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 2228    4    0    0    0
##          B    2 1506    8    0    0
##          C    2    3 1357   15    0
##          D    0    5    3 1271    5
##          E    0    0    0    0 1437
##
## Overall Statistics
```

```
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9921   0.9920   0.9883   0.9965
## Specificity            0.9993   0.9984   0.9969   0.9980   1.0000
## Pos Pred Value         0.9982   0.9934   0.9855   0.9899   1.0000
## Neg Pred Value         0.9993   0.9981   0.9983   0.9977   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1919   0.1730   0.1620   0.1832
## Detection Prevalence   0.2845   0.1932   0.1755   0.1637   0.1832
## Balanced Accuracy      0.9987   0.9953   0.9944   0.9932   0.9983

#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 2036  659  630  575  201
##          B   30  494   49  223  204
##          C  156  365  689  488  405
##          D    0    0    0    0    0
##          E   10    0    0    0  632
##
## Overall Statistics
##
##                Accuracy : 0.4908
##                  95% CI : (0.4797, 0.502)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

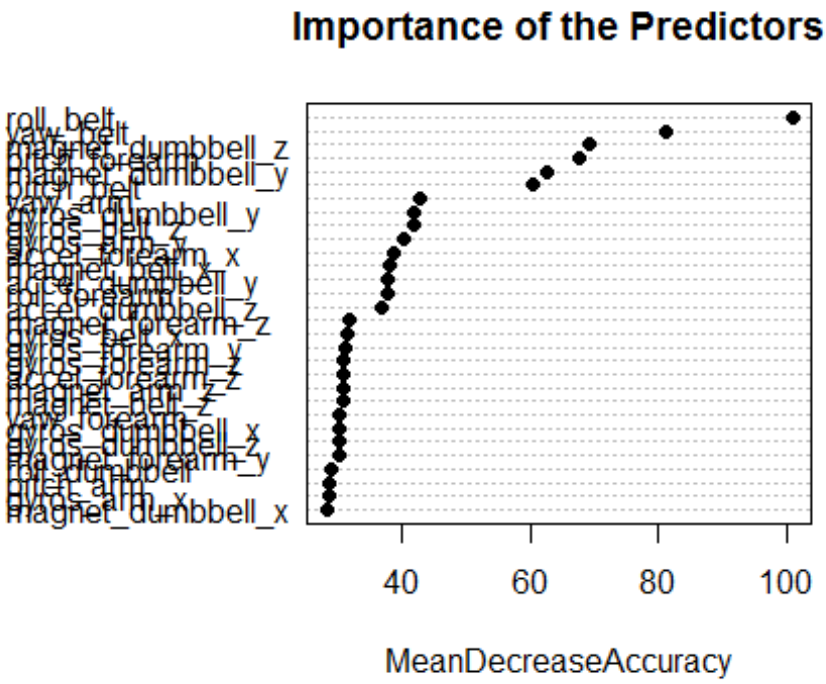
```
##
##                   Kappa : 0.3343
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9122  0.32543  0.50365   0.0000  0.43828
## Specificity            0.6322  0.92004  0.78172   1.0000  0.99844
## Pos Pred Value         0.4965  0.49400  0.32763      NaN  0.98442
## Neg Pred Value         0.9477  0.85042  0.88177   0.8361  0.88756
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2595  0.06296  0.08782   0.0000  0.08055
## Detection Prevalence   0.5227  0.12745  0.26803   0.0000  0.08183
## Balanced Accuracy      0.7722  0.62273  0.64269   0.5000  0.71836

#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
##      0.9940097      0.9924231      0.9920420      0.9955953      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN

rf_confusion$overall['Accuracy']

##  Accuracy
## 0.9940097

rf_confusion$overall['AccuracyUpper']

## AccuracyUpper
##     0.9955953

rf_confusion$overall['AccuracyLower']
```

```
## AccuracyLower
##      0.992042
```

```
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
```