Liver Disease Classification

Nabeel Khan

16-May-2020

Contents

1 Introduction

This report is part of the 'HarvardX: PH125.9x Data Science: Capstone' course. In this report, we chose a dataset of our choice and apply various machine learning techniques to perform binary classification to diagnose liver disease.

1.1 Background

The liver plays an important role in keeping us healthy. The main job of liver is to filter the blood coming from the digestive tract, before passing it to the rest of the body. The liver also turns nutrients into chemicals our body needs, turns food into energy, and filters out poisons. So, malfunctioing of liver affects the whole body.

The classification techniques are used in various automatic medical diagnoses tools[1].The problems with liver patients are not easily discovered in an early stage. An early diagnosis of liver problems will help in increasing the survival rate of patiets. We can detect the liver disease by analyzing the levels of enzymes in the blood [2]. A classification algorithm capable of automatically detecting the liver disease can assisst the doctors.

1.2 Aim of Project

The patients with liver disease are on the rise because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. The aim of this proect is to develop a bianry classifier, which can use blood enzymes information to diagnose liver disease.

2 Dataset and Evaluation Metrics

We use the Liver Patient Records, which are collected from North East of Andhra Pradesh, India. The data set contains:

1. 416 liver patient records and 167 non-liver patient records.

2.1 Download Data

The dataset is publically available online both at Kaggle and UCI repository. We download data from the website. Then, we split data into a training and validation sets.

- 10

```
################################
#  Install packages (if not installed)
################################
# Note: this process could take a couple of minutes
repos_path<- "http://cran.us.r-project.org"
if(!require(tidyverse)) install.packages("tidyverse", repos =repos_path)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = repos_path)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos =repos_path)
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(lubridate)) install.packages("lubridate", repos = repos_path)
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

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

```
##     date
```

```
if(!require(dplyr)) install.packages("dplyr", repos = repos_path)
if(!require(sjmisc)) install.packages("dplyr", repos = repos_path)
```

```
## Loading required package: sjmisc
```

```
## Learn more about sjmisc with 'browseVignettes("sjmisc")'.
```

```
##
## Attaching package: 'sjmisc'
```

```
## The following object is masked from 'package:purrr':
##
##     is_empty
```

```
## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```
## The following object is masked from 'package:tibble':
##
##     add_case
```

```
if(!require(sjmisc)) install.packages("scales", repos = repos_path)

#################################
# Load libraries
#################################
library(lubridate)
library(tidyverse)
library(dplyr)
library(lubridate)
library(sjmisc)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
#################################
# Downloading data
#################################
# Indian Live Patient Records :
 # https://www.kaggle.com/uciml/indian-liver-patient-records/
 # https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian Liver Patient Dataset (ILPD).

url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian Liver Patient Dataset (I
# Download csv
liverData <- read.csv(url)

# Rename columns of csv
```

```r
colnames(liverData)<- c("Age","Gender","Total_Bilirubin","Direct_Bilirubin", "Alkaline_Phosphotase","Ala

################################
# Creating training and validation sets
################################

# Validation set will be 10% of whole data
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
test_index <- createDataPartition(y = liverData$Dataset, times = 1, p = 0.1, list = FALSE)

training <- liverData[-test_index,]
validation <- liverData[test_index,]

 # Removing the objects from environment as no longer required
rm(liverData)
```

2.2 Metrics

To evaluate the performance of classifiers, we will use following metrics:

1. **Accuracy** It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Truepositives + Truenegatives}{TotalPredictions} \tag{1}$$

2. **Sensitivity** It is also referred as true positive rate or recall. It is the proportion of true positives that are correctly identified.

$$Sensitivity = \frac{Numberoftruepositives}{Numberoftruepositives + Numberoffalsenegatives} \tag{2}$$

3. **Precision** It is defined as the proportion of the true positives against all the positive results.

$$Precision = \frac{Numberoftruepositives}{Numberoftruepositives + Numberoffalsepositives} \tag{3}$$

4. **Specificity** It is the True negative rate. It is the proportion of true negatives that are correctly identified.

$$Specificity = \frac{Numberoftruenegatives}{Numberoftruenegatives + Numberoffalsepositives} \tag{4}$$

5. **F1 Score** One metric that is preferred over overall accuracy is the average of specificity and sensitivity, referred to as balanced accuracy. Because specificity and sensitivity are rates, it is more appropriate to compute the harmonic average. In fact, the F1-score is widely used to compute harmonic average of precision and recall.

$$F1Score = 2 * \frac{Precision - Recall}{Precision + Recall} \tag{5}$$

References

[1] Ethan Du-Crowa, Lucy Warrenb, Susan M Astleya and Johan Hullemanc,"Is there a safety-net effect with Computer-Aided Detection (CAD)?", Medical Imaging 2019.

[2] Eugene, R., Sorrell, Michael F.; Maddrey, Willis C., "Schiff's Diseases of the Liver", 10th Edition, Lippincott Williams & Wilkins by Schiff.