Assessment of the efficacy of Exercise Activity

Loading the required Libraries

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: lattice
## Loading required package: ggplot2
```

Executive Summary The exercise activity data provided was analyzed to find out whether it was possible to predict how well the exercise was performed. Using the randomforest algorithm yielded a 99% accuracy in predicting how well the exercises was performed

Introduction Devices such as Jawbone Up, Nike FuelBand, and Fitbit have been utilized by tech geek health enthusiasts to collect exercise data in an effort to improve their health. While the amount of exercise is often collected, how well the exercise is done is seldom quantified. The current data tracks 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.

Data Description The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Trainging data came from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data came from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data consisted of 160 fields consisting of actual mesurement data together with derived data such as standard deviation, max, min and variances. The data was collected from from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

Study Design *The model was generated as follows:*

The pml-training data was split 60:40 into a training set and a testing set. The training file was used to trian the model using the random forest algorithm. The tesing file was used to test the model generated by the random forest algorithm. The random forest algorithm was chosen for the following reasons (modified from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#overview)

1. Compared to other algorithms,it has high accuracy
2. Runs efficiently on large data sets
3. It can handle large numbers of of input variables

Cross validation was performed as follows:

- Use the entire training set
- Split it into training/test sets
- Build a model on the training set
- Evaluate on the test set

The project was done in the following phases:

1. Feature selection
2. Build the model using the training data
3. Running the model against the testing set
4. Run the model against the 20 data cases
5. Evaluate the model statistics

1.0 Feature Selection Features were selected using the following process

1. Removing the first eight columns since they contain pesonal identification data.
2. Remove all calculated fields such as standard deviation since they are derived from the existing data
3. Remove all columns that mostly contain NAs

The above process resulted in the selection of 36 features

```
tr <- read.csv("pml-training.csv", na.strings=c("#DIV/0!","NA") )
tr <- tr[,c(9:160)]
r <- grep('kur*|std*|skew*|max*|min*|var*|avg*|tot*|user|new|X|num',colnames(tr))
tr <- tr[-r]
nacols <- c((colSums(!is.na(tr[,-ncol(tr)])) >= 0.6*nrow(tr)))
tr <- tr[,nacols]
tr$classe <- factor(tr$classe)
```

2. Build the model using the training data

The following results were obtained when the training data set was used to generate the model

```
set.seed(157)
trainIndex <- createDataPartition(tr$classe, p=0.60, list=FALSE)
training<- tr[ trainIndex,]
testing<- tr[-trainIndex,]
model <- randomForest(classe~.,data=training)
testclass <- predict(model, testing)
cfMatrix <- confusionMatrix(testclass, testing$classe)
model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3341    4    2    0    1 0.002090800
## B   16 2254    5    0    4 0.010969724
## C    1   14 2032    6    1 0.010710808
## D    0    1   14 1912    3 0.009326425
## E    1    0    2    7 2155 0.004618938
```

3. Running the model against the testing set Running the model aginst the the testing set yielded the following statistics for the confusion matrix

```
cfMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2220   16    0    0    0
##          B    9 1495    7    0    0
##          C    1    7 1357   14    2
##          D    2    0    4 1272    8
##          E    0    0    0    0 1432
##
## Overall Statistics
##
##                Accuracy : 0.9911
##                  95% CI : (0.9887, 0.993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9887
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9848   0.9920   0.9891   0.9931
## Specificity            0.9971   0.9975   0.9963   0.9979   1.0000
## Pos Pred Value         0.9928   0.9894   0.9826   0.9891   1.0000
## Neg Pred Value         0.9979   0.9964   0.9983   0.9979   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2829   0.1905   0.1730   0.1621   0.1825
## Detection Prevalence   0.2850   0.1926   0.1760   0.1639   0.1825
## Balanced Accuracy      0.9959   0.9912   0.9941   0.9935   0.9965
```

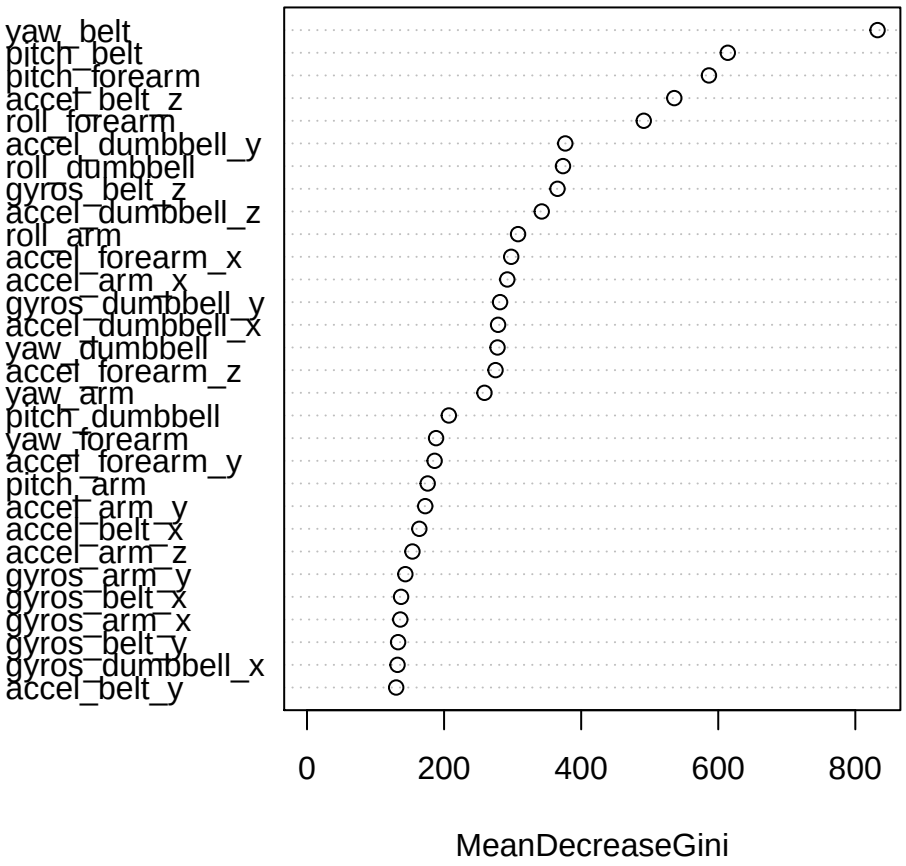
4. Run the model against the 20 data cases

The ml-testing.csv file contained 20 test cases. The file was subjected to the same data cleaning techniques as was used with the pml-training.csv file that was used to generate the model

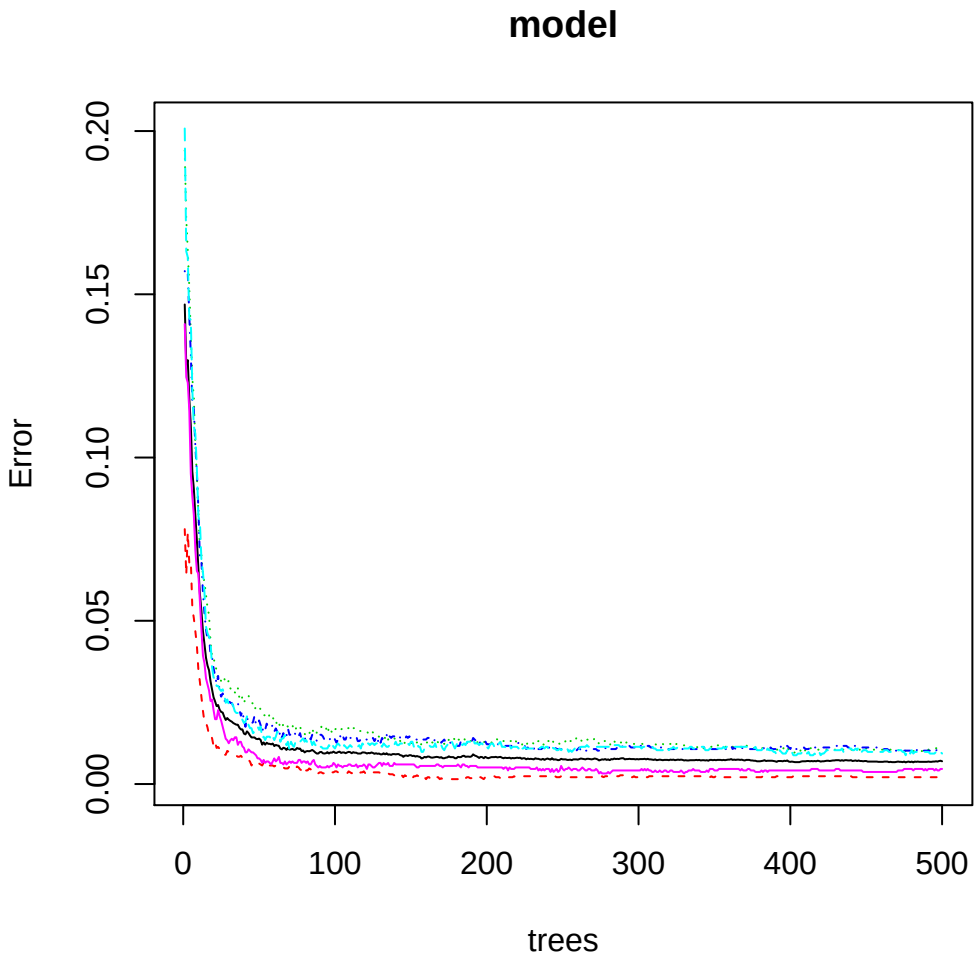
```
ev <- read.csv("pml-testing.csv", na.strings=c("#DIV/0!") )
ev <- ev[,c(9:160)]
ev <- ev[-r]
ev <- ev[,nacols]
testclass1 <- predict(model, ev)
testclass1
```

```
##  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
```

5. Evaluate the model statistics



The matrix indicates how important that variable is in classifying the data. The plot shows the importance of each variable with the most importanct variable at the top of the list.



The plot above traces the error rates (out-of-bag, and by each response category) as the number of trees increases.

Conclusion The goal of the study was to findout if was possible predict how well specific exercises were performed.Using the random forest algorithm, it was possible to predict how well the exercise regimen was performed with a high degree of accuracy.