Machine Learning Prediction Assignment

Pirkko M.

Thursday, February 19, 2015

Summary:

Based on data from: http://groupware.les.inf.puc-rio.br/har. Test subjects were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The task was to predict the manner in which they did the exercise (the "classe" variable in the training set).

The model I ended up using had two variations, as the data had different variables available based on value of "new_window"-variable.

Accuracy of the model with "new_window" as "yes" was in sample: 0.912 and out sample: 0.796

Accuracy of the model with "new_window" as "no" was

in sample: 0.997 and out sample: 0.993

Exploratory data analysis of "training"-data:

run str(training), names(training)

Problems with this many variables that it takes forever to check the correlations within each pairing. In real life, would still do some of that, but don't have time know. Moving on.

Some data processing:

There was a problem with code for NA:s in the data:it can be NA, "#DIV/0!", or " ", or "", also with factors: 0 level was market either 0.00 or 0.0000. I took care of that. (using for instance

```
training [training == "#DIV/0!"| training == " "| training == ""] <- NA
```

I also made sure numeric variables were numeric and not factors:

```
train_index <-as.vector(which(sapply(training, nlevels)>10))
fac_to_num <- names(which(sapply(training, nlevels)>10))

training3 <- training

for (i in train_index[train_index>7]){
      x<- sapply(training[,i],as.character)
      training3[,i]<- sapply(x,as.numeric)
}
```

Partitioning for cross-validation

I then partitioned the training data into training "train" and testing "test0" set (60:40):

```
inTrain <- createDataPartition(y = training3$classe, p=0.60, list= FALSE)
train <- training3[inTrain,]
test0 <- training3[-inTrain,]
```

Then i got rid of factors with only one level

```
isfactor <- sapply(train, is.factor)
badfactor <- sapply(train, nlevels)
train2 <- train[, which (badfactor >1 | isfactor ==  FALSE)] #left 152
variables
```

and column that only had NA's

```
nas <- colSums(sapply(train2, is.na))
noNas <- nrow(train2)-nas
```

there seem to be about 2 groups here: 243 obs., 11776 obs.#this seems to depend new_window being 0 or 1

Let's also get rid of variable no 4 for group1 (cvtd timestamp, which is all NA)

```
group1 <- train2[,c(1:4, 6:7, which(noNas < 500 & noNas>0), ncol(train2) )] #
99 variables(classe+ 7 + 91)
group2 <- train2[,which(noNas > 500)] #60 variables (classe + 7 +52)
```

Then I took only complete cases for each group into g1 and g2:

G1: 124 obs. of 98 variables, G2: 11527 obs. of 68 variables

```
g1_comp <- g1[complete.cases(g1),]
g2_comp <- g2[complete.cases(g2),]
```

Model building

Then I built the models (without timestamps etc. which seem to be specific to this specific data, rownumbers etc. "X""raw_timestamp_part_1" "raw_timestamp_part_2" "num_window")

```
fitG1 <- randomForest(classe~., data = g1_comp[,-c(1,3,4,5)])
fitG2 <- randomForest(classe~., data = g2_comp[,-c(1,3,4,5,6)])
```

I made "model"-function that did the same data processing for any data and run the fitG1 and fitG2 according to new_window- variable.

```
model <- function(data){
      data [data == "#DIV/0!"| data == " "| data == ""] <- NA
      data [as.character(data) == "#DIV/0!"| as.character(data) == "
"|as.character(data) == ""] <- NA
      data[as.character(data) == "0.00"| as.character(data) ==
"0.000"|as.character(data) == "0.0000"] <- "0.00"
      data <- droplevels(data)
```

```
      for (i in fac_to_num){
            x<- sapply(data[,i],as.character)
            data[,i]<- sapply(x,as.numeric)
      }



#

      data1 <- subset (data, data$new_window == "yes")[,
                        intersect(names(subset (data, data$new_window ==
"yes")),
                                            names(g1_comp[,-c(1,3:5)]))]

if (nrow(data1)>0){
      #which columns only have only na.s
      nas1 <- colSums(sapply(data1, is.na))
      noNas1 <- nrow(data1)-nas1
      data1 <- data1[,which(noNas1 > 0)] # no all-Na variables

      data1 <- data1[complete.cases(data1),]
      if (nrow(data1)>0){
            pred1 <- predict(fitG1,data1, na.action = na.omit)
            predRight1 <- pred1 == data1[,ncol(data1)]
      print(sum(predRight1)/nrow(data1)) #accuracy
      print(1-sum(predRight1)/nrow(data1)) #error rate
            print(table (pred1, data1[,ncol(data1)])) #print table for accuracy
calculation
      }
}

      data2 <- subset (data, data$new_window == "no")[,
                  c(intersect(names(subset (data, data$new_window == "no")),
                  names(g2_comp[,-c(1,3:6)])), names(data)[ncol(data)])]
if (nrow(data2)>0){

      #which columns only have only na.s
      nas2 <- colSums(sapply(data2, is.na))
      noNas2 <- nrow(data2)-nas2
      data2 <- data2[,which(noNas2 > 0)] # no all-Na variables

      data2 <- data2[complete.cases(data2),]

      if (nrow(data2)>0){
      pred2 <- predict(fitG2,data2, na.action = na.omit)
      predRight2 <- pred2 == data2[,ncol(data2)]
      print(sum(predRight2)/nrow(data2)) #accuracy
      print(1-mean(predRight2)) #error rate
      print(table (pred2, data2[,ncol(data2)]))#print table for accuracy
calculation
      #return(as.character(pred2)) #the predictions
      }#data2 2nd if
#       print(head(data1))
```

```
#       print(str(data1))

} #data2 first if
} #model
```

Results

```
model(training)
```

```
## [1] 0.9124424
## [1] 0.0875576
##
## pred1  A  B  C  D  E
##     A 50  5  4  1  1
##     B  3 43  0  1  3
##     C  0  0 33  0  0
##     D  1  0  0 33  0
##     E  0  0  0  0 39
## [1] 0.9971378
## [1] 0.002862198
##
## pred2    A    B    C    D    E
##     A 5469    7    0    0    0
##     B    1 3709   15    0    0
##     C    0    2 3336   19    1
##     D    0    0    1 3127    7
##     E    1    0    0    1 3520
```

```
model(test0)
```

```
## [1] 0.9142857
## [1] 0.08571429
##
## pred1  A  B  C  D  E
##     A 16  2  2  0  0
##     B  2 24  0  1  2
##     C  0  0 17  0  0
##     D  0  0  0 18  0
##     E  0  0  0  0 21
## [1] 0.9971294
## [1] 0.002870564
##
## pred2    A    B    C    D    E
##     A 2186    1    0    0    0
##     B    1 1480    4    0    0
##     C    0    1 1329    7    0
##     D    0    0    1 1248    5
##     E    1    0    0    1 1399
```

In of sample accuracy (model(training)):

Group 1: accuracy 0.912, Group 2: accuracy 0.997 Group 1: in of sample error rate 0.088, Group 2: in of sample error rate 0.003

Testing with data set test0 (model(test0)) (for cross-validation):

Group 1: accuracy 0.796, Group 2: accuracy 0.993 Group 1: out of sample error rate 0.01, Group 2: out of sample error rate 0.204

(got all the testing cases correct in submission-part)

Here we can see how the model is over-fitted for Group 1 (fitG1), which is not surprising as 124 obs. to fit model of 93 variables is ridiculous. Would probably be a good idea to do some pre-processing to cut down the number of variables, maybe PCA as well. No time for that :(