PML Course Project

Rogerli

7/6/2020

1.Overview

This report is used to predict the manner in which the 6 participants did the exercise. This dataset comes from the paper **Qualitative Activity Recognition of Weight Lifting Exercises**. Many thanks to the authors for their generosity in contributing this data source.

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

3.Data Process

3.1 Load the Package

First, we will load the machine learning associated packages.

```
library(caret)
```

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

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

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

```
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:rattle':
##
##     importance
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

3.2 Data Cleaning

The data cleaning process will focus on the trainning dataset since the final model will use the model derived from trainning set to predict testset.Here, we need to cleanning near-zero-variation variables and ID associated variables which contains no useful information.Besides, we also remove the missing variables.

```
Train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train<-read.csv(url(Train_url))
test<-read.csv(url(Test_url))

inTrain  <- createDataPartition(train$classe, p=0.7, list=FALSE)
trainset <- train[inTrain, ]
testset <- train[-inTrain, ]

## Remove near-zero-variation data
nzv<-nearZeroVar(trainset)
trainset<-trainset[,-nzv]
testset<-testset[,-nzv]

## Remove ID associated variables, which is first five columns
trainset<-trainset[,-(1:5)]
testset<-testset[,-(1:5)]
```

```r
## Remove variables with missing values
trainset<-trainset[,colSums(is.na(trainset))==0]
testset<-testset[,colSums(is.na(testset))==0]

## summarize data diomensions
dim(trainset)
```

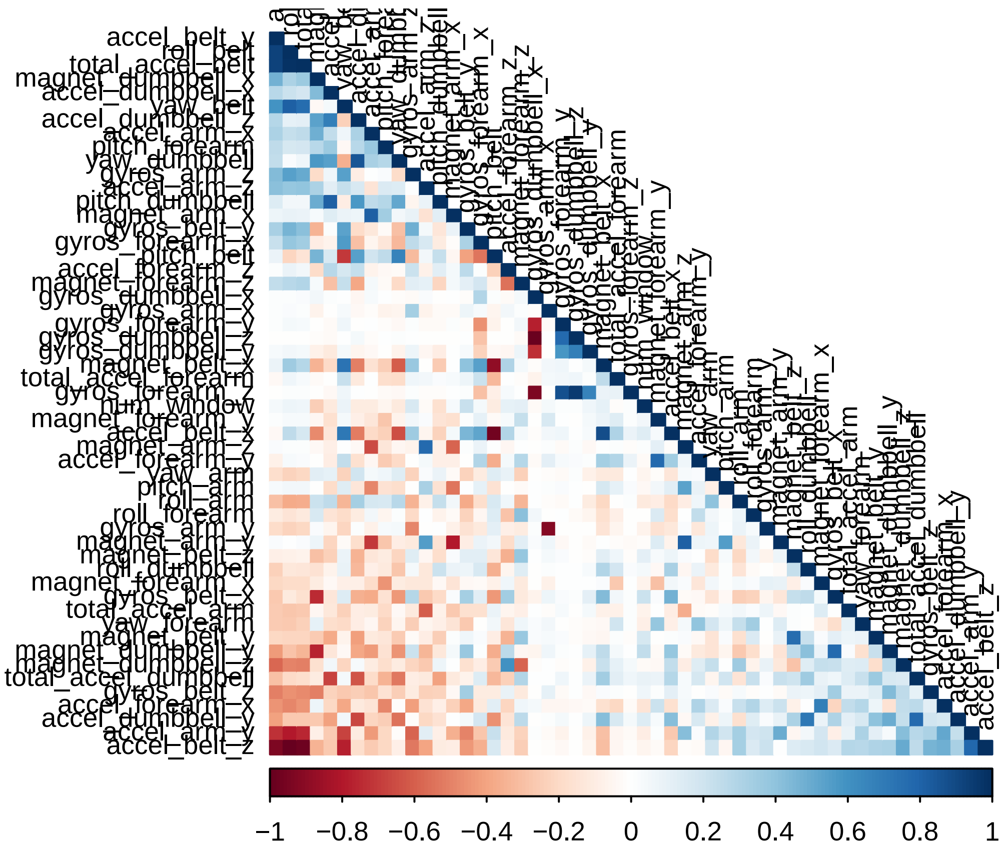
```
## [1] 13737    54
```

```r
dim(testset)
```

```
## [1] 5885   54
```

4. Correlation Analysis

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



As the correlation matrix has shown, there are some variables are highly correlated. The next step, we will separate out the variables with more than 80% correlation.

```r
highcorrelation <- findCorrelation(cormax, cutoff=0.8)
names(trainset)[highcorrelation]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_dumbbell_z"
##  [5] "accel_belt_x"     "pitch_belt"       "accel_arm_x"      "accel_dumbbell_x"
##  [9] "magnet_arm_y"     "gyros_forearm_y"  "gyros_dumbbell_x" "gyros_dumbbell_z"
## [13] "gyros_arm_x"
```

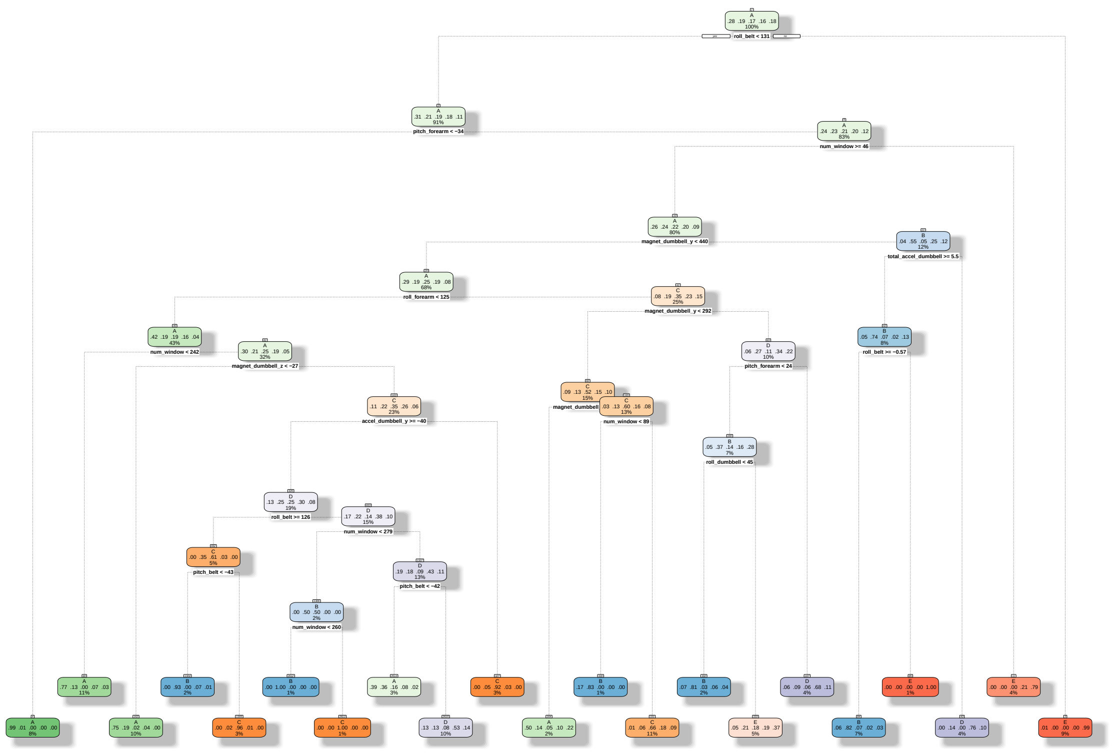
5. Model Selection

5.1 Classfication trees model

First, we will fit a tree model to the trainning set.

```r
set.seed(1111)
treemod <- rpart(classe ~ ., data=trainset, method="class")
fancyRpartPlot(treemod)
```

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



Rattle 2020−Jul−08 22:41:44 lyh2019

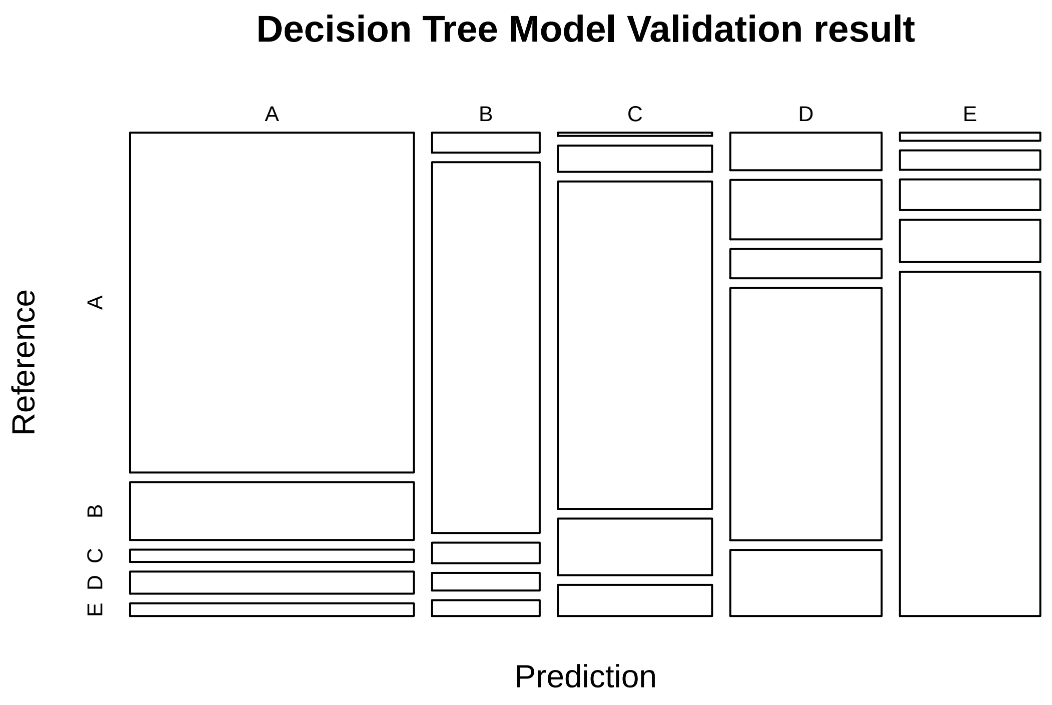
Then, we will use the testset to validate the "decision tree model".

```
pred_treemod<-predict(treemod,testset,type="class")
result_tree<-confusionMatrix(pred_treemod,testset$classe)
result_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1524  259   55   99   57
##          B   34  631   35   30   27
##          C    8   64  798  138   76
##          D   90  142   70  603  158
##          E   18   43   68   94  764
##
## Overall Statistics
##
##                Accuracy : 0.7341
##                  95% CI : (0.7226, 0.7453)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6617
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104   0.5540   0.7778   0.6255   0.7061
## Specificity            0.8884   0.9735   0.9411   0.9065   0.9536
## Pos Pred Value         0.7643   0.8336   0.7362   0.5673   0.7741
## Neg Pred Value         0.9614   0.9009   0.9525   0.9251   0.9351
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2590   0.1072   0.1356   0.1025   0.1298
## Detection Prevalence   0.3388   0.1286   0.1842   0.1806   0.1677
## Balanced Accuracy      0.8994   0.7637   0.8595   0.7660   0.8298
```

Then result of validation shows that the accuracy of the tree model is 0.8024 on the testset. The detail result of accuracy was shown below.

```
plot(result_tree$table, col = result_tree$byClass,
     main = "Decision Tree Model Validation result")
```



5.2 Random Forest model

```r
cont_rf <- trainControl(method="cv", number=3, verboseIter=FALSE)
mod_rf <- train(classe ~ ., data=trainset, method="rf", trControl=cont_rf)
mod_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: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2647    5    0    0 0.0041384500
## C    0    6 2389    1    0 0.0029215359
## D    0    0    7 2244    1 0.0035523979
## E    0    0    0    6 2519 0.0023762376
```

After we get the random forest, we use it to predict the validation (testset). The result of the validation shows the accuracy is 0.9969, which is pretty high

```r
pred_rf <- predict(mod_rf,newdata=testset)
result_rf <- confusionMatrix(pred_rf,testset$classe)
result_rf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    3    0    3
##          C    0    0 1023    6    0
##          D    0    0    0  958    2
##          E    0    0    0    0 1077
## 
## Overall Statistics
## 
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9961
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9971   0.9938   0.9954
## Specificity            0.9991   0.9987   0.9988   0.9996   1.0000
## Pos Pred Value         0.9976   0.9947   0.9942   0.9979   1.0000
## Neg Pred Value         1.0000   0.9992   0.9994   0.9988   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1738   0.1628   0.1830
## Detection Prevalence   0.2851   0.1939   0.1749   0.1631   0.1830
## Balanced Accuracy      0.9995   0.9976   0.9979   0.9967   0.9977
```

5.3 Generalized Boosted Regression Model

```r
set.seed(1111)
cont_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
mod_gbm  <- train(classe ~ ., data=trainset, method = "gbm",
                  trControl = cont_gbm, verbose = FALSE)
mod_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.
```

```r
print(mod_gbm)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10988, 10989, 10991, 10990, 10990
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7657396  0.7028913
##   1                  100      0.8341679  0.7900769
##   1                  150      0.8697647  0.8351627
##   2                   50      0.8826503  0.8513739
##   2                  100      0.9385592  0.9222496
##   2                  150      0.9636744  0.9540411
##   3                   50      0.9355008  0.9183567
##   3                  100      0.9712448  0.9636163
##   3                  150      0.9863141  0.9826879
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Next, we will use the Generalized Boosted Regression Model to validate the testset data. It shows a very high accuracy, which is 0.9854.

```r
pred_gbm <- predict(mod_gbm,newdata=testset)
result_gbm <- confusionMatrix(pred_gbm,testset$classe)
result_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    5    0    0    1
##          B    0 1114   11    9    2
##          C    0   20 1011   22    1
##          D    0    0    4  932   11
##          E    2    0    0    1 1067
##
## Overall Statistics
##
##                Accuracy : 0.9849
##                  95% CI : (0.9814, 0.9878)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9809
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9781   0.9854   0.9668   0.9861
## Specificity            0.9986   0.9954   0.9912   0.9970   0.9994
## Pos Pred Value         0.9964   0.9806   0.9592   0.9842   0.9972
## Neg Pred Value         0.9995   0.9947   0.9969   0.9935   0.9969
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1893   0.1718   0.1584   0.1813
## Detection Prevalence   0.2851   0.1930   0.1791   0.1609   0.1818
## Balanced Accuracy      0.9987   0.9867   0.9883   0.9819   0.9928
```

6.Select best model to predtict the test

According to the previous model, the random forest model generate the highest accuracy and we will apply it to the test dataset.

```
pred_test<-predict(mod_rf,newdata=test)
pred_test
```

```
##  [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
```