Prediction Assignment Writeup

Marina. K

2018.10.31

Introduction

This is a report for assignment of Practical Machine Learning course in Coursera. XXX

0. Environmental Setting

Loading packages that I will use in this analysis.

```
library(caret)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## バージョン 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## 'rattle()' と入力して、データを多角的に分析します。
```

```
library(rpart)
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:dplyr':
##
##     combine
```

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

1. Data Preparetion

1-1.Data Loading

```
training_data <- read.csv("./data/pml-training.csv", header = T)
testing_data <- read.csv("./data/pml-testing.csv", header = T)
```

1-2.Data Cleansing

There are some colums in testing data. It would be not used in our practice, hence I will remove those colums from testing data set and training data set. In addition to that, I want to use "randomForest" pacakage in the following model building, the package has limitation of data(53 factors), so I will compress the data to the 53colums(variables). Hence I will cut off first 7 columns that seems to be non-quantitive values.

And also, I will remove all NA rows from training data because it is harmful to building the model.

```
eliminateFactors <- names(testing_data[,colSums(is.na(testing_data)) == 0])[8:59]
training_data1 <- training_data[,c(eliminateFactors,"classe")]
testing_data1 <- testing_data[,c(eliminateFactors,"problem_id")]
training_data1 <- na.omit(training_data1)
dim(training_data1)
```

```
## [1] 19622    53
```

```
dim(testing_data1)
```

```
## [1] 20 53
```

Then, column names in data are following;

```
names(testing_data1)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "problem_id"
```

Next I will split the training data set to portion of “training set” and “testing set”. By using the method of “Random subsampling”, I will do cross validation. Then, I will split data as training set 60% of all training data, and difine the remainings are testing set. This “testing set” is simlilar name with “testing data” that I loaded before, but it explicityly different.

```
set.seed(1123)
inTrain <- createDataPartition(training_data1$classe, p=0.6, list=FALSE)
training <- training_data1[inTrain,]
testing <- training_data1[-inTrain,]

dim(training)
```

```
## [1] 11776    53
```

```
dim(testing)
```

```
## [1] 7846   53
```

2. Model Building

2-1. With Random Forest Method

I tried to use “caret” but it does not work well because of heavy load. Then, instead of caret, I will use “randomForest” package that directly lead the prediction by using random forest method.

```
set.seed(1123)
modFitRFM <- randomForest(classe ~ ., data = training, ntree = 1000)
modFitRFM
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.62%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    1    0    0 0.0005973716
## B   12 2259    8    0    0 0.0087757789
## C    0   14 2039    1    0 0.0073028238
## D    0    0   22 1905    3 0.0129533679
## E    0    0    3    8 2154 0.0050808314
```

```
prediction <- predict(modFitRFM, testing, type = "class")
confusionMatrix(prediction, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    8    0    0    0
##          B    0 1510   11    0    0
##          C    0    0 1357   17    0
##          D    0    0    0 1268    6
##          E    0    0    0    1 1436
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.9926, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9920   0.9860   0.9958
## Specificity            0.9986   0.9983   0.9974   0.9991   0.9998
## Pos Pred Value         0.9964   0.9928   0.9876   0.9953   0.9993
## Neg Pred Value         1.0000   0.9987   0.9983   0.9973   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1925   0.1730   0.1616   0.1830
## Detection Prevalence   0.2855   0.1939   0.1751   0.1624   0.1832
## Balanced Accuracy      0.9993   0.9965   0.9947   0.9925   0.9978
```

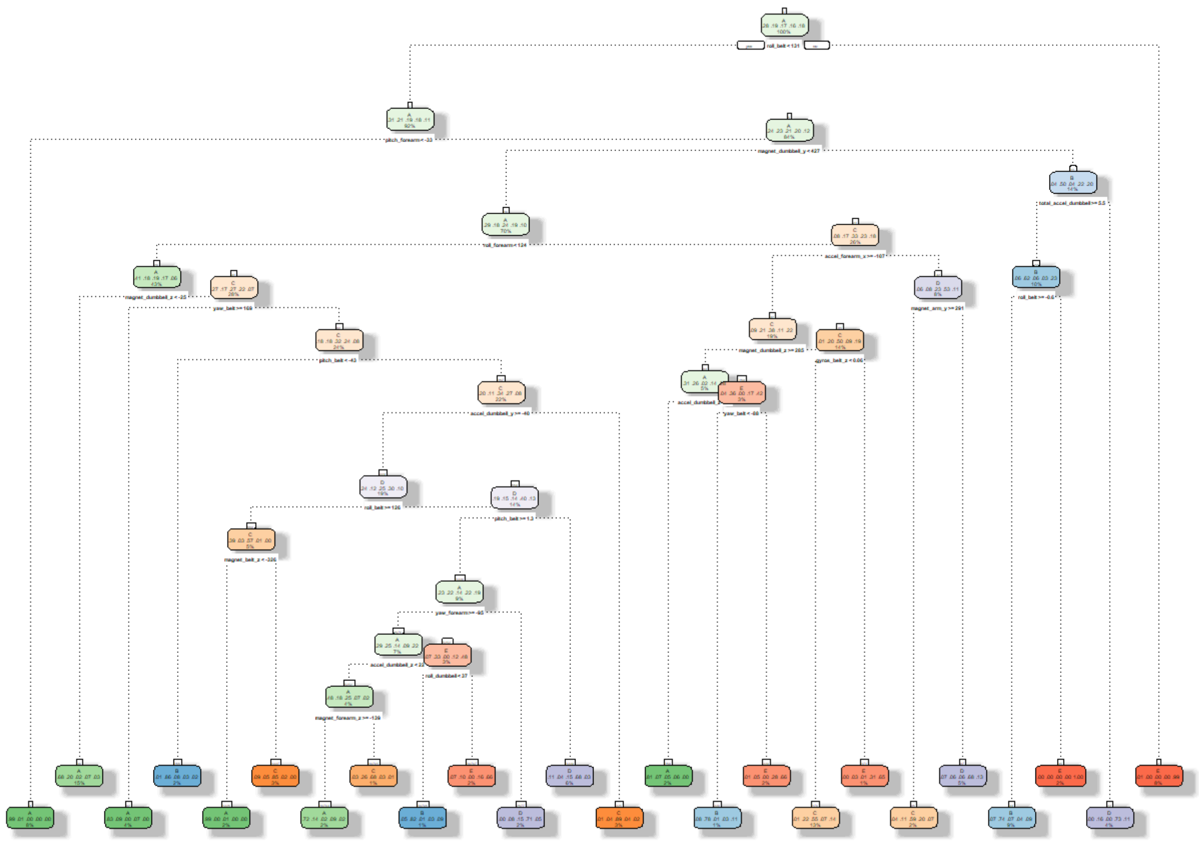
From abve consequence, “Balanced Accuracy” is 0.9990, it is very high. Accuracy of 99.9% is almost 100%. It seems to be great model to predicting, but I still try to find out whether much better model is exit or not.

2-2. With Decision Tree Model

Next, I will use “Decision Tree Model” to construct a predictive model.

```
set.seed(1123)
modFitDTM <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(modFitDTM)
```

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



Rattle 2018-11-01 19:41:39 Marina

```
modFitDTM
```

```
## n= 11776 
## 
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
## 
##     1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)  
##       2) roll_belt< 130.5 10782 7444 A (0.31 0.21 0.19 0.18 0.11)  
##         4) pitch_forearm< -33.15 943   10 A (0.99 0.011 0 0 0) *
##         5) pitch_forearm>=-33.15 9839 7434 A (0.24 0.23 0.21 0.2 0.12)  
##          10) magnet_dumbbell_y< 426.5 8186 5851 A (0.29 0.18 0.24 0.19 0.1)  
##            20) roll_forearm< 123.5 5071 2982 A (0.41 0.18 0.19 0.17 0.056)  
##              40) magnet_dumbbell_z< -25.5 1755  564 A (0.68 0.2 0.021 0.07 0.034) *
##              41) magnet_dumbbell_z>=-25.5 3316 2410 C (0.27 0.17 0.27 0.22 0.068)  
##                82) yaw_belt>=168.5 444   76 A (0.83 0.092 0.0023 0.072 0.0045) *
##                83) yaw_belt< 168.5 2872 1967 C (0.18 0.18 0.32 0.24 0.078)  
##                 166) pitch_belt< -43.15 277   38 B (0.011 0.86 0.079 0.025 0.022) *
##                 167) pitch_belt>=-43.15 2595 1712 C (0.2 0.11 0.34 0.27 0.084)  
##                   334) accel_dumbbell_y>=-40.5 2219 1547 D (0.24 0.12 0.25 0.3 0.096)  
##                     668) roll_belt>=125.5 541  233 C (0.39 0.031 0.57 0.013 0)  
##                      1336) magnet_belt_z< -326 178    1 A (0.99 0 0.0056 0 0) *
##                      1337) magnet_belt_z>=-326 363   56 C (0.088 0.047 0.85 0.019 0) *
##                     669) roll_belt< 125.5 1678 1013 D (0.19 0.15 0.14 0.4 0.13)  
##                      1338) pitch_belt>=1.305 1024  785 A (0.23 0.22 0.14 0.22 0.19)  
##                        2676) yaw_forearm>=-95.15 818  579 A (0.29 0.25 0.14 0.093 0.22)  
##                          5352) accel_dumbbell_z< 21.5 450  235 A (0.48 0.18 0.25 0.069 0.018)  
##                           10704) magnet_forearm_z>=-139 292   81 A (0.72 0.14 0.017 0.092 0.024) *
##                           10705) magnet_forearm_z< -139 158   50 C (0.025 0.26 0.68 0.025 0.0063) *
##                          5353) accel_dumbbell_z>=21.5 368  192 E (0.065 0.33 0.0027 0.12 0.48)  
##                           10706) roll_dumbbell< 36.68983 117   21 B (0.051 0.82 0.0085 0.034 0.085)
*
##                           10707) roll_dumbbell>=36.68983 251   85 E (0.072 0.1 0 0.16 0.66) *
##                        2677) yaw_forearm< -95.15 206   59 D (0 0.083 0.15 0.71 0.053) *
##                      1339) pitch_belt< 1.305 654  212 D (0.11 0.038 0.15 0.68 0.026) *
##                   335) accel_dumbbell_y< -40.5 376   41 C (0.011 0.037 0.89 0.043 0.019) *
##            21) roll_forearm>=123.5 3115 2077 C (0.079 0.17 0.33 0.23 0.18)  
##              42) accel_forearm_x>=-107.5 2204 1372 C (0.088 0.21 0.38 0.11 0.22)  
##                84) magnet_dumbbell_z>=284.5 567  394 A (0.31 0.26 0.019 0.14 0.28)  
##                 168) accel_dumbbell_z< 28.5 194   36 A (0.81 0.072 0.052 0.062 0) *
##                 169) accel_dumbbell_z>=28.5 373  216 E (0.04 0.36 0.0027 0.17 0.42)  
##                   338) yaw_belt< -88.15 160   36 B (0.081 0.78 0.0062 0.031 0.11) *
##                   339) yaw_belt>=-88.15 213   73 E (0.0094 0.052 0 0.28 0.66) *
##                85) magnet_dumbbell_z< 284.5 1637  816 C (0.013 0.2 0.5 0.095 0.19)  
##                 170) gyros_belt_z< 0.06 1478  658 C (0.014 0.22 0.55 0.072 0.14) *
##                 171) gyros_belt_z>=0.06 159   55 E (0 0.031 0.0063 0.31 0.65) *
##              43) accel_forearm_x< -107.5 911  427 D (0.057 0.078 0.23 0.53 0.11)  
##                86) magnet_arm_y>=291 285  117 C (0.035 0.11 0.59 0.2 0.067) *
##                87) magnet_arm_y< 291 626  198 D (0.067 0.062 0.061 0.68 0.13) *
##          11) magnet_dumbbell_y>=426.5 1653  831 B (0.042 0.5 0.045 0.22 0.2)  
##            22) total_accel_dumbbell>=5.5 1202  451 B (0.058 0.62 0.061 0.03 0.23)  
##              44) roll_belt>=-0.6 1019  268 B (0.069 0.74 0.072 0.035 0.087) *
##              45) roll_belt< -0.6 183    0 E (0 0 0 0 1) *
##            23) total_accel_dumbbell< 5.5 451  123 D (0 0.16 0.0022 0.73 0.11) *
##       3) roll_belt>=130.5 994   10 E (0.01 0 0 0 0.99) *
```

```
prediction2 <- predict(modFitDTM, newdata=testing, type="class")
cM_DTM <- confusionMatrix(prediction2, testing$classe)
cM_DTM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2020  318   25  143   37
##          B   61  789   67   34   92
##          C   51  280 1172  120  194
##          D   82  103  101  883  102
##          E   18   28    3  106 1017
##
## Overall Statistics
##
##                Accuracy : 0.7496
##                  95% CI : (0.7398, 0.7591)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6821
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9050   0.5198   0.8567   0.6866   0.7053
## Specificity            0.9068   0.9599   0.9004   0.9409   0.9758
## Pos Pred Value         0.7943   0.7565   0.6450   0.6947   0.8677
## Neg Pred Value         0.9600   0.8928   0.9675   0.9387   0.9363
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2575   0.1006   0.1494   0.1125   0.1296
## Detection Prevalence   0.3241   0.1329   0.2316   0.1620   0.1494
## Balanced Accuracy      0.9059   0.7398   0.8786   0.8137   0.8405
```

From abve consequence, “Balanced Accuracy” is 0.9059, it is high enogh but smaller than with “random forest model’. And also, R warns that”there is over fitting“, hence I choose this former one.

3. Testing with Random Forest Model

I decided to choose the prediction model that made by using “Random Forest Model”, because of high accuracy. And finally, I will carry out prediction on 20 “test case”.

```
prediction_RFM <- predict(modFitRFM, testing_data1, type = "class")
prediction_RFM
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

The answer is above. It’s accuracy is 99.9%.