Practical Machine Learning - Project

karunesh bhardwaj

October 14, 2016

1. Overview

This paper is the final report of the project from Coursera's course Practical Machine Learning, which is part of Data Science specialization track. This report was created using RStudio, using markdown file function knitr and published in html format. The outcome of this analysis/study is to predict the course quiz questions. This is described by the variable "classe" in training set. The machine learning algorithm explained here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

2. Background

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 (see the section on the Weight Lifting Exercise Dataset).

3. Data Loading and Exploratory Analysis

i) 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

ii) Prepare Environment

Populate and load all R libraries

```r
rm (list=ls())  # clean up memory
setwd("C:/Rproject/kb/MachineLearning")
library(knitr)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(rpart.plot)
library(rattle)

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

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)
set.seed(12000)
```

iii) Data Loading and Cleaning

```r
# Configure URL for data download

kbTrainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
kbTestUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

# Download the data

kbTraining <- read.csv(url(kbTrainUrl))
kbTesting <- read.csv(url(kbTestUrl))

# create a partition with the training dataset

kbinTraining <- createDataPartition(kbTraining$classe, p=0.7, list=FALSE)
kbTrainSet <- kbTraining[kbinTraining, ]
```

```r
kbTestSet <- kbTraining[-kbinTraining, ]
dim(kbTrainSet)

## [1] 13737   160

dim(kbTestSet)

## [1] 5885  160
```

Both kbTrainSet and kbTestSet have 160 variables. Any NA in these variable can be cleaned by using following procedure. The Near Zero Variance (NZV) and ID variables are also removed.

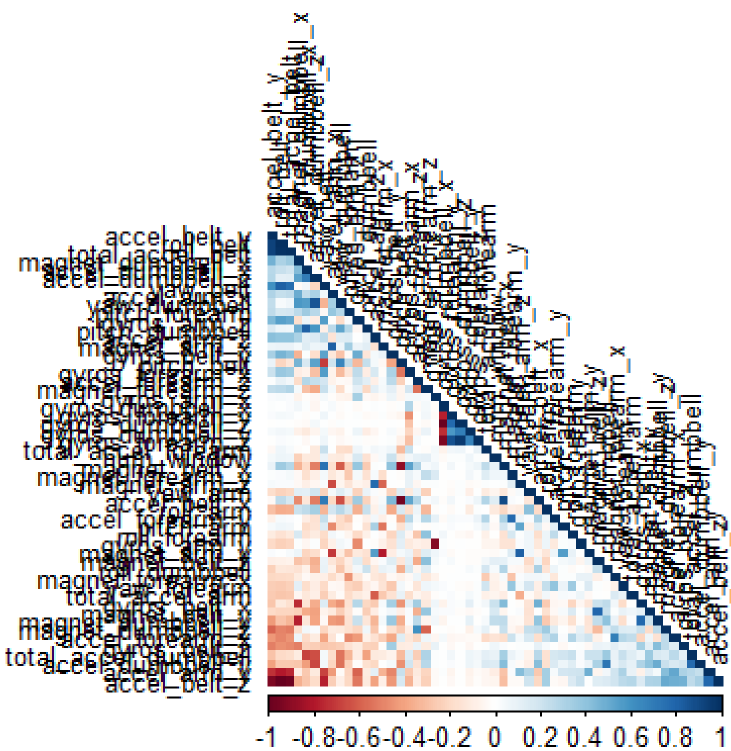
```r
# remove Near Zero Variance variables

NVZ <- nearZeroVar(kbTrainSet)
kbTrainSet <- kbTrainSet[, -NVZ]
kbTestSet <- kbTestSet[, -NVZ]
dim(kbTrainSet)

## [1] 13737   101

dim(kbTestSet)

## [1] 5885  101

# remove any NA variables

NAAny <- sapply(kbTrainSet, function(x) mean(is.na(x))) > 0.95
kbTrainSet <- kbTrainSet[, NAAny==FALSE]
kbTestSet <- kbTestSet[, NAAny==FALSE]
dim(kbTrainSet)

## [1] 13737    59

dim(kbTestSet)

## [1] 5885   59

# remove ID variables (col 1 to 5)
kbTrainSet <- kbTrainSet[, -(1:5)]
kbTestSet <- kbTestSet[, -(1:5)]
dim(kbTrainSet)

## [1] 13737    54

dim(kbTestSet)

## [1] 5885   54
```

By above cleaning process, the total number of variable for the analysis has been reduced to 54 variables.

iv) Correlation Analysis

Before proceeding to the modeling procedure, a correlation between variables must be analysed.

```
kbCorMatrix <- cor(kbTrainSet[, -54])

corrplot(kbCorMatrix, order = "FPC", method = "color", type = "lower", tl.cex
= 0.8, tl.col = rgb(0, 0, 0))
```



Dark color in the grap shows highly correlated variables, which are very few. Therefore, there is no need to preprocess the data with Principal Components Analysis (PCA).

4. Building Prediction Model

Three methods will be applied to model the regressions to the Training Dataset and the best one, which give higher accuracy will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model.

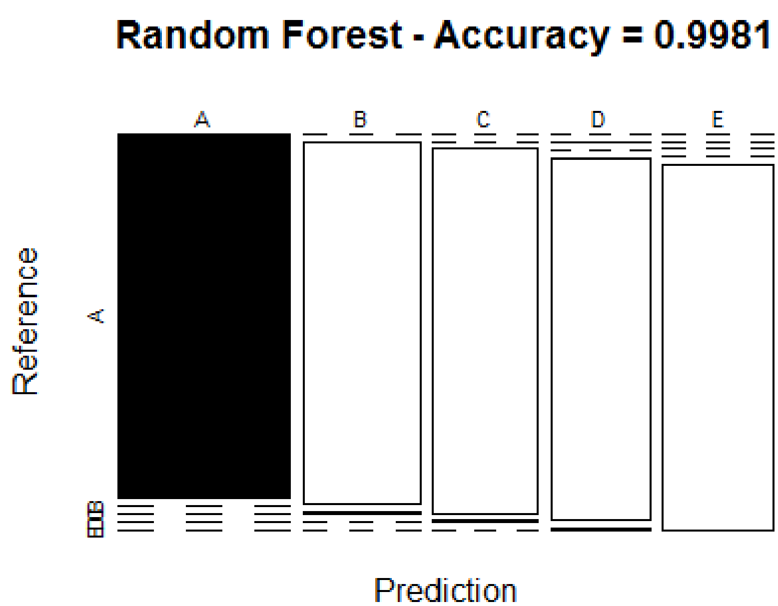
i) Random Forest Method

```
# Fit Model
```

```
set.seed(12000)
RFcontrol <- trainControl(method="cv", number=3, verboseIter = FALSE)
FitRmodandForest <- train(classe ~ ., data=kbTrainSet, method="rf",
trControl=RFcontrol)
FitRmodandForest$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 3905    0    0    0    1 0.0002560164
## B    5 2651    2    0    0 0.0026335591
## C    0    5 2391    0    0 0.0020868114
## D    0    0    7 2244    1 0.0035523979
## E    0    1    0    5 2519 0.0023762376

# prediction on Test dataset

RandForestprediction <- predict(FitRmodandForest, newdata=kbTestSet)
RandForestConfMatrix <- confusionMatrix(RandForestprediction,
kbTestSet$classe)
RandForestConfMatrix

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1138    5    0    0
##          C    0    0 1021    2    0
##          D    0    1    0  962    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

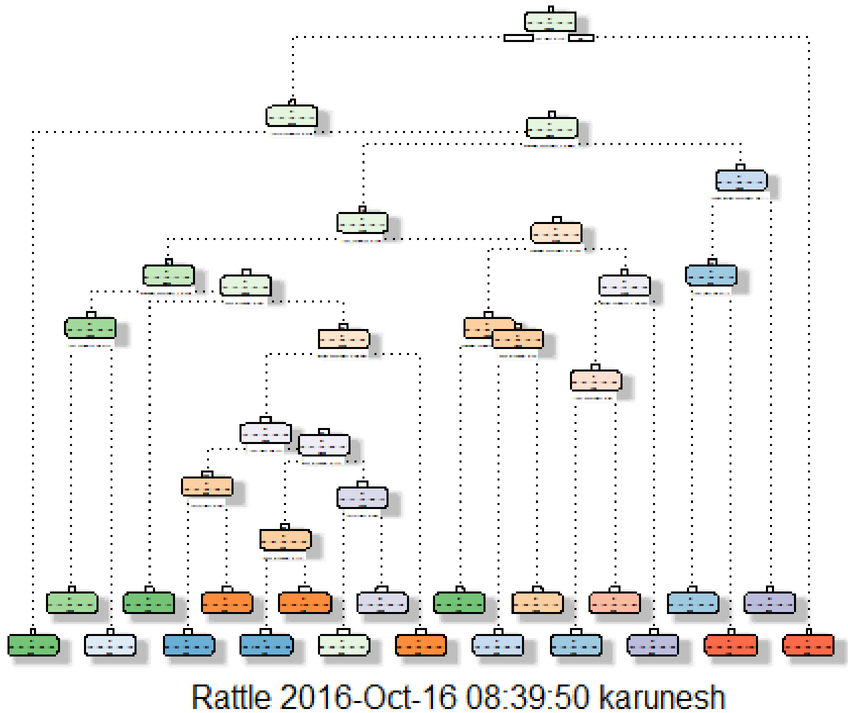
```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9951   0.9979   0.9972
## Specificity            1.0000   0.9989   0.9996   0.9992   1.0000
## Pos Pred Value         1.0000   0.9956   0.9980   0.9959   1.0000
## Neg Pred Value         1.0000   0.9998   0.9990   0.9996   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1735   0.1635   0.1833
## Detection Prevalence   0.2845   0.1942   0.1738   0.1641   0.1833
## Balanced Accuracy      1.0000   0.9990   0.9974   0.9986   0.9986
```

```
# Plot Matrix Result

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

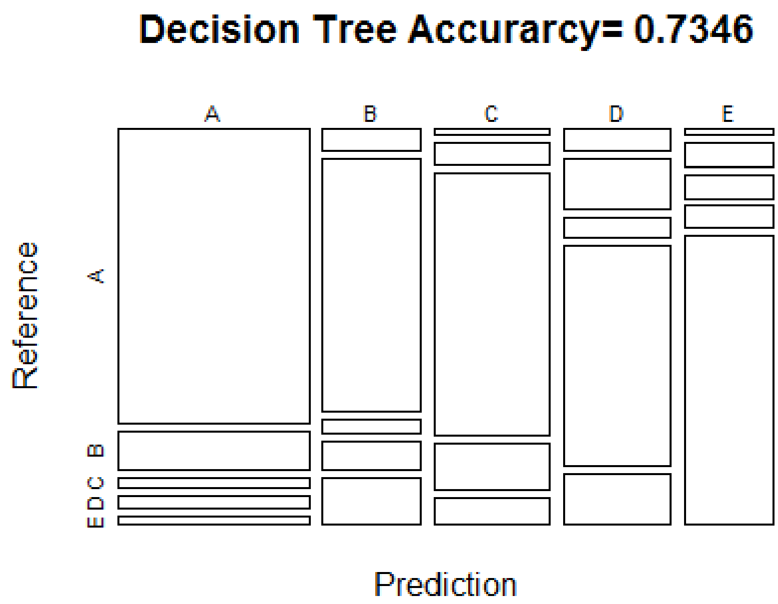


ii) Decisioin Tree Method

```
# Fit Model
set.seed(12000)
DecTreeFitModel <- rpart(classe ~ ., data = kbTrainSet, method="class")
fancyRpartPlot(DecTreeFitModel)

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



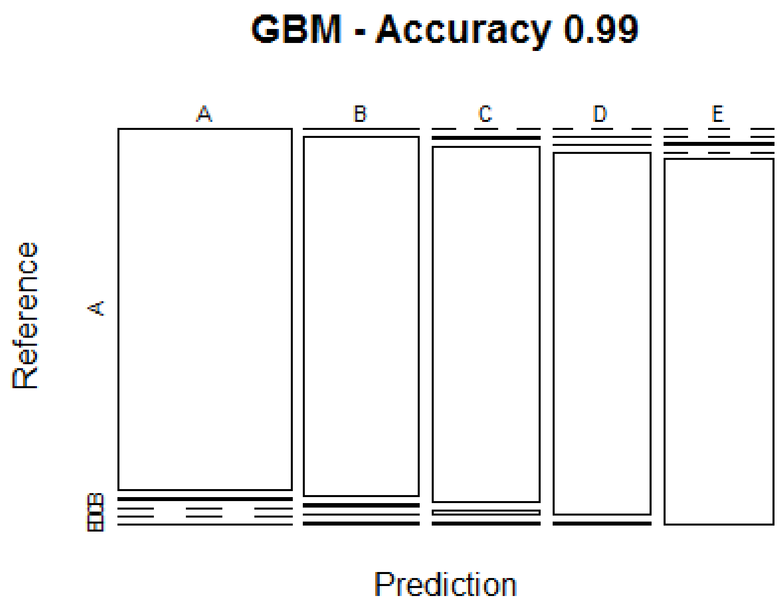

```r
# Test dataset prediction

DecTreePrediction <- predict(DecTreeFitModel, newdata = kbTestSet,
type="class")
DecTreeConfMatrix <- confusionMatrix(DecTreePrediction, kbTestSet$classe)

# Plot matrix results

plot(DecTreeConfMatrix$table, col = DecTreeConfMatrix$byClass, main =
paste("Decision Tree Accurarcy=",
round(DecTreeConfMatrix$overall['Accuracy'], 4)))
```



iii) Generalized Boosted Model Method

```
# Fit Model
set.seed(12000)
GBMControl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
GBMModFit <- train(classe ~ ., data=kbTrainSet, method = "gbm", trControl =
GBMControl, verbose = FALSE)

## Loading required package: gbm

## 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 2.1.1

## Loading required package: plyr

GBMModFit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 40 had non-zero influence.

# Test Data prediction

GMBPrediction <- predict(GBMModFit, newdata=kbTestSet)
GBMConfMatrix <- confusionMatrix(GMBPrediction, kbTestSet$classe)
GBMConfMatrix

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   11    0    0    2
##          B    3 1117    4    1    5
##          C    0   10 1020   10    4
##          D    0    1    1  953    6
##          E    0    0    1    0 1065
##
## Overall Statistics
##
##                Accuracy : 0.99
##                  95% CI : (0.9871, 0.9924)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9873
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9807   0.9942   0.9886   0.9843
## Specificity            0.9969   0.9973   0.9951   0.9984   0.9998
## Pos Pred Value         0.9923   0.9885   0.9770   0.9917   0.9991
## Neg Pred Value         0.9993   0.9954   0.9988   0.9978   0.9965
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1898   0.1733   0.1619   0.1810
## Detection Prevalence   0.2862   0.1920   0.1774   0.1633   0.1811
## Balanced Accuracy      0.9976   0.9890   0.9946   0.9935   0.9920

# Matril result plot

plot(GBMConfMatrix$table, col=GBMConfMatrix$byClass, main = paste("GBM -
Accuracy", round(GBMConfMatrix$overall['Accuracy'], 4)))
```



5. Applying the Selected Model to the Test Data set

The accuracy of the Three regression methods above are

i) Radom Forest = 0.9981
ii) Decision Tree = 0.7346
iii) GBM = 0.99

The Random Forest model will be applied to predict the 20 quiz results as below:

```
TestPrediction <- predict(FitRmodandForest, newdata = kbTesting)
TestPrediction

##  [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
```