Machine Learning Final Project

Haoyu Wang

2017-09-4

1. Import data and package

```
library(AppliedPredictiveModeling)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rattle)

## Rattle: A free graphical interface for data mining with R.
## XXXX 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.

library(rpart.plot)

## Loading required package: rpart

library(randomForest)

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

setwd("C:/Users/May/Documents")
df_training<-read.csv("pml-training.csv", na.strings=c("NA",""))
df_testing<-read.csv("pml-testing.csv", na.strings=c("NA",""))
colnames_train <- colnames(df_training)
colnames_test <- colnames(df_testing)
set.seed(1213)
```

2.Clean Data

```
# Count the number of non-NAs.
nonNAs <- function(x) {
    as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}
```

```
# drop NA columns.
colcnts <- nonNAs(df_training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(df_training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}

# Drop NA data and the first 7 columns are unnecessary for predicting.
df_training <- df_training[,!(names(df_training) %in% drops)]
df_training <- df_training[,8:length(colnames(df_training))]

df_testing <- df_testing[,!(names(df_testing) %in% drops)]
df_testing <- df_testing[,8:length(colnames(df_testing))]
```

3.partition Data

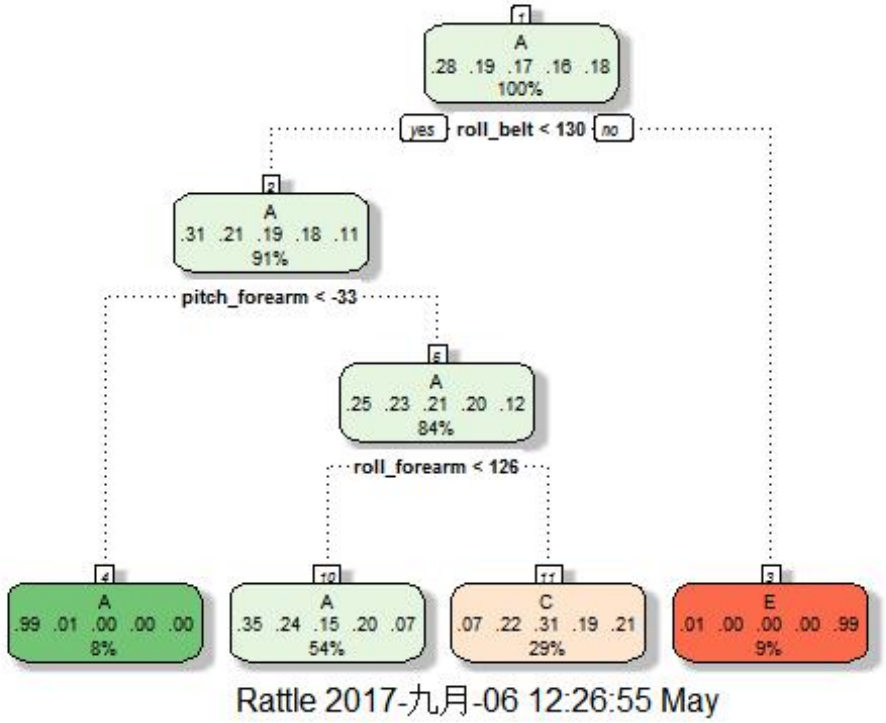
as the performence of my PC, I can only set P to 0.25.

```
intrain<-createDataPartition(y=df_training$classe,p=0.25,list=FALSE)
training<-df_training[intrain,]
testing<-df_training[-intrain,]
```

4.Try different models

1. try predict tree

```
modTree<-train(classe~., data=training,method="rpart")
fancyRpartPlot(modTree$finalModel)
```




```
predTree<-predict(modTree,testing)
confusionMatrix(predTree,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##         A 3878 1939 1254 1471  558
##         B    0    0    0    0    0
##         C  298  908 1312  941  933
##         D    0    0    0    0    0
##         E    9    0    0    0 1214
##
## Overall Statistics
##
##               Accuracy : 0.4352
##                 95% CI : (0.4272, 0.4433)
##    No Information Rate : 0.2844
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.2537
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity           0.9266   0.0000  0.51130   0.0000  0.44880
```

```
## Specificity            0.5041   1.0000  0.74648   1.0000  0.99925
## Pos Pred Value         0.4262      NaN  0.29872      NaN  0.99264
## Neg Pred Value         0.9453   0.8065  0.87852   0.8361  0.88949
## Prevalence             0.2844   0.1935  0.17438   0.1639  0.18383
## Detection Rate         0.2635   0.0000  0.08916   0.0000  0.08250
## Detection Prevalence   0.6184   0.0000  0.29847   0.0000  0.08311
## Balanced Accuracy      0.7154   0.5000  0.62889   0.5000  0.72402
```

As we can see the predict tree result is not so good. The Accuracy is only`0.4352`. it is not so good.

2. try random forest

```
modRF<-train(classe~., data=training,method="rf",prox=TRUE)
predRF<-predict(modRF,testing)
confusionMatrix(predRF,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 4142   42    0    4    0
##          B   32 2759   48    6   13
##          C    9   36 2493   53   18
##          D    1    8   25 2344   21
##          E    1    2    0    5 2653
##
## Overall Statistics
##
##               Accuracy : 0.978
##                 95% CI : (0.9755, 0.9803)
##    No Information Rate : 0.2844
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9721
##  Mcnemar's Test P-Value : 2.228e-09
##
## Statistics by Class:
##
##                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9897   0.9691   0.9716   0.9718   0.9808
## Specificity            0.9956   0.9917   0.9905   0.9955   0.9993
## Pos Pred Value         0.9890   0.9654   0.9555   0.9771   0.9970
## Neg Pred Value         0.9959   0.9926   0.9940   0.9945   0.9957
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2815   0.1875   0.1694   0.1593   0.1803
## Detection Prevalence   0.2846   0.1942   0.1773   0.1630   0.1808
## Balanced Accuracy      0.9927   0.9804   0.9810   0.9837   0.9901
```

the Acuracy is 0.978. the model is good to use.

CONCLUSION

```
predFinal<-predict(modRF,df_testing)

predFinal
```

```
##  [1] B A B A A E D D A A B C B A E E A B B B
## Levels: A B C D E
```

the result of prediction of testing set is "B A B A A E D B A A B C B A E E A B B B"