# Predicting Exercise Quality = Class

*steve dubois*

*4/1/2019*

### Introduction & 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. 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 to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. .

### Project Mission

The goal of this project is to predict the manner in which the people did the exercises, which is defined in the "classe" variable in the training dataset. The goal is also describing how the prediction model is built, how it is cross validated, evaluation of the expected out of sample error, and explaining the reasons of the choices made to build this model. The prediction model will be used to predict 20 different test cases.

### Sources for Project Data

The training data for this project can be found here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data for this project can be found here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

### Loading requisite R libraries/packages

```
library(randomForest)
library(knitr)
library(rpart)
library(h2o)
library(rattle)
library(caret)
library(gmodels)
library(data.table)
```

### Loading Training_data & Testing_data, & then replace invalid strings as NA

```
training_data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing_data  <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
dim(training_data)
```

```
## [1] 19622   160
```

```
dim(testing_data)
```

```
## [1]  20 160
```

### Processing and Compression of Data

```
# Delete columns with NA in testing & training datasets
training_data <- training_data[, colSums(is.na(training_data)) == 0]
testing_data  <- testing_data[, colSums(is.na(testing_data)) == 0]
dim(training_data)
```

```
## [1] 19622    93
```

```
dim(testing_data)
```

```
## [1] 20 60
```

### Remove variables with low variance

```
nzv            <- nearZeroVar(training_data)
training_data          <- training_data[, -nzv]
testing_data           <- testing_data[, -nzv]
dim(training_data)
```

```
## [1] 19622    59
```

```
dim(testing_data)
```

```
## [1] 20 35
```

### Deleting some non-significant variables: user_name, raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, num_window – mainly first 6 columns

```
training_data  <- training_data[, -c(1:6)]
testing_data   <- testing_data[, -c(1:6)]
dim(training_data)
```

```
## [1] 19622    53
```

```
dim(testing_data)
```

```
## [1] 20 29
```

Compresion of Data reduced to 53 Variables/Predictors

### Cross Validation & Data to Training, Testing Data

Setting seed to preserve reproducibility

```
set.seed(6217)
```

Divide data into a training_data (60%) and testing_data (40%)

```
intrain  <- createDataPartition(y = training_data$classe, p = 0.60, list = FALSE)
training_data2  <- training_data[intrain,]
testing_data2   <- training_data[-intrain,]
dim(training_data2)
```

```
## [1] 11776    53
```

```
dim(testing_data2)
```

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

### Convert dependent ("classe") variable to a factor variable

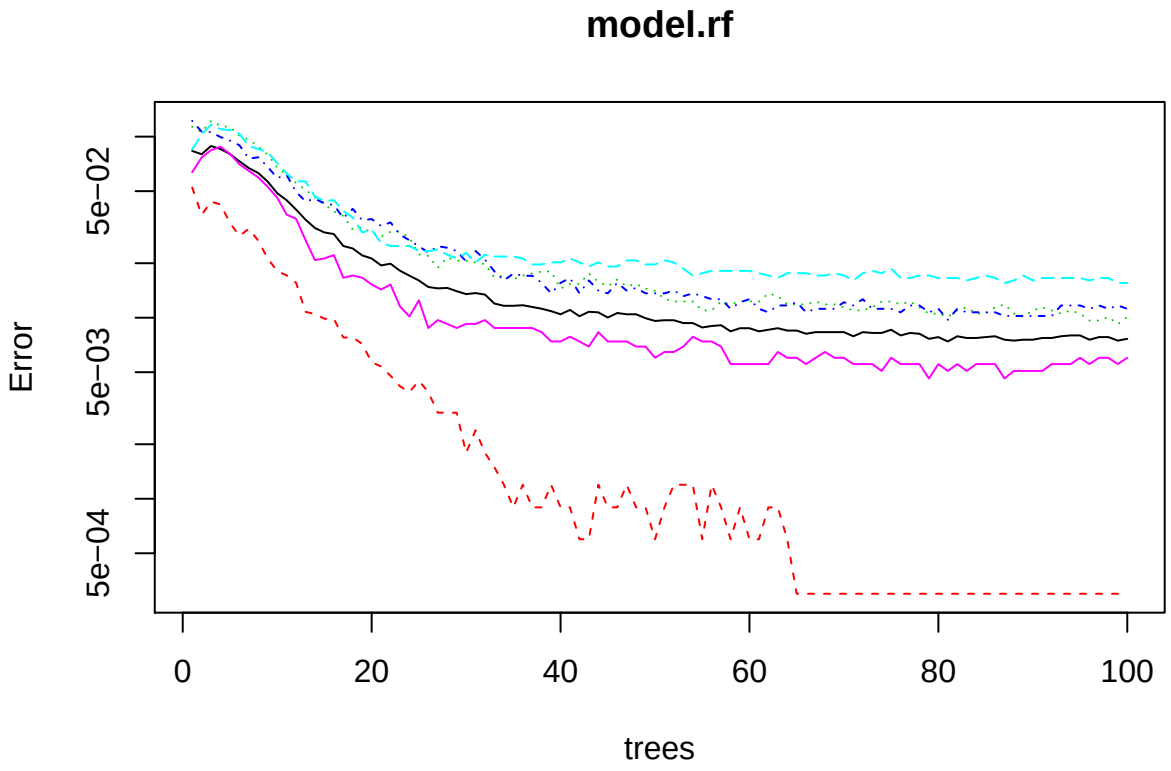
```
training_data2$classe <- as.factor(training_data2$classe)
```

### Random Forest Model

```
model.rf = randomForest(classe~., data = training_data2, ntree=100)
predRF  <- predict(model.rf, testing_data2, type = "response")
confus_rand_forest    <- confusionMatrix(predRF, testing_data2$classe)
confus_rand_forest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   21    0    0    0
##          B    4 1493   17    0    0
##          C    1    4 1346   11    0
##          D    0    0    5 1274    4
##          E    0    0    0    1 1438
##
## Overall Statistics
##
##                Accuracy : 0.9913
##                  95% CI : (0.989, 0.9933)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.989
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9835   0.9839   0.9907   0.9972
## Specificity            0.9963   0.9967   0.9975   0.9986   0.9998
## Pos Pred Value         0.9907   0.9861   0.9883   0.9930   0.9993
## Neg Pred Value         0.9991   0.9961   0.9966   0.9982   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1903   0.1716   0.1624   0.1833
## Detection Prevalence   0.2865   0.1930   0.1736   0.1635   0.1834
## Balanced Accuracy      0.9970   0.9901   0.9907   0.9946   0.9985
```

```
plot(model.rf, log="y")
```




```
tab <- table("PREDICTION" = predRF, "ACTUAL" = testing_data2$classe)
tab
```

```
##            ACTUAL
## PREDICTION    A    B    C    D    E
##          A 2227   21    0    0    0
##          B    4 1493   17    0    0
##          C    1    4 1346   11    0
##          D    0    0    5 1274    4
##          E    0    0    0    1 1438
```

```
CrossTable(predRF, testing_data2$classe)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  7846
##
##
##              | testing_data2$classe
```

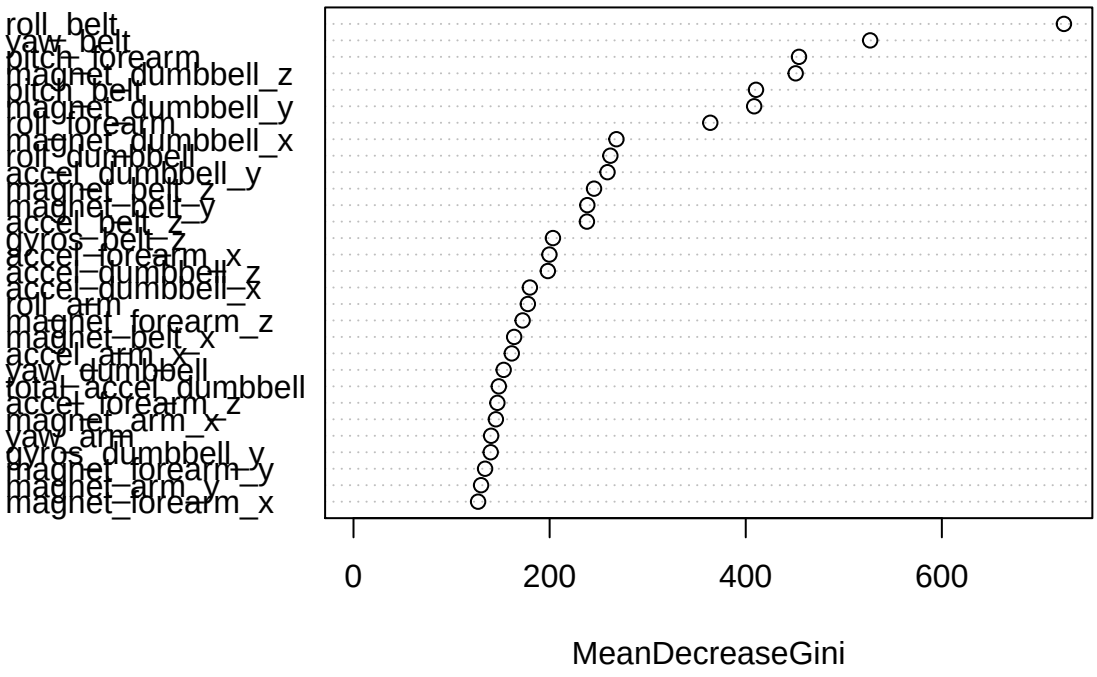
```
##       predRF |         A |         B |         C |         D |         E | Row Total |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            A |      2227 |        21 |         0 |         0 |         0 |      2248 |
##              |  3940.796 |   393.944 |   391.953 |   368.459 |   413.155 |           |
##              |     0.991 |     0.009 |     0.000 |     0.000 |     0.000 |     0.287 |
##              |     0.998 |     0.014 |     0.000 |     0.000 |     0.000 |           |
##              |     0.284 |     0.003 |     0.000 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            B |         4 |      1493 |        17 |         0 |         0 |      1514 |
##              |   422.734 |  4916.668 |   231.070 |   248.152 |   278.255 |           |
##              |     0.003 |     0.986 |     0.011 |     0.000 |     0.000 |     0.193 |
##              |     0.002 |     0.984 |     0.012 |     0.000 |     0.000 |           |
##              |     0.001 |     0.190 |     0.002 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            C |         1 |         4 |      1346 |        11 |         0 |      1362 |
##              |   385.459 |   255.573 |  5174.607 |   201.781 |   250.319 |           |
##              |     0.001 |     0.003 |     0.988 |     0.008 |     0.000 |     0.174 |
##              |     0.000 |     0.003 |     0.984 |     0.009 |     0.000 |           |
##              |     0.000 |     0.001 |     0.172 |     0.001 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            D |         0 |         0 |         5 |      1274 |         4 |      1283 |
##              |   364.983 |   248.228 |   213.811 |  5380.553 |   227.868 |           |
##              |     0.000 |     0.000 |     0.004 |     0.993 |     0.003 |     0.164 |
##              |     0.000 |     0.000 |     0.004 |     0.991 |     0.003 |           |
##              |     0.000 |     0.000 |     0.001 |     0.162 |     0.001 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            E |         0 |         0 |         0 |         1 |      1438 |      1439 |
##              |   409.361 |   278.410 |   250.899 |   233.864 |  5207.269 |           |
##              |     0.000 |     0.000 |     0.000 |     0.001 |     0.999 |     0.183 |
##              |     0.000 |     0.000 |     0.000 |     0.001 |     0.997 |           |
##              |     0.000 |     0.000 |     0.000 |     0.000 |     0.183 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
## Column Total |      2232 |      1518 |      1368 |      1286 |      1442 |      7846 |
##              |     0.284 |     0.193 |     0.174 |     0.164 |     0.184 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##
##
```

```r
randomForest::importance(model.rf, type = 1)
```

```
##
## roll_belt
## pitch_belt
## yaw_belt
## total_accel_belt
## gyros_belt_x
## gyros_belt_y
## gyros_belt_z
## accel_belt_x
## accel_belt_y
## accel_belt_z
## magnet_belt_x
## magnet_belt_y
## magnet_belt_z
## roll_arm
```

```
## pitch_arm
## yaw_arm
## total_accel_arm
## gyros_arm_x
## gyros_arm_y
## gyros_arm_z
## accel_arm_x
## accel_arm_y
## accel_arm_z
## magnet_arm_x
## magnet_arm_y
## magnet_arm_z
## roll_dumbbell
## pitch_dumbbell
## yaw_dumbbell
## total_accel_dumbbell
## gyros_dumbbell_x
## gyros_dumbbell_y
## gyros_dumbbell_z
## accel_dumbbell_x
## accel_dumbbell_y
## accel_dumbbell_z
## magnet_dumbbell_x
## magnet_dumbbell_y
## magnet_dumbbell_z
## roll_forearm
## pitch_forearm
## yaw_forearm
## total_accel_forearm
## gyros_forearm_x
## gyros_forearm_y
## gyros_forearm_z
## accel_forearm_x
## accel_forearm_y
## accel_forearm_z
## magnet_forearm_x
## magnet_forearm_y
## magnet_forearm_z
```

```
impPlot <- varImpPlot(model.rf)
```




```
library(DT)
datatable(tab)
```

Show 10 entries Search:

| | PREDICTION | ACTUAL | Freq |
|---|---|---|---|
| A | A | A | 2227 |
| B | B | A | 4 |
| C | C | A | 1 |
| D | D | A | 0 |
| E | E | A | 0 |
| A | A | B | 21 |
| B | B | B | 1493 |
| C | C | B | 4 |
| D | D | B | 0 |
| E | E | B | 0 |

Showing 1 to 10 of 25 entries Previous 1 2 3 Next

**Key Observations:**

From the confusion matrix, one can see that, in implementing a Random Forest model/algorythm, the number of misclassified predictions is relatively low. Additionally, the predictive accuracy of the model is 99.15%, which is very good. On the flip side, the out of sample error is 100% minus 99.15% which equals 0.85%.

### Decision-Classification Tree Model

```
control <- trainControl(method = "cv", number = 10)
Dtree_model <- rpart(classe ~., training_data2)

pred_Dtree   <- predict(Dtree_model, testing_data2, type = "class")
summary(pred_Dtree)
```

```
##    A    B    C    D    E
## 2676 1146 1736 1003 1285
```

```
confus_Dtree <- confusionMatrix(pred_Dtree, testing_data2$classe)

confus_Dtree
```

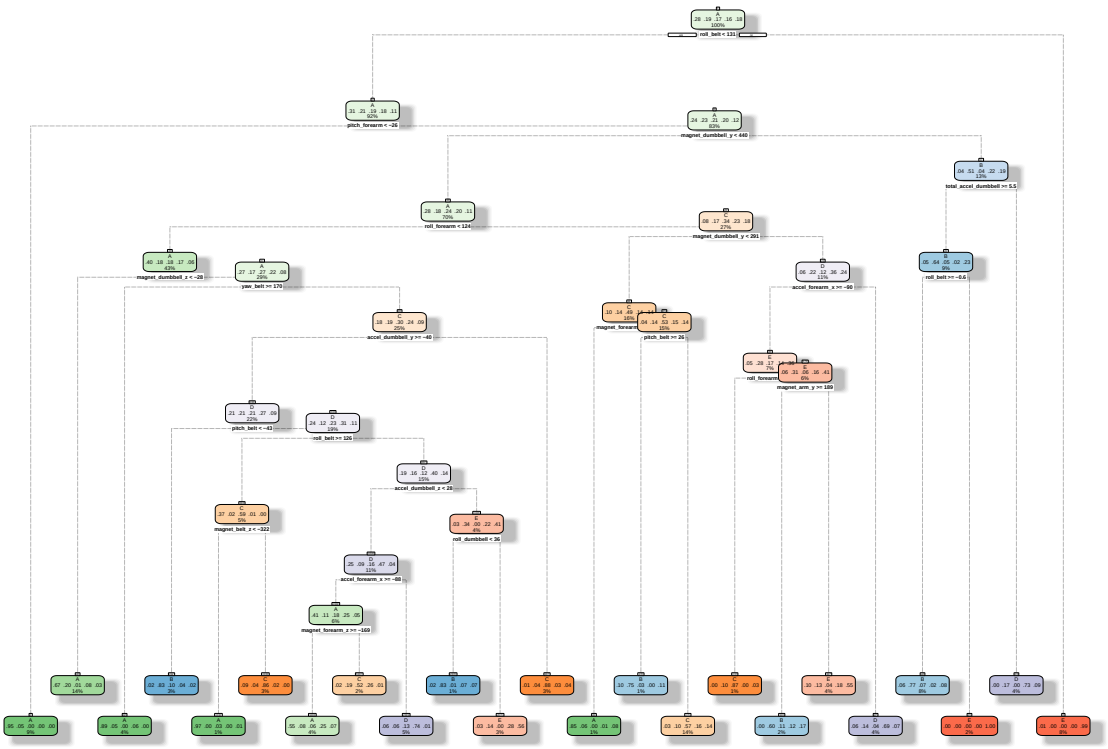
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2040  354   40  177   65
##          B   42  851   97   56  100
##          C   59  142 1121  236  178
##          D   53  108   98  672   72
##          E   38   63   12  145 1027
##
## Overall Statistics
##
##                Accuracy : 0.7279
##                  95% CI : (0.7179, 0.7377)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6534
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.5606   0.8194  0.52255   0.7122
## Specificity            0.8867   0.9534   0.9051  0.94954   0.9597
## Pos Pred Value         0.7623   0.7426   0.6457  0.66999   0.7992
## Neg Pred Value         0.9629   0.9004   0.9596  0.91027   0.9367
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2600   0.1085   0.1429  0.08565   0.1309
## Detection Prevalence   0.3411   0.1461   0.2213  0.12784   0.1638
## Balanced Accuracy      0.9003   0.7570   0.8623  0.73605   0.8360
```

```
CrossTable(pred_Dtree, testing_data2$classe)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
```

```
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  7846
##
##
##              | testing_data2$classe
##   pred_Dtree |         A |         B |         C |         D |         E | Row Total |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            A |      2040 |       354 |        40 |       177 |        65 |      2676 |
##              |  2147.997 |    51.783 |   390.007 |   156.038 |   370.407 |           |
##              |     0.762 |     0.132 |     0.015 |     0.066 |     0.024 |     0.341 |
##              |     0.914 |     0.233 |     0.029 |     0.138 |     0.045 |           |
##              |     0.260 |     0.045 |     0.005 |     0.023 |     0.008 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            B |        42 |       851 |        97 |        56 |       100 |      1146 |
##              |   247.421 |  1785.984 |    52.902 |    92.531 |    58.100 |           |
##              |     0.037 |     0.743 |     0.085 |     0.049 |     0.087 |     0.146 |
##              |     0.019 |     0.561 |     0.071 |     0.044 |     0.069 |           |
##              |     0.005 |     0.108 |     0.012 |     0.007 |     0.013 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            C |        59 |       142 |      1121 |       236 |       178 |      1736 |
##              |   382.899 |   111.906 |  2212.361 |     8.280 |    62.361 |           |
##              |     0.034 |     0.082 |     0.646 |     0.136 |     0.103 |     0.221 |
##              |     0.026 |     0.094 |     0.819 |     0.184 |     0.123 |           |
##              |     0.008 |     0.018 |     0.143 |     0.030 |     0.023 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            D |        53 |       108 |        98 |       672 |        72 |      1003 |
##              |   189.174 |    38.162 |    33.797 |  1567.310 |    68.461 |           |
##              |     0.053 |     0.108 |     0.098 |     0.670 |     0.072 |     0.128 |
##              |     0.024 |     0.071 |     0.072 |     0.523 |     0.050 |           |
##              |     0.007 |     0.014 |     0.012 |     0.086 |     0.009 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##            E |        38 |        63 |        12 |       145 |      1027 |      1285 |
##              |   293.502 |   138.579 |   200.691 |    20.443 |  2648.189 |           |
##              |     0.030 |     0.049 |     0.009 |     0.113 |     0.799 |     0.164 |
##              |     0.017 |     0.042 |     0.009 |     0.113 |     0.712 |           |
##              |     0.005 |     0.008 |     0.002 |     0.018 |     0.131 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
## Column Total |      2232 |      1518 |      1368 |      1286 |      1442 |      7846 |
##              |     0.284 |     0.193 |     0.174 |     0.164 |     0.184 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|
##
##
```

```
fancyRpartPlot(Dtree_model, caption = NULL)
```



```
library(rpart.plot)
```

**Key Observations:**

From the confusion matrix, one can see that, in the implementation of a Decision-Classification model/algorythm, the number of misclassified predictions is quite high. Additionally, the predictive accuracy of the model is just 49.26%, which is significantly lower than the Random Forest model. On the flip side, the out of sample error is 100% minus 49.26% which equals 50.74%, which is quite high and indicative of a poorly performing model.

h3 { font-size: 38px; color: DarkRed; text-align: center; }

**Compare models <- choosing model with the best accuracy (random forest)**

```
compare <- data.frame(confus_rand_forest$overall, confus_Dtree$overall)
data.table(compare)
```

```
##    confus_rand_forest.overall confus_Dtree.overall
## 1:                  0.9913332         7.278868e-01
## 2:                  0.9890346         6.533963e-01
## 3:                  0.9890255         7.178920e-01
## 4:                  0.9932638         7.377121e-01
## 5:                  0.2844762         2.844762e-01
## 6:                  0.0000000         0.000000e+00
## 7:                        NaN        6.458906e-119
```

Again whith an accuracy rate of approximatly 99.%, the random forest model outperforms the decision tree model (69% accuracy). Also, the out of sample error is 100% minus the accuracy rate (very small). Great, but this may be a sign of overfitting.

h3 { font-size: 38px; color: DarkRed; text-align: center; }

**Application of best model ("ie., random forest") to predict outcomes of 20 observations in the testing_data set.**

**Source:**

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).

**Credits:**

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.