Model_Test Assignment: 6 [Group: PRJ 4]

2023-07-14

Installing and Loading the required packages

```
# install.packages("tidyverse")
# install.packages("caret")
# install.packages("data.table")
# install.packages("tidyr")
# install.packages("stringr")
# install.packages("forcats")
# install.packages("ggplot2")
# install.packages("kableExtra")
#install.packages("kernlab")
#install.packages("gam")
#install.packages("kknn")
#install.packages("igraph")
#install.packages("randomForest")
#install.packages("RcppEigen")
#install.packages("ranger")
#install.packages("wsrf")
#install.packages("RSNNS")

# Loading all needed libraries
library(kernlab)
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.22-2
```

```
library(kknn)
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
library(RcppEigen)
library(ranger)
```

```
##
## Attaching package: 'ranger'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(wsrf)
```

```
## Loading required package: parallel
```

```
## Loading required package: Rcpp
```

```
## wsrf: An R Package for Scalable Weighted Subspace Random Forests.
```

```
## Version 1.7.30
```

```
## Use C++ standard thread library for parallel computing
```

```
##
## Attaching package: 'wsrf'
```

```
## The following object is masked from 'package:ranger':
##
##     importance
```

```
## The following objects are masked from 'package:randomForest':
##
##     combine, importance
```

```
## The following object is masked from 'package:igraph':
##
##     strength
```

```
library(RSNNS)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:wsrf':
##
##     combine

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v ggplot2   3.4.2     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%--%()       masks igraph::%--%()
## x purrr::accumulate()     masks foreach::accumulate()
## x ggplot2::alpha()        masks kernlab::alpha()
## x tibble::as_data_frame() masks dplyr::as_data_frame(), igraph::as_data_frame()
## x dplyr::combine()        masks wsrf::combine(), randomForest::combine()
## x purrr::compose()        masks igraph::compose()
## x purrr::cross()          masks kernlab::cross()
## x tidyr::crossing()       masks igraph::crossing()
## x dplyr::filter()         masks stats::filter()
## x dplyr::lag()            masks stats::lag()
## x ggplot2::margin()       masks randomForest::margin()
## x purrr::simplify()       masks igraph::simplify()
## x purrr::when()           masks foreach::when()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
##
## The following objects are masked from 'package:RSNNS':
##
##     confusionMatrix, train
##
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)
library(splines)
library(foreach)
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
##
## Attaching package: 'mgcv'
##
## The following objects are masked from 'package:gam':
##
##     gam, gam.control, gam.fit, s
```

```r
library(nlme)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

Loading the Heart dieseas(HD) Dataset

```r
HD <- read_csv("C:/Users/yashp/Downloads/archive/Hearts.csv",show_col_types = FALSE)

# changing the column name for making it more descriptive.

names(HD) <- c("age", "sex", "chest_pain_type", "resting_blood_pressure",
                   "cholesterol", "fasting_blood_sugar", "rest_ecg",
                   "max_heart_rate_achieved", "exercise_induced_angina",
                   "st_depression", "st_slope", "num_major_vessels",
                   "thallium_stress_test", "disease")

# converts disease as  "0" or "1" for using it with confusionMatrix()

HD <- mutate_at(HD, vars(disease), as.factor)
```

Spliting HD dataset in HDX and validation sets

```r
set.seed(1)
# Validation set will be 20% of HD data because it is a little dataset

test_index <- createDataPartition(y = HD$disease,
                                  times = 1, p = 0.2, list = FALSE)
HDX <- HD[-test_index,]
validation <- HD[test_index,] # we will use it only to do final test
```

Spliting HDX dataset in train_set and test_set

```
set.seed(1)

test_index <- createDataPartition(y = HDX$disease,
                                  times = 1, p = 0.2,
                                  list = FALSE)  # test_set 20%
train_set <- HDX[-test_index,]
test_set <- HDX[test_index,]


#  We will use it to train ours models
```

List of models and usinf trainControl function for tuning parameter

```
# model list
models <- c("glm", "lda", "naive_bayes", "svmLinear",
            "gamLoess", "knn", "kknn","gam",
            "rf", "ranger", "wsrf", "mlp")

# using trainControl function for control tuning parameters of models

control <- trainControl(method = "cv",   # cross validation
                        number = 10,     # 10 k-folds or number
                        # of resampling iterations
                        )
```

Initializing varialble and using loop

```
train_data <- train_set
test_data <-  test_set
correct_value <- test_set$disease # Correct outcome from test_set

# loop to use train and test set first, then HDX, then validation

for(i in 1:2) {
  fits <- lapply(models, function(model){
    #    print(model)  # it´s used to debug code
    set.seed(1)
    train(disease ~ .,
          method = model,
          preProcess=c("center", "scale"),   # to normalize the data
          data = train_data,
          trControl = control)
  })

names(fits) <- models

  # to be sure that the actual value of the output do not have influence on the prediction
```

```r
vali2 <- test_data %>% select(-disease)

pred <- sapply(fits, function(object) # predicting outcome
predict(object, newdata = vali2))

  # avg predicted values if equals to true values

  if (i == 1) acc <- colMeans(pred == correct_value)

  train_data <- HDX               # last value for data parameter
  test_data <-  validation        # last we´ll use HDX and validation
  correct_value <- validation$disease  # true outcome from validation set

}
```

```
## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : eval 3.8858

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : upperlimit 3.1512

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : eval -2.9145

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : lowerlimit -2.2719

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval 2.0556

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : upperlimit 1.8568

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval 2.3656

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : upperlimit 1.8568

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending
```

```
## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : eval -2.2613

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : lowerlimit -1.9254

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## eval 4.925

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## upperlimit 3.04

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : eval 2.4286

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : upperlimit 2.3458

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, : eval
## -2.2747

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## lowerlimit -2.2207

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval -2.5864

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : lowerlimit -2.5248

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, : eval
## 6.3464

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## upperlimit 3.448
```

```
## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : eval 4.2534

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : upperlimit 3.7906

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval -3.295

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : lowerlimit -2.5985

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : eval 2.402

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : upperlimit 2.3192

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval -3.4077

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : lowerlimit -2.6918

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## eval 4.692

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## upperlimit 2.8778

## Warning in gam.lo(data[["lo(st_depression, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : eval -2.8135
```

```
## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : lowerlimit -2.2875

## Warning in gam.lo(data[["lo(age, span = 0.5, degree = 1)"]], z, w, span = 0.5,
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : eval 2.2588

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : upperlimit 1.9891

## Warning in gam.lo(data[["lo(max_heart_rate_achieved, span = 0.5, degree =
## 1)"]], : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : eval 3.9987

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : upperlimit 3.5613

## Warning in gam.lo(data[["lo(resting_blood_pressure, span = 0.5, degree = 1)"]],
## : extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, : eval
## 6.4971

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## upperlimit 3.5247

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, : eval
## -2.4465

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## lowerlimit -2.3736

## Warning in gam.lo(data[["lo(cholesterol, span = 0.5, degree = 1)"]], z, :
## extrapolation not allowed with blending
```

Results of different models on 2 different datasets: first train/test_set, and edx/validation

```
acc    # all accuracy values with Train and Test set
```

```
##         glm         lda naive_bayes   svmLinear    gamLoess         knn
##   0.7916667   0.7916667   0.8541667   0.7708333   0.8125000   0.8125000
##        kknn         gam          rf      ranger        wsrf         mlp
##   0.8541667   0.7916667   0.8125000   0.7916667   0.8125000   0.7500000
```

```r
acc2 <- colMeans(pred == correct_value) # avg predicted values

acc2   # all accuracy values with HDX and Validation set
```

```
##         glm         lda naive_bayes   svmLinear    gamLoess         knn
##   0.7833333   0.8166667   0.8166667   0.8000000   0.8000000   0.8500000
##        kknn         gam          rf      ranger        wsrf         mlp
##   0.8333333   0.7500000   0.8333333   0.8333333   0.7833333   0.7666667
```

```r
results <- acc2 - acc # accuracy diff by model


results
```

```
##          glm          lda  naive_bayes    svmLinear     gamLoess          knn
## -0.008333333  0.025000000 -0.037500000  0.029166667 -0.012500000  0.037500000
##         kknn          gam           rf       ranger         wsrf          mlp
## -0.020833333 -0.041666667  0.020833333  0.041666667 -0.029166667  0.016666667
```

Computing balance accuracy, sensitivity, specificity,prevalence with confusionMatrix

```r
confmat_val_HD<- confusionMatrix(as.factor(pred[,11]),
                                 validation$disease, positive = "1")
confmat_val_HD
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28  9
##          1  4 19
##
##                Accuracy : 0.7833
##                  95% CI : (0.658, 0.8793)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 5.405e-05
##
##                   Kappa : 0.5598
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.6786
##             Specificity : 0.8750
##          Pos Pred Value : 0.8261
##          Neg Pred Value : 0.7568
##              Prevalence : 0.4667
##          Detection Rate : 0.3167
##    Detection Prevalence : 0.3833
```

```
##       Balanced Accuracy : 0.7768
##
##        'Positive' Class : 1
##
```

Using KNN Algorithm As it is overall the best Algorithm/model/method

```
# to be sure that the actual value of the output has not influence on the prediction

vali02 <- validation %>% select(-disease)

# trainControl function for control iteration model
# we test differents parameters and choose that ones that improve accuracy

control02 <- trainControl(method = "cv",    # cross validation
                          number = 30)    # optimum k-folds or number 30
# of resampling iterations

# training KNN model

set.seed(1)
knn_model <- train(disease ~., data = HDX,
                       method = "knn",  # KNN model
                       preProcess=c("center", "scale"),    # to normalize the data
                       trControl = control02)

# predicting outcome

prediction_knn <- predict(knn_model, newdata = vali02)

# Check results with confusionMatrix() function and validation set

confmat_knn<- confusionMatrix(prediction_knn,
                                    validation$disease, positive = "1")
confmat_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  5
##          1  3 23
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 4.403e-08
##
##                   Kappa : 0.7309
##
##  Mcnemar's Test P-Value : 0.7237
##
```

```
##             Sensitivity : 0.8214
##             Specificity : 0.9062
##          Pos Pred Value : 0.8846
##          Neg Pred Value : 0.8529
##              Prevalence : 0.4667
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.4333
##       Balanced Accuracy : 0.8638
##
##        'Positive' Class : 1
##
```