Assessment of the eficacy 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 wa sperformed

Introduction

Data Description

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)
```

How model was built 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

Running the model against the tesing file The folloing results were obtained when the trained model was run against the testing file

```
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)
cfMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2224   10    0    1    0
##          B    2 1505    7    0    1
##          C    4    3 1353   30    2
##          D    2    0    6 1254    2
##          E    0    0    2    1 1437
##
## Overall Statistics
##
##                Accuracy : 0.9907
##                  95% CI : (0.9883, 0.9927)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9882
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9914   0.9890   0.9751   0.9965
## Specificity            0.9980   0.9984   0.9940   0.9985   0.9995
## Pos Pred Value         0.9951   0.9934   0.9720   0.9921   0.9979
## Neg Pred Value         0.9986   0.9979   0.9977   0.9951   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2835   0.1918   0.1724   0.1598   0.1832
## Detection Prevalence   0.2849   0.1931   0.1774   0.1611   0.1835
## Balanced Accuracy      0.9972   0.9949   0.9915   0.9868   0.9980
```

Test 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
```