R Notebook

Pull Data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.3
```

```
## 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
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.4.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

We begin by pulling the data from the excel files

```
pml_training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0", ""))

pml_testing <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0", ""))
```

Splitting the training data

We now split the training data into a training group (70%) and a testing group (30%).

```
set.seed(2121)

train <- createDataPartition(y=pml_training$classe, p=0.7, list=FALSE)

in_train <- pml_training[train,]

in_test <- pml_training[-train,]
```

Cleaning the data

In order to create a viable prediction model, we need to cut out the useless variables.

The first seven variables are of no use to us and should be removed.

```
in_train <- in_train[, -c(1:7)]
in_test <- in_test[, -c(1:7)]

dim(in_train)
```

```
## [1] 13737   153
```

```
dim(in_test)
```

```
## [1] 5885  153
```

Next, we delete columns that have scarce data. We want columns of which at least 10% of their values are not "NA".

```
in_train <- in_train[, colSums(is.na(in_train))==0]
in_test <- in_test[, colSums(is.na(in_test))==0]

dim(in_train)
```

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

```
dim(in_test)
```

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

We now have the data sets cut down to 53 variables.

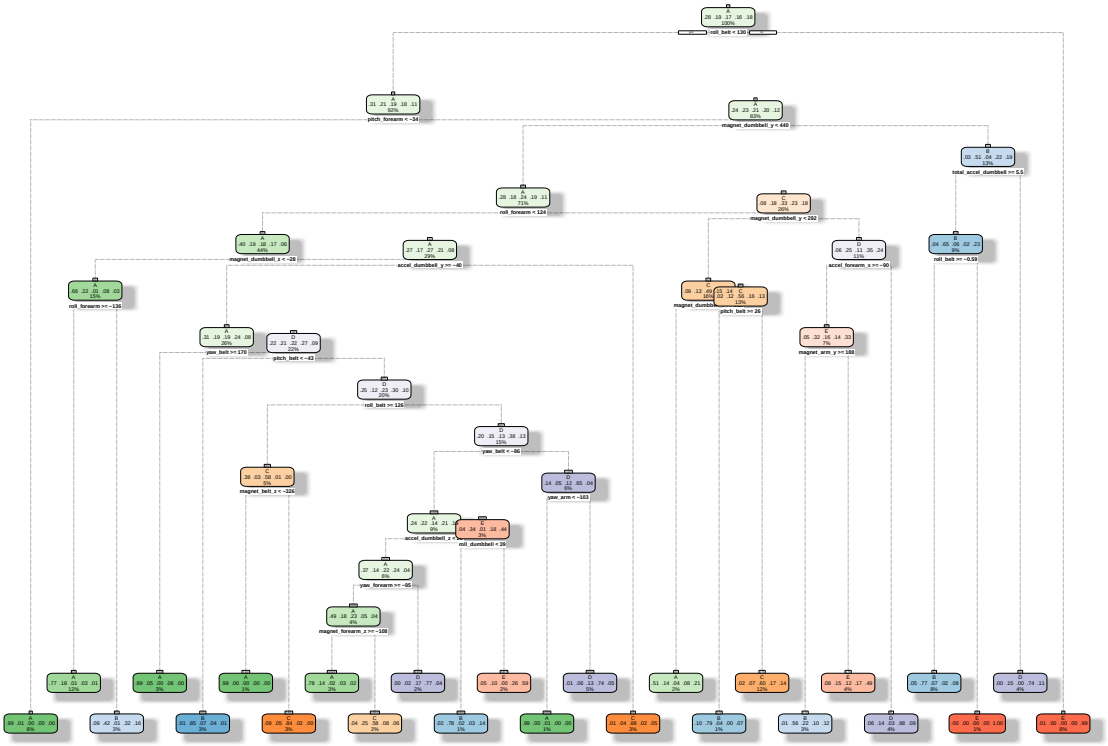
Model 1: Decision tree

```
dec_tre_intr <- rpart(classe ~., data = in_train, method = "class")
```

plot of decision tree:

```
fancyRpartPlot(dec_tre_intr)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2018−Jan−14 15:19:25 mmads

```
predict_dt <- predict(dec_tre_intr, in_test,  type = "class")

confusionMatrix(predict_dt, in_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1528  169   21   53   43
##          B   59  711   93   76   97
##          C   43  116  818  141  104
##          D   18   89   67  609   51
##          E   26   54   27   85  787
##
## Overall Statistics
##
##                Accuracy : 0.7567
##                  95% CI : (0.7455, 0.7676)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6913
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9128   0.6242   0.7973   0.6317   0.7274
## Specificity            0.9321   0.9315   0.9169   0.9543   0.9600
## Pos Pred Value         0.8423   0.6863   0.6694   0.7302   0.8039
## Neg Pred Value         0.9641   0.9117   0.9554   0.9297   0.9399
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2596   0.1208   0.1390   0.1035   0.1337
## Detection Prevalence   0.3082   0.1760   0.2076   0.1417   0.1664
## Balanced Accuracy      0.9224   0.7779   0.8571   0.7930   0.8437
```

Our decision tree model has an accuracy of 0.7567.

Model 2: Random Forest

```
ran_fr_intr <- randomForest(classe ~., data = in_train, method = "class")
predict_rf <- predict(ran_fr_intr, in_test,  type = "class")

confusionMatrix(predict_rf, in_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    0 1128    4    0    0
##          C    0    3 1020   12    0
##          D    0    0    2  952    3
##          E    1    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9903   0.9942   0.9876   0.9972
## Specificity            0.9981   0.9992   0.9969   0.9990   0.9998
## Pos Pred Value         0.9952   0.9965   0.9855   0.9948   0.9991
## Neg Pred Value         0.9998   0.9977   0.9988   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1917   0.1733   0.1618   0.1833
## Detection Prevalence   0.2856   0.1924   0.1759   0.1626   0.1835
## Balanced Accuracy      0.9988   0.9947   0.9955   0.9933   0.9985
```

Conclusion

In comparrison we see that our random forest model is more accuarate than our decision tree model (decision tree accuracy: 75.67%, random forest accuracy: 99.44%).

We will use the random forest model for our predictions.

The expected error range is: 0.9961-0.9921 = 0.004, which is less than 1%. So out of 20 different test cases, it is highly unlikely that any of our predictions will be incorrect.