Model

Ronaldo Echeverria

20/2/2022

Prediction

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Preparing Data

```
## Loading required package: ggplot2

## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.1.0

##
## 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
```

Drop columns with na values in test data, this colums will drop in train data.

```r
#replace NA values
colNames <- names(test)
quitar <- c("X")

for(i in 1:length(colNames)){
  if(sum(is.na(test[colNames[i]]))>0){
    quitar <-c(quitar, colNames[i])
  }
}

training <- training[ , !(names(training) %in% quitar)]
test <- test[ , !(names(test) %in% quitar)]
```

Create model

Generate model by 70% to training and 30% to test.

```r
entrenamiento <- createDataPartition(training$classe, p=0.7, list = FALSE)
trainModel <- training[entrenamiento,]
trainTest <- training[-entrenamiento,]

modelo <- train(classe ~ ., data = trainModel, method = "rf", trControl = trainControl(method = "cv", 5

modelo
```

```
## Random Forest
##
## 13737 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10989, 10988, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9724099  0.9650774
##   41    0.9979616  0.9974217
##   80    0.9965787  0.9956725
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 41.
```

Prediction

Get a predicion by test data.

```r
predTrain <- predict(modelo, trainTest)
confusionMatrix(trainTest$classe, predTrain)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    2 1136    1    0    0
##          C    0    0 1025    1    0
##          D    0    0    3  958    3
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   1.0000   0.9961   0.9979   0.9972
## Specificity            1.0000   0.9994   0.9998   0.9988   0.9998
## Pos Pred Value         1.0000   0.9974   0.9990   0.9938   0.9991
## Neg Pred Value         0.9995   1.0000   0.9992   0.9996   0.9994
## Prevalence             0.2848   0.1930   0.1749   0.1631   0.1842
## Detection Rate         0.2845   0.1930   0.1742   0.1628   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9994   0.9997   0.9980   0.9983   0.9985
```

```
exactitud <- postResample(predTrain, trainTest$classe)
exactitud
```

```
##  Accuracy     Kappa
## 0.9981308 0.9976356
```

```
predTest <- predict(modelo, test)
predTest
```

```
##  [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
```