Course 8 project

xueyan zhang

7/29/2020

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.

Data loading

```
urltraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- read.csv(urltraining,header = TRUE,na.strings=c("NA","#DIV/0!",""))
urltesting <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- read.csv(urltesting,header = TRUE,na.strings=c("NA","#DIV/0!",""))
library(caret)
library(rattle)
library(randomForest)
set.seed(112233)
#install.packages('e1071', dependencies=TRUE)
```

Data cleaning

```
# delete first 7 columns which are identifier
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]
# delete variables with too many NA (70%)
training1 <- training
for(i in 1:length(training)) {
        if( sum( is.na( training[, i] ) ) /nrow(training) >= .7 ) {
        for(j in 1:length(training1)) {
            if( length( grep(names(training[i]), names(training1)[j]) ) ==1)  {
                training1 <- training1[ , -j]
            }
        }
    }
}
training <- training1
testing1 <- testing
for(i in 1:length(testing)) {
        if( sum( is.na( testing[, i] ) ) /nrow(testing) >= .7 ) {
        for(j in 1:length(testing1)) {
            if( length( grep(names(testing[i]), names(testing1)[j]) ) ==1)  {
                testing1 <- testing1[ , -j]
            }
        }
    }
}
testing <- testing1
```

Generate new training and testing set with training data

```
intrain <- createDataPartition(y=training$classe,p=0.75,list=FALSE)
newtraining <- training[intrain,]
newtesting <- training[-intrain,]
```

Predict with trees

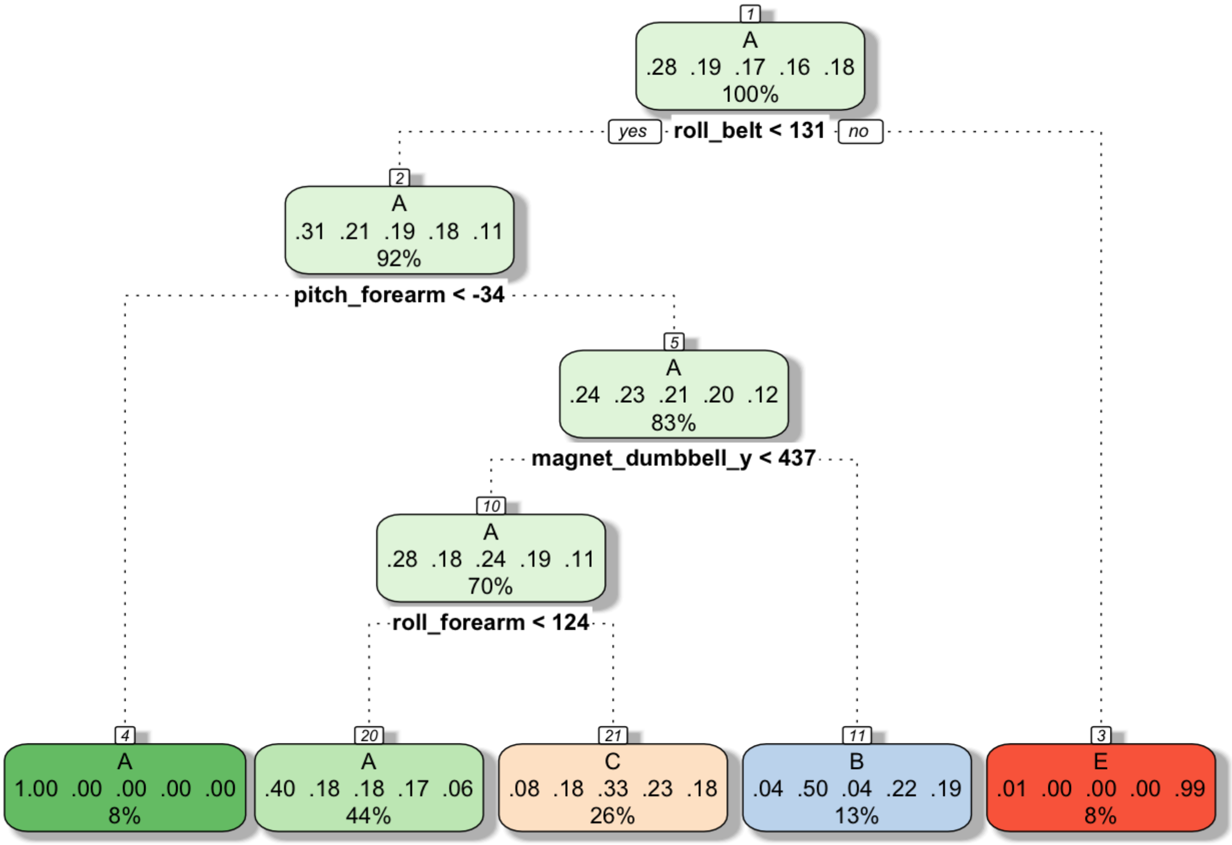
```
fit1 <- train(classe~.,method="rpart",data=newtraining)
print(fit1$finalModel)
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 13467  9296 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.35 1192     3 A (1 0.0025 0 0 0) *
##      5) pitch_forearm>=-34.35 12275  9293 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436.5 10354  7446 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6487  3875 A (0.4 0.18 0.18 0.17 0.062) *
##         21) roll_forearm>=123.5 3867  2579 C (0.077 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=436.5 1921   956 B (0.039 0.5 0.045 0.22 0.19) *
##    3) roll_belt>=130.5 1251    14 E (0.011 0 0 0 0.99) *
```

```
pred1 <- predict(fit1,newdata=newtesting)
newtesting$classe <- as.factor(newtesting$classe)
confusionMatrix(pred1,newtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1265  380  393  344  118
##          B   21  337   23  154  125
##          C  109  232  439  306  264
##          D    0    0    0    0    0
##          E    0    0    0    0  394
##
## Overall Statistics
##
##                Accuracy : 0.4965
##                  95% CI : (0.4824, 0.5106)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3429
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068  0.35511  0.51345   0.0000  0.43729
## Specificity            0.6480  0.91833  0.77501   1.0000  1.00000
## Pos Pred Value         0.5060  0.51061  0.32519      NaN  1.00000
## Neg Pred Value         0.9459  0.85580  0.88295   0.8361  0.88758
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2580  0.06872  0.08952   0.0000  0.08034
## Detection Prevalence   0.5098  0.13458  0.27529   0.0000  0.08034
## Balanced Accuracy      0.7774  0.63672  0.64423   0.5000  0.71865
```

```
fancyRpartPlot(fit1$finalModel)
```



Rattle 2020-Jul-31 01:47:24 Administrator

Predict with random forests

```
fit2 <- randomForest(as.factor(classe)~.,data=newtraining)
pred2 <- predict(fit2,newdata=newtesting)
confusionMatrix(pred2,newtesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    4    0    0    0
##          B    1  945   10    0    0
##          C    2    0  845    8    0
##          D    0    0    0  796    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9936
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9958   0.9883   0.9900   1.0000
## Specificity            0.9989   0.9972   0.9975   1.0000   1.0000
## Pos Pred Value         0.9971   0.9885   0.9883   1.0000   1.0000
## Neg Pred Value         0.9991   0.9990   0.9975   0.9981   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1927   0.1723   0.1623   0.1837
## Detection Prevalence   0.2847   0.1949   0.1743   0.1623   0.1837
## Balanced Accuracy      0.9984   0.9965   0.9929   0.9950   1.0000
```

The accuracy of tree is 0.4965 [0.4824, 0.5106].The accuracy of random forest is 0.9949 [0.9925, 0.9967]. So random forest model is used for predict with testing data.The expected out-of-sample error is estimated at 0.005.

Predict with original testing data

```
predict(fit2,newdata=testing)
```

```
##  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
```