Practical Machine Learning Course Project

Yasuhiro HARA

2021/9/23

Summary

In this project, we would predict the “classe” variable in “Human Activity Recognition” data set. We used Random Forest approach to predict the variable, and our prediction model showed 99.23% accuracy.

Note: The data is can be downloaded from the link below.

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

Data preprocessing

```
#rm(list = ls())
library(data.table)
library(doParallel)
library(caret)
library(corrplot)
library(rattle)
set.seed(0)
```

Data Loading

```
train_raw <- read.csv("pml-training.csv")
test_raw <- read.csv("pml-testing.csv")
```

Data Cleaning

First, we remove variables which includes NAs and near zero variance.

```
train_cleaned <- train_raw[,colMeans(is.na(train_raw)) < .9]
train_cleaned <- train_cleaned[,-c(1:7)]
```

```
nzv <- nearZeroVar(train_cleaned)
train_cleaned <- train_cleaned[,-nzv]
dim(train_cleaned)
```

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

Data slicing

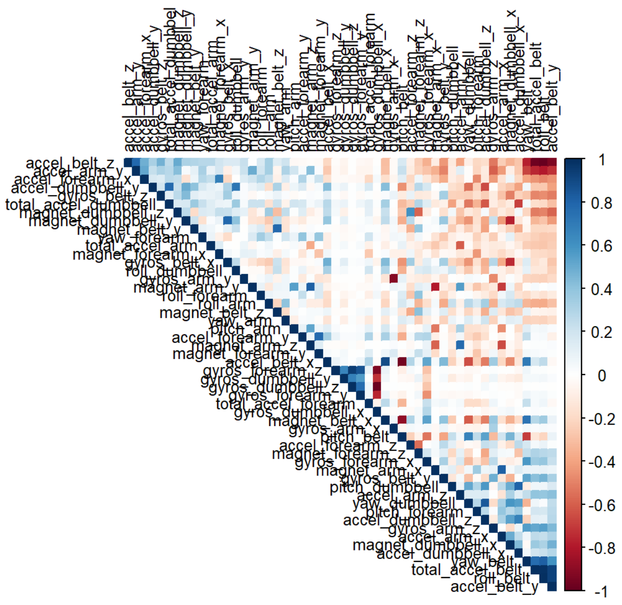
Then, we split the training set in order to validate our model later on.

```
inTrain <- createDataPartition(y = train_cleaned$classe, p = 0.75, list = F)
train <- train_cleaned[inTrain,]
valid <- train_cleaned[-inTrain,]
```

EDA

Now that we have cleaned the dataset off absolutely useless varibles, we shall look at the dependence of these variables on each other through a correlation plot.

```
cor_mat <- cor(train[, -53])
corrplot(cor_mat, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



Model Selection

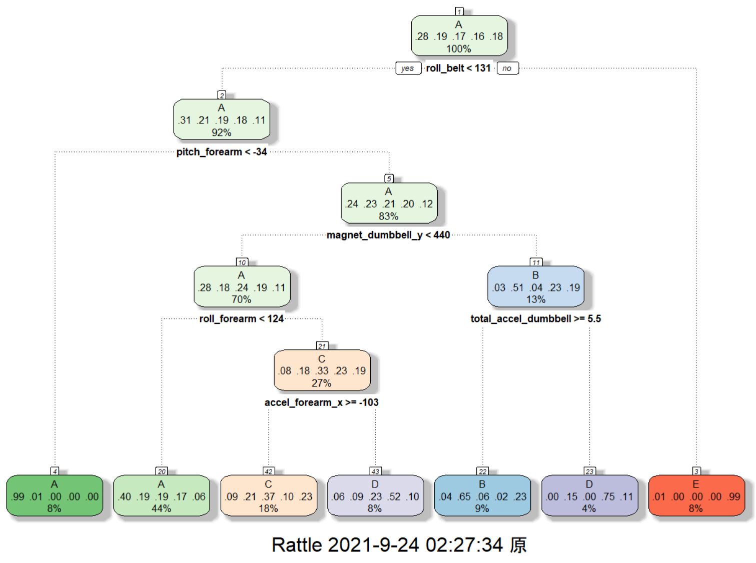
We will use 2 methods to model the training set and thereby choose the one having the best accuracy to predict the outcome variable in the testing set. The methods are Decision Tree and Random Forest.

```
control <- trainControl(method="cv", number=3, verboseIter=F)
```

Decision Tree

Model:

```
mod_trees <- train(classe~., data=train, method="rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(mod_trees$finalModel)
```



Prediction:

```
pred_trees <- predict(mod_trees, valid)
cmtrees <- confusionMatrix(pred_trees, factor(valid$classe))
cmtrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1273  383  385  364  157
##          B   21  295   27    8  107
##          C   79  202  354  112  182
##          D   18   69   89  320   63
##          E    4    0    0    0  392
##
## Overall Statistics
##
##                Accuracy : 0.5371
##                  95% CI : (0.523, 0.5511)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3963
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9125  0.31085  0.41404  0.39801  0.43507
## Specificity            0.6327  0.95879  0.85799  0.94171  0.99900
## Pos Pred Value         0.4969  0.64410  0.38105  0.57245  0.98990
## Neg Pred Value         0.9479  0.85290  0.87396  0.88861  0.88709
## Prevalence             0.2845  0.19352  0.17435  0.16395  0.18373
## Detection Rate         0.2596  0.06015  0.07219  0.06525  0.07993
## Detection Prevalence   0.5224  0.09339  0.18944  0.11399  0.08075
## Balanced Accuracy      0.7726  0.63482  0.63601  0.66986  0.71704
```

Random Forest

Model:

```
mod_rf <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)
```

Prediction:

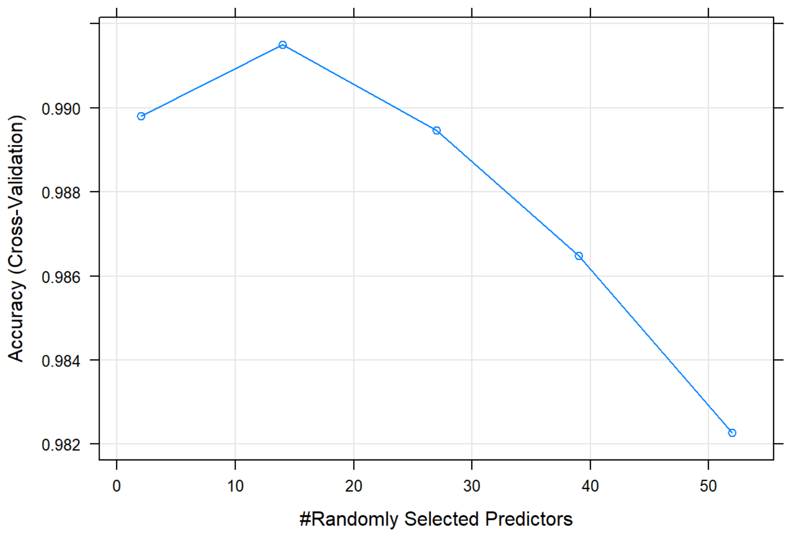
```
pred_rf <- predict(mod_rf, valid)
cmrf <- confusionMatrix(pred_rf, factor(valid$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    4    0    0    0
##          B    1  942    9    0    0
##          C    1    3  843   15    1
##          D    0    0    3  789    3
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9889, 0.9942)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9926   0.9860   0.9813   0.9956
## Specificity            0.9989   0.9975   0.9951   0.9985   1.0000
## Pos Pred Value         0.9971   0.9895   0.9768   0.9925   1.0000
## Neg Pred Value         0.9994   0.9982   0.9970   0.9963   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1921   0.1719   0.1609   0.1829
## Detection Prevalence   0.2849   0.1941   0.1760   0.1621   0.1829
## Balanced Accuracy      0.9987   0.9950   0.9905   0.9899   0.9978
```

Since Random Forest showed much higher accuracy, we choose Random Forest method to predict the data.

However, we plot the model just in case it might be overfitting.

```
plot(mod_rf)
```



Prediction

The prediction using the Random Forest method is shown below.

```
pred <- predict(mod_rf, test_raw)
print(pred)
```

```
##  [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
```