# COursera Assignment

#### Francesca

#### 11 March 2017

# Executive Summary

There are data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants; they measure how the different body parts and the dumbell itself are moving as the participant is attempting to lift it.

Participants were asked to lift the dumbell in 5 different ways, 1 correct way and 4 ‘wrong’ ways. Our aim is to predict the manner in which the participants exercise such as “how well” an exercise is taking place.

## load libraries

library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

library(ggplot2)

## Get data - download csv then load

Url\_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

Url\_testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(Url\_training), na.strings=c("NA","#DIV/0!",""))

testing <- read.csv(url(Url\_testing), na.strings=c("NA","#DIV/0!",""))

## check data

dim(training)

## [1] 19622 160

dim(testing)

## [1] 20 160

it seems first 7 variables have no predictive value

## remove variables with many NAs and variables that seem to have no predictive value

NA\_Count = sapply(1:dim(training)[2],function(x)sum(is.na(training[,x])))

NA\_Count

## [1] 0 0 0 0 0 0 0 0 0 0 0

## [12] 19226 19248 19622 19225 19248 19622 19216 19216 19226 19216 19216

## [23] 19226 19216 19216 19226 19216 19216 19216 19216 19216 19216 19216

## [34] 19216 19216 19216 0 0 0 0 0 0 0 0

## [45] 0 0 0 0 0 19216 19216 19216 19216 19216 19216

## [56] 19216 19216 19216 19216 0 0 0 0 0 0 0

## [67] 0 0 19294 19296 19227 19293 19296 19227 19216 19216 19216

## [78] 19216 19216 19216 19216 19216 19216 0 0 0 19221 19218

## [89] 19622 19220 19217 19622 19216 19216 19221 19216 19216 19221 19216

## [100] 19216 19221 0 19216 19216 19216 19216 19216 19216 19216 19216

## [111] 19216 19216 0 0 0 0 0 0 0 0 0

## [122] 0 0 0 19300 19301 19622 19299 19301 19622 19216 19216

## [133] 19300 19216 19216 19300 19216 19216 19300 0 19216 19216 19216

## [144] 19216 19216 19216 19216 19216 19216 19216 0 0 0 0

## [155] 0 0 0 0 0 0

NA\_list = which(NA\_Count>0)

## remove unnecesary columns in training and test data sets then transform class into a factor

training\_cleaning <- training[,-NA\_list]

training\_cleaning <- training\_cleaning[,-c(1:7)]

training\_cleaning$classe = factor(training\_cleaning$classe)

inTrain <-createDataPartition(training\_cleaning$classe, p=0.60, list=FALSE)

training\_clean = training\_cleaning[inTrain,]

validation\_clean = training\_cleaning[-inTrain,]

testing\_clean <- testing[,-NA\_list]

testing\_clean <- testing\_clean[,-c(1:7)]

# head(testing\_clean) # not shown in output

## build models and decide which one performs best

## this is a classification problem, and i will try random forest and classification tree

These are methods used for supervised learning which are relevant for the class I am trying to predict since the class is known.

A decision tree performs by running through all variables and picking the best split within the data set. This best split therefore should mean there are 2 distinct groups.

The process of splitting each subset is repeated until the tree has reached a maximum depth, or the benefit of splitting the subset groups any further cannot be distinguished.

The key difference between standard classification tree and random forest is that random forest builds many trees and combines them, usually by voting. The random forest is less likely to be influenced by quirks in the data (overfitting issue). But, on large datasets, it can be resource intensive and maybe hard to explain.

Therefore, i will cross validate the random forest 3 times. I will expect random forest to perform better as the dataset is small.

set.seed(2593)

Random Forest

rfFit <- train(classe ~ ., method = "rf", data = training\_clean, importance = T, trControl = trainControl(method = "cv", number = 3))

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.3.2

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##

## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':

##

## margin

#validation performance

validation\_rf\_pred <- predict(rfFit, newdata=validation\_clean)

rf\_confusion <-confusionMatrix(validation\_rf\_pred,validation\_clean$classe)

rf\_confusion

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 2230 9 0 0 0

## B 1 1506 8 0 0

## C 0 3 1357 21 0

## D 0 0 3 1265 2

## E 1 0 0 0 1440

##

## Overall Statistics

##

## Accuracy : 0.9939

## 95% CI : (0.9919, 0.9955)

## No Information Rate : 0.2845

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.9923

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.9991 0.9921 0.9920 0.9837 0.9986

## Specificity 0.9984 0.9986 0.9963 0.9992 0.9998

## Pos Pred Value 0.9960 0.9941 0.9826 0.9961 0.9993

## Neg Pred Value 0.9996 0.9981 0.9983 0.9968 0.9997

## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838

## Detection Rate 0.2842 0.1919 0.1730 0.1612 0.1835

## Detection Prevalence 0.2854 0.1931 0.1760 0.1619 0.1837

## Balanced Accuracy 0.9988 0.9953 0.9941 0.9915 0.9992

#looks good

## Random Forest Results seems satisfactory

Classification Tree

rpartFit <- train(classe ~ ., method = "rpart", data = training\_clean)

## Loading required package: rpart

#training performance

validation\_rpart\_pred <- predict(rpartFit, newdata=validation\_clean)

confusionMatrix(validation\_rpart\_pred,validation\_clean$classe)

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 2029 639 653 587 208

## B 32 497 42 205 197

## C 164 382 673 494 378

## D 0 0 0 0 0

## E 7 0 0 0 659

##

## Overall Statistics

##

## Accuracy : 0.4917

## 95% CI : (0.4806, 0.5028)

## No Information Rate : 0.2845

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.3353

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.9091 0.32740 0.49196 0.0000 0.45700

## Specificity 0.6283 0.92478 0.78111 1.0000 0.99891

## Pos Pred Value 0.4930 0.51079 0.32186 NaN 0.98949

## Neg Pred Value 0.9456 0.85145 0.87924 0.8361 0.89095

## Prevalence 0.2845 0.19347 0.17436 0.1639 0.18379

## Detection Rate 0.2586 0.06334 0.08578 0.0000 0.08399

## Detection Prevalence 0.5246 0.12401 0.26651 0.0000 0.08488

## Balanced Accuracy 0.7687 0.62609 0.63653 0.5000 0.72796

#not as good

## Regressions trees are not as good as random forest in this case.

## Therfore random forest is selected

## important variables, expected error (1-accuracy) and predictions for test data:

#Important Variables

imp\_rf <- varImp(rfFit)$importance

varImpPlot(rfFit$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex = 1, main = "Importance of the Predictors")

#accuracy and expected error

attributes(rf\_confusion)

## $names

## [1] "positive" "table" "overall" "byClass" "mode" "dots"

##

## $class

## [1] "confusionMatrix"

rf\_confusion$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull

## 0.9938822 0.9922608 0.9918968 0.9954859 0.2844762

## AccuracyPValue McnemarPValue

## 0.0000000 NaN

rf\_confusion$overall['Accuracy']

## Accuracy

## 0.9938822

rf\_confusion$overall['AccuracyUpper']

## AccuracyUpper

## 0.9954859

rf\_confusion$overall['AccuracyLower']

## AccuracyLower

## 0.9918968

testing\_rf\_pred <- predict(rfFit, newdata=testing\_clean)

testing\_rf\_pred

## [1] 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

## writing out the predictions

pml\_write\_files = function(x){

n = length(x)

for(i in 1:n){

filename = paste0("./assignm\_ml\_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)

}

}

pml\_write\_files(testing\_rf\_pred)

testing\_rf\_pred

## [1] 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