Machine Learning Project

Toto Kamari

12/12/2020

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. More information is available from the website here: 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).

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Data

```
download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',destfile = './pml-training.cs
v', method = 'curl')

training <- read.csv('pml-training.csv')

download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',destfile = './pml-test.csv', m
ethod = 'curl')

testing <- read.csv('pml-test.csv')

library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

```
## Loading required package: lattice
```

The variable we are trying to predict is “classe”. Let’s first start removing variables that don’t have enough variance.

```
removevars <- nearZeroVar(training)
training <- training[,-removevars]
testing <- testing[,-removevars]
```

Let’s now remove any variables that have significant number of NA’s in them.

```
training <- training[,colSums(is.na(training)) < .05*nrow(training)]
testing <- testing[,colSums(is.na(testing)) < .05*nrow(testing)]

training <- training[,-(1:6)]
testing <- testing[,-(1:6)]
```

Running Prediction Models

Let's break the training data set into a training and validation set.

```
intrain <- createDataPartition(y=training$classe, p=.75, list=FALSE)
training <- training[intrain,]
validation <- training[-intrain,]
```

Now let's start out with a random forest.

```
set.seed(2184)

modfitone <- train(classe~., data=training,method="rf",trControl=trainControl(method="cv",number=5))
print(modfitone$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.65%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4181    2    1    0    1 0.0009557945
## B   20 2821    6    1    0 0.0094803371
## C    0   14 2546    7    0 0.0081807557
## D    0    2   27 2380    3 0.0132669983
## E    0    2    3    6 2695 0.0040650407
```

And now we'll predict on the validation set.

```
modfitone_prediction <- predict(modfitone, newdata=validation)
confusionMatrix(modfitone_prediction, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1016    0    0    0    0
##          B    0  763    0    0    0
##          C    0    0  611    0    0
##          D    0    0    0  609    0
##          E    0    0    0    0  664
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2774
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2774   0.2083   0.1668   0.1663   0.1813
## Detection Rate         0.2774   0.2083   0.1668   0.1663   0.1813
## Detection Prevalence   0.2774   0.2083   0.1668   0.1663   0.1813
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Let us now use the boosting method.

```
set.seed(2184)

modfittwo <- train(classe~., data=training,method="gbm",verbose=FALSE,
                   trControl=trainControl(method="repeatedcv",number=5))
print(modfittwo$finalModel)
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

And now we'll predict on the validation set.

```
modfittwo_prediction <- predict(modfittwo, newdata=validation)
confusionMatrix(modfittwo_prediction, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1010   21    0    1    3
##          B    6  723   15    2    6
##          C    0   18  589   15    7
##          D    0    0    4  591    4
##          E    0    1    3    0  644
##
## Overall Statistics
##
##                Accuracy : 0.9711
##                  95% CI : (0.9651, 0.9762)
##     No Information Rate : 0.2774
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9634
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9476   0.9640   0.9704   0.9699
## Specificity            0.9906   0.9900   0.9869   0.9974   0.9987
## Pos Pred Value         0.9758   0.9614   0.9364   0.9866   0.9938
## Neg Pred Value         0.9977   0.9863   0.9927   0.9941   0.9934
## Prevalence             0.2774   0.2083   0.1668   0.1663   0.1813
## Detection Rate         0.2757   0.1974   0.1608   0.1613   0.1758
## Detection Prevalence   0.2826   0.2053   0.1717   0.1635   0.1769
## Balanced Accuracy      0.9923   0.9688   0.9754   0.9839   0.9843
```

Final Prediction

It appears we get better results using the random forest method. So now we will use that on our test set.

```
modfitfinal_prediction <- predict(modfitone, newdata=testing)
modfitfinal_prediction
```

```
##  [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
```