Practical Machine Learning Project

Karuna Raghuwanshi

04/05/2020

Overview

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.

The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set.Three machine learning models are 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.

Data Loading and Cleaning

The next step is loading the dataset from the URL provided above. The training dataset is then partinioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset will be used for the quiz results generation.

```
# Loading the Libraries
rm(list=ls())
library(knitr)
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)

## Warning: package 'rpart' was built under R version 3.6.3

library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3

library(rattle)

## Warning: package 'rattle' was built under R version 3.6.3

## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

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

## 
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
## 
##     importance

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

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded

library(gbm)

## Warning: package 'gbm' was built under R version 3.6.3

## Loaded gbm 2.1.5

set.seed(12345)

# downloading the data
testing  <- read.csv("pml-testing.csv", head=TRUE)
training <- read.csv("pml-training.csv", head=TRUE)

# To create a partition with the training dataset

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

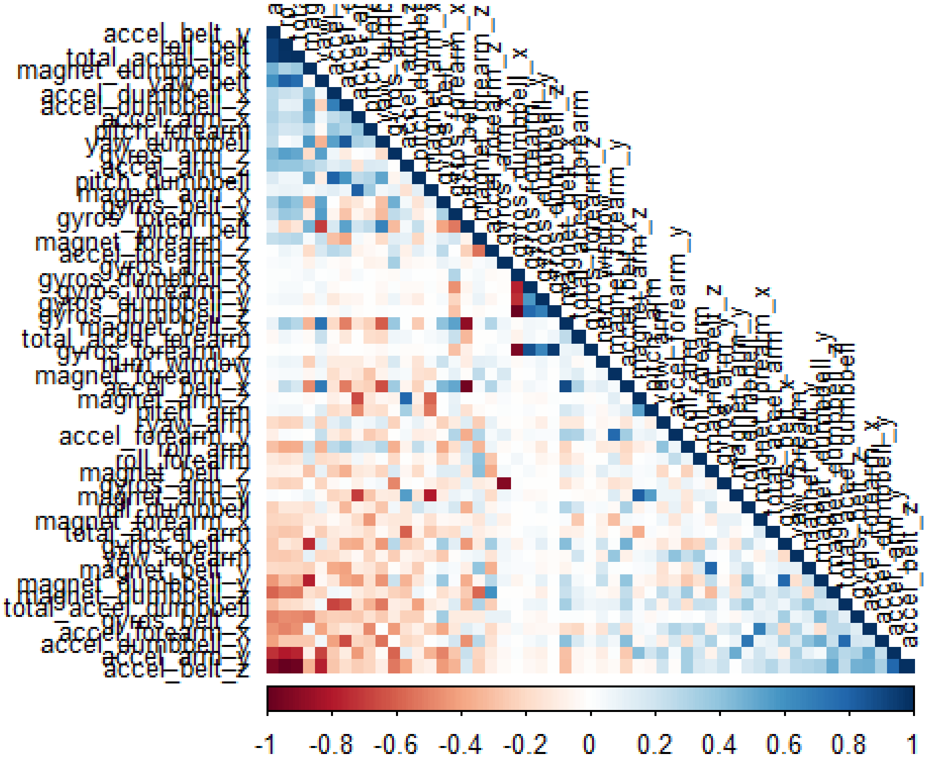
```
## [1] 13737   160

dim(TestSet)

## [1] 5885  160

# To remove variables with Nearly Zero Variance from the dataset

NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)

## [1] 13737   104

dim(TestSet)

## [1] 5885  104

# To remove variables that are mostly NA

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

## [1] 13737    59

dim(TestSet)

## [1] 5885   59

# To 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
```

After cleaning , the number of variables for the analysis has been reduced to 54 only.

Correlation Analysis

Before proceeding to the modeling procedures, a correlation among variables is analysed

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



This plot shows highly correlated variables with dark colors.

Prediction Model Building

Below three models will be applied to model the regressions (in the Train dataset) 1. Random Forests 2. Decision Tree 3. Generalized Boosted Model

A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models. The model with higher accuracy(Best Fit) will be applied to the Test dataset

1. Random Forest Model

```
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
Model_RF <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
Model_RF$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: 0.23%
## Confusion matrix:
```

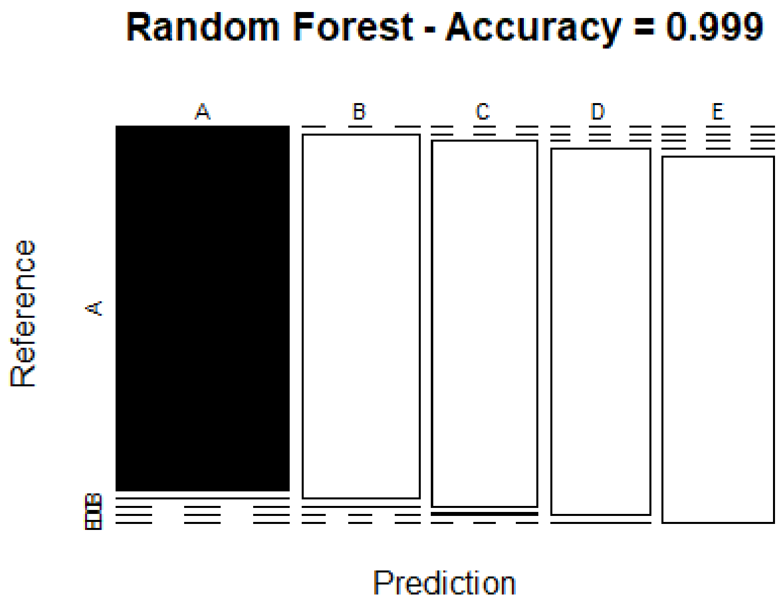
```
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B    6 2647    4    1    0 0.0041384500
## C    0    5 2391    0    0 0.0020868114
## D    0    0    9 2243    0 0.0039964476
## E    0    0    0    5 2520 0.0019801980
```

prediction on Test dataset

```
predictRandForest <- predict(Model_RF, newdata=TestSet)
Confusion_Mat_RF <- confusionMatrix(predictRandForest, TestSet$classe)
Confusion_Mat_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9979   0.9991
## Specificity            0.9998   0.9996   0.9996   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1837
## Detection Prevalence   0.2846   0.1937   0.1743   0.1636   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9989   0.9995
```

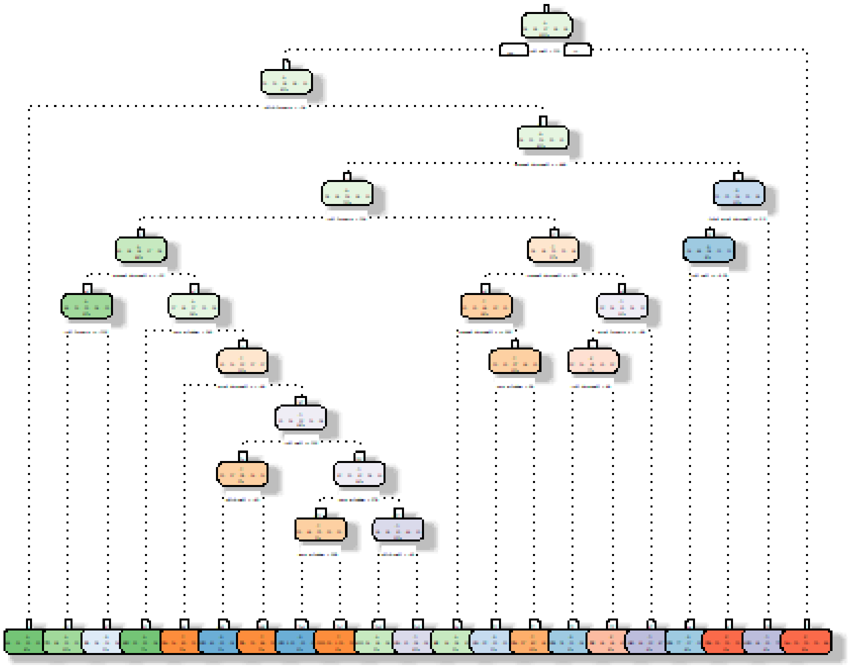
```
plot(Confusion_Mat_RF$table, col = Confusion_Mat_RF$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(Confusion_Mat_RF$overall['Accuracy'], 4)))
```



2. Decision Trees Model

```
set.seed(12345)
Model_DecisionTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(Model_DecisionTree)
```

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



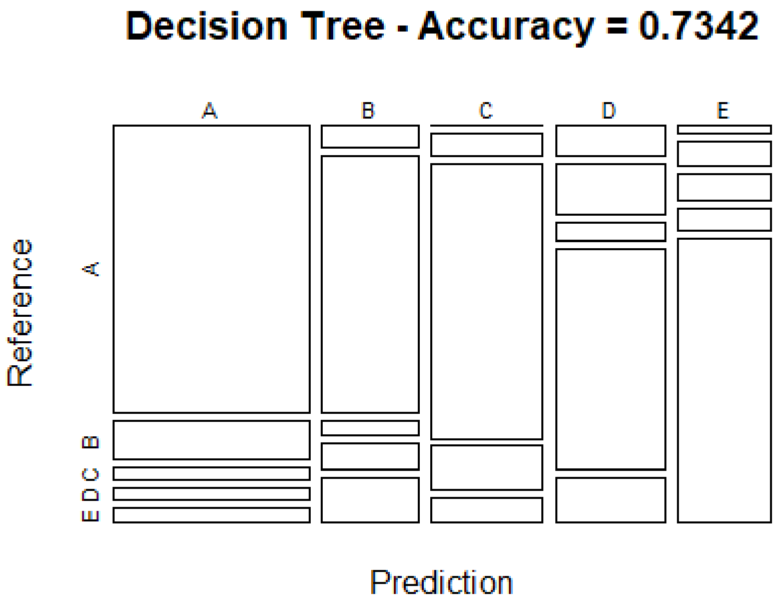
Rattle 2020-May-06 19:33:33 vivek raghuwanshi

prediction on Test dataset

```
predictDecTree <- predict(Model_DecisionTree, newdata=TestSet, type="class")
Confusion_Mat_DT <- confusionMatrix(predictDecTree, TestSet$classe)
Confusion_Mat_DT

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  201   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## Overall Statistics
##
##                Accuracy : 0.7342
##                  95% CI : (0.7228, 0.7455)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973   0.5795   0.7943   0.6722   0.6433
## Specificity            0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value         0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value         0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence   0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy      0.9011   0.7610   0.8693   0.7936   0.8006
```

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



3. Generalized Boosted Model

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

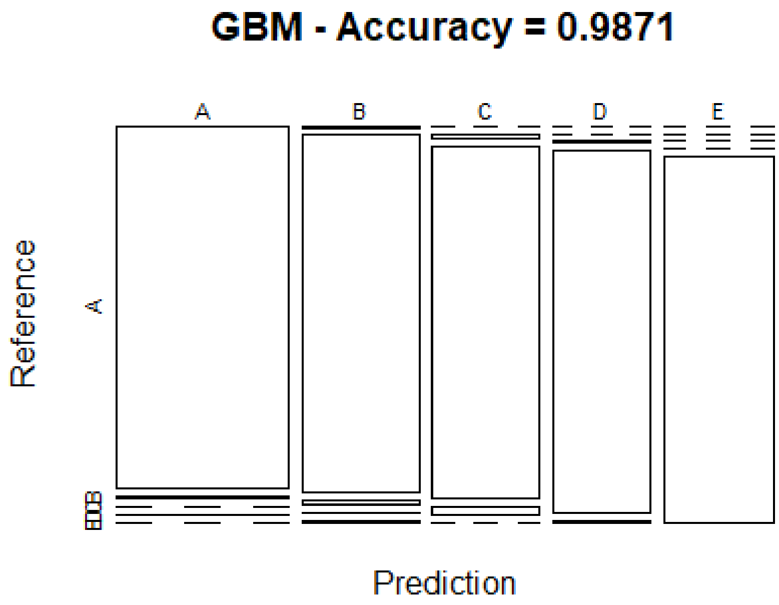
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

prediction on Test dataset

```
predictGBM <- predict(Model_GB, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   12    0    1    0
##          B    6 1115   12    1    3
##          C    0   12 1012   21    0
##          D    0    0    2  941    6
##          E    0    0    0    0 1073
## 
## Overall Statistics
## 
##                Accuracy : 0.9871
##                  95% CI : (0.9839, 0.9898)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9837
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9789   0.9864   0.9761   0.9917
## Specificity            0.9969   0.9954   0.9932   0.9984   1.0000
## Pos Pred Value         0.9923   0.9807   0.9684   0.9916   1.0000
## Neg Pred Value         0.9986   0.9949   0.9971   0.9953   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1895   0.1720   0.1599   0.1823
## Detection Prevalence   0.2856   0.1932   0.1776   0.1613   0.1823
## Balanced Accuracy      0.9967   0.9871   0.9898   0.9873   0.9958
```

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



Best Model to test data

The accuracy of the 3 regression modeling methods above are:

Random Forest : 0.9963 Decision Tree : 0.7368 GBM : 0.9839 Hence, the Random Forest model will be applied to predict testing dataset:

```
predictTEST <- predict(Model_RF, 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
```