 160 variables.

We now need to apply some data cleaning to the data. In this case, it is mainly to remove the column containing mostly NA values. THe ID column can also be removed as it is not required.

```
training_data <- training_data[,-1]
testingdata <- testingdata[,-1]
cleaned_training_data <- training_data
cleaned_training_data <- cleaned_training_data[, unlist(lapply(cleaned_training_data, function(x) !any(is.na(x
))))]
cleaned_testing_data <- testing_data
cleaned_testing_data <- cleaned_testing_data[, unlist(lapply(cleaned_testing_data, function(x) !any(is.na(x)))
)]
```

This reduces the number of variables significantlly for both training and testing fdata i.e. 60 as opposed to 160 previously.

## Create a cross-validation partition set

Set seed and partition the data into roughly 2/3 and 1/3.

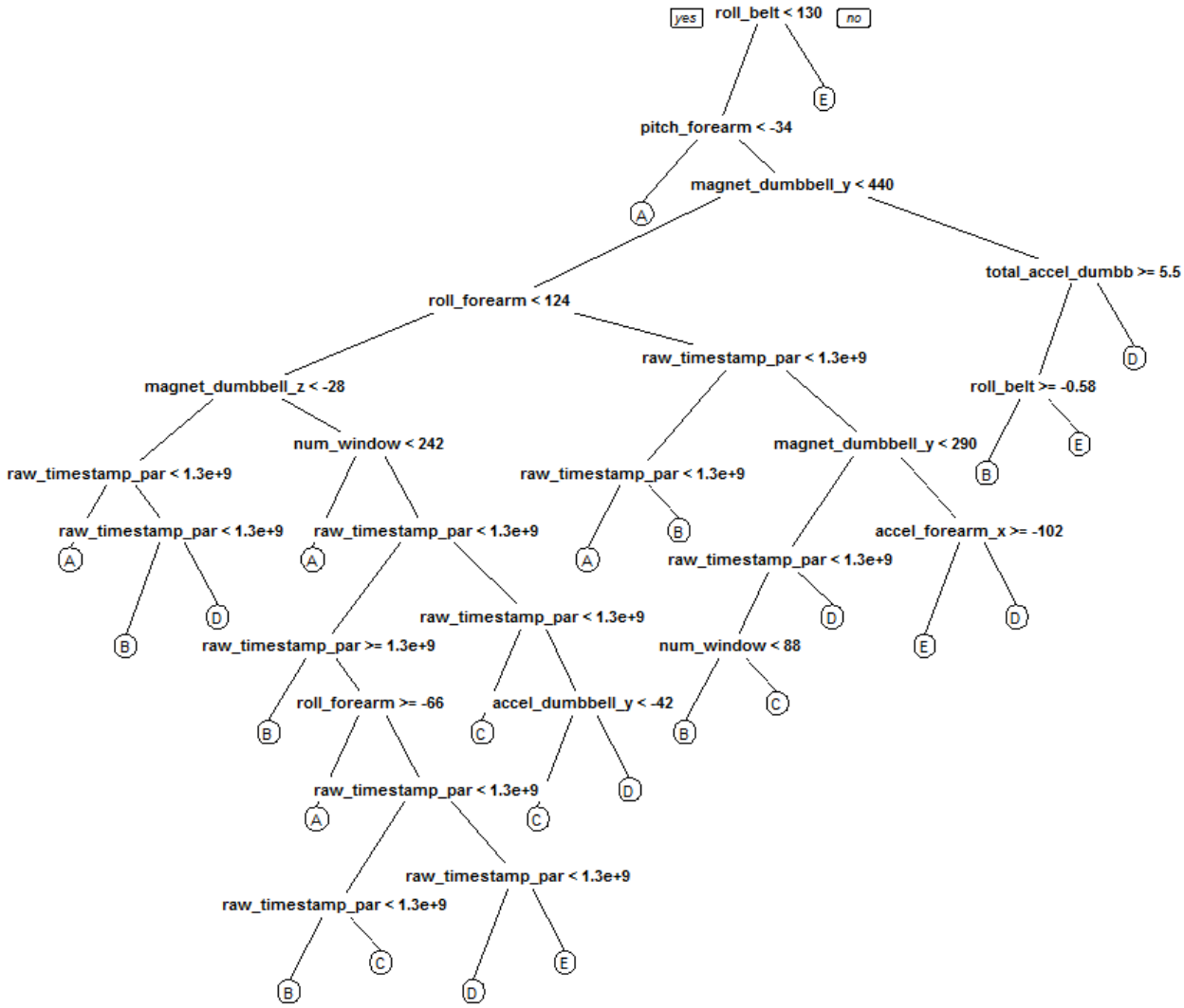
```
set.seed(1508)
training_partition <-createDataPartition(y=cleaned_training_data$classe,p=0.66,list=FALSE)
```

```
train_subset <- cleaned_training_data[training_partition,]
testing_subset <- cleaned_training_data[-training_partition,]
```

## Decision Tree

I first used a Decision Tree to examine the data

```
Decision_Tree_Model<- rpart(classe ~. , data = trainData, method="class")
Decision_Tree_Prediction <- predict(Decision_Tree_Model, testData, type = "class")
rpart.plot(Decision_Tree_Model)
```




## Random Forest Model

I prefer to use Random Forest to create my model as I believe it is more accurate and appropriate for this task.

```
randomForest(classe~., data=train_subset, importance=TRUE, keep.forest=TRUE)
```

```
    Call:
     randomForest(formula = classe ~ ., data = trainData, importance = TRUE,      keep.forest = TRUE)
                   Type of random forest: classification
                         Number of trees: 500
    No. of variables tried at each split: 7

            OOB estimate of  error rate: 0.12%
    Confusion matrix:
         A    B    C    D    E  class.error
    A 4184    1    0    0    0 0.0002389486
    B    2 2846    0    0    0 0.0007022472
    C    0    5 2562    0    0 0.0019477990
```

```
D    0    0    7 2404    1 0.0033167496
E    0    0    0    2 2704 0.0007390983
```

```
rand_forest_control <- trainControl(method="cv", 7)
random_forest_model <- train(classe ~ ., data=train_subset, method="rf", trControl=rand_forest_control, ntree=
250)
```

```
Random Forest

14718 samples
   57 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (7 fold)

Summary of sample sizes: 12614, 12616, 12614, 12617, 12616, 12616, ...

Resampling results across tuning parameters:

  mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
   2    0.9955842  0.9944142  0.0017066907  0.0021591702
  31    0.9993204  0.9991404  0.0007192687  0.0009098263
  61    0.9980978  0.9975942  0.0015761724  0.0019931250

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 31.
```

Applying the model to our test data, we can determine the out of sample error.

# Random Forest Prediction

```
random_forest_prediction <- predict(random_forest_model, test_subset)
confusionMatrix(test_subset$classe, random_forest_prediction)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
        A 1395    0    0    0    0
        B    0  949    0    0    0
        C    0    2  853    0    0
        D    0    0    2  802    0
        E    0    0    0    0  901

Overall Statistics

               Accuracy : 0.9992
                 95% CI : (0.9979, 0.9998)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.999
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            1.0000   0.9979   0.9977   1.0000   1.0000
Specificity            1.0000   1.0000   0.9995   0.9995   1.0000
Pos Pred Value         1.0000   1.0000   0.9977   0.9975   1.0000
Neg Pred Value         1.0000   0.9995   0.9995   1.0000   1.0000
Prevalence             0.2845   0.1939   0.1743   0.1635   0.1837
Detection Rate         0.2845   0.1935   0.1739   0.1635   0.1837
Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
Balanced Accuracy      1.0000   0.9989   0.9986   0.9998   1.0000
```

Accuracy can be calculated as follows:

```
postResample(random_forest_prediction, test_subset$classe)
```

```
 Accuracy     Kappa
```

```
    0.9991843 0.9989683
```

Out of sample error can be calculated as follows:

```
1 - as.numeric(confusionMatrix(test_subset$classe, random_forest_prediction)$overall[1])
```

```
    [1] 0.0008156607
```

It is then a simple task to produce the final output, using the predict function with my model and the test data.

```
predict(random_forest_model, testing_data)
```

This returns the 20 (A,B,C,D,E) values that need to be output individually into files for submission.

# Conclusion - Accuracy, out of sample error, expected out of sample error and cross validation.

I created a model using Random Forest as it produced better results. Using this Model, we achieved an accuracy of 0.999 (99.9%) and an out of sample error of 0.0008. The expected out of sample eror is 1-accuracy in the cross validation data. THe cross validation was done by creating two sample sets of data from the training data in the ratio of 66% to 34% for training ang testing subsets respectively. PLease note that in order to reproduce the results, the seed value should be the same and the packages used e.g. RandomForest will need to be downloaded and installed. The correlation Matrix Visualisation can be seen below.

My Random Forest Model was run on my original test data (20 observations) and these all appear to be correct. I have created the files using the R code supplied and submitted these files separately as required.