Practical Machine Learning Assignment

A. Heflin October 17, 2017

## Introduction

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har>

## The 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>

## Loading the Data

# First we will load the packages  
library(knitr)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)  
library(rpart.plot)  
#library(rattle)  
library(randomForest)

## 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(corrplot)

## corrplot 0.84 loaded

set.seed(12345)  
  
# set the URL  
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
UrlTest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
  
# download the datasets  
training <- read.csv(url(UrlTrain))  
testing <- read.csv(url(UrlTest))

## Cleaning the Data

We will create a partition in the Training dataset (70%) for modeling and a Test ste (30%) for validation.

inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)  
TrainSet <- training[inTrain, ]  
TestSet <- training[-inTrain, ]  
dim(TrainSet)

## [1] 13737 160

dim(TestSet)

## [1] 5885 160

# remove Nearly Zero and NA variables  
NZV <- nearZeroVar(TrainSet)  
TrainSet <- TrainSet[, -NZV]  
TestSet <- TestSet[, -NZV]  
dim(TrainSet)

## [1] 13737 106

dim(TestSet)

## [1] 5885 106

AllNA <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95  
TrainSet <- TrainSet[, AllNA==FALSE]  
TestSet <- TestSet[, AllNA==FALSE]  
dim(TrainSet)

## [1] 13737 59

dim(TestSet)

## [1] 5885 59

# remove identification only variables (columns 1 to 5)  
TrainSet <- TrainSet[, -(1:5)]  
TestSet <- TestSet[, -(1:5)]  
dim(TrainSet)

## [1] 13737 54

dim(TestSet)

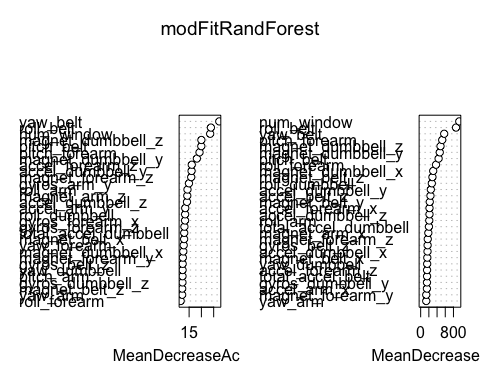
## [1] 5885 54

## Prediction Model Building

We will apply the following three methods to the Test dataset that will be used for the quiz predictions: Random Foreset, Decision Tree and Generalized Boosted Model.

#### Random Forest

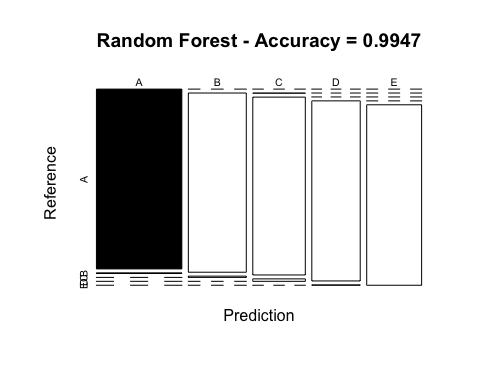
set.seed(12345)  
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)  
modFitRandForest <- randomForest(classe ~ ., data=TrainSet, importance=TRUE, ntree=100)  
varImpPlot(modFitRandForest)



predictRandForest <- predict(modFitRandForest, newdata=TestSet)  
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)  
confMatRandForest

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1674 6 0 0 0  
## B 0 1132 8 0 0  
## C 0 1 1018 14 0  
## D 0 0 0 950 2  
## E 0 0 0 0 1080  
##   
## Overall Statistics  
##   
## Accuracy : 0.9947   
## 95% CI : (0.9925, 0.9964)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9933   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9939 0.9922 0.9855 0.9982  
## Specificity 0.9986 0.9983 0.9969 0.9996 1.0000  
## Pos Pred Value 0.9964 0.9930 0.9855 0.9979 1.0000  
## Neg Pred Value 1.0000 0.9985 0.9984 0.9972 0.9996  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2845 0.1924 0.1730 0.1614 0.1835  
## Detection Prevalence 0.2855 0.1937 0.1755 0.1618 0.1835  
## Balanced Accuracy 0.9993 0.9961 0.9946 0.9925 0.9991

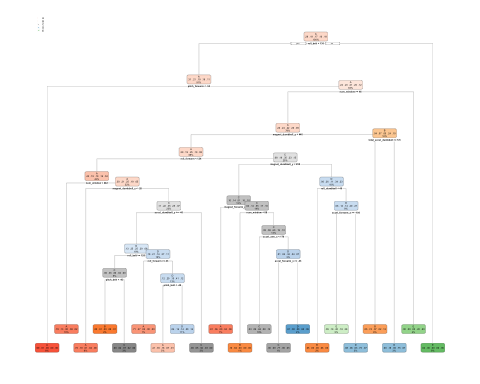
plot(confMatRandForest$table, col = confMatRandForest$byClass,   
 main = paste("Random Forest - Accuracy =",  
 round(confMatRandForest$overall['Accuracy'], 4)))



#### Decision Tree

set.seed(12345)  
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")  
rpart.plot(modFitDecTree)

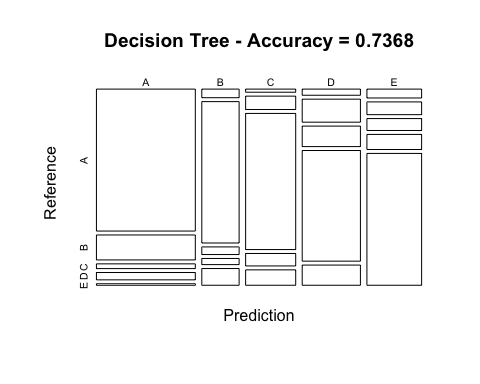
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")  
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)  
confMatDecTree

## 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(confMatDecTree$table, col = confMatDecTree$byClass,   
 main = paste("Decision Tree - Accuracy =",  
 round(confMatDecTree$overall['Accuracy'], 4)))



#### Generalized Boosted Model

set.seed(12345)  
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)  
modFitGBM <- train(classe ~ ., data=TrainSet, method = "gbm",  
 trControl = controlGBM, verbose = FALSE)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

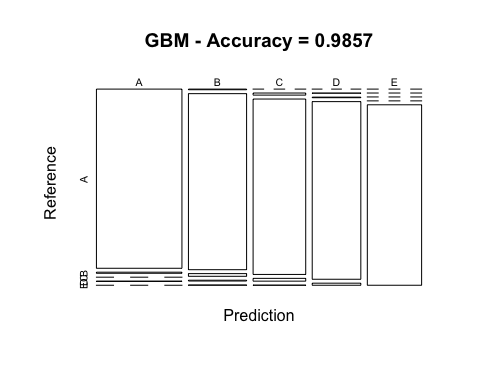
modFitGBM$finalModel

## 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=TestSet)  
confMatGBM <- confusionMatrix(predictGBM, TestSet$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(confMatGBM$table, col = confMatGBM$byClass,   
 main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))



## Apply the Selected Model to the Test Data

We will apply the Random Forest model to the quiz because, although all models were accurate, Random Forest model was the most accurate: Random Forest : 0.9963 Decision Tree : 0.7368 GBM : 0.9839

predictTEST <- predict(modFitRandForest, newdata=testing)  
predictTEST

## 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