Statistical Machine Learning(STAT30270)

Nisarga G-19203753

05/04/2020

Objective: the central goal is to obtain a machine learning model which will perform well at predicting the target variable on new unseen inputs.

Load and prepare the data:The data is divided into test(dat_test) and train sets,validation set (dat).

```
#install.packages("mlbench")
library(mlbench)
data("Satellite")

# this will re-order alphabetically class labels and remove spacing
Satellite$classes <- gsub(" ", "_", Satellite$classes)
Satellite$classes <- factor( as.character(Satellite$classes) )

# to have the same initial split
set.seed(777222)
D <- nrow(Satellite)
keep <- sample(1:D, 5500)
test <- setdiff(1:D, keep)
dat <- Satellite[keep,] #training and validation
dat_test <- Satellite[test,]#testing
```

To predict the classfication of the images we use : Multinomial Logistic Regression: multinomial logistic regression is an extension of the standard logistic regression model to the case where the response variable can take more than two classes. The framework is usually applied for multi-class classification problems. To fit a multinomial logistic regression we can use the function multinom available in the package nnet.

Random Forest : The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. Random forests are implemented in R in the package randomForest via the function with the same name.

```
#load the packages for Random Forest and multinomial regression
#install.packages("nnet")
#install.packages("randomForest")

library(nnet)
library(randomForest)

## randomForest 4.6-14
```

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

Approach for choosing an appropriate classifier is to split the labeled data into three non overlapping sets: Training set: Data points whose labels are used in the classifier fitting procedure. Validation set: Data points used to test each of the classifiers to see which is best.Each classifier in turn is tested on these data. Test set: Data points used to estimate the generalization error of the chosen classifier.

```
# split the training data (dat)
# 75% of dat are used as training data
# 25% of dat are used as validation data
#dat_test will be employed for testing.
set.seed(19203753)
N <- nrow(dat)
train <- sample(1:N, size = 0.75*N)
val <- setdiff(1:N, train)
```

Out of all the cross validation strategies we use Holdout sample because large number of observations is available, usually a simple hold-out procedure works fine.Leave-one-out and k-fold are preferred in the case of not so large sample size.

```
# fit classifiers to only the training data
# multinomial logistic regression
fit1 <- multinom(classes ~ ., data = dat,maxit =300,subset = train)

## # weights:  228 (185 variable)
## initial  value 7391.007811
## iter  10 value 3860.903289
## iter  20 value 3537.522358
## iter  30 value 3510.527341
## iter  40 value 3280.448786
## iter  50 value 2835.688308
## iter  60 value 2722.182126
## iter  70 value 2694.379747
## iter  80 value 2535.118796
## iter  90 value 2311.675804
## iter 100 value 2261.958987
## iter 110 value 2113.800881
## iter 120 value 2061.047315
## iter 130 value 1817.464226
## iter 140 value 1766.868243
## iter 150 value 1679.606858
## iter 160 value 1646.184442
## iter 170 value 1600.334090
## iter 180 value 1526.812314
## iter 190 value 1463.137294
## iter 200 value 1294.877778
## iter 210 value 1270.946642
## iter 220 value 1267.595213
## iter 230 value 1267.284961
```

```
## iter 240 value 1267.223611
## iter 250 value 1267.206280
## iter 260 value 1267.202083
## final  value 1267.201833
## converged

#Random Forest
fit2 <-randomForest(classes~.,data =dat,subset = train)

# classify the validation data observations

#multinomial regression
pred1 <- predict(fit1, type = "class", newdata = dat[val,])
tab1 <- table(dat$classes[val], pred1)
tab1

##                       pred1
##                        cotton_crop damp_grey_soil grey_soil red_soil
##   cotton_crop                  169              1         0        0
##   damp_grey_soil                 0             37        29        2
##   grey_soil                      0             24       268        3
##   red_soil                       0              2         6      298
##   vegetation_stubble             9              1         0        6
##   very_damp_grey_soil            0             28         5        1
##                       pred1
##                        vegetation_stubble very_damp_grey_soil
##   cotton_crop                          12                   0
##   damp_grey_soil                        2                  43
##   grey_soil                             0                   1
##   red_soil                              5                   0
##   vegetation_stubble                  126                  21
##   very_damp_grey_soil                   7                 269

#Accuracy
acc1 <- sum(diag(tab1))/sum(tab1)

#Random forest
pred2 <- predict(fit2, type = "class", newdata = dat[val,])
tab2 <- table(dat$classes[val], pred2)
tab2

##                       pred2
##                        cotton_crop damp_grey_soil grey_soil red_soil
##   cotton_crop                  179              1         0        0
##   damp_grey_soil                 0             68        26        1
##   grey_soil                      0              8       282        2
##   red_soil                       1              0         2      302
##   vegetation_stubble             2              1         0        3
##   very_damp_grey_soil            0             17         7        0
##                       pred2
```

```
##                       vegetation_stubble very_damp_grey_soil
##   cotton_crop                         1                    1
##   damp_grey_soil                      0                   18
##   grey_soil                           0                    4
##   red_soil                            6                    0
##   vegetation_stubble                143                   14
##   very_damp_grey_soil                 5                  281
```

```
#Accuracy
acc2 <- sum(diag(tab2))/sum(tab2)

# print accuracy
acc <- c(Multinomial_regression = acc1, random_forest = acc2)
acc
```

```
## Multinomial_regression          random_forest
##              0.8487273              0.9127273
```

```
#to assess the general performance of a model, we aim to estimate its general
prediction error,  which is denoted as "generalization error".
# use the method that did best on the validation data
# to predict the test data
best <- names( which.max(acc) )
switch(best,
       Multinomial_regression = {
         predTestml <- predict(fit1, type = "class", newdata = dat_test)
         tabTestml <- table(dat_test$classes, predTestml)
         accBest <- sum(diag(tabTestml))/sum(tabTestml)
       },
       random_forest = {
         predTestrf <- predict(fit2, type = "class", newdata = dat_test)
         tabTestrf <- table(dat_test$classes, predTestrf)
         accBest <- sum(diag(tabTestrf))/sum(tabTestrf)
       }
)
best
```

```
## [1] "random_forest"
```

```
accBest
```

```
## [1] 0.9026738
```

It is good practice to replicate the procedure of training-validation-testing a number of times to account for sampling variation and assess uncertainity.

```
# replicate the process a number of times
R <- 100
out <- matrix(NA, R, 4)
colnames(out) <- c("val_mul_log", "val_rf", "best", "test")
out <- as.data.frame(out)
```

```r
for ( r in 1:R ) {

  # split the data
  train <- sample(1:N, size = 0.75*N)             # 75% of data points are
used as training data
  val <- setdiff(1:N,train)                 # 25% of data points are used as
val data

  # fit classifiers to only the training data
  fitrf <-randomForest(classes~.,data =dat,subset = train)    #
classification tree
  fitLog <- multinom(classes ~ ., data = dat,maxit =300,subset = train)#
multinomial logistic regression

  # classify the validation data observations
  predValrf <- predict(fitrf, type = "class", newdata = dat[val,])    #
classification tree
  tabValrf <- table(dat$classes[val], predValrf)
  #tabValCt
  accRf <- sum(diag(tabValrf))/sum(tabValrf)
  #
  predValLog <- predict(fitLog, type = "class", newdata = dat[val,])  #
logistic regression
  tabValLog <- table(dat$classes[val], predValLog)
  #tabValLog
  accLog <- sum(diag(tabValLog))/sum(tabValLog)

  # accuracy
  acc <- c(random_forest = accRf, logistic = accLog)
  out[r,1] <- accRf
  out[r,2] <- accLog

  # use the method that did best on the validation data
  # to predict the test data
  best <- names( which.max(acc) )
  switch(best,
         random_forest = {
           predTestrf <- predict(fitrf, type = "class", newdata = dat_test)
           tabTestrf<- table(dat_test$classes, predTestrf)
           accBest <- sum(diag(tabTestrf))/sum(tabTestrf)
         },
         logistic = {
           predTestLog <- predict(fitLog, type = "class", newdata = dat_test)
           tabTestLog <- table(dat_test$classes, predTestLog)
           accBest <- sum(diag(tabTestLog))/sum(tabTestLog)
         }
  )
  out[r,3] <- best
```

```
  out[r,4] <- accBest

  print(r)
}
```

The output of the above code has been deleted to make the report concise.

Evaluate the generalization performance of the best model on the test data: to fairly evaluate a classifier performance, the model needs to be tested on data not used in the model fitting procedure.The predictive performance can be equivalently assessed also in terms of accuracy rather than error.

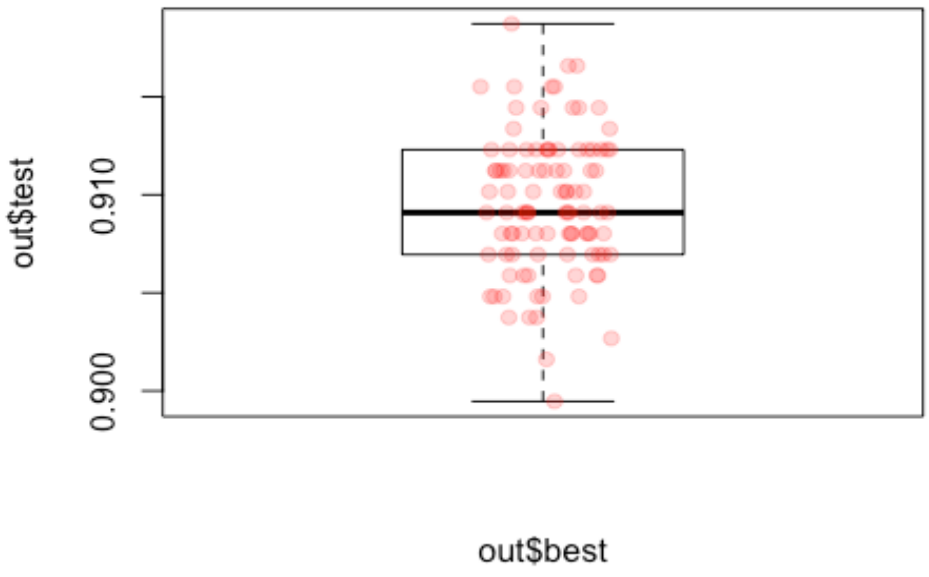
```
# check out the error rate summary statistics
table(out[,3])

##
## random_forest
##           100

tapply(out[,4], out[,3], summary)

## $random_forest
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8995  0.9070  0.9091  0.9096  0.9123  0.9187

boxplot(out$test ~ out$best)
stripchart(out$test ~ out$best, add = TRUE, vertical = TRUE,
           method = "jitter", pch = 19, col = adjustcolor("red", 0.2))
```



From the error rate summary statistics we can infer that when the process is replicated 100 times , every time Random Forest classification was chosen the best method as seen in the output out[,3]. Also from the output of out[,4] which records the accuracy rate of all the 100 repitions we can conclude that the minimum accuracy rate was 89.95% while highest being 91.87% and the typical values of accuracy lies between 90.70%(First Quantile) - 91.23%(Third Quantile). The average accuracy rate is 90.96%.

Interpretation of boxplot:All the accuracy rates recorded in 100 repititions lies between 0.905 and 0.920.The typical values are those points lying inside the box and they range from 0.907-0.9123. Thereby concluding that the accuracy is over 90% .

Comments on the predictive performance of the best classifier:

```
best

## [1] "random_forest"

accBest

## [1] 0.9048128
```

CONCLUSION : The best classifer used to classify images of sattelitte data is “Random Forest” with accuracy over 90%.