Course Project Practical ML

AryaVa

9/22/2020

OVERVIEW

This is the final course project of the course "Practical Machine Learning". The machine learning algorithm described in this project will be used to predict the outcomes of 20 test cases in the test data. 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. We will use the data of 6 participants in this project to find the manner in which they performed the exercises as described.

DATA PRE-PROCESSING: DATA ACQUISITION AND EXPLORATORY ANALYSIS

a) Dataset summary

The training dataset for the assignment is available at: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The testing dataset is available at: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv The data for the entire project is available at: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har The data used is a courtesy of: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H.: Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013. In this project, the 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 goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We may use any of the other variables to predict with.

b) Loading the libraries and setting up the environment

In this part of the analysis, we first clear up the memory space to download the data, load the required libraries and set the seed.

```
knitr::opts_chunk$set(echo = TRUE)
## Free up memory to download datasets
rm(list=ls())
## Load the required libraries
library(knitr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

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

## Warning: package 'rpart.plot' was built under R version 4.0.2

library(rattle)

## Warning: package 'rattle' was built under R version 4.0.2

## Loading required package: tibble

## Warning: package 'tibble' was built under R version 4.0.2

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.0.2

## 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 4.0.2

## corrplot 0.84 loaded

library(RColorBrewer)
library(gbm)

## Warning: package 'gbm' was built under R version 4.0.2

## Loaded gbm 2.1.8

library(e1071)

## Warning: package 'e1071' was built under R version 4.0.2

## Set the seed
set.seed(12345)
```

c) Getting and Cleaning data and Creating test dataset for Cross Validation

Download and read the training and testing datasets using the URLs. Create a partition in the training dataset in the ratio 70:30 (70% for modelling the data and 30% for cross validation). The testing dataset will be used for quiz and is not changed in any way.

```
## Download and read the datasets
training<- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing<- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
## Create a partition in the training dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
## Create a training set and a testing set out of the downloaded training dataset
TrainSet <- training[inTrain, ] ## For modelling/training the algorithms
TestSet  <- training[-inTrain, ] ## For cross validation
## Check the dimensions of the newly created training and testing sets
dim(TrainSet)

## [1] 13737   160

dim(TestSet)

## [1] 5885  160
```

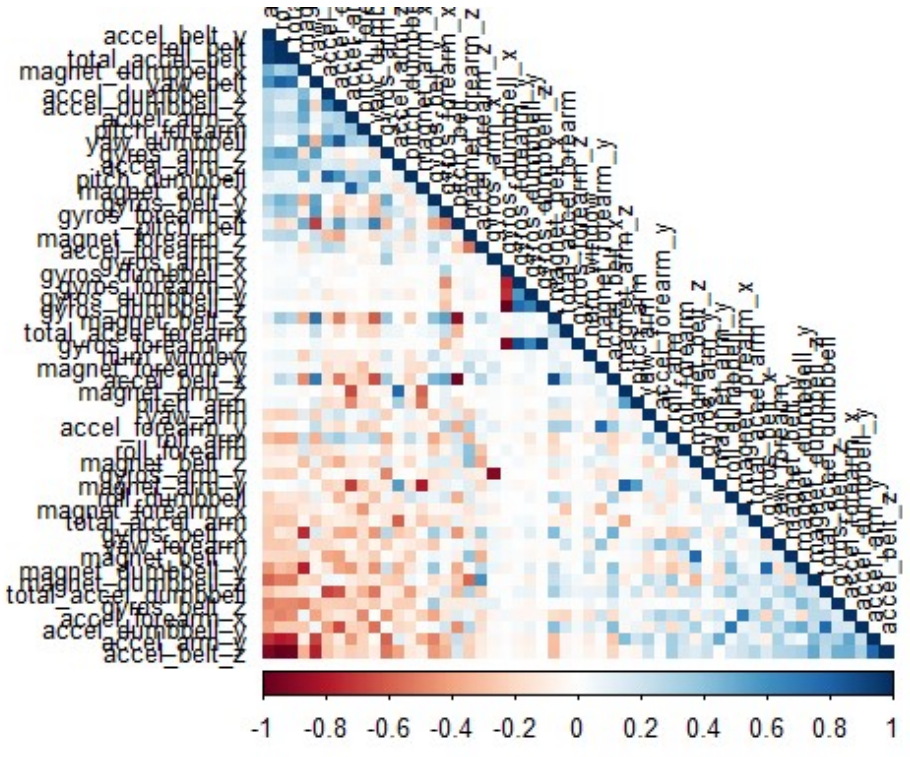
The newly created datasets have 160 variables each. We remove all the NA values, ID variables and the near zero variance variables.

```
## For removing near zero variance variables
nzv <- nearZeroVar(TrainSet)
TrainSet<- TrainSet[, -nzv]
TestSet<- TestSet[, -nzv]
dim(TrainSet)

## [1] 13737   104

dim(TestSet)

## [1] 5885  104

## For removing all the NA values
NAs<- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet<- TrainSet[, NAs==FALSE]
TestSet<- TestSet[, NAs==FALSE]
dim(TrainSet)

## [1] 13737    59

dim(TestSet)

## [1] 5885   59

## For removing the ID variables
TrainSet<- TrainSet[, -(1:5)]
TestSet<- TestSet[, -(1:5)]
dim(TrainSet)

## [1] 13737    54

dim(TestSet)

## [1] 5885   54
```

d) Correlation Analysis

It is important to analyze the correlation among all the variables before proceeding to prediction modelling procedures. The following code is used to do the same using the function “corrplot”.

```
Matrix1<- cor(TrainSet[, -54])
corrplot(Matrix1, order = "FPC", method = "color", type = "lower", tl.cex =
0.8, tl.col = rgb(0, 0, 0))
```



According to the graph, highly correlated variables are shown in dark colours and less correlated variables are shown in light colours.

BUILDING PREDICTION MODELS

Three prediction models, namely, (i) Random Forest, (ii) Classification/decision tree, (iii) Gneralized Boosted model, are used in this project for modelling the training set. The model with the highest accuracy will be applied to the test set which will be used for prediction quiz.

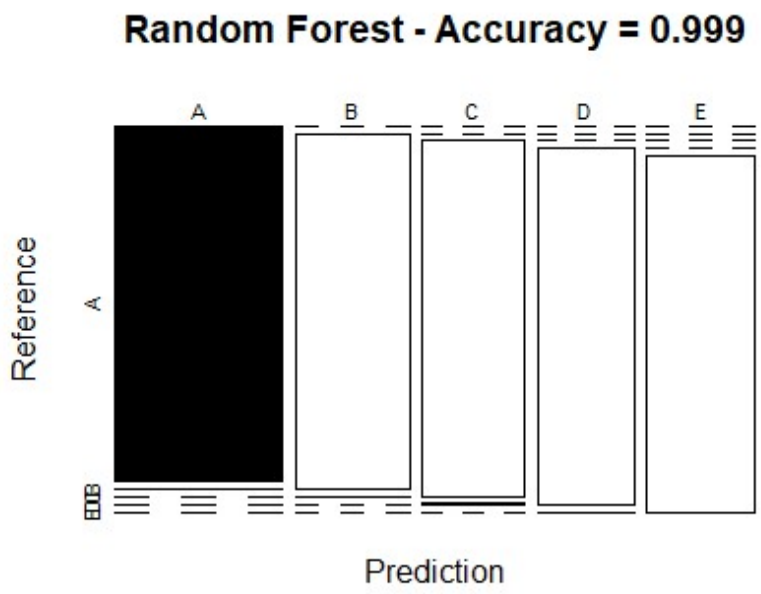
a) Random Forest

```
## model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRF <- train(classe ~ ., data=TrainSet, 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: 0.23%
## 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
```

```r
## prediction on Test dataset
predictRF <- predict(modFitRF, newdata=TestSet)
CMRF<- confusionMatrix(predictRF, as.factor(TestSet$classe))
CMRF
```

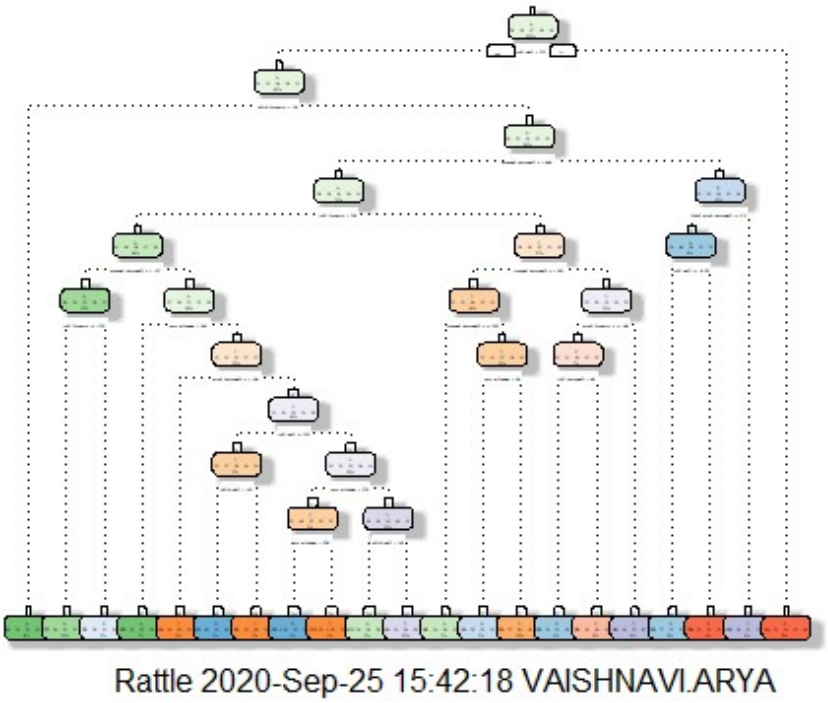
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## 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
```

```r
## plotting confusion matrix results
plot(CMRF$table, col = CMRF$byClass, main = paste("Random Forest - Accuracy
=", round(CMRF$overall['Accuracy'], 4)))
```



b) Classification Tree

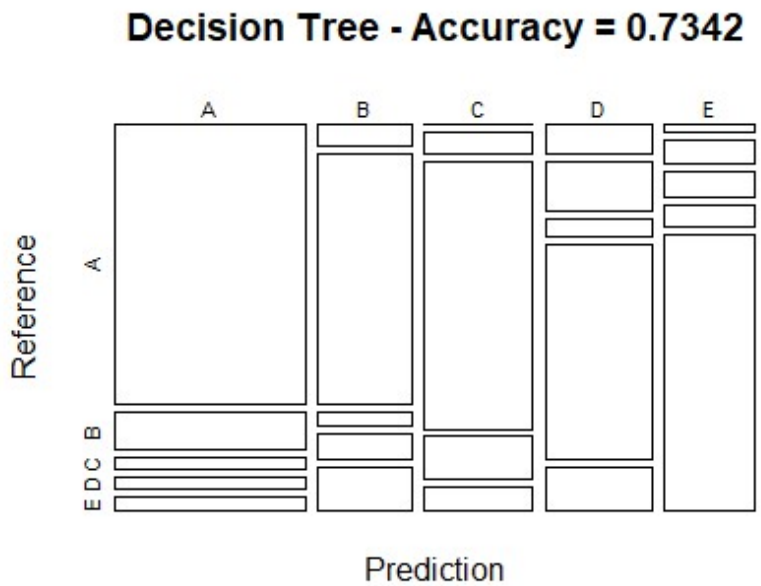
```
## model fit
set.seed(12345)
modFitCT <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitCT)

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




```
## prediction on Test dataset
predictCT <- predict(modFitCT, newdata=TestSet, type="class")
CMCT <- confusionMatrix(predictCT, as.factor(TestSet$classe))
CMCT
```

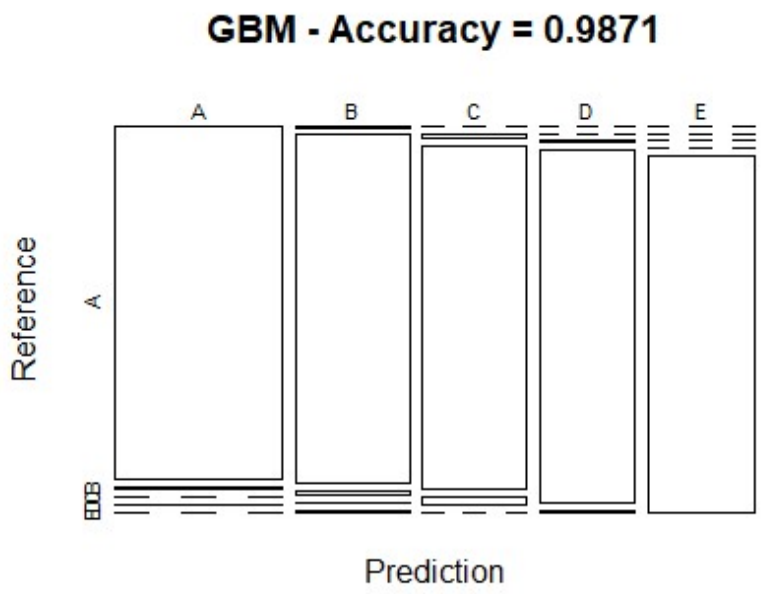
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  201   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## 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
```

```
## plotting confusion matrix results
plot(CMCT$table, col = CMCT$byClass, main = paste("Decision Tree - Accuracy
=", round(CMCT$overall['Accuracy'], 4)))
```



c) Generalized Boosted model

```
## model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, 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 53 had non-zero influence.

## prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
CMGBM <- confusionMatrix(predictGBM, as.factor(TestSet$classe))
CMGBM

## 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

## plotting confusion matrix results
plot(CMGBM$table, col = CMGBM$byClass, main = paste("GBM - Accuracy =",
round(CMGBM$overall['Accuracy'], 4)))
```



APPLYING THE MODEL WITH HIGHEST ACCURACY TO THE TEST DATASET

The accuracy of the three prdiction models is as follows: 1) Random Forest: 0.9990 2) Classification Tree: 0.7342 3) Generalized Boosted model: 0.9871 Therefore, the Random Forest model will be applied to the 20 test cases in the test dataset to solve the prediction quiz.

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

```
##  [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
```