# Practical Machine Learning - Prediction Assignment Writeup

In order to determine what activity an individual perform, I made use of caret and randomForest, this allowed me to generate correct answers for each of the 20 test data cases provided in this assignment. I made use of a seed value for consistent results.

library(Hmisc)

## Warning: package 'Hmisc' was built under R version 3.1.3

## Loading required package: grid  
## Loading required package: lattice  
## Loading required package: survival  
## Loading required package: splines  
## Loading required package: Formula

## Warning: package 'Formula' was built under R version 3.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.1.3

##   
## Attaching package: 'Hmisc'  
##   
## The following objects are masked from 'package:base':  
##   
## format.pval, round.POSIXt, trunc.POSIXt, units

library(caret)

## Warning: package 'caret' was built under R version 3.1.3

##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:survival':  
##   
## cluster

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:Hmisc':  
##   
## combine

library(foreach)  
library(doParallel)

## Warning: package 'doParallel' was built under R version 3.1.3

## Loading required package: iterators  
## Loading required package: parallel

set.seed(2048)  
options(warn=-1)

I loaded the data both from the provided training and test data provided by COURSERA.

training\_data <- read.csv("pml-training.csv", na.strings=c("#DIV/0!") )  
evaluation\_data <- read.csv("pml-testing.csv", na.strings=c("#DIV/0!") )

I casted all columns 8 to the end to be numeric.

for(i in c(8:ncol(training\_data)-1)) {training\_data[,i] = as.numeric(as.character(training\_data[,i]))}  
  
for(i in c(8:ncol(evaluation\_data)-1)) {evaluation\_data[,i] = as.numeric(as.character(evaluation\_data[,i]))}

I chose a feature set that only included complete columns. We also remove user name, timestamps and windows.

Determine and display out feature set.

feature\_set <- colnames(training\_data[colSums(is.na(training\_data)) == 0])[-(1:7)]  
model\_data <- training\_data[feature\_set]  
feature\_set

## [1] "roll\_belt" "pitch\_belt" "yaw\_belt"   
## [4] "total\_accel\_belt" "gyros\_belt\_x" "gyros\_belt\_y"   
## [7] "gyros\_belt\_z" "accel\_belt\_x" "accel\_belt\_y"   
## [10] "accel\_belt\_z" "magnet\_belt\_x" "magnet\_belt\_y"   
## [13] "magnet\_belt\_z" "roll\_arm" "pitch\_arm"   
## [16] "yaw\_arm" "total\_accel\_arm" "gyros\_arm\_x"   
## [19] "gyros\_arm\_y" "gyros\_arm\_z" "accel\_arm\_x"   
## [22] "accel\_arm\_y" "accel\_arm\_z" "magnet\_arm\_x"   
## [25] "magnet\_arm\_y" "magnet\_arm\_z" "roll\_dumbbell"   
## [28] "pitch\_dumbbell" "yaw\_dumbbell" "total\_accel\_dumbbell"  
## [31] "gyros\_dumbbell\_x" "gyros\_dumbbell\_y" "gyros\_dumbbell\_z"   
## [34] "accel\_dumbbell\_x" "accel\_dumbbell\_y" "accel\_dumbbell\_z"   
## [37] "magnet\_dumbbell\_x" "magnet\_dumbbell\_y" "magnet\_dumbbell\_z"   
## [40] "roll\_forearm" "pitch\_forearm" "yaw\_forearm"   
## [43] "total\_accel\_forearm" "gyros\_forearm\_x" "gyros\_forearm\_y"   
## [46] "gyros\_forearm\_z" "accel\_forearm\_x" "accel\_forearm\_y"   
## [49] "accel\_forearm\_z" "magnet\_forearm\_x" "magnet\_forearm\_y"   
## [52] "magnet\_forearm\_z" "classe"

I have the model data built from our feature set.

idx <- createDataPartition(y=model\_data$classe, p=0.75, list=FALSE )  
training <- model\_data[idx,]  
testing <- model\_data[-idx,]

I build 5 random forests with 150 trees each. We make use of parallel processing to build this model. I found several examples of how to perform parallel processing with random forests in R, this provided a great speedup.

registerDoParallel()  
x <- training[-ncol(training)]  
y <- training$classe  
  
rf <- foreach(ntree=rep(150, 6), .combine=randomForest::combine, .packages='randomForest') %dopar% {  
 randomForest(x, y, ntree=ntree)   
}

Provide error reports for both training and test data.

predictions1 <- predict(rf, newdata=training)  
confusionMatrix(predictions1,training$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 4185 0 0 0 0  
## B 0 2848 0 0 0  
## C 0 0 2567 0 0  
## D 0 0 0 2412 0  
## E 0 0 0 0 2706  
##   
## Overall Statistics  
##   
## Accuracy : 1   
## 95% CI : (0.9997, 1)  
## No Information Rate : 0.2843   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 1.0000 1.0000 1.0000 1.0000  
## Specificity 1.0000 1.0000 1.0000 1.0000 1.0000  
## Pos Pred Value 1.0000 1.0000 1.0000 1.0000 1.0000  
## Neg Pred Value 1.0000 1.0000 1.0000 1.0000 1.0000  
## Prevalence 0.2843 0.1935 0.1744 0.1639 0.1839  
## Detection Rate 0.2843 0.1935 0.1744 0.1639 0.1839  
## Detection Prevalence 0.2843 0.1935 0.1744 0.1639 0.1839  
## Balanced Accuracy 1.0000 1.0000 1.0000 1.0000 1.0000

predictions2 <- predict(rf, newdata=testing)  
confusionMatrix(predictions2,testing$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1395 4 0 0 0  
## B 0 942 5 0 0  
## C 0 3 849 9 2  
## D 0 0 1 795 2  
## E 0 0 0 0 897  
##   
## Overall Statistics  
##   
## Accuracy : 0.9947   
## 95% CI : (0.9922, 0.9965)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9933   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9926 0.9930 0.9888 0.9956  
## Specificity 0.9989 0.9987 0.9965 0.9993 1.0000  
## Pos Pred Value 0.9971 0.9947 0.9838 0.9962 1.0000  
## Neg Pred Value 1.0000 0.9982 0.9985 0.9978 0.9990  
## Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Detection Rate 0.2845 0.1921 0.1731 0.1621 0.1829  
## Detection Prevalence 0.2853 0.1931 0.1760 0.1627 0.1829  
## Balanced Accuracy 0.9994 0.9957 0.9948 0.9940 0.9978

## Conclusions and Test Data Submit

you can see the confusion matrix, this model looks like accurate. I did experiment with PCA and other models, but did not get as good of accuracy. Because my test data was around 99% accurate I expected nearly all of the submitted test cases to be correct. It turned out they were all correct.

pml\_write\_files = function(x){  
 n = length(x)  
 for(i in 1:n){  
 filename = paste0("problem\_id\_",i,".txt")  
 write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)  
 }  
}  
  
  
x <- evaluation\_data  
x <- x[feature\_set[feature\_set!='classe']]  
answers <- predict(rf, newdata=x)  
  
answers

## 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

pml\_write\_files(answers)