Practical Machine Learning Project

Yong-Hao Bai

7/10/2020

Load the required packages

```
library(caret); library(rattle); library(rpart); library(rpart.plot); library(randomForest); library(repmi
s);
library(lattice); library(ggplot2); library(readr); library(gbm)
```

Load the Data, divide the data

```
set.seed(19)
trainurl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testurl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(trainurl, "pml-training.csv")
download.file(testurl, "pml-testing.csv")
training <- read.csv("pml-training.csv",  na.strings=c("NA","#DIV/0!", ""))
testing <- read.csv("pml-testing.csv",  na.strings=c("NA","#DIV/0!", ""))
#update datasets to exclude those variables with NA values
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

Remove irrelevant variables to the prediction

```
newtraining <- training[,-c(1:7)]
newtesting <- testing[, -c(1:7)]
```

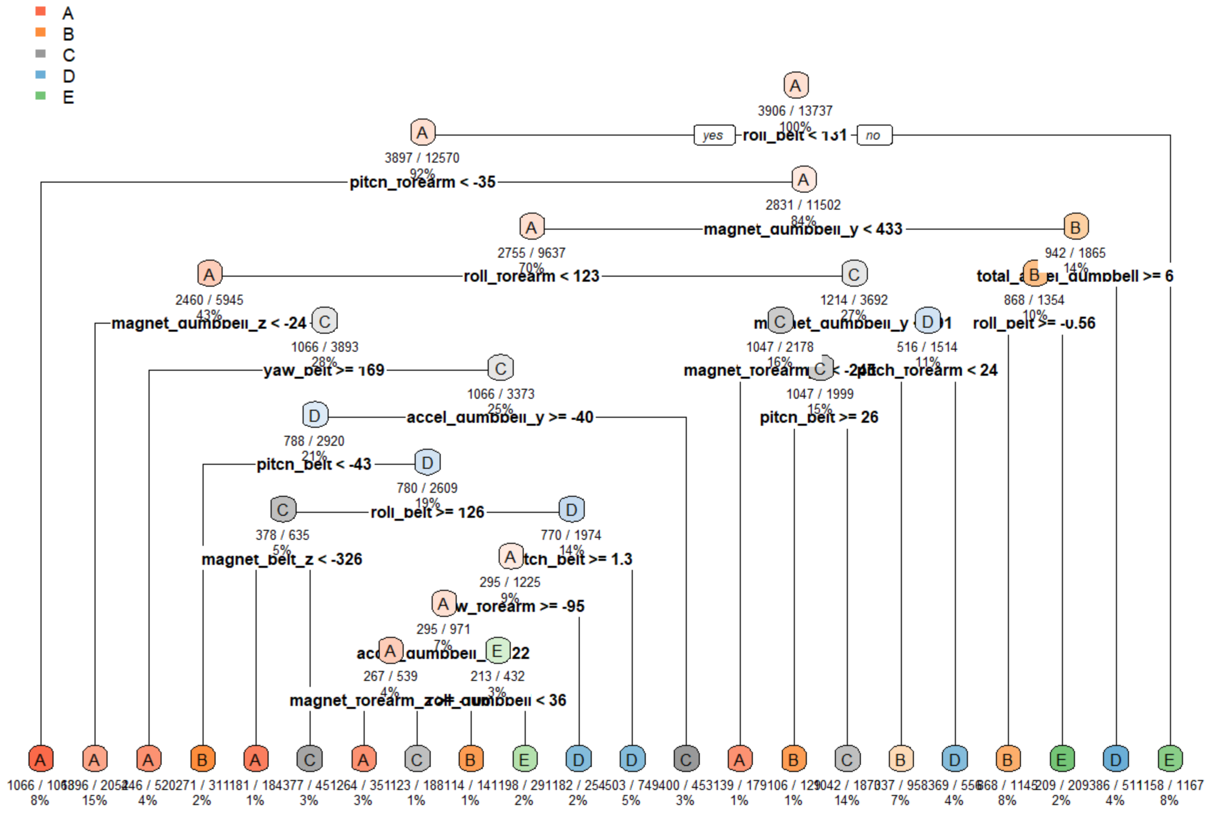
For cross validation purpose, the training data will be split into training training and training testing.

Data Modeling

Test the predictive power by trying different methods

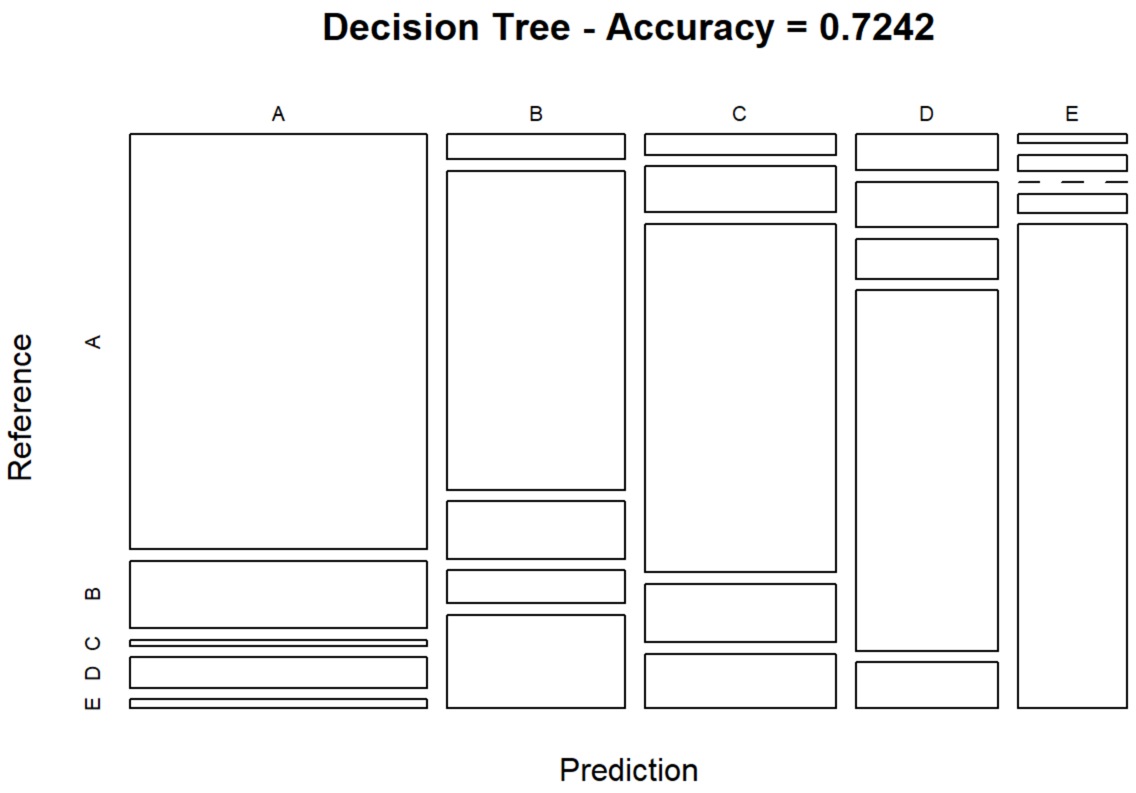
Decision Tree

Classification Tree



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1499  243   23  109   33
##          B   54  689  124   71  202
##          C   47  107  810  136  127
##          D   62   79   69  623   79
##          E   12   21    0   25  641
##
## Overall Statistics
##
##                Accuracy : 0.7242
##                  95% CI : (0.7126, 0.7356)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6495
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8955   0.6049   0.7895   0.6463   0.5924
## Specificity            0.9031   0.9050   0.9142   0.9413   0.9879
## Pos Pred Value         0.7861   0.6044   0.6601   0.6831   0.9170
## Neg Pred Value         0.9560   0.9052   0.9536   0.9314   0.9150
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2547   0.1171   0.1376   0.1059   0.1089
## Detection Prevalence   0.3240   0.1937   0.2085   0.1550   0.1188
## Balanced Accuracy      0.8993   0.7549   0.8518   0.7938   0.7902
```

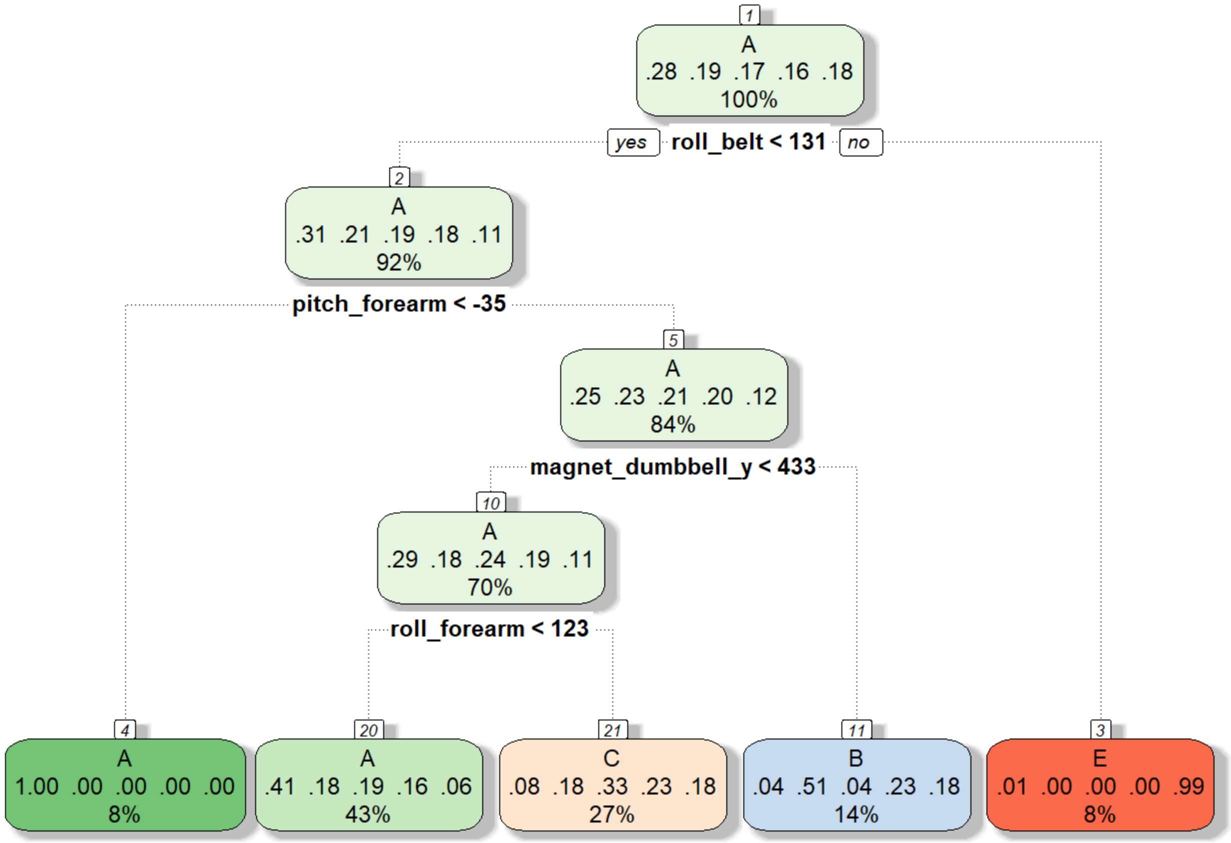
```
##  Accuracy
## 0.7242141
```



The accuracy rate of the model is low: 0.7242.

Classification tree

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03550  0.5112    0.36227
##   0.06052  0.4414    0.25185
##   0.11688  0.2998    0.02353
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0355.
```



Rattle 2020-Jul-11 23:33:41 josep

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1526   31  112    0    5
##          B  490  379  270    0    0
##          C  476   38  512    0    0
##          D  436  174  354    0    0
##          E  158  157  294    0  473
## 
## Overall Statistics
## 
##                Accuracy : 0.4911          
##                  95% CI : (0.4782, 0.5039)
##     No Information Rate : 0.5244          
##     P-Value [Acc > NIR] : 1               
## 
##                   Kappa : 0.3344          
## 
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4945   0.4865   0.3320       NA  0.98954
## Specificity            0.9471   0.8512   0.8816   0.8362  0.88737
## Pos Pred Value         0.9116   0.3327   0.4990       NA  0.43715
## Neg Pred Value         0.6295   0.9157   0.7880       NA  0.99896
## Prevalence             0.5244   0.1324   0.2620   0.0000  0.08122
## Detection Rate         0.2593   0.0644   0.0870   0.0000  0.08037
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638  0.18386
## Balanced Accuracy      0.7208   0.6688   0.6068       NA  0.93845
```

```
## Accuracy 
## 0.491079
```

The accuracy rate of the model is even lower.

Boosted Logistic Regression

```
## Boosted Logistic Regression
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10991, 10989
## Resampling results across tuning parameters:
##
##   nIter  Accuracy   Kappa
##   11     0.8194597  0.7688882
##   21     0.8704476  0.8344241
##   31     0.8958802  0.8674851
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was nIter = 31.
```

```
##  Accuracy
## 0.8967284
```

The accuracy rate of the model has improved from the prior 2 models.

Gradient Boosting

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10990, 10988, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7473965  0.6795041
##   1                  100      0.8189556  0.7707698
##   1                  150      0.8516405  0.8122365
##   2                   50      0.8528048  0.8134939
##   2                  100      0.9060922  0.8811091
##   2                  150      0.9303334  0.9118307
##   3                   50      0.8956102  0.8678282
##   3                  100      0.9430734  0.9279625
##   3                  150      0.9595977  0.9488814
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1653   36    0    0    0
##          B   13 1063   30    8   16
##          C    6   38  983   27    8
##          D    1    2   10  923   20
##          E    1    0    3    6 1038
## 
## Overall Statistics
## 
##                Accuracy : 0.9618
##                  95% CI : (0.9565, 0.9665)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9516
## 
##  Mcnemar's Test P-Value : 1.357e-08
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9875   0.9333   0.9581   0.9575   0.9593
## Specificity            0.9915   0.9859   0.9837   0.9933   0.9979
## Pos Pred Value         0.9787   0.9407   0.9256   0.9655   0.9905
## Neg Pred Value         0.9950   0.9840   0.9911   0.9917   0.9909
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2809   0.1806   0.1670   0.1568   0.1764
## Detection Prevalence   0.2870   0.1920   0.1805   0.1624   0.1781
## Balanced Accuracy      0.9895   0.9596   0.9709   0.9754   0.9786
```

```
##  Accuracy 
## 0.9617672
```

The Accuracy is getting better.

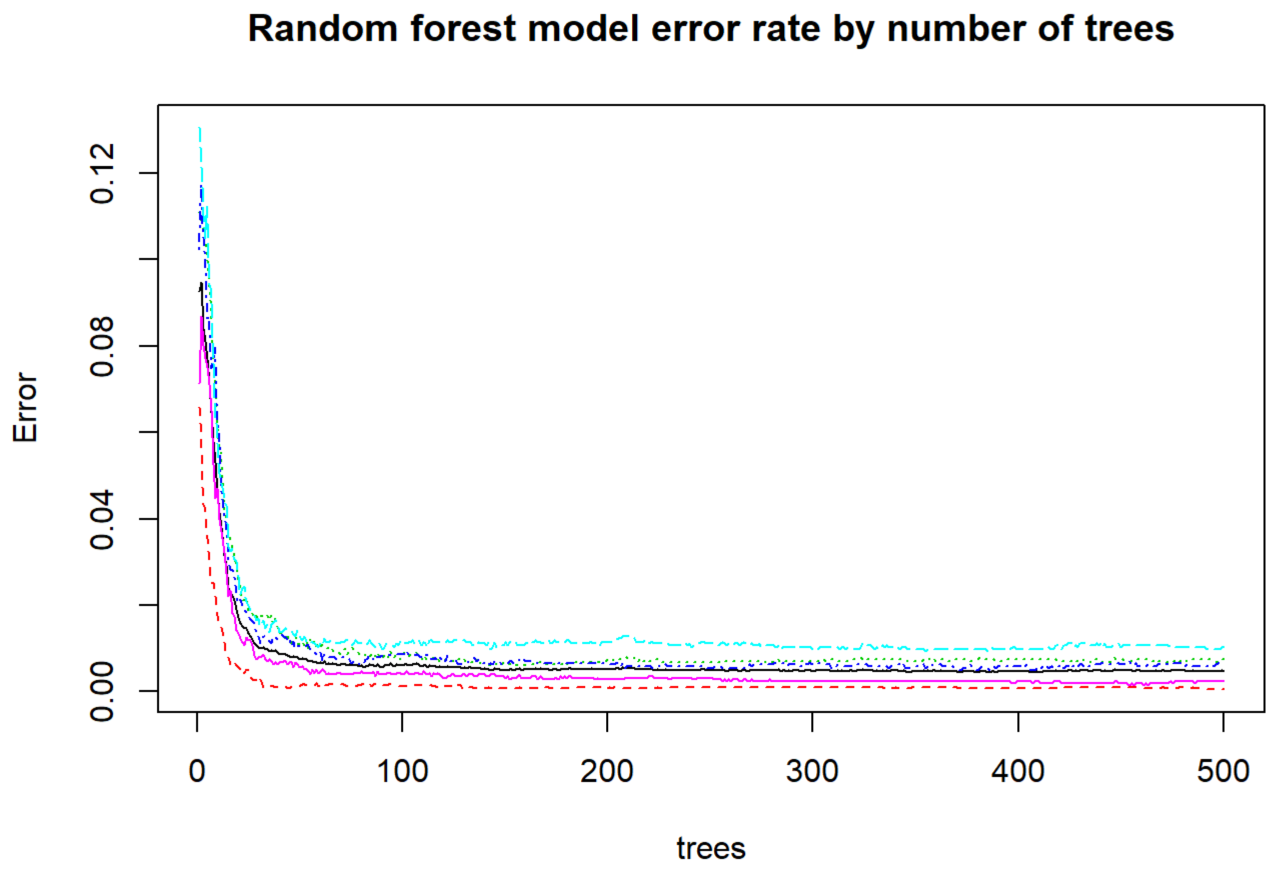
Random Forest

```
## 
## Call:
##  randomForest(formula = classe ~ ., data = training_train, method = "class") 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
## 
##         OOB estimate of  error rate: 0.48%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B   12 2638    8    0    0 0.0075244545
## C    0   15 2381    0    0 0.0062604341
## D    0    0   22 2229    1 0.0102131439
## E    0    0    2    4 2519 0.0023762376
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    0 1132    2    0    0
##          C    0    1 1022   13    1
##          D    0    0    2  950    0
##          E    1    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9939   0.9961   0.9855   0.9991
## Specificity            0.9986   0.9996   0.9969   0.9996   0.9996
## Pos Pred Value         0.9964   0.9982   0.9855   0.9979   0.9982
## Neg Pred Value         0.9998   0.9985   0.9992   0.9972   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1924   0.1737   0.1614   0.1837
## Detection Prevalence   0.2853   0.1927   0.1762   0.1618   0.1840
## Balanced Accuracy      0.9990   0.9967   0.9965   0.9925   0.9993
```

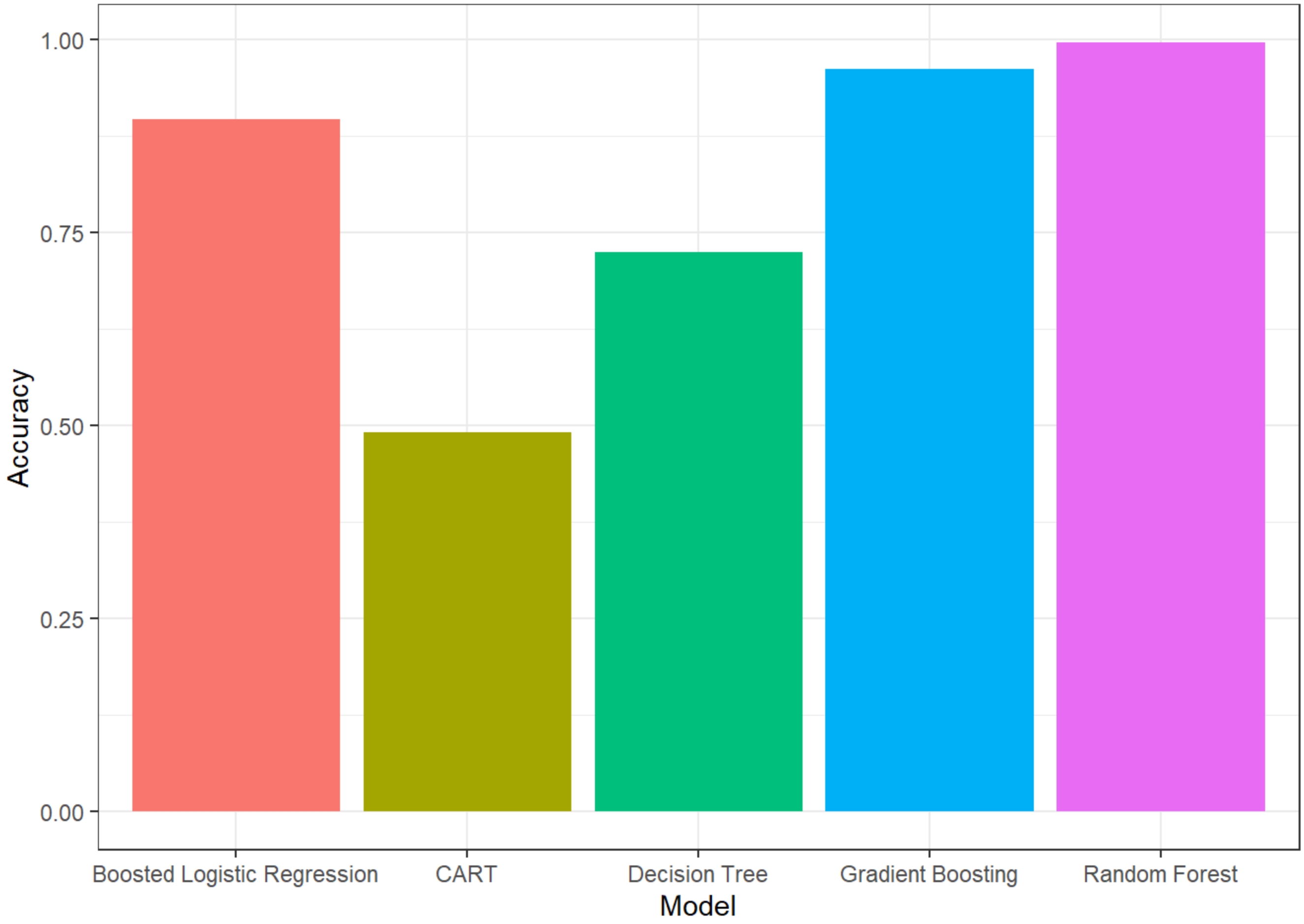
```
##  Accuracy
## 0.9954121
```

Looking at the results, clearly, the random forest model provides a more accurate prediction of classe. The expected out-of-sample error is estimated at 0.005.

plot of the model error rate by number of trees and 20 most important variables (out of 52)



Accurary comparison among models



Random Forest has the highest accurary.

Prediction on Testing

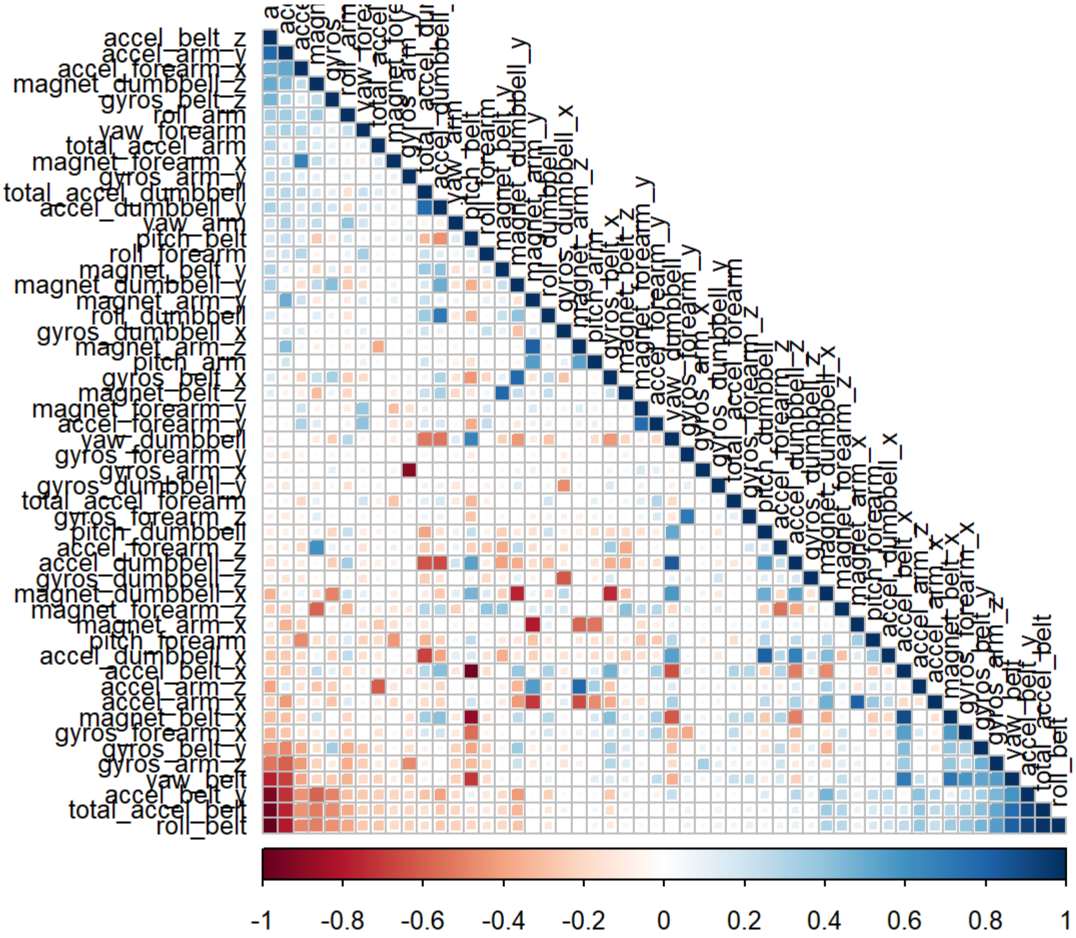
Based on Random Forest prediction:

```
##  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
```

Appendix

check for correlation

```
## corrplot 0.84 loaded
```



Variable Importance

```
##                        Overall
## roll_belt            869.54426
## pitch_belt           480.80822
## yaw_belt             634.18014
## total_accel_belt     158.91033
## gyros_belt_x          63.94466
## gyros_belt_y          76.19367
## gyros_belt_z         219.52569
## accel_belt_x          80.27219
## accel_belt_y          82.95947
## accel_belt_z         268.10200
## magnet_belt_x        186.61420
## magnet_belt_y        275.78663
## magnet_belt_z        293.60114
## roll_arm             213.62676
## pitch_arm            124.15713
## yaw_arm              161.48501
## total_accel_arm       67.54977
## gyros_arm_x           96.47692
## gyros_arm_y           96.40537
## gyros_arm_z           44.92191
## accel_arm_x          162.16946
## accel_arm_y          113.43433
## accel_arm_z           94.60188
## magnet_arm_x         183.19255
## magnet_arm_y         163.13118
## magnet_arm_z         129.01364
## roll_dumbbell        293.17185
## pitch_dumbbell       126.78872
## yaw_dumbbell         177.74233
## total_accel_dumbbell 194.09640
## gyros_dumbbell_x      90.94863
## gyros_dumbbell_y     169.50741
## gyros_dumbbell_z      63.29426
## accel_dumbbell_x     177.25603
## accel_dumbbell_y     298.05300
## accel_dumbbell_z     246.44862
## magnet_dumbbell_x    339.16801
## magnet_dumbbell_y    477.30535
## magnet_dumbbell_z    523.25405
## roll_forearm         421.36225
## pitch_forearm        534.13935
## yaw_forearm          116.62986
## total_accel_forearm   75.09184
## gyros_forearm_x       54.18130
## gyros_forearm_y       86.85268
## gyros_forearm_z       56.60624
## accel_forearm_x      219.74014
## accel_forearm_y      101.00889
## accel_forearm_z      166.67266
## magnet_forearm_x     150.19272
## magnet_forearm_y     156.75580
## magnet_forearm_z     203.17271
```