Course 8 project

Niamh Devitt

Analysis of the quality of exercise taken, using data from a number of different exercise recording devices

Executive Summary

This report explores not what kind of activity is taken but how well it is taken. To do this I've used data on 6 participants who were asked to do a series of exercises correctly and incorrectly.

Data Cleaning and Exploration

First we download the training and test datasets

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url_train, destfile = "train_data.csv")
download.file(url_test, destfile = "test_data.csv")

train_data <- read.csv("~/DataScience/Course8_PracticalMachineLearning/train_data.csv")
test_data <- read.csv("~/DataScience/Course8_PracticalMachineLearning/test_data.csv")
```

Data cleaning; we remove a number of variables with near zero variance that will have little impact in our prediction. First 7 columns are not predictors and we'll remove columns that are mostly blank before we build our models.

```
train_clean <- train_data[,-c(1:7)]
remove_blanks <- which(colSums(is.na(train_clean) |train_clean=="")>0.9*dim(train_clean)[1])
train_clean <- train_clean[,-remove_blanks]

test_clean <- test_data[,-c(1,7)]
remove_blanks2 <- which(colSums(is.na(test_clean) |test_clean=="")>0.9*dim(test_clean)[1])
test_clean <- test_clean[,-remove_blanks2]
```

First we will split the training data into a training dataset and a validation set where we will test our final model before applying to the test set and submitting.

```
set.seed(383)
inTrain <- createDataPartition(train_clean$classe, p=0.7, list= FALSE)
training <- train_clean[inTrain,]
validation <- train_clean[-inTrain,]

dim(training)
```

```
## [1] 13737    53
```

```
dim(validation)
```

```
## [1] 5885   53
```

```
levels(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

Model Building

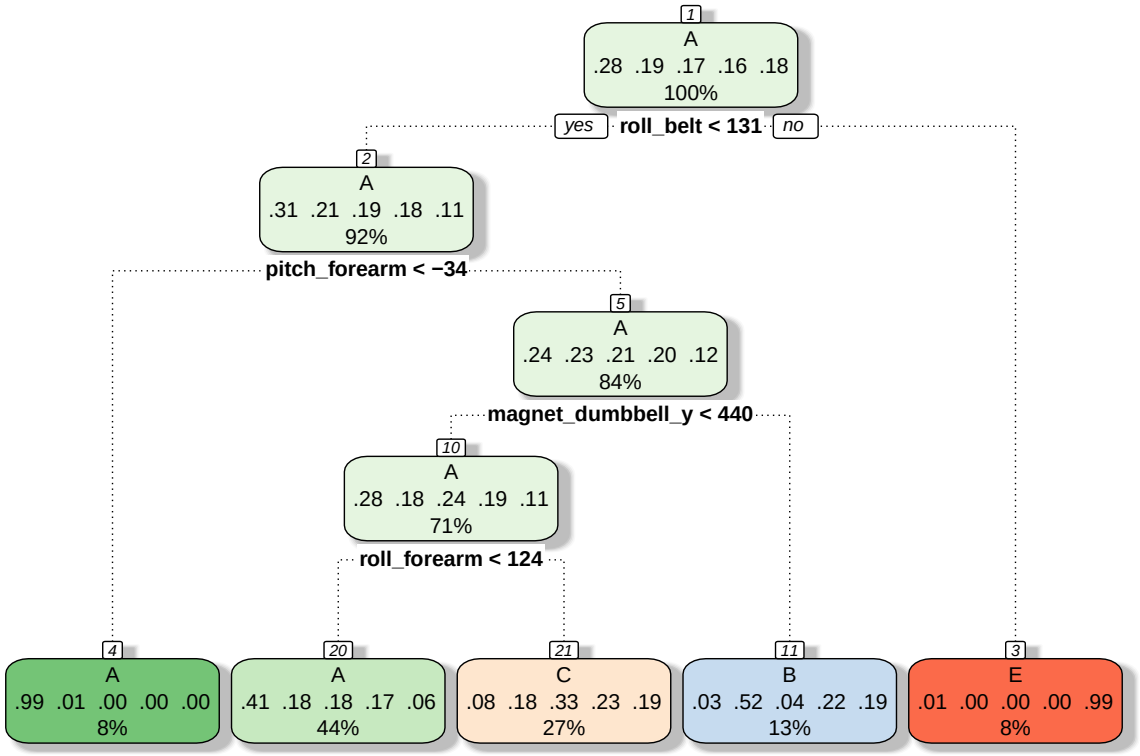
In order to avoid overfitting and to assess the effectiveness of our models we will use cross-validation in all of our models. We will fit a number of models and assess their accuracy giving us a sense of their out of sample error rate.

Model 1: Classification Tree

```
set.seed(111)
mod_cv <- trainControl(method="cv", number=3, verboseIter=FALSE)
modfit_ct<- train(classe~., data=training, method="rpart", trControl=mod_cv)
modfit_ct$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12591 8695 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1091    8 A (0.99 0.0073 0 0 0) *
##      5) pitch_forearm>=-33.95 11500 8687 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 9723 6970 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6069 3597 A (0.41 0.18 0.18 0.17 0.062) *
##         21) roll_forearm>=123.5 3654 2450 C (0.077 0.18 0.33 0.23 0.19) *
##       11) magnet_dumbbell_y>=439.5 1777  860 B (0.034 0.52 0.043 0.22 0.19) *
##    3) roll_belt>=130.5 1146   10 E (0.0087 0 0 0 0.99) *
```

```
rattle::fancyRpartPlot(modfit_ct$finalModel)
```



Model 2: Random Forest

```
set.seed(456)
mod_cv <- trainControl(method="cv", number=3, verboseIter=FALSE)
modfit_rf <- train(classe ~., data = training, method = "rf", trControl = mod_cv)
modfit_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.79%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    3    1    0    1 0.001280082
## B   21 2627    9    1    0 0.011662904
## C    0   11 2373   12    0 0.009599332
```

```
## D    0    2   30 2217    3 0.015541741
## E    0    1    5    8 2511 0.005544554
```

Model 3: Gradient Boosting

```
set.seed(444)
modfit_gbm <- train(classe ~., data = training, method = "gbm", verbose=FALSE, trControl = mod_cv)
modfit_gbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

Model Evaluation and Selection

The accuracy of our classification tree is low at 49.5%. This means in-sample error is c.50% and we can expect out of sample error to be worse meaing this is model than random selection.

The random forest model has 99.2% accuracy with cross validation 3 times in predicting the validation set, expected out of sample error will be higher than in-sample error so will be slightly bigger than 1% in sample error however this is still a very strong model.

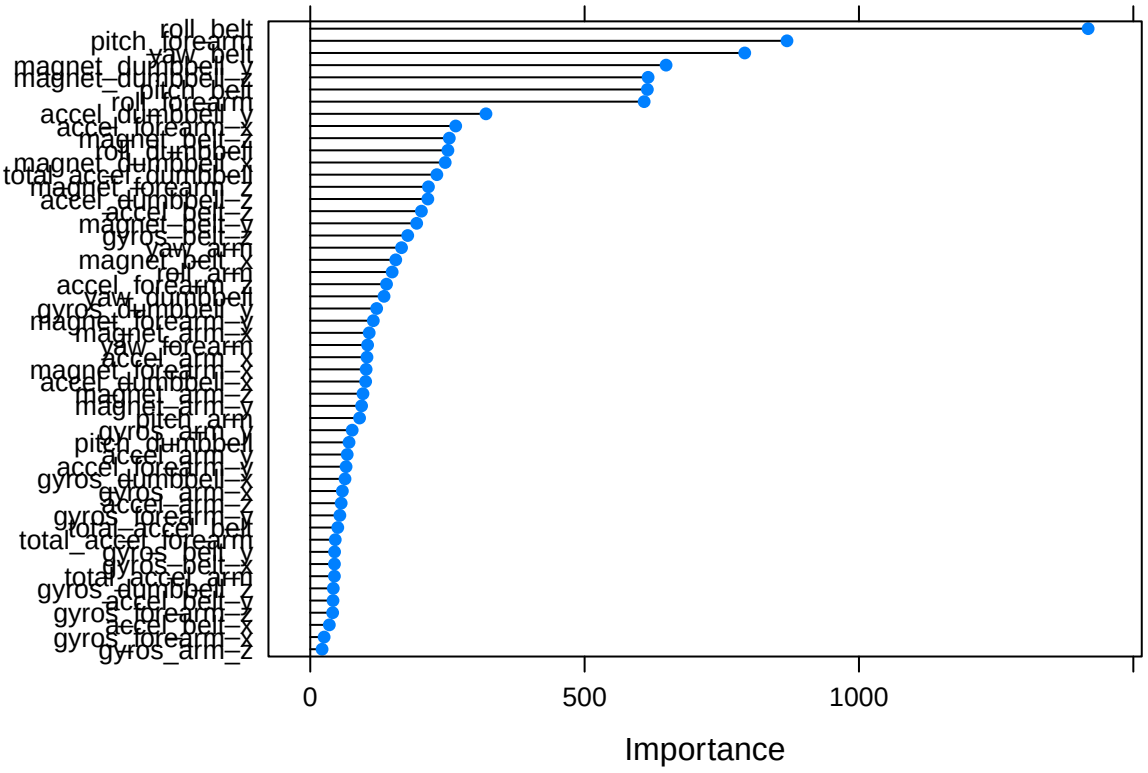
The gradient boosting model has 96% accuracy with cross validation. Our final model will be the random forest model.

```
pred_ct <- predict(modfit_ct, newdata=validation)
acc_ct <- confusionMatrix(pred_ct, validation$classe)$overall['Accuracy']

pred_rf <- predict(modfit_rf, newdata=validation)
acc_rf <- confusionMatrix(pred_rf, validation$classe)$overall['Accuracy']

pred_gbm <- predict(modfit_gbm, newdata=validation)
acc_gbm <- confusionMatrix(pred_gbm, validation$classe)$overall['Accuracy']

accuracy_vec <- c(acc_ct, acc_rf, acc_gbm)
names(accuracy_vec) <- c("classification", "random forest", "gradient boost")
accuracy_vec
```

```
## classification  random forest gradient boost
##      0.4948173      0.9933730      0.9612574
```

We will look at variable importance for our selected model. We can see high importance in the top 8 predictor before a significant drop. We could streamline the predictors if we wanted.

```
rfImp <- varImp(modfit_rf, scale = FALSE)
plot(rfImp)
```



Using our Final Model to predict on the Clean Test set

```
FinalTest_pred <- predict(modfit_rf,newdata=test_clean)
FinalTest_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
```

The results wll be tested in Course Project Quiz.