MachineLearning_Project

Ronald Armando

4/26/2018

Data Loading

Dataset Overview

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project can be found here: http://groupware.les.inf.puc-rio.br/har.

A short description of the datasets content from the authors' website:

"Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

Getting and loading the data

We first upload the R libraries that are necessary for the complete analysis.

```
rm(list=ls())

library(caret)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(knitr)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
library(rattle)
```

```
## 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
```

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

set.seed(12345)

#Set URL for data download
train_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

#Download datasets
training <- read.csv(url(train_Url), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(test_Url), na.strings=c("NA","#DIV/0!",""))

# Create a partition with the training dataset
# 70% Training Data & 30% Test Data
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainD <- training[inTrain, ]
TestD  <- training[-inTrain, ]
```

Let's see the dimensions for both training and test sets and have an overview of the dataset with the str() function

```
dim(TrainD)
```

```
## [1] 13737   160
```

```
dim(TestD)
```

```
## [1] 5885  160
```

```
str(TrainD)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  2 3 4 5 6 7 8 12 13 14 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  808298 820366 120339 196328 304277 368296 440390 528316 560359 576
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.42 1.42 ...
##  $ pitch_belt              : num  8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.18 8.2 8.21 ...
```

```
##  $ yaw_belt               : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt       : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x           : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_belt_y           : num  0 0 0 0.02 0 0 0 0 0 0 ...
##  $ gyros_belt_z           : num  -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 ...
##  $ accel_belt_x           : int  -22 -20 -22 -21 -21 -22 -22 -22 -22 -22 ...
##  $ accel_belt_y           : int  4 5 3 2 4 3 4 2 4 4 ...
##  $ accel_belt_z           : int  22 23 21 24 21 21 21 23 21 21 ...
##  $ magnet_belt_x          : int  -7 -2 -6 -6 0 -4 -2 -2 -3 -8 ...
##  $ magnet_belt_y          : int  608 600 604 600 603 599 603 602 606 598 ...
##  $ magnet_belt_z          : int  -311 -305 -310 -302 -312 -311 -313 -319 -309 -310 ...
##  $ roll_arm               : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm              : num  22.5 22.5 22.1 22.1 22 21.9 21.8 21.5 21.4 21.4 ...
##  $ yaw_arm                : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm        : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x            : num  0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y            : num  -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.02 0 ...
##  $ gyros_arm_z            : num  -0.02 -0.02 0.02 0 0 0 0 0 -0.02 -0.03 ...
##  $ accel_arm_x            : int  -290 -289 -289 -289 -289 -289 -289 -288 -287 -288 ...
```

```
##  $ accel_arm_y             : int  110 110 111 111 111 111 111 111 111 111 ...
##  $ accel_arm_z             : int  -125 -126 -123 -123 -122 -125 -124 -123 -124 -124 ...
##  $ magnet_arm_x            : int  -369 -368 -372 -374 -369 -373 -372 -363 -372 -371 ...
##  $ magnet_arm_y            : int  337 344 344 337 342 336 338 343 338 331 ...
##  $ magnet_arm_z            : int  513 513 512 506 513 509 510 520 509 523 ...
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 12.9 13.4 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.6 -70.3 -70.4 -70.4 -70.8 ...
##  $ yaw_dumbbell            : num  -84.7 -85.1 -84.9 -84.9 -84.5 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Cleaning the data

We can see from the previous step that the datasets have 160 variables, and also that many of their variables have NA values. The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```
# Remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainD)
TrainD <- TrainD[, -NZV]
TestD  <- TestD[, -NZV]
dim(TrainD)
```

```
## [1] 13737   130
```

```
dim(TestD)
```

```
## [1] 5885  130
```

```
# remove variables that are mostly NA
AllNAval    <- sapply(TrainD, function(x) mean(is.na(x))) > 0.95
TrainD <- TrainD[, AllNAval==FALSE]
TestD  <- TestD[, AllNAval==FALSE]
dim(TrainD)
```

```
## [1] 13737    59
```

```
dim(TestD)
```

```
## [1] 5885   59
```

Now we need to remove the identification only variables, which correspond to columns 1 to 5

```
TrainD <- TrainD[, -(1:5)]
TestD  <- TestD[, -(1:5)]
dim(TrainD)
```

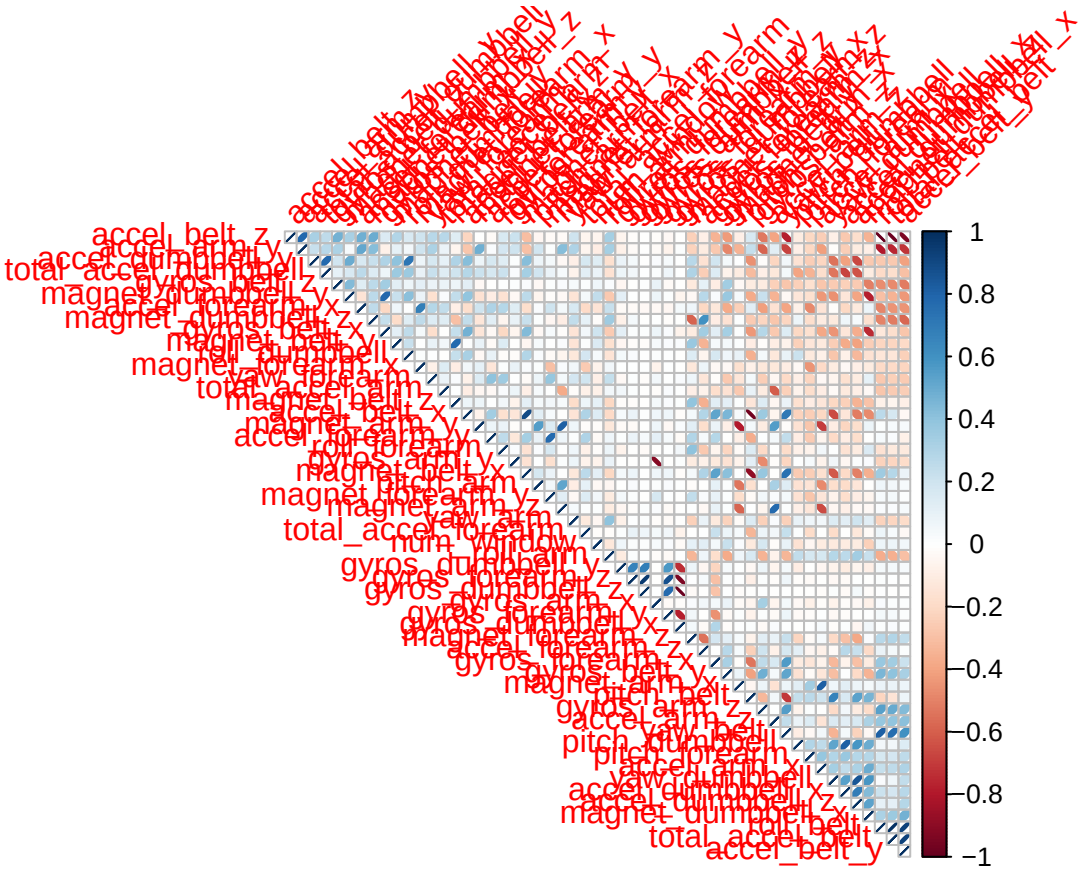
```
## [1] 13737    54
```

```
dim(TestD)
```

```
## [1] 5885   54
```

After all these steps, we see that we were able to reduce the number of variables from 160 to 54.

Correlation Analysis



The highly correlated variables are shown in dark colors in the graph above: blue if they are positively correlated and red if they are negativily correlated.

Prediction Model Building

Three methods will be applied to model the regressions and the best one, with higher accuracy when applied to the Test dataset, will be used for the quiz predictions. The methods chosen for are: Random Forests, Decision Tree and Generalized Boosted Model. At the end of each analysis, a confusion matrix will be plotted to better visualize the accuracy of the each model.

Random Forest

```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRF <- train(classe ~ ., data=TrainD, method="rf",
                          trControl=controlRF)
modFitRF$finalModel
```

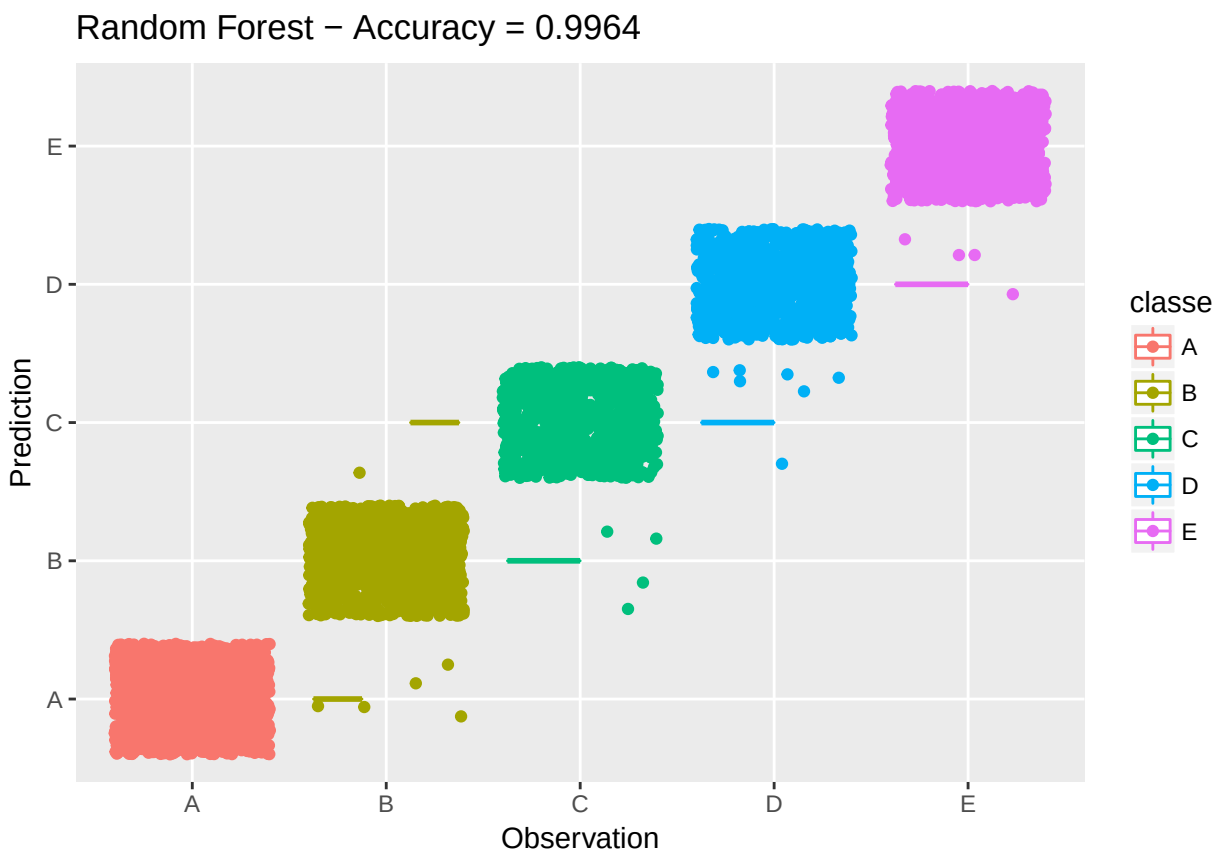
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2649    2    1    0 0.0033860045
## C    0    4 2391    1    0 0.0020868114
## D    0    0    7 2245    0 0.0031083481
## E    0    0    0    5 2520 0.0019801980
```

```
# prediction on Test dataset
predictRF <- predict(modFitRF, newdata=TestD)
confMatRF <- confusionMatrix(predictRF, TestD$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    4    0    0
##          C    0    1 1022    7    0
##          D    0    0    0  957    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9964
##                  95% CI : (0.9946, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9961   0.9927   0.9963
## Specificity            0.9988   0.9992   0.9984   0.9992   1.0000
## Pos Pred Value         0.9970   0.9965   0.9922   0.9958   1.0000
## Neg Pred Value         1.0000   0.9987   0.9992   0.9986   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1737   0.1626   0.1832
## Detection Prevalence   0.2853   0.1932   0.1750   0.1633   0.1832
## Balanced Accuracy      0.9994   0.9969   0.9972   0.9960   0.9982
```

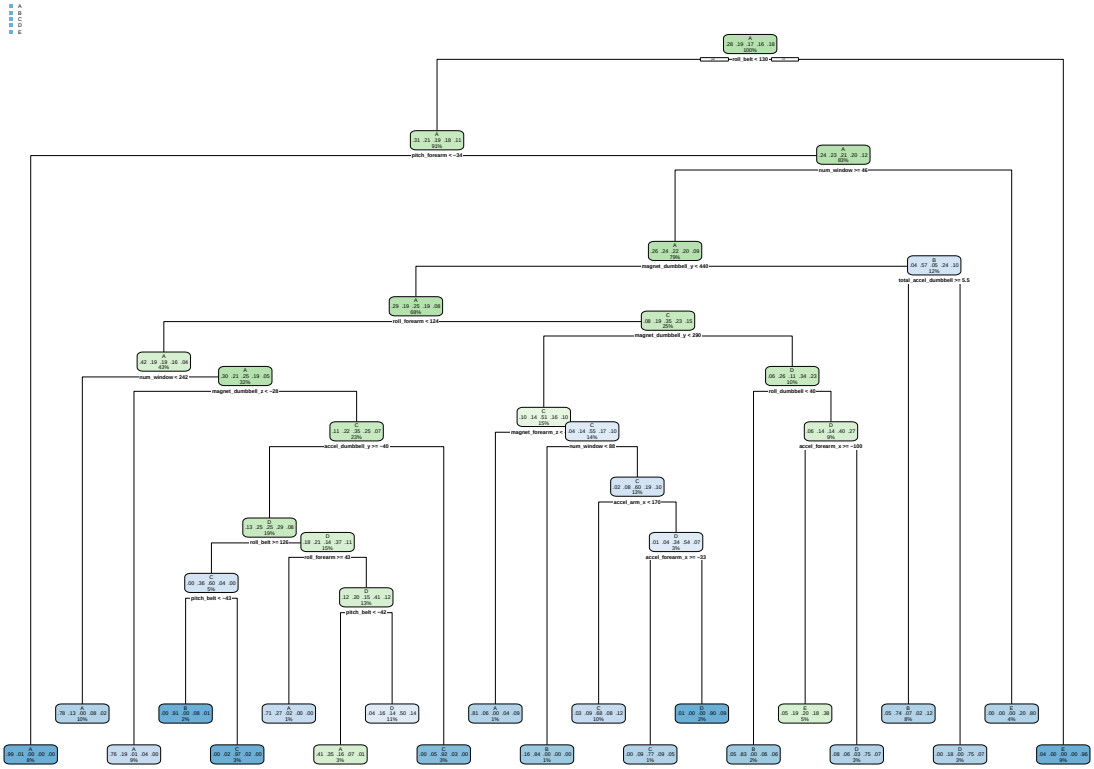
```
# Plot matrix results
qplot(classe, predictRF, data=TestD,  colour= classe, geom = c("boxplot", "jitter"), main = paste("Rand
```



Decision Trees

```r
# Model Fit
set.seed(12345)
modFitDT <- rpart(classe ~ ., data=TrainD, method="class")
rpart.plot(modFitDT, box.palette = "GnBu")
```

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

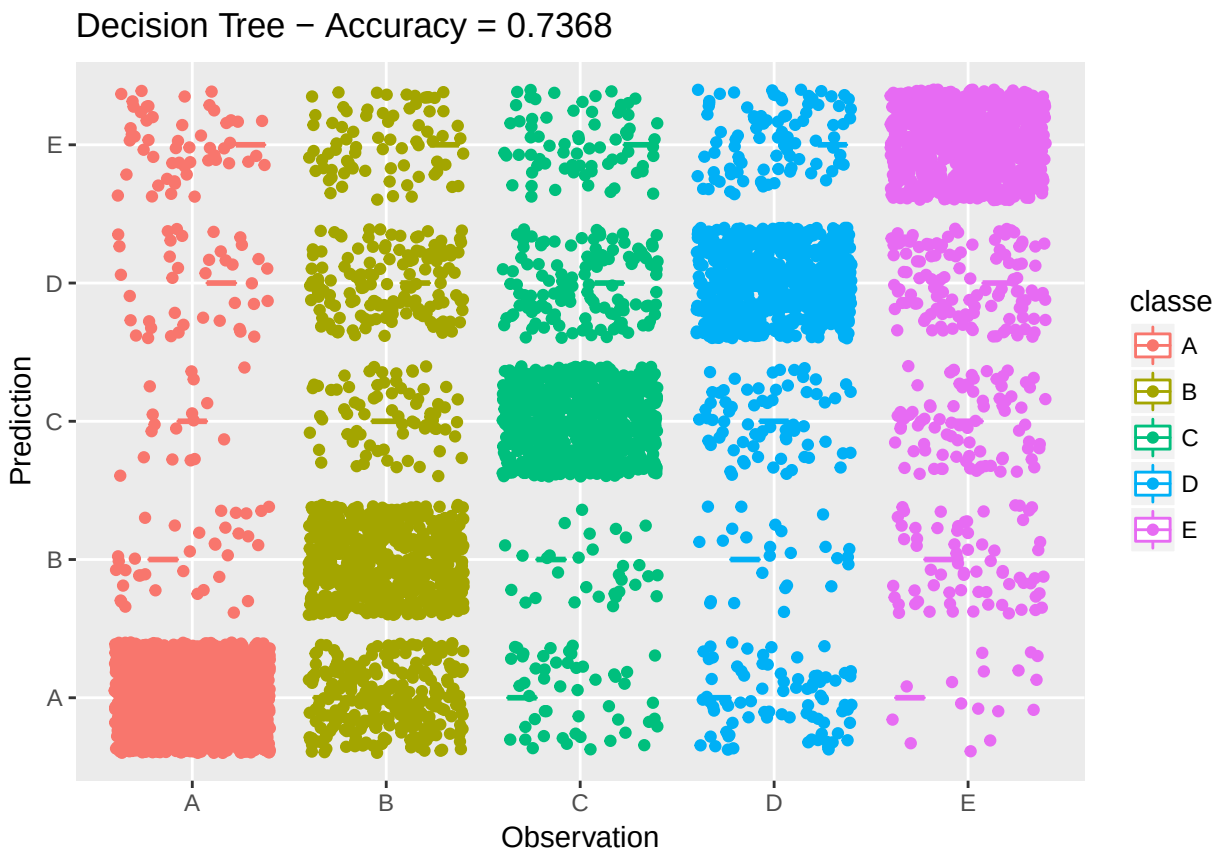


```r
predictDT <- predict(modFitDT, newdata=TestD, type="class")
confMatDT <- confusionMatrix(predictDT, TestD$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  269   51   79   16
##          B   35  575   31   25   68
##          C   17   73  743   68   84
##          D   39  146  130  702  128
##          E   53   76   71   90  786
##
## Overall Statistics
##
##                Accuracy : 0.7368
##                  95% CI : (0.7253, 0.748)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6656
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.50483   0.7242   0.7282   0.7264
## Specificity            0.9014  0.96650   0.9502   0.9100   0.9396
## Pos Pred Value         0.7866  0.78338   0.7543   0.6131   0.7305
## Neg Pred Value         0.9635  0.89051   0.9422   0.9447   0.9384
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.3305  0.12472   0.1674   0.1946   0.1828
## Balanced Accuracy      0.9077  0.73566   0.8372   0.8191   0.8330
```

```
# Plot matrix results
qplot(classe, predictDT, data=TestD,  colour= classe, geom = c("boxplot", "jitter"), main = paste("Deci
```



Generalized Boosted Model

```
set.seed(12345)
controlGbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGbm  <- train(classe ~ ., data=TrainD, method = "gbm",
                    trControl = controlGbm, verbose = FALSE)
modFitGbm$finalModel
```

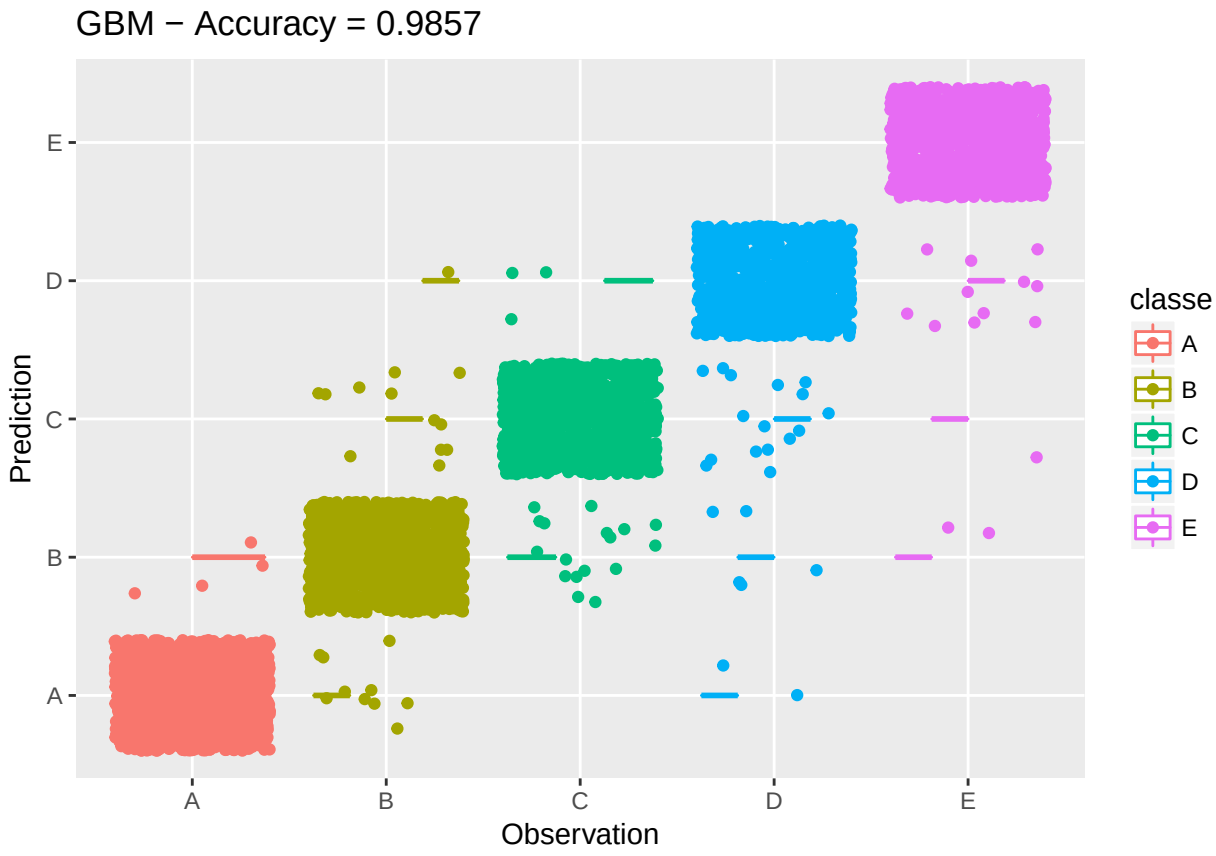
```
## A gradient boosted model with multinomial loss function.
```

```
## 150 iterations were performed.
## There were 53 predictors of which 41 had non-zero influence.
```

```
predictGbm <- predict(modFitGbm, newdata=TestD)
confMatGbm <- confusionMatrix(predictGbm, TestD$classe)
confMatGbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   10    0    2    0
##          B    4 1116   17    5    2
##          C    0   12 1006   16    1
##          D    0    1    3  941   11
##          E    0    0    0    0 1068
##
## Overall Statistics
##
##                Accuracy : 0.9857
##                  95% CI : (0.9824, 0.9886)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9819
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9798   0.9805   0.9761   0.9871
## Specificity            0.9972   0.9941   0.9940   0.9970   1.0000
## Pos Pred Value         0.9929   0.9755   0.9720   0.9843   1.0000
## Neg Pred Value         0.9990   0.9951   0.9959   0.9953   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1896   0.1709   0.1599   0.1815
## Detection Prevalence   0.2858   0.1944   0.1759   0.1624   0.1815
## Balanced Accuracy      0.9974   0.9870   0.9873   0.9865   0.9935
```

```
# Plot matrix results
qplot(classe, predictGbm, data=TestD,  colour= classe, geom = c("boxplot", "jitter"), main = paste("GBM
```



Applying Best Model to Test Data

The accuracy of the 3 regression modeling methods are:

Random Forest : 0.9964 Decision Tree : 0.7368 *GBM : 0.9857

Since the Random Forest model had the best accuracy, it will be applied to predict the 20 quiz results, as follows:

```
predictTEST <- predict(modFitRF, newdata=testing)
predictTEST
```

```
##  [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
```