Prediction Assignment Writeup

Chongyang Wang

10/18/2022

Introduction

This document is the report of the Peer Assessment project for Coursera’s Practical Machine Learning. The Knit to HTML doesn’t work. The result HTML is blank. The Knit to WORD works. It is a RStudio markdown file to be published in word format. It contains the code and analysis for the course quiz. The machine learning algorithm described here is deployed 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.

# load libraries  
  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(corrplot)

## corrplot 0.90 loaded

library(corrplot)  
library(data.table)  
library(gbm)

## Loaded gbm 2.1.8

library(ggplot2)  
library(knitr)  
library(lattice)  
library(plotly)

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

library(rattle)

## Loading required package: tibble

## 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.

##   
## Attaching package: 'rattle'

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

library(RColorBrewer)  
library(rpart)  
library(rpart.plot)  
  
data\_training <- read.csv("pml-training.csv")  
data\_quiz <- read.csv("pml-testing.csv")  
dim(data\_training)

## [1] 19622 160

dim(data\_quiz)

## [1] 20 160

# remove NA  
in\_training <- createDataPartition(data\_training$classe, p=0.70, list=FALSE)  
training\_set <- data\_training[ in\_training, ]  
testing\_set <- data\_training[-in\_training, ]  
dim(training\_set)

## [1] 13737 160

dim(testing\_set)

## [1] 5885 160

# remove NZV variables  
nzv\_var <- nearZeroVar(training\_set)  
training\_set <- training\_set[ , -nzv\_var]  
testing\_set <- testing\_set [ , -nzv\_var]  
dim(training\_set)

## [1] 13737 109

# remove mostly NA variables  
na\_var <- sapply(training\_set, function(x) mean(is.na(x))) > 0.95  
training\_set <- training\_set[ , na\_var == FALSE]  
testing\_set <- testing\_set [ , na\_var == FALSE]  
dim(training\_set)

## [1] 13737 59

dim(testing\_set)

## [1] 5885 59

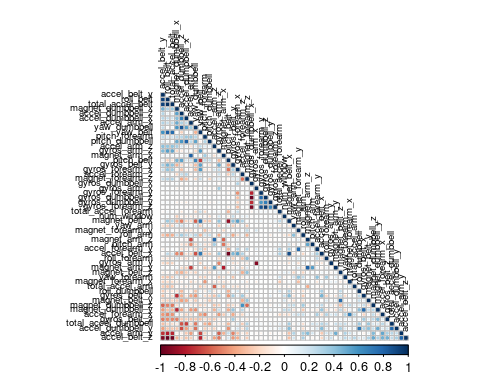
# remove columns 1 to 5 identification variables   
training\_set <- training\_set[ , -(1:5)]  
testing\_set <- testing\_set [ , -(1:5)]  
dim(training\_set)

## [1] 13737 54

dim(testing\_set)

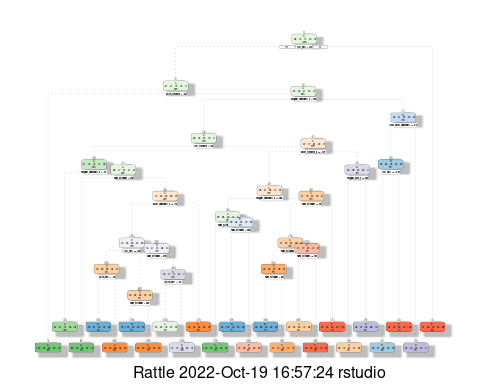
## [1] 5885 54

# correlation among variables  
corr\_matrix <- cor(training\_set[ , -54])  
corrplot(corr\_matrix, order = "FPC", method = "circle", type = "lower",  
 tl.cex = 0.6, tl.col = rgb(0, 0, 0))



# try Decision Tree model  
set.seed(1967)  
fit\_DT <- rpart(classe ~ ., data = training\_set, method="class")  
fancyRpartPlot(fit\_DT)

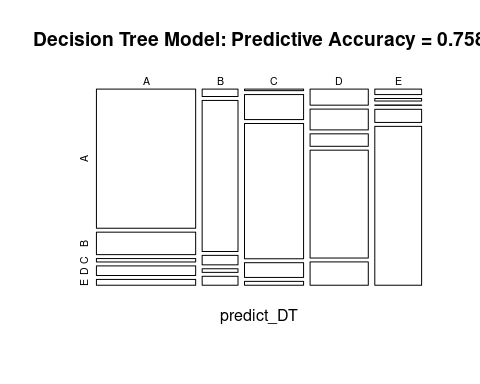
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



predict\_DT <- predict(fit\_DT, newdata = testing\_set, type="class")  
conf\_matrix\_DT <- confusionMatrix(table(predict\_DT, testing\_set$classe))  
conf\_matrix\_DT

## Confusion Matrix and Statistics  
##   
##   
## predict\_DT A B C D E  
## A 1505 243 37 104 63  
## B 29 590 37 14 35  
## C 10 161 871 95 25  
## D 102 132 79 684 148  
## E 28 13 2 67 811  
##   
## Overall Statistics  
##   
## Accuracy : 0.758   
## 95% CI : (0.7469, 0.7689)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6927   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8990 0.5180 0.8489 0.7095 0.7495  
## Specificity 0.8938 0.9758 0.9401 0.9063 0.9771  
## Pos Pred Value 0.7710 0.8369 0.7496 0.5974 0.8806  
## Neg Pred Value 0.9570 0.8940 0.9672 0.9409 0.9454  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2557 0.1003 0.1480 0.1162 0.1378  
## Detection Prevalence 0.3317 0.1198 0.1975 0.1946 0.1565  
## Balanced Accuracy 0.8964 0.7469 0.8945 0.8079 0.8633

plot(conf\_matrix\_DT$table, col = conf\_matrix\_DT$byClass,   
 main = paste("Decision Tree Model: Predictive Accuracy =",  
 round(conf\_matrix\_DT$overall['Accuracy'], 4)))



# try Generalized Boosted Model   
set.seed(1967)  
ctrl\_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)  
fit\_GBM <- train(classe ~ ., data = training\_set, method = "gbm",  
 trControl = ctrl\_GBM, verbose = FALSE)  
fit\_GBM$finalModel

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

predict\_GBM <- predict(fit\_GBM, newdata = testing\_set)  
conf\_matrix\_GBM <- confusionMatrix(table(predict\_GBM, testing\_set$classe))  
conf\_matrix\_GBM

## Confusion Matrix and Statistics  
##   
##   
## predict\_GBM A B C D E  
## A 1669 19 0 0 0  
## B 4 1104 7 6 2  
## C 0 16 1011 19 0  
## D 1 0 6 938 7  
## E 0 0 2 1 1073  
##   
## Overall Statistics  
##   
## Accuracy : 0.9847   
## 95% CI : (0.9812, 0.9877)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9807   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9970 0.9693 0.9854 0.9730 0.9917  
## Specificity 0.9955 0.9960 0.9928 0.9972 0.9994  
## Pos Pred Value 0.9887 0.9831 0.9665 0.9853 0.9972  
## Neg Pred Value 0.9988 0.9927 0.9969 0.9947 0.9981  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2836 0.1876 0.1718 0.1594 0.1823  
## Detection Prevalence 0.2868 0.1908 0.1777 0.1618 0.1828  
## Balanced Accuracy 0.9963 0.9826 0.9891 0.9851 0.9955

# try Random Forest  
set.seed(1967)  
ctrl\_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)  
fit\_RF <- train(classe ~ ., data = training\_set, method = "rf",  
 trControl = ctrl\_RF, verbose = FALSE)  
fit\_RF$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)), verbose = FALSE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 27  
##   
## OOB estimate of error rate: 0.26%  
## Confusion matrix:  
## A B C D E class.error  
## A 3904 1 0 0 1 0.0005120328  
## B 7 2646 5 0 0 0.0045146727  
## C 0 5 2391 0 0 0.0020868114  
## D 0 0 12 2239 1 0.0057726465  
## E 0 0 0 4 2521 0.0015841584

predict\_RF <- predict(fit\_RF, newdata = testing\_set)  
conf\_matrix\_RF <- confusionMatrix(table(predict\_RF, testing\_set$classe))  
conf\_matrix\_RF

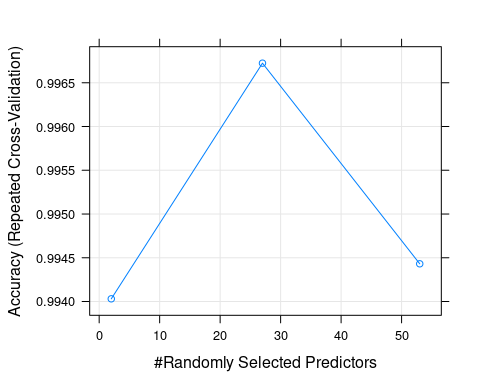
## Confusion Matrix and Statistics  
##   
##   
## predict\_RF A B C D E  
## A 1674 8 0 0 0  
## B 0 1129 1 0 1  
## C 0 1 1022 6 0  
## D 0 1 3 958 2  
## E 0 0 0 0 1079  
##   
## Overall Statistics  
##   
## Accuracy : 0.9961   
## 95% CI : (0.9941, 0.9975)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9951   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9912 0.9961 0.9938 0.9972  
## Specificity 0.9981 0.9996 0.9986 0.9988 1.0000  
## Pos Pred Value 0.9952 0.9982 0.9932 0.9938 1.0000  
## Neg Pred Value 1.0000 0.9979 0.9992 0.9988 0.9994  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2845 0.1918 0.1737 0.1628 0.1833  
## Detection Prevalence 0.2858 0.1922 0.1749 0.1638 0.1833  
## Balanced Accuracy 0.9991 0.9954 0.9973 0.9963 0.9986

Applying the Best Predictive Model to the Test Data To summarize, the predictive accuracy of the three models evaluated is the followings:

* Decision Tree Model: is the worst, has the low mean and the highest standard deviation.
* GBM Model: has a good mean accuracy but a lower accuracy than RF.
* Random Fores Model: has the highest mean accuracy and lowest standard deviation

Checking if there are anything to gain from increasing the number of boosting iterations.

plot(fit\_RF)



print(fit\_RF$bestTune)

## mtry  
## 2 27

The predictive accuracy of the Random Forest model is the best at 99.5 %. Accuracy is not going to be better and further tuning would only yield a little gain.

Make Predictions Deciding to predict with this model. Decision Tree Model: 75.8 % Generalized Boosted Model: 98.5 % Random Forest Model: 99.5 %

The Random Forest model is selected to make predictions on the 20 data points from the original testing dataset (data\_quiz).

cat("Predictions: ", paste(predict(fit\_RF, data\_quiz)))

## Predictions: B A B A A E D B A A B C B A E E A B B B