Practical Machine Learning Prediction Assignment

Troy Huffman

4/19/2022

Model Building

'Classe' is a factor variable that corresponds to the following exercise methods:

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)

throwing the hips to the front (Class E)

Class A is the correct exercise method whereas the other classes are commonly made mistakes. The prediction model will be based on maximizing the accuracy and minimizing the out-of-sample error. All available variables after cleaning will be used for prediction. Random Forest algorithm will be used.

Cross-validation

Cross-validation will be performed by subsampling our training data set randomly without replacement according to training data (70% of the original data set) and testing data (30%). The Random Forest algorithm will be fitted on the training data set and tested on the testing data.

Expected Out-Of-Sample Error

The expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the test data set. Therefore the expected out-of-sample error is calculated by 1 - Accuracy (as reported from the cross-validation data set).

```
data <- read.csv("pml-training.csv")
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(randomForest)

## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed(411)
#partioning the training data into train and test
train <- createDataPartition(y=data$classe,p=.70,list=F)
training <- data[train,]
testing <- data[-train,]

Cl <- grep("name|timestamp|window|X", colnames(training), value=F)
trainingCl <- training[,-Cl]
#excluding variables with excessive missing data
trainingCl[trainingCl==""] <- NA
highNA <- apply(trainingCl, 2, function(x) sum(is.na(x)))/nrow(trainingCl)
trainingCl <- trainingCl[!(highNA>0.95)]
trainingCl$classe = factor(trainingCl$classe)

#performing principal component analysis
preProc <- preProcess(trainingCl[,1:52],method="pca",thresh=.8)
preProc <- preProcess(trainingCl[,1:52],method="pca",thresh=.9)
preProc <- preProcess(trainingCl[,1:52],method="pca",thresh=.95)
preProc <- preProcess(trainingCl[,1:52],method="pca",pcaComp=25)
preProc$rotation

trainingPC <- predict(preProc,trainingCl[,1:52])

modFitRF <- randomForest(trainingCl$classe ~ .,   data=trainingPC,
do.trace=F)
print(modFitRF)

##
## Call:
##  randomForest(formula = trainingCl$classe ~ ., data = trainingPC,
do.trace = F)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 2.36%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3871   11   10   11    3 0.008960573
## B   33 2580   42    1    2 0.029345372
## C    5   29 2336   23    3 0.025041736
## D    5    3   99 2141    4 0.049289520
## E    1   11   14   14 2485 0.015841584
```

```r
#running model on partitioned test data
testingCl <- testing[,-Cl]
testingCl[testingCl==""] <- NA
NArate <- apply(testingCl, 2, function(x) sum(is.na(x)))/nrow(testingCl)
testingCl <- testingCl[!(NArate>0.95)]
testingPC <- predict(preProc,testingCl[,1:52])
testingCl$classe = factor(testingCl$classe)
confusionMatrix(testingCl$classe,predict(modFitRF,testingPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1656    5   10    2    1
##          B   29 1084   23    0    3
##          C    0   16  995   15    0
##          D    2    0   32  928    2
##          E    0    1   12    7 1062
##
## Overall Statistics
##
##                Accuracy : 0.9728
##                  95% CI : (0.9683, 0.9768)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9656
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9816   0.9801   0.9282   0.9748   0.9944
## Specificity            0.9957   0.9885   0.9936   0.9927   0.9958
## Pos Pred Value         0.9892   0.9517   0.9698   0.9627   0.9815
## Neg Pred Value         0.9926   0.9954   0.9842   0.9951   0.9988
## Prevalence             0.2867   0.1879   0.1822   0.1618   0.1815
## Detection Rate         0.2814   0.1842   0.1691   0.1577   0.1805
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9887   0.9843   0.9609   0.9837   0.9951
```

###Discussion: The model showed an overall accuracy of 97% for the testing set. The model was used for the course project prediction quiz and correctly identified 18 of the 20 test cases.