Machine Learning Assignment

EC

2/10/2020

Summary

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 (see the section on the Weight Lifting Exercise Dataset).

The goal of your project is to predict the manner in which the test subjects did the exercise. We preprocess the data, fit a random forest model and test the overall accuracy reached.

Data loading and preprocessing

First, we download and load the files. Then, we exclude the first 7 variables (containing metadata) and all variables that contains more than 50% NAs or empty values, or with near zero variance.

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "training.csv",me
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "testing.csv",metho

training <- read.csv("training.csv")
testing <- read.csv("testing.csv")


library(dplyr)
library(caret)
training <- training[,-nearZeroVar(training)]
training <- training %>%
  select(-(1:7)) %>%
  select(where(function(x) {mean(is.na(x)) < 0.5})) %>%
  select(where(function(x) {mean(x=="") < 0.5}))
```

We then further split the training database in a training and testing subsample.

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

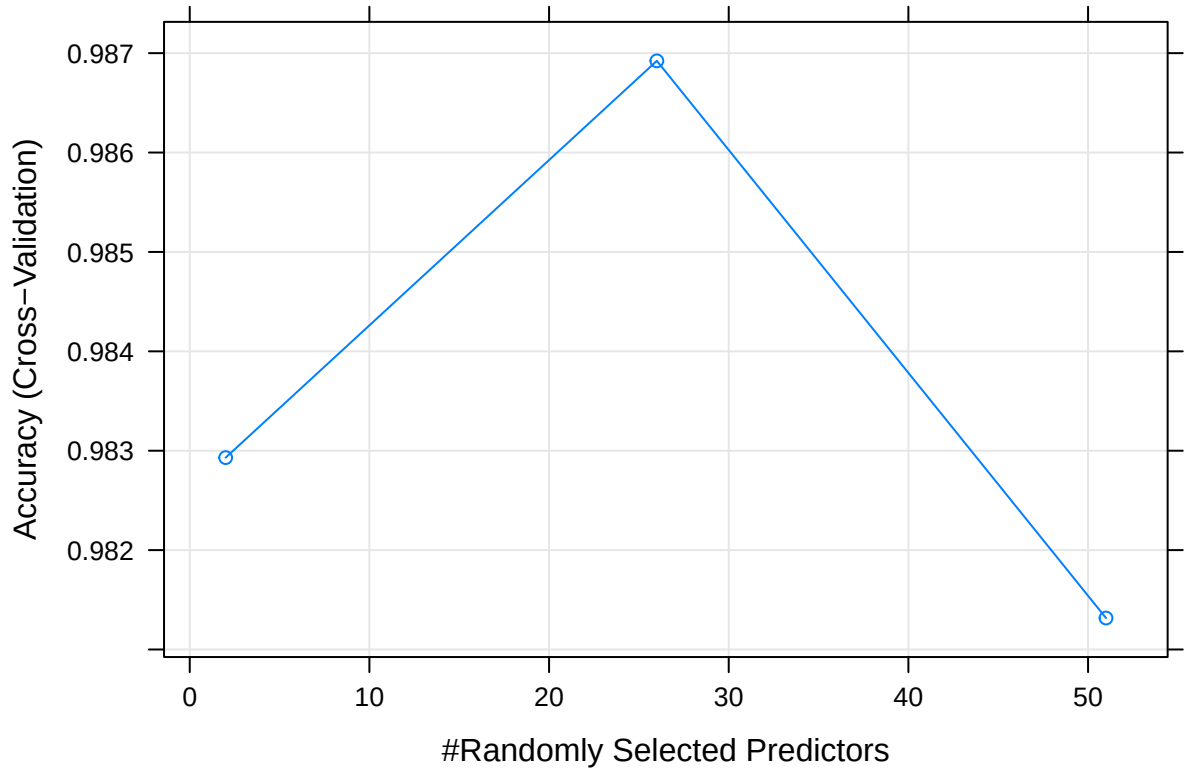
Machine learning training

Before training the model, we set the seed. We will use a random forest model, because of the categorical nature of the outcome variable, with k-fold cross-validation.

```
library(parallel)
library(doParallel)
set.seed(48)
cluster <- makeCluster(7)
registerDoParallel(cluster)
trControl <- trainControl(method = "cv", number = 3, verboseIter=FALSE, allowParallel = TRUE)

trainFit <- train(classe ~ ., data = train,model="rf", trControl = trControl)
stopCluster(cluster)
```

```
plot(trainFit)
```

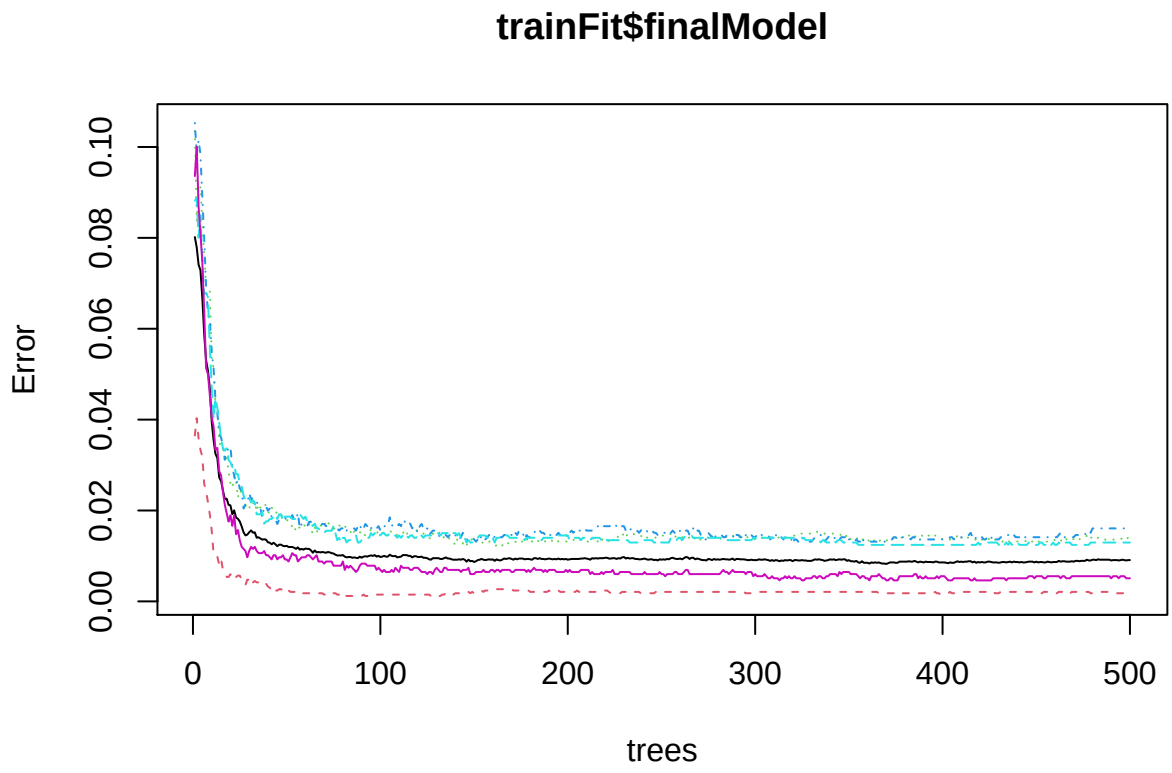



```
trainFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "rf")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
```

```
##
##         OOB estimate of  error rate: 0.91%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3342    4    1    1    0 0.001792115
## B   23 2248    7    0    1 0.013602457
## C    0   25 2020    8    1 0.016553067
## D    0    0   22 1905    3 0.012953368
## E    0    0    4    7 2154 0.005080831
```

```
plot(trainFit$finalModel)
```



The final model seems to have an acceptable error rate. We now see how it performs on the test portion of the data.

```
predictions <- predict(trainFit,test)
confusionMatrix(predictions,as.factor(test$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   18    0    0    0
##          B    4 1494   10    0    0
##          C    0    6 1354   18    0
##          D    0    0    4 1268    6
```

```
##          E    1    0    0    0 1436
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9892, 0.9934)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9842   0.9898   0.9860   0.9958
## Specificity            0.9968   0.9978   0.9963   0.9985   0.9998
## Pos Pred Value         0.9920   0.9907   0.9826   0.9922   0.9993
## Neg Pred Value         0.9991   0.9962   0.9978   0.9973   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1904   0.1726   0.1616   0.1830
## Detection Prevalence   0.2861   0.1922   0.1756   0.1629   0.1832
## Balanced Accuracy      0.9973   0.9910   0.9930   0.9922   0.9978
```

We reached an accuracy of 99.2%, which can be considered acceptable for this project.

Validation

Finally, we use the prediction model on the testing dataframe.

```
predictions2 <- predict(trainFit,testing)
predictions2
```

```
##  [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
```