Machine Learning Project

Project Introduction

Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and try to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We will compare the accuracy of the models and choose the best model to apply on the given test set; the obtained results will be chosen as answers to the multiple choice quiz.

Data

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

```r
#set library path
.libPaths("D:/R/R-3.5.0/library")

#Load Packages
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(knitr)
```

Load the training and testing data.

```r
#LOAD DATA
pml_training_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
pml_testing_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

#read data and replace the empty/non existing cells with NA
pml_training <- read.csv(url(pml_training_url), na.strings=c("NA","#DIV/0!",""))
pml_testing <- read.csv(url(pml_testing_url), na.strings=c("NA","#DIV/0!",""))

rm(pml_training_url,pml_testing_url)
```

Split the pml_training data into train data & test data; our outcome is the "classe" variable.

pml_training: we will allocate 60% of to be our training data and the rest; 40% to be our test data **pml_testing:** we will apply our selected best prediction model on this data and obtain the results we need to insert later in the multiple choice quiz.

```r
inTrain <- createDataPartition(pml_training$classe, p=0.6, list=FALSE) # 60% training and 40% testing
my_training <- pml_training[inTrain, ]
my_testing <- pml_training[-inTrain, ]
```

```r
dim(my_training);
```

```
## [1] 11776   160
```

```r
dim(my_testing)
```

```
## [1] 7846  160
```

```r
rm(inTrain)
```

Cleaning data

```r
# Remove the extra first column; equivalent to row names.
my_training <- my_training[c(-1)]


#Remove predictors that have one unique value (i.e: zero variance predictors) or those that satisfy the
#1. Very few unique values relative to the number of samples
#2. Ratio of the frequency of the most common value to the frequency of the second most common value is
nzv <- nearZeroVar(my_training, saveMetrics=TRUE)
my_training <- my_training[,nzv$nzv==FALSE]


nzv<- nearZeroVar(my_testing,saveMetrics=TRUE)
my_testing <- my_testing[,nzv$nzv==FALSE]



# Clean variables with more than 70% NA
index_to_rm <- NULL
for (i in 1:ncol(my_training))
{
if  ( length(which(is.na(my_training[,i])==T)) / nrow(my_training) >= 0.7 )
index_to_rm <- append(index_to_rm, i)
}

my_training <- my_training[,-index_to_rm]

rm(i, index_to_rm,nzv)




# Specify which columns to include in each of pml_testing & my_testing data frames for consistency with
columns_to_include_in_mytesting <- colnames(my_training)          # we will classe variable to build th
columns_to_include_in_pmltesting <- colnames(my_training[, -58])  # since the classe variable doesn't e

my_testing <- my_testing[columns_to_include_in_mytesting]         # allow only variables in mytesting t
pml_testing <- pml_testing[columns_to_include_in_pmltesting]      # allow only variables in pml_testing


rm(columns_to_include_in_mytesting,columns_to_include_in_pmltesting)
```

```r
#  print the dimensions of my_testing and pml_testing
dim(my_testing)
```

```
## [1] 7846   58
```

```r
dim(pml_testing)
```

```
## [1] 20 57
```

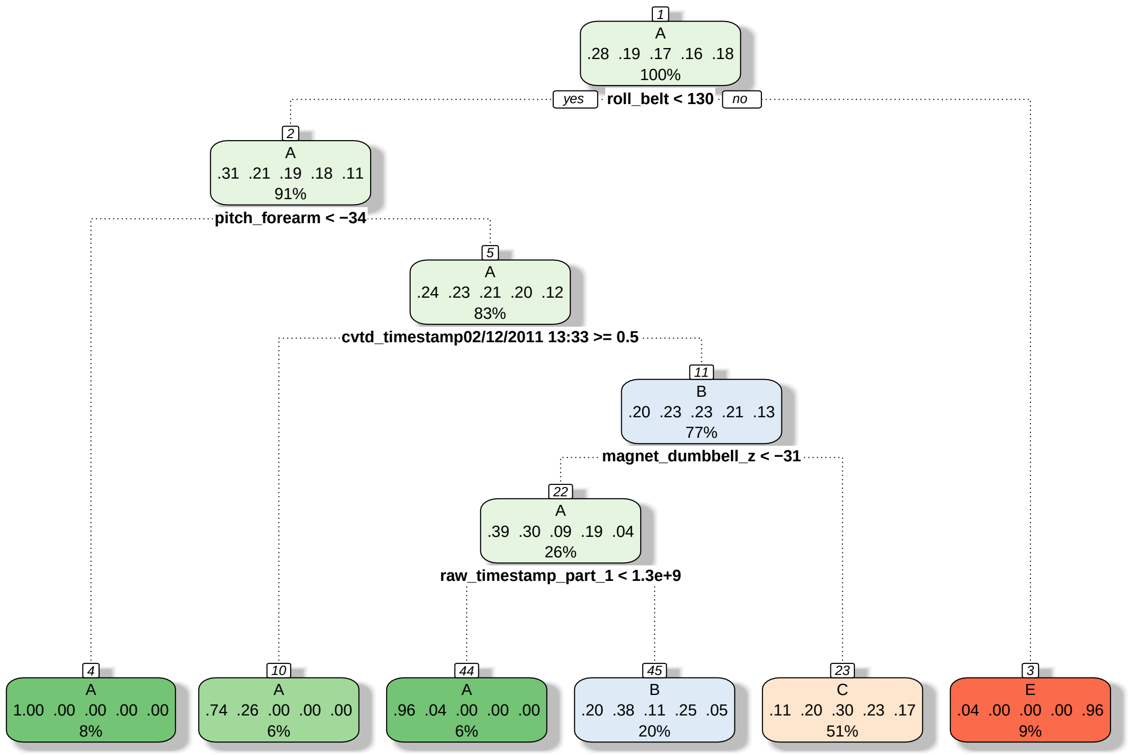
```r
#  Unifying data structures between my_training, my_testing and pml_testing by converting all integer v



# To get the same class between testing and mytraining
my_testing <- rbind(my_training[2, ] , my_testing)
my_testing <- my_testing[-1,]



# To get the same class between testing and mytraining
pml_testing <- rbind(my_training[2, -58] , pml_testing)
pml_testing <- pml_testing[-1,]
```

Prediction

We will experiment with 3 prediction models and choose the one that gives us the highest accuracy and apply it on plm_testing in order to obtain our results for the quiz section.

Decision Trees

```r
set.seed(1)
model_1<- train(classe ~ ., data=my_training, method="rpart") #use all variables to predict the classe
fancyRpartPlot(model_1$finalModel)
```



Rattle 2019−Jan−16 16:21:55 RDajani

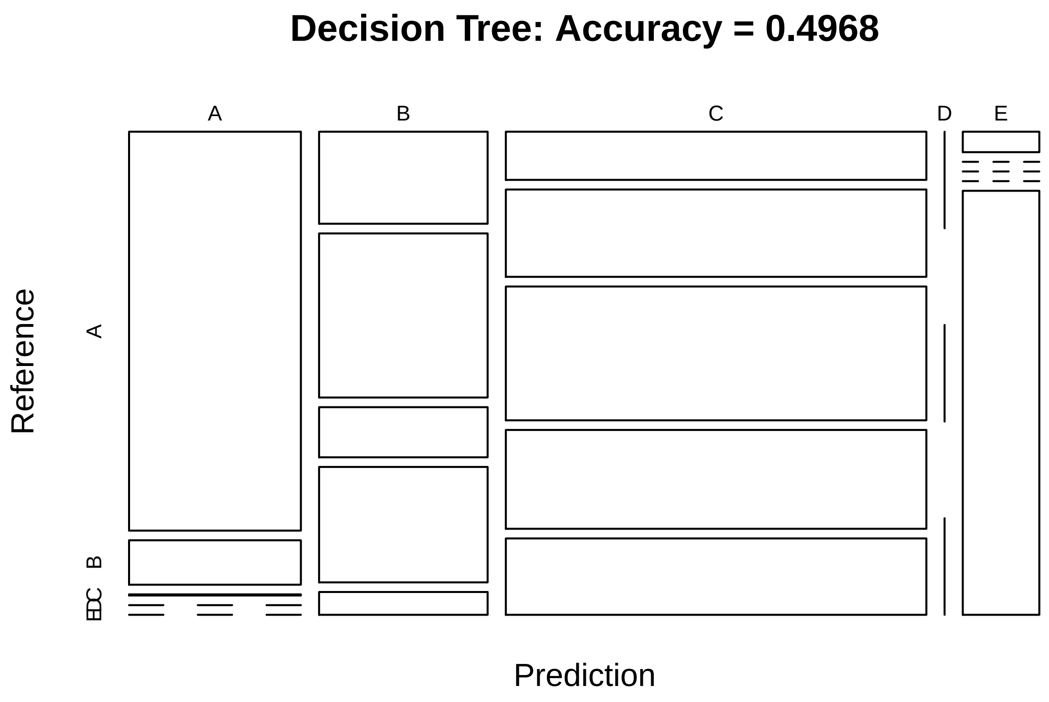
In order to test accuracy apply the model on the test set: my_testing

```
predictions1 <- predict(model_1, my_testing) #apply model on my_testing

cmtree <- confusionMatrix(predictions1, my_testing$classe)     #insepect accuracy and other statistics v
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1445  161    4    0    0
##          B  327  583  178  410   81
##          C  427  774 1186  876  677
##          D    0    0    0    0    0
##          E   33    0    0    0  684
##
## Overall Statistics
##
##                Accuracy : 0.4968
##                  95% CI : (0.4857, 0.5079)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3697
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6474  0.38406   0.8670   0.0000  0.47434
## Specificity            0.9706  0.84260   0.5749   1.0000  0.99485
## Pos Pred Value         0.8975  0.36922   0.3010      NaN  0.95397
## Neg Pred Value         0.8738  0.85081   0.9534   0.8361  0.89367
## Prevalence             0.2845  0.19347   0.1744   0.1639  0.18379
## Detection Rate         0.1842  0.07431   0.1512   0.0000  0.08718
## Detection Prevalence   0.2052  0.20125   0.5022   0.0000  0.09138
## Balanced Accuracy      0.8090  0.61333   0.7209   0.5000  0.73459
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree: Accuracy =", round(cmtree$overall
```

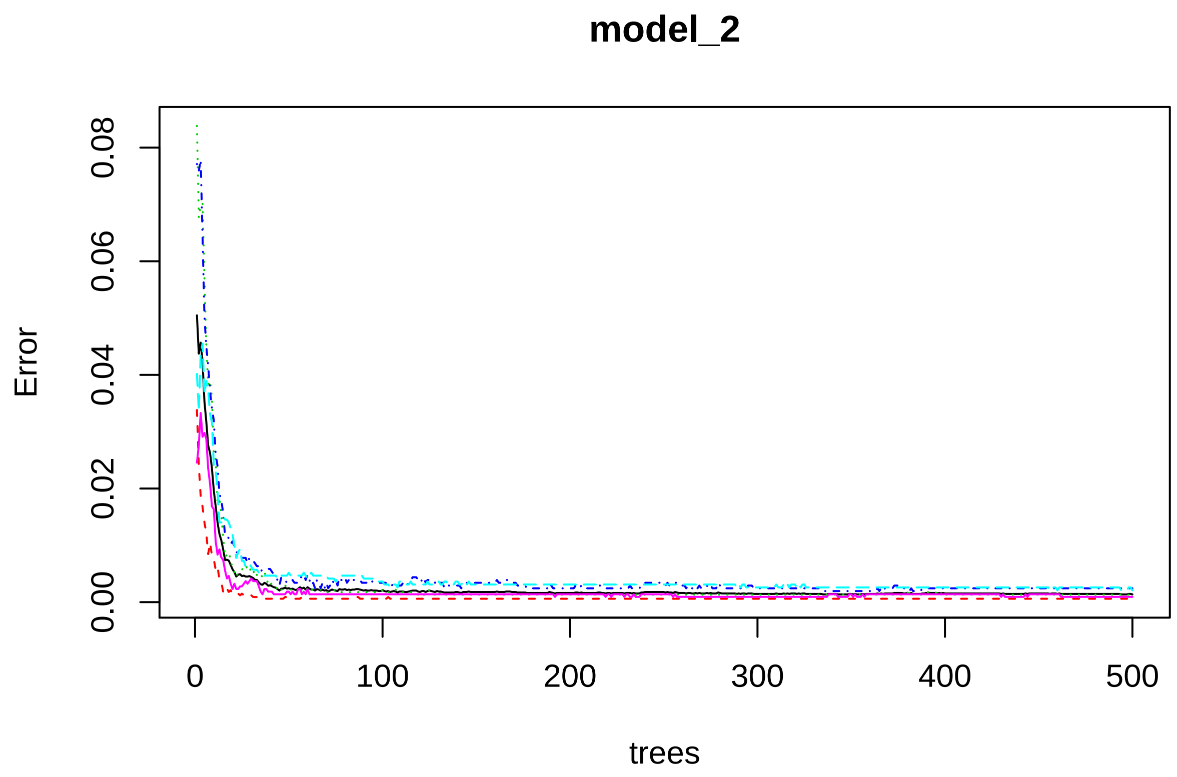


Random Forests

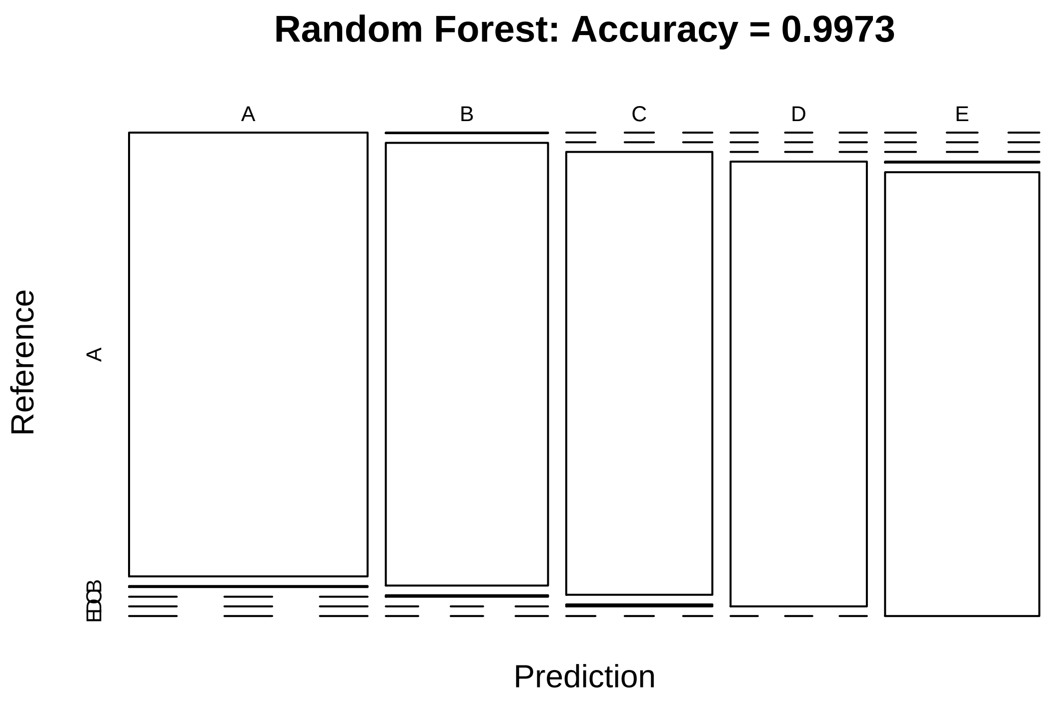
```
set.seed(1)

model_2 <- randomForest(classe ~ ., data=my_training)
prediction2 <- predict(model_2, my_testing, type = "class")
cmrf <- confusionMatrix(prediction2, my_testing$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    5    0    0    0
##          B    2 1513    5    0    0
##          C    0    0 1363    6    0
##          D    0    0    0 1277    0
##          E    0    0    0    3 1442
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9959, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9967   0.9963   0.9930   1.0000
## Specificity            0.9991   0.9989   0.9991   1.0000   0.9995
## Pos Pred Value         0.9978   0.9954   0.9956   1.0000   0.9979
## Neg Pred Value         0.9996   0.9992   0.9992   0.9986   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1928   0.1737   0.1628   0.1838
## Detection Prevalence   0.2849   0.1937   0.1745   0.1628   0.1842
## Balanced Accuracy      0.9991   0.9978   0.9977   0.9965   0.9998
```

```r
plot(model_2)
```



```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest: Accuracy =", round(cmrf$overall['Ac
```

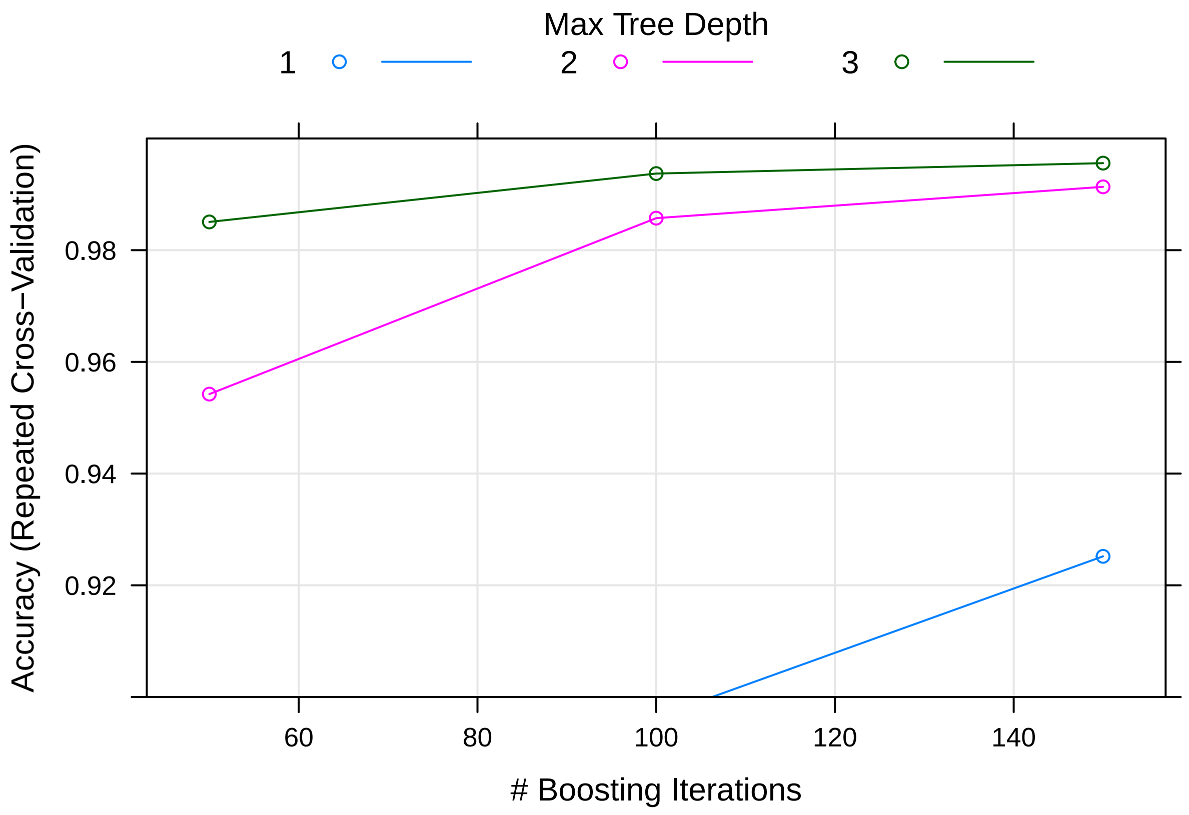


Boosting with Trees

```r
set.seed(1)


fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

model_3 <- train(classe ~ ., data=my_training, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)



prediction3 <- predict(model_3, newdata=my_testing)
cmGBM <- confusionMatrix(prediction3, my_testing$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    8    0    0    0
##          B    1 1507    2    0    0
##          C    0    3 1362    5    0
##          D    0    0    4 1279    3
##          E    0    0    0    2 1439
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.9948, 0.9976)
##     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            0.9996   0.9928   0.9956   0.9946   0.9979
## Specificity            0.9986   0.9995   0.9988   0.9989   0.9997
## Pos Pred Value         0.9964   0.9980   0.9942   0.9946   0.9986
## Neg Pred Value         0.9998   0.9983   0.9991   0.9989   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1921   0.1736   0.1630   0.1834
## Detection Prevalence   0.2854   0.1925   0.1746   0.1639   0.1837
## Balanced Accuracy      0.9991   0.9961   0.9972   0.9967   0.9988
```

```
plot(model_3, ylim=c(0.9, 1))
```



Selecting the best model to apply on the Test Data

By comparing the accuracy among the 3 models, we notice that the Random Forests model gave us the highest accuracy 99.81%; thus the expected out-of-sample error is 100-99.81 = 0.19%. We choose RF to apply on the test data: plm_testing. The obtained results are the ones to be selected as answers to the multiple choice quiz.

```
prediction_model_2 <- predict(model_2, pml_testing, type = "class")
prediction_model_2
```

```
##  1  2 31  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
```