practical machine learning-project

Shaghayegh Kazemlou

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

Goal

The goal is to predict “classe” variable in the training set which is the way they exercise.

Setup and load data

```
library(rpart)
library(caret)
library(rattle)
library(randomForest)
library(RColorBrewer)
library(knitr)
library(rpart.plot)
set.seed(1200)

setwd("E:/Shaghayegh/Coursera")
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
dim(training);dim(testing)
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

data cleaning

remove NA,remove low variance variables

```
training <- training[, colSums(is.na(training)) == 0]
nzv <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[,nzv$nzv==FALSE]

dim(training)
```

```
## [1] 19622    59
```

Split data

split the training set into a training set (train, 60%) for prediction and a validation set (valid 40%) to compute the out-of-sample errors.

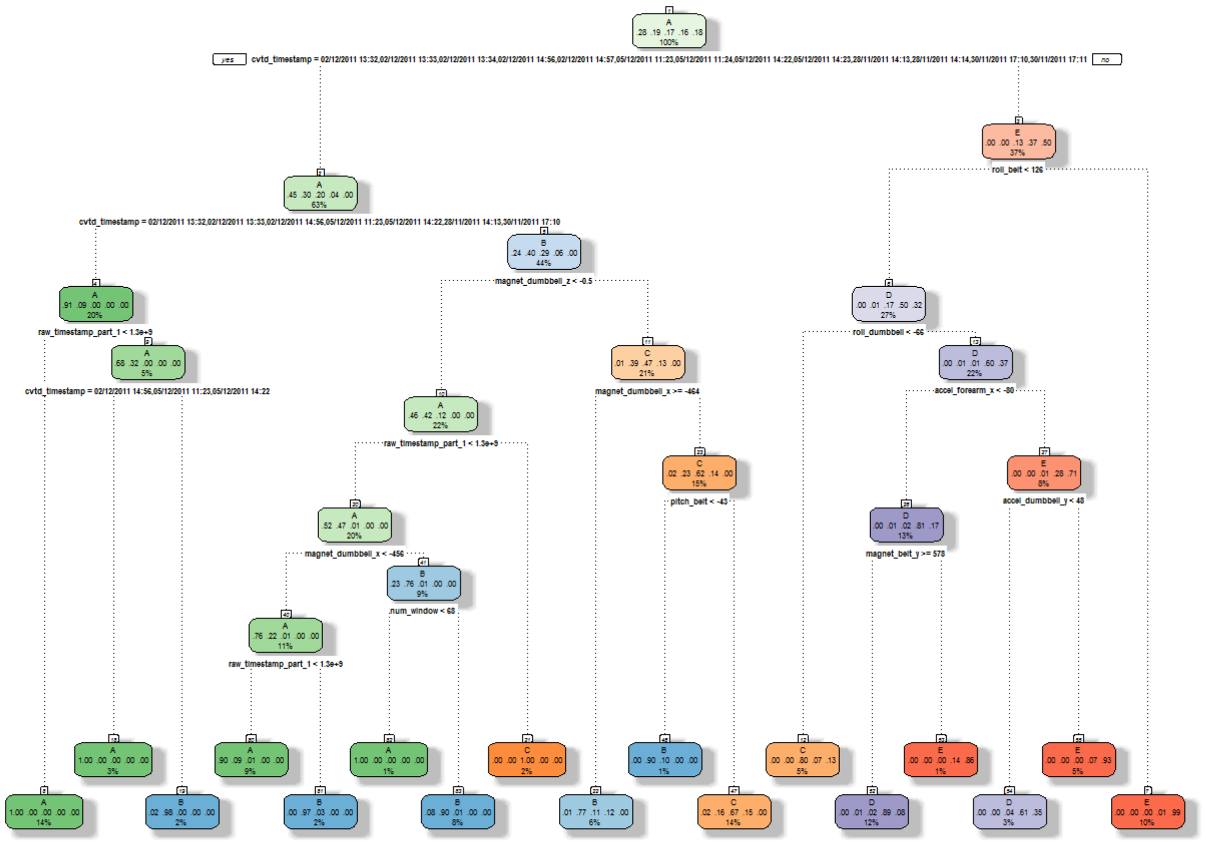
```
train <- createDataPartition(training$classe,p=0.6,list=FALSE)
trainset <- training[train, ]
testset <- training[-train, ]


trainset <- trainset[c(-1)]
clean1 <- colnames(trainset)
clean2 <- colnames(trainset[, -58]) #remove classe column
testset <- testset[clean1]
testing <- testing[clean2]
```

Coerce to the same type

```
for (i in 1:length(testing) ) {
  for(j in 1:length(trainset)) {
    if( length( grep(names(trainset[i]), names(testing)[j]) ) == 1)  {
      class(testing[j]) <- class(trainset[i])
    }
  }
}
testing <- rbind(trainset[2, -58] , testing)
testing <- testing[-1,]
```

Decision tree prediction

```
model1 <- rpart(classe ~ ., data=trainset, method="class")
fancyRpartPlot(model1)
```



Rattle 2016-Nov-02 08:15:16 Shaghayegh

Predict validation set, decision tree

```
predict1 <- predict(model1, testset, type = "class")
(conf1 <- confusionMatrix(predict1, testset$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2154   63    6    6    0
##          B   60 1271  100   65    0
##          C   18  176 1240  207   58
##          D    0    8   22  953  193
##          E    0    0    0   55 1191
## 
## Overall Statistics
##                                           
##                Accuracy : 0.8678          
##                  95% CI : (0.8601, 0.8753)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.8329          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9651   0.8373   0.9064   0.7411   0.8259
## Specificity            0.9866   0.9644   0.9291   0.9660   0.9914
## Pos Pred Value         0.9664   0.8496   0.7298   0.8104   0.9559
## Neg Pred Value         0.9861   0.9611   0.9792   0.9501   0.9620
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2745   0.1620   0.1580   0.1215   0.1518
## Detection Prevalence   0.2841   0.1907   0.2165   0.1499   0.1588
## Balanced Accuracy      0.9758   0.9009   0.9178   0.8535   0.9087
```

```
(accuracy1<- conf1$overall[1])
```

```
##  Accuracy 
## 0.8678307
```

the accuracy rate is 0.87, now lets try random forest method and compare the results.

Random Forest prediction

```
model2 <- randomForest(classe ~ ., data=trainset, method="class")
```

Predict validation set, random forest

```
predict2 <- predict(model2, testset, type = "class")
(conf2 <- confusionMatrix(predict2, testset$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    2    0    0    0
##          B    2 1516    1    0    0
##          C    0    0 1362    8    0
##          D    0    0    5 1278    1
##          E    0    0    0    0 1441
## 
## Overall Statistics
## 
##                Accuracy : 0.9976
##                  95% CI : (0.9962, 0.9985)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9969
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9987   0.9956   0.9938   0.9993
## Specificity            0.9996   0.9995   0.9988   0.9991   1.0000
## Pos Pred Value         0.9991   0.9980   0.9942   0.9953   1.0000
## Neg Pred Value         0.9996   0.9997   0.9991   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1932   0.1736   0.1629   0.1837
## Detection Prevalence   0.2845   0.1936   0.1746   0.1637   0.1837
## Balanced Accuracy      0.9994   0.9991   0.9972   0.9964   0.9997
```

```
(accuracy2<- conf2$overall[1])
```

```
##  Accuracy 
## 0.9975784
```

as we can see, the random forest method is much better than decision tree method. The accuracy rate is 0.99.

Prediction on testing set

```
predict(model2, testing, type = "class")
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 
##  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
```