edX HarvardX PH125.9x
Data Science: Capstone
Classification System for
Diagnosis of Liver Disease causing by Hepatitis C Virus (HCV)

Chan Kwok Leung

12/27/2020

Contents

FOREWORD

This report is part of the fulfillment for the course HarvardX PH125.9x Data Science: Capstone.
We are going to present the development of a machine learning classification system which help diagnosis of Liver Disease from various of patient information and laboratory test result.
The soft copy of this pdf report, data source and related programming code (R and Rmd) is available in GitHub. Here come the url for reference. (GitHub https://github.com/klchanhenry/edx-HCV-diag)

OVERVIEW

Life is priceless. Medical resources is always a scarce resources in any society. This include medical professional, medical laboratories and various professional support teams. With a system providing instant diagnosis advice, this will not only help enhancing the efficiency of various resources and give an extra level of guard to prevent missing of potential patient. The cost of a missed diagnosis is very serious, it can cost hundreds folds of social resources on curing the disease due to the delayed treatment and may even cost a precious life.

GOAL OF THE PROJECT

For a classification system to identify possible patient, in this project, we are trying to achieve a system with:-
1. *high accuracy* with *very low false negative rate.* This act as a safe guard to ensure patient need medical attention will be well awared.
2. Low False positive can be accepted. The lower the false positive rate, the better efficiency on spending medical system cost which mean the less health people will be examined again and again.

To make this system easily available for any society, in additional to the classification performance, we try to complete this project without fancy computer equipment. We will finish this project with an entry level PC. This PC is having CPU with 2 cores 4 threads processor within 5 years old, 16GBytes RAM and running Windows 7 or above.

CHALLENGE OF THE PROJECT

When we talking about Data Science, Data is obviously a very critical component. In order to gather confident statistic information for good machine learning result. The number of reliable sample is very important. The bigger the sample size, the more likely we can come out with a reliable model to tackle the problem.
However, in this project, the sample size is very small. For this or similar kind of project, experiencing this obstacle can due to different possible reasons such as patient privacy, legal eligibility of patient data usage constrains, high cost of formulate data. In some case, this can be something we had to accept.

Another problem which is data imbalance. This mean the sample is having severe skew in the class distribution, it can be 1:10 or even 1:1000. This is a kind of sample distribution which is unfavorable in creating data science model. However, it is very common realistic scenario in some domains such as our project.
To simply illustrate, for example, it is common sense for a particular cancer screaming, lots of testing will be done to thousands of patient but the positive diagnosis is rare just because the fact that particular cancer is not common as flu however it is still fatal. To extract cue from the limited information and avoid distraction from the majority class outcome is the challenge.

To make this case even more challenging, our project is a multi classes classification system on the severe imbalance data set. This mean the target class is minority, and out of this rare cases, we need to handle the classification of the different type of rare cases.
For instance, our target is not only differentiate whether the patient is having cancer A or not, we need to classify the patient is having “No cancer A”, “Cancer A(type I)”, “Cancer A(type II)” or “Cancer A(type III)” under the fact that the cancer A is rare in our sample.

Finally, for this project, similar in all medical research fields, it is a highly professional industry. We, as data science teammate, usually may not have the very in depth knowledge on every particular field we are working as we may participate on many different areas at the same time. This may consume us some days just to understand the problem. However, this also bring us advantage of review the problem from a newly fresh angle without any field perception upfront.

ABOUT THE DATASET

The data source of this project store at UC Irvine Machine Learning Repository named “HCV data” and here come the url for the details of this data set (https://archive.ics.uci.edu/ml/datasets/HCV+data)

According to the data set provider, the data have 615 samples each with 14 attributes. These six hundreds samples are classified into five types, from normal blood donor to the end stage cirrhosis. The attributes can be divided to two categories, general patient information and laboratory data. There are 3 types of general patient information and 10 types of laboratory data per sample.

Classificaion Target:-
Category:
Diagnosis values are ‘0=Blood Donor’, ‘0s=suspect Blood Donor’, ‘1=Hepatitis’, ‘2=Fibrosis’, ‘3=Cirrhosis’

Attributes (General patient information):-
X (Patient ID/No.)
Age (in years)
Sex (f,m)

Attributes (Laboratory data):-
ALB (albumin)
ALP (alkaline phosphatase)
ALT (alanine amino-transferase)
AST (aspartate amino-transferase)
BIL (bilirubin)
CHE (choline esterase)
CHOL (cholesterol)
CREA (creatinine)
GGT (γ-glutamyl-transferase)
PROT (total protein)

(Note: Since the laboratory test name is not directly available from the data set, we collect the related information from the Internet with best effort accuracy. If you need further details information about the laboratory test, you can consider to communicate with the data source contact which can be obtain from the above link.)

KEY STEPS

We are going to apply a common framework for data science, **OSEMN** framework, which covers major steps on data science project lifecycle.

STEP 1. (O) Obtaining data
In this project, the data is stored in UC Irvine Machine Learning Repository and we can directly obtain data from their web site.

STEP 2. (S) Scrubbing data
In some case, like our project, there are missing values data, possible inconsistency or other errors. We need to perform cleaning such as filling missing values by some technical method or correct errors before further analysis begin. For those data can be highly likely unreliable entry, we may consider drop the row of data.

STEP 3. (E) Exploring data
Once the clean data is ready, we will start exploring data. By visualization of data, we may find the trend, pattern and possible to extract the features by statistics tools. This give us the detail insight on the set of data and more understand on the problem.

STEP 4. (M) Modeling data
Different machine learning techniques/models will give significantly varies in performance and also the computation cost (calculation time consume and equipment cost) can be a large different. In machine learning, method with higher computation cost do not imply a better performance.
To figure out which is the cost effective method to achieve good performance involve experience on data insight and technical knowledge in machine learning.

STEP 5. (N) iNterpreting data
The final step and an important one is about communicate with the project sponsor or project audiences. This can be a storytelling process via appropriate level of technical language which is this report trying to deliver.
Audiences understanding of the development progress step by step can help convincing the support on the result findings. We know getting project sponsor buy-in is usually equally importance as reaching the good performance.

ANALYSIS AND METHODS

Obtaining data

In this project, we are going to directly collect our data source from UC Irvine Machine Learning Repository and load to R for further processing.
For simplicity and tidiness, we will load all the necessary libraries upfront similar to common practice in general programming.

```
# Loading various library needed upfront
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(doParallel)) install.packages("doParallel", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rattle)) install.packages("rattle", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")

library(dplyr)
library(tidyverse)
library(caret)
library(data.table)
library(doParallel)

library(corrplot)
library(RColorBrewer)

library(rpart.plot)
library(rpart)
library(rattle)
library(randomForest)

library(xgboost)

# Load the data set to variable from the UC Irvine Machine Learning Repository
hcv <- fread("https://archive.ics.uci.edu/ml/machine-learning-databases/00571/hcvdat0.csv")

#
# In case you cannot access the url above due to any reason, you can download the hcvdat0.csv
# file from the github state in the FORWORD section and put it in your current working directory.
# you can check your working directory by getwd().
# Then you can uncomment the following line the it will load the data from the local file.
#hcv <- fread("hcvdat0.csv")
```

Scrubbing data

Let us have a quick look on how the data look like.

```r
# Show the current data structure of the data set
str(hcv)
```

```
## Classes 'data.table' and 'data.frame':   615 obs. of  14 variables:
##  $ V1      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Category: chr  "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" ...
##  $ Age     : int  32 32 32 32 32 32 32 32 32 32 ...
##  $ Sex     : chr  "m" "m" "m" "m" ...
##  $ ALB     : num  38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
##  $ ALP     : num  52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
##  $ ALT     : num  7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
##  $ AST     : num  22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
##  $ BIL     : num  7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
##  $ CHE     : num  6.93 11.17 8.84 7.33 9.15 ...
##  $ CHOL    : num  3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
##  $ CREA    : num  106 74 86 80 76 111 70 109 83 81 ...
##  $ GGT     : num  12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
##  $ PROT    : num  69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Show a few row of the data set so that we have some idea of what the data look like.
head(hcv)
```

V1	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106	12.1	69.0
2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74	15.6	76.5
3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86	33.2	79.3
4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80	33.8	75.7
5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76	29.9	68.7
6	0=Blood Donor	32	m	41.6	43.3	18.5	19.7	12.3	9.92	6.05	111	91.0	74.0

Seems we can remove the "V1" as it only present the row number or possible patient ID and also we need to make sum type change in the "Category" and "Sex" as they are currently is chr type which is not suitable for the up coming data manipulation.

Then, we need to check for missing data on all attributes.

```r
# Show the numbner of NA, Not Available, data under each attribute
sapply(hcv, function(x) sum(is.na(x)))
```

```
##       V1 Category      Age      Sex      ALB      ALP      ALT      AST 
##        0        0        0        0        1       18        1        0 
##      BIL      CHE     CHOL     CREA      GGT     PROT 
##        0        0       10        0        0        1
```

There are some missing data out of the 615 samples. All the missing are under laboratory data attributes and there is no very serious missing in any particular attribute. We will try to use statistical method to fill them up.

Then here come the tidy and clean up tasks. We are going to remove the V1 attribute, factor the Category and Sex. After that we will replace the missing value with average of respective attribute.
Note: For the completeness of data set, we keep Sex and Age attribute at this moment. However, we will consider not including them into analysis as we aim on the laboratory test derived result instead of general population statistic related to the disease.

```r
# removing the attribute V1 which only present the row number
hcv$V1 <- NULL

# change the Category and Sex attribute form char to factor
hcv <- as.data.frame(hcv) %>%
  mutate(Category = as.factor(Category),
         Sex = as.factor(Sex))

# Show the structure of the data set after modification
str(hcv)
```

```
## 'data.frame':    615 obs. of  13 variables:
##  $ Category: Factor w/ 5 levels "0=Blood Donor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Age     : int  32 32 32 32 32 32 32 32 32 32 ...
##  $ Sex     : Factor w/ 2 levels "f","m": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ALB     : num  38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
##  $ ALP     : num  52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
##  $ ALT     : num  7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
##  $ AST     : num  22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
##  $ BIL     : num  7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
##  $ CHE     : num  6.93 11.17 8.84 7.33 9.15 ...
##  $ CHOL    : num  3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
##  $ CREA    : num  106 74 86 80 76 111 70 109 83 81 ...
##  $ GGT     : num  12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
##  $ PROT    : num  69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
```

```r
# locate the missing value and replace with the mean of respective attribute
NA2mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))
hcv[] <- lapply(hcv, NA2mean)

# check again for any missing value
sapply(hcv, function(x) sum(is.na(x)))
```

```
## Category      Age      Sex      ALB      ALP      ALT      AST      BIL
##        0        0        0        0        0        0        0        0
##      CHE     CHOL     CREA      GGT     PROT
##        0        0        0        0        0
```

After the data is clean and ready, we will split the data into training set (named train_set) and validation set (named validation_set).

```r
# fix the seed so that process result related to random is repeatable
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use 'set.seed(1)'

# random create index of the data set into 7 to 3 ratio
index = createDataPartition(hcv$Category, p = .7, list = F)
```

```r
# Assign the training set and validation set in the ratio of 7 to 3
train_set = hcv[index, ]
validation_set = hcv[-index, ]
```

The **train__set** is all the data we can perform any kind of manipulation as we want using any kind of statistic and machine learning method. We can try to extract any cue from this training set to formulate different models for upcoming prediction.
The **validation__set** from now on will be set aside until the resulting phase for challenging our performance(e.g accuracy and other indicators) of models built.
Since the actual number of sample is small and it is severe imbalance. We need a reasonable amount of data in the validation__set in order to provide a good enough challenge to our model. We had decided to split the train__set:validation__set = 7:3.
We understand this will highly reduce the train set and possible have negative impact to the model we produce but without a reasonable size validation set, it is hard to conclude the performance of models. This is a constrain we have to due with.

Exploring data

Before we started, please be noted that we are intentionally NOT to evaluate the validation set of data in order to provide a more fair and trustworthy result.

Firstly, let us have a quick look on the training set including Category and Attribute:-

```r
# showing statistical summary of the training set
summary(train_set)
```

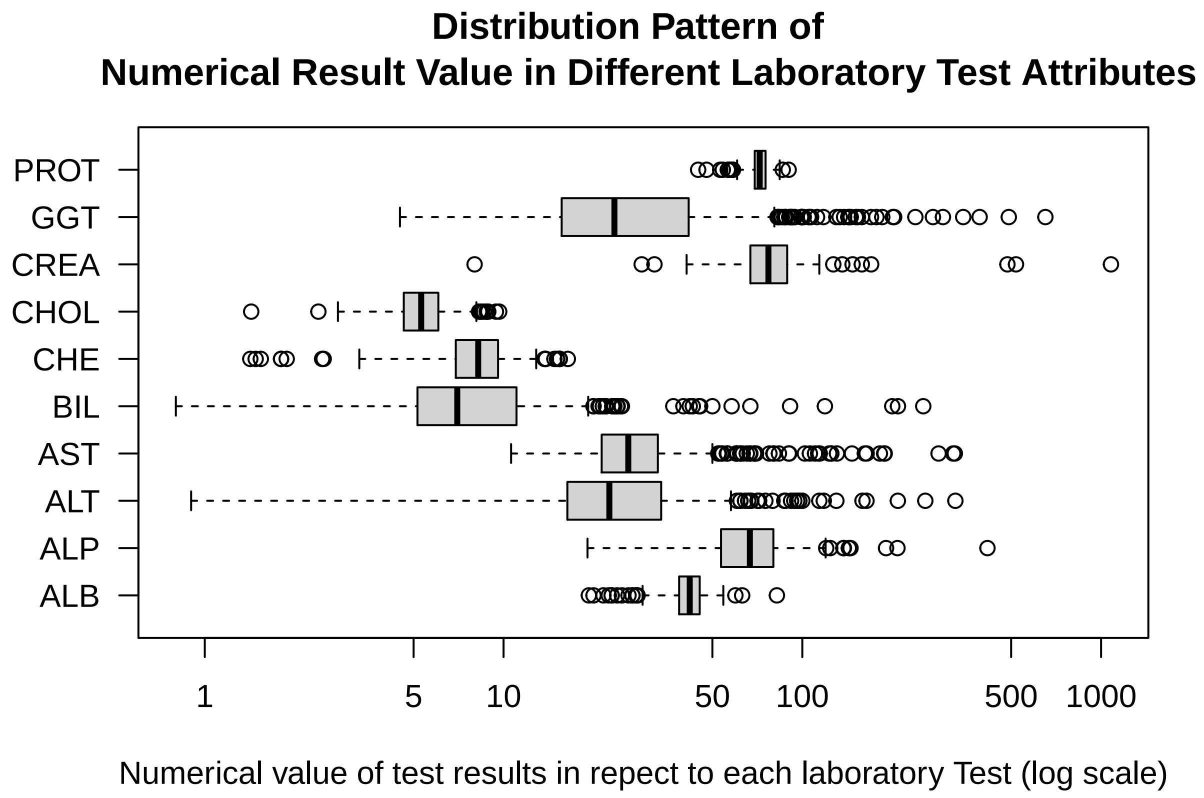
```
##                    Category        Age         Sex          ALB       
##  0=Blood Donor          :374   Min.   :19.00   f:168   Min.   :19.30  
##  0s=suspect Blood Donor:  5   1st Qu.:39.00   m:264   1st Qu.:38.70  
##  1=Hepatitis            : 17   Median :47.00           Median :42.00  
##  2=Fibrosis             : 15   Mean   :47.29           Mean   :41.65  
##  3=Cirrhosis            : 21   3rd Qu.:54.00           3rd Qu.:45.33  
##                                Max.   :77.00           Max.   :82.20  
##       ALP              ALT              AST              BIL         
##  Min.   : 19.10   Min.   :  0.90   Min.   : 10.60   Min.   :  0.800  
##  1st Qu.: 53.48   1st Qu.: 16.38   1st Qu.: 21.30   1st Qu.:  5.175  
##  Median : 66.75   Median : 22.60   Median : 26.15   Median :  7.000  
##  Mean   : 68.74   Mean   : 29.09   Mean   : 34.42   Mean   : 11.150  
##  3rd Qu.: 79.95   3rd Qu.: 33.55   3rd Qu.: 32.83   3rd Qu.: 11.025  
##  Max.   :416.60   Max.   :325.30   Max.   :324.00   Max.   :254.000  
##       CHE              CHOL            CREA              GGT        
##  Min.   : 1.420   Min.   :1.430   Min.   :   8.00   Min.   :  4.50  
##  1st Qu.: 6.920   1st Qu.:4.638   1st Qu.:  67.00   1st Qu.: 15.68  
##  Median : 8.225   Median :5.300   Median :  77.00   Median : 23.50  
##  Mean   : 8.204   Mean   :5.378   Mean   :  82.75   Mean   : 40.75  
##  3rd Qu.: 9.578   3rd Qu.:6.050   3rd Qu.:  89.00   3rd Qu.: 41.62  
##  Max.   :16.410   Max.   :9.670   Max.   :1079.10   Max.   :650.90  
##       PROT      
##  Min.   :44.80  
##  1st Qu.:69.30  
##  Median :72.00  
##  Mean   :71.97  
##  3rd Qu.:75.33  
##  Max.   :90.00
```

The data set is severe imbalance which can be confirm from the the Category. Every minority class(e.g 1=Hepatitis) is less than 6% compare to the majority class(0=Blood Donor) and the worse one(0s=suspect Blood Donor) is only 2% when compare to the majority class.

The result value seems to be likely very widely spread. We can quickly found this from the interquartile range and the minimum and maximum values. This will be illustrate in details later below.

To explore the laboratory test attributes, we are going to use boxplot.

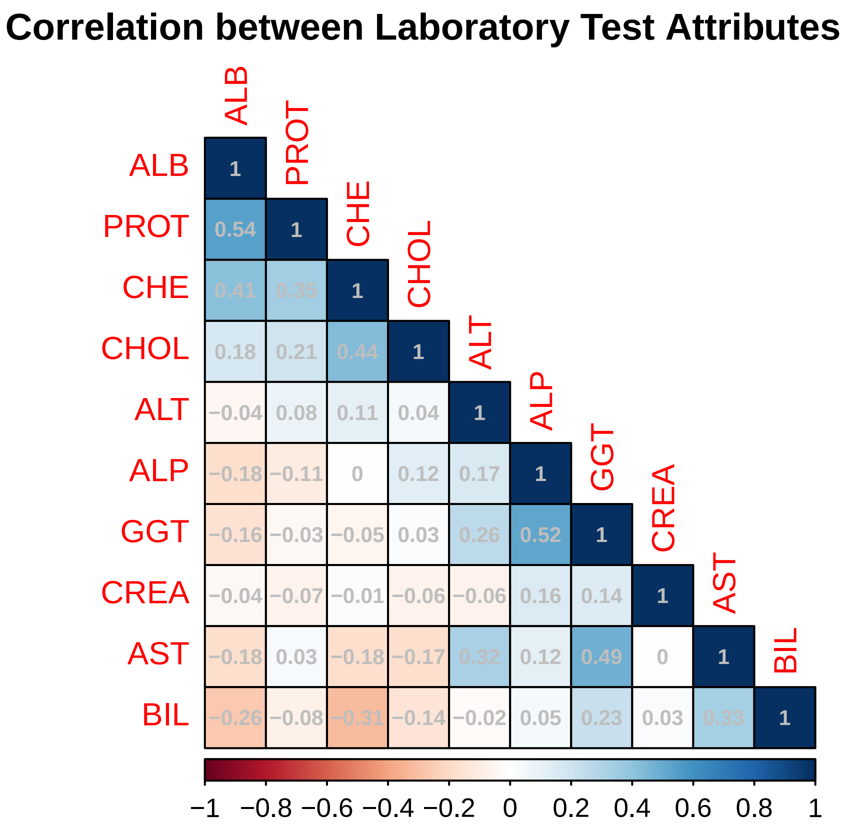
```r
# create a boxplot chart for lab test attributes
boxplot(x = as.list(train_set[,c(-1,-2,-3)]),
        horizontal=TRUE,las=1,
        log = "x",
        main="Distribution Pattern of   \n Numerical Result Value in Different Laboratory Test Attribut
        xlab= "Numerical value of test results in repect to each laboratory Test (log scale)")
```



Before start reviewing the chart, we would like to remind that the boxplot present here should be focus on the distribution of each attribute only. We should not use this chart for comparing different between tests. Also, this chart present the numerical result value only without unit of result. Once again, our focus of this chart is the distribution of numerical result value.
We can found the result of each test have obvious amount of outliers and most likely they group into one end (either max or min end). This may be a good sign telling those tests can possible give us cue on the classify different category. In contrast, hypothetically, if most the results of tests are lying near their mean, the max and min is narrow. You can imagine it will be hard to distinguish the bad from good as they all stick closely together.
Although we cannot draw any conclusion by simply viewing the boxplot, this give us insights of promising in finding some ways reaching reasonable solutions for this classification task.

Finally, let us check for any significant relationship between 10 laboratory data attributes:

```r
#calculate correlation between lab test attribute and plot chart
corr_matrix <-cor(train_set[,c(-1,-2,-3)])
corrplot(corr_matrix,type="lower", order="hclust",
         method="color",outline = T,
         addCoef.col = "grey", number.digits = 2,
         number.cex= 7/ncol(corr_matrix),
         title="Correlation between Laboratory Test Attributes",
         mar=c(0,0,2,0))
```



	ALB	PROT	CHE	CHOL	ALT	ALP	GGT	CREA	AST	BIL
ALB	1									
PROT	0.54	1								
CHE	0.41	0.35	1							
CHOL	0.18	0.21	0.44	1						
ALT	−0.04	0.08	0.11	0.04	1					
ALP	−0.18	−0.11	0	0.12	0.17	1				
GGT	−0.16	−0.03	−0.05	0.03	0.26	0.52	1			
CREA	−0.04	−0.07	−0.01	−0.06	−0.06	0.16	0.14	1		
AST	−0.18	0.03	−0.18	−0.17	0.32	0.12	0.49	0	1	
BIL	−0.26	−0.08	−0.31	−0.14	−0.02	0.05	0.23	0.03	0.33	1

Seems there is no highly correlated(correlation > 0.75) laboratory data attributes. Therefore it does not likely to have redundant features which mean each attribute is not very similar to others and may have certain degree of unique contribution to the classification process.

Modelling data

Before we start modeling, let us recap what kind of problem we are facing and what are those major constrain or difficulty we need to tackle.

We need to classify the different category of patient by the laboratory test result. This is a supervised classification problem and since the number category class is more than two, it is supervised multiclass classification problem.

The main difficulty we are facing, firstly, the size of sample is small. After splitting the whole set of data into training set and validation set, the training set data left only around 430 samples. Secondly, it is a severe imbalance data set. The ratio of number of sample in majority class versus minority class is high. This severe imbalance data set usually lead the classification machine learning algorithms in general to a biased result.

For supervised multiclass classification problem, there are lots of different machine learning method to approach the problem. As long as appropriated method had been selected, there is no right or wrong but there is always good or better way to solve the problem. We will pick some methods which are designed to be tackle classification (i.e. appropriate) and see how they are going to perform. The selection is also base on our knowledge and experiences of similar kind of problem in the past. However, if it turns out all results are below our expectation, we may need to review all over again and this is a unavoidable scenario happening sometime in machine learning projects.

We will try to following methods which are coming from different groups of machine learning algorithms.

Classification Algorithms - Decision Tree
Neural Networks - Learning Vector Quantization (LVQ)
Ensemble - Random Forest, XGBoost (Extreme Gradient Boosting)

We will provide brief descriptions about each method under modeling framework section upcoming.

Regarding to the problem of size of sample is small, theoretically, we should always come to the root of the problem and talk with the data source provider and find the way on getting a bigger data set. However, in commercial reality, it is still worth to trying but usually it is a dead end. As some domain of problems, small sample size is basically one of the characteristic of the problem. We have to accept and deal with it. However, on the data set imbalance, we still have some technical way to make our shape better. In this project, we use the function "upSample" from the "caret" library which can do samples with replacement to make class distributions equal.

Before upSample, severe imbalance:-

```r
# show the number of sample under different category in training set
table(train_set$Category)
```

```
##
##          0=Blood Donor 0s=suspect Blood Donor            1=Hepatitis
##                    374                      5                     17
##             2=Fibrosis            3=Cirrhosis
##                     15                     21
```

After upSample, artificial balanced:-

```r
# Creating artifical balanced data set by upSample
train_set <- upSample(x = train_set[, -1],y = train_set$Category, yname = "Category")

# show the number of sample under different category in training set after upSample
table(train_set$Category)
```

```
##
##          0=Blood Donor 0s=suspect Blood Donor            1=Hepatitis
##                    374                    374                    374
##             2=Fibrosis            3=Cirrhosis
##                    374                    374
```

This is an artificial way to adjust the distribution via random resampling which hopefully can help avoid those machine learning algorithms apply on them being biased.

Modelling Framework

In this section, we will illustrate the development of models via different machine learning methods. To make this section clear and more readable, we will use caret package (Classification And REgression Training). This is a set of functions which provide a uniform interface accessing different machine learning methods. So, it will be much more simple to follow by reader.

Also due to the small of sample size which mentioned several times, we will use the k-fold cross-validation to perform optimization of our different machine learning methods. Once again, the *validation set will NOT be accessed during the model development phase* which ensure the fair and trustworthiness of those models we construct.

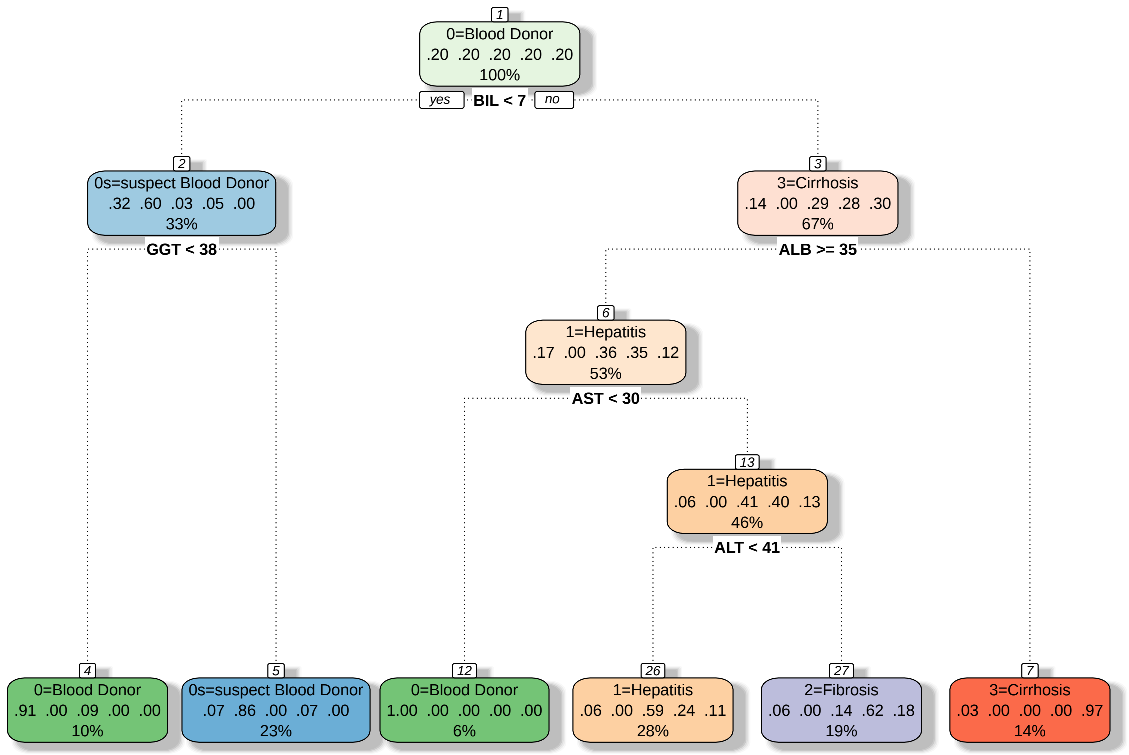
Decision Tree
A decision tree classification algorithm uses a training dataset to stratify or segment the predictor space into multiple regions. Each such region has only a subset of the training dataset. Decision tree classification models can easily handle qualitative predictors.

```r
#
# In this Modelling Framework section, there are computational intensive tasks.
# You are welcome to run the Rmd / R script with a non-critical machine since
# this process may take you around 20 minutes and using most of your machine
# CPU power during this period of time.
#

# As this machine learning method demand for more computing power
# We are creating a parallel socket cluster to utilize most of the CPU core
# One CPU core intentionally reserved in order to keep the system responsive
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

# fix the seed so that process result related to random is repeatable
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

# Cross-Validated (10 folds, repeated 3 times)
control_dt <- trainControl(method="repeatedcv", number=10, repeats=3)

# create decision tree model
model_dt <- train(Category ~ . -Age -Sex ,
                  data=train_set,
                  method="rpart",
                  trControl=control_dt,
                  tuneLength=5)

# Stop the parallel socket cluster after use
stopCluster(cl)

# Show the decision tree model built
print(model_dt)
```

```
## CART
##
## 1870 samples
##   12 predictor
##    5 classes: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1682, 1683, 1684, 1683, 1683, 1684, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02673797  0.8029252  0.75366476
##   0.09458556  0.6893230  0.61170462
##   0.11363636  0.5641138  0.45516828
##   0.16176471  0.4573940  0.32183058
##   0.25000000  0.2710192  0.09078161
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02673797.
```

```r
# plot the decision tree
fancyRpartPlot(model_dt$finalModel, sub="Decision Tree model")
```



Decision Tree model

Learning Vector Quantization (LVQ)
The model is prepared by using an input pattern to adjust the vectors most similar to it. Repeated performance of this procedure results in a distribution of vectors which provide a fair representation of the input space. The advantage of LVQ is that it is non-parametric - it does not make any assumptions about the data.

```
# As this machine learning method demand for more computing power
# We are creating a parallel socket cluster to utilize most of the CPU core
# One CPU core intentionally reserved in order to keep the system responsive
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

# fix the seed so that process result related to random is repeatable
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use ‘set.seed(1)‘

# Cross-Validated (10 folds, repeated 3 times)
control_lvq <- trainControl(method="repeatedcv", number=10, repeats=3)

# create Learning Vector Quantization (LVQ) model
model_lvq <- train(Category ~ . -Age -Sex ,
                   data=train_set,
                   method="lvq",
                   trControl=control_lvq,
                   tuneLength=5)

# Stop the parallel socket cluster after use
stopCluster(cl)

# Show the LVQ model built
print(model_lvq)
```

```
## Learning Vector Quantization
##
## 1870 samples
##   12 predictor
##    5 classes: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1682, 1683, 1684, 1683, 1683, 1684, ...
## Resampling results across tuning parameters:
##
##   size  k   Accuracy   Kappa
##   18     1  0.6710508  0.5888376
##   18     6  0.6719032  0.5898835
##   18    11  0.6613708  0.5767207
##   18    16  0.6581873  0.5727269
##   18    21  0.6597602  0.5747382
##   22     1  0.6567107  0.5708994
##   22     6  0.6681499  0.5851498
##   22    11  0.6645358  0.5806599
##   22    16  0.6582698  0.5728393
##   22    21  0.6521111  0.5651534
##   27     1  0.7008875  0.6260967
##   27     6  0.6831606  0.6039566
##   27    11  0.6866262  0.6082539
##   27    16  0.7081190  0.6351525
##   27    21  0.6798195  0.5997102
##   31     1  0.7232496  0.6540669
##   31     6  0.7100131  0.6375352
##   31    11  0.7072825  0.6340793
##   31    16  0.7166430  0.6458192
##   31    21  0.7012901  0.6266563
##   36     1  0.7258270  0.6572665
##   36     6  0.7387434  0.6734526
##   36    11  0.7355116  0.6693701
##   36    16  0.7409301  0.6761315
##   36    21  0.7192100  0.6490020
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 36 and k = 16.
```

Random Forest A random forest is comprised of a set of decision trees, each of which is trained on a random subset of the training data. These trees predictions can then be aggregated to provide a single prediction from a series of predictions.

```r
# As this machine learning method demand for more computing power
# We are creating a parallel socket cluster to utilize most of the CPU core
# One CPU core intentionally reserved in order to keep the system responsive
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

# fix the seed so that process result related to random is repeatable
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use ‘set.seed(1)‘

# Cross-Validated (10 folds, repeated 3 times)
control_rf <- trainControl(method="repeatedcv", number=10, repeats=3)

# create random forest model
model_rf <- train(Category ~ . -Age -Sex ,
                  data=train_set,
                  method="rf",
                  trControl=control_rf,
                  tuneLength=5)

# Stop the parallel socket cluster after use
stopCluster(cl)

# Show the random forest model built
print(model_rf)
```

```
## Random Forest
##
## 1870 samples
##   12 predictor
##    5 classes: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1682, 1683, 1684, 1683, 1683, 1684, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    1.0000000  1.0000000
##    4    0.9996435  0.9995543
##    6    0.9989286  0.9986606
##    8    0.9966103  0.9957627
##   10    0.9930432  0.9913035
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

XGBoost (Extreme Gradient Boosting) Gradient Boosted Decision Trees (GBDT) is a machine learning algorithm that iterative constructs an ensemble of weak decision tree learners through boosting. XGBoost is an implementation of GBDT designed for speed and performance. For further idea about XGBoost, you can visit this link (https://xgboost.readthedocs.io/en/latest/tutorials/model.html) for details.

```
# As this machine learning method demand for more computing power
# We are creating a parallel socket cluster to utilize most of the CPU core
# One CPU core intentionally reserved in order to keep the system responsive
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

# fix the seed so that process result related to random is repeatable
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use ‘set.seed(1)‘

# Cross-Validated (10 folds, repeated 3 times)
control_xgb <- trainControl(method="repeatedcv", number=10, repeats=3)

# create xgBoost model
model_xgb <- train(Category ~ . -Age -Sex ,
               data=train_set,
               method="xgbTree",
               tuneLength=5,
               trControl=control_xgb)

# Stop the parallel socket cluster after use
stopCluster(cl)

# Show the xgBoost model built
options(max.print = 100)
print(model_xgb)
```

```
## eXtreme Gradient Boosting
##
## 1870 samples
##   12 predictor
##    5 classes: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1682, 1683, 1684, 1683, 1683, 1684, ...
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa
##   0.3  1          0.6               0.500       50      0.9779168  0.9723956
##   0.3  1          0.6               0.500      100      0.9948362  0.9935451
##   0.3  1          0.6               0.500      150      0.9960831  0.9951037
##   0.3  1          0.6               0.500      200      0.9967923  0.9959902
##   0.3  1          0.6               0.500      250      0.9969725  0.9962155
##   0.3  1          0.6               0.625       50      0.9782534  0.9728160
##   0.3  1          0.6               0.625      100      0.9937638  0.9922045
##   0.3  1          0.6               0.625      150      0.9957227  0.9946533
##   0.3  1          0.6               0.625      200      0.9966169  0.9957710
##   0.3  1          0.6               0.625      250      0.9975063  0.9968828
##   0.3  1          0.6               0.750       50      0.9770132  0.9712665
##   0.3  1          0.6               0.750      100      0.9942986  0.9928730
##   0.3  1          0.6               0.750      150      0.9951880  0.9939848
##   0.3  1          0.6               0.750      200      0.9964367  0.9955458
##  [ reached getOption("max.print") -- omitted 486 rows ]
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 250, max_depth = 3, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 0.5.
```

```r
options(max.print = 1000)
```

RESULTS

We got 4 models built and it is time to examine the result performance by challenge the models with validation data set.

Before we move on, we would like to remind that since the original data set is small and this lead to the validation set is small. Also the data set is severe imbalance, we would like to say the performance measure is by the best effort base on what we have available.

To recap, we would like to the model with high accuracy and very low false negative rate. Low false positive can be accepted. In other words, we would like to pick nearly all the real patient from the pool and we do not mind to pick a little bit more healthy people. However, the less excessive we pick, the more efficient the system.

Note: if the system can successfully picking the patient but putting not exactly to the right disease type, we will consider it is a good try as this already serve the main purpose of the system. The only main concern is we missed a unhealthy patient without perform further medical check and follow up.

```r
# using the decision tree model we built previously and challenge with the validation set
predict_dt <- predict(model_dt, validation_set, type = "raw")
# formulate confusion matrix for performance review
cm_dt <- confusionMatrix(predict_dt, validation_set$Category)

# using the LVQ model we built previously and challenge with the validation set
predict_lvq <- predict(model_lvq, validation_set, type = "raw")
# formulate confusion matrix for performance review
cm_lvq <- confusionMatrix(predict_lvq, validation_set$Category)

# using the random forest model we built previously and challenge with the validation set
predict_rf <- predict(model_rf, validation_set, type = "raw")
# formulate confusion matrix for performance review
cm_rf <- confusionMatrix(predict_rf, validation_set$Category)

# using the xgboost model we built previously and challenge with the validation set
predict_xgb <- predict(model_xgb, validation_set, type = "raw")
# formulate confusion matrix for performance review
cm_xgb <- confusionMatrix(predict_xgb, validation_set$Category)
```

Confusion Matrix table for decision tree model

```r
#confusion matrix table for decision tree model
print(cm_dt$table)
```

```
##                          Reference
## Prediction                0=Blood Donor 0s=suspect Blood Donor 1=Hepatitis
##   0=Blood Donor                     122                      0           1
##   0s=suspect Blood Donor             11                      1           1
##   1=Hepatitis                        12                      0           2
##   2=Fibrosis                         12                      0           3
##   3=Cirrhosis                         2                      1           0
##                          Reference
## Prediction                2=Fibrosis 3=Cirrhosis
##   0=Blood Donor                   0           0
##   0s=suspect Blood Donor          1           0
##   1=Hepatitis                     4           4
##   2=Fibrosis                      1           0
##   3=Cirrhosis                     0           5
```

```r
print(cm_dt$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.7158470      0.2993668      0.6446214      0.7799331      0.8688525
## AccuracyPValue  McnemarPValue
##      1.0000000            NaN
```

Confusion Matrix table for Learning Vector Quantization (LVQ) model

```r
#confusion matrix table for LVQ model
print(cm_lvq$table)
```

```
##                          Reference
## Prediction                0=Blood Donor 0s=suspect Blood Donor 1=Hepatitis
##   0=Blood Donor                     149                      0           2
##   0s=suspect Blood Donor              1                      0           0
##   1=Hepatitis                         5                      1           2
##   2=Fibrosis                          4                      1           3
##   3=Cirrhosis                         0                      0           0
##                          Reference
## Prediction                2=Fibrosis 3=Cirrhosis
##   0=Blood Donor                   0           0
##   0s=suspect Blood Donor          0           0
##   1=Hepatitis                     4           2
##   2=Fibrosis                      2           3
##   3=Cirrhosis                     0           4
```

```r
print(cm_lvq$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8579235      0.4865098      0.7987943      0.9050449      0.8688525
## AccuracyPValue  McnemarPValue
##      0.7143667            NaN
```

Confusion Matrix table for random forest model

```
#confusion matrix table for random forest model
print(cm_rf$table)
```

```
##                          Reference
## Prediction               0=Blood Donor 0s=suspect Blood Donor 1=Hepatitis
##   0=Blood Donor                    159                      1           5
##   0s=suspect Blood Donor             0                      1           0
##   1=Hepatitis                        0                      0           1
##   2=Fibrosis                         0                      0           1
##   3=Cirrhosis                        0                      0           0
##                          Reference
## Prediction               2=Fibrosis 3=Cirrhosis
##   0=Blood Donor                   3           2
##   0s=suspect Blood Donor          0           0
##   1=Hepatitis                     1           0
##   2=Fibrosis                      2           0
##   3=Cirrhosis                     0           7
```

```
print(cm_rf$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.928961749    0.626060987    0.881585865    0.961635169    0.868852459
## AccuracyPValue  McnemarPValue
##    0.007181343            NaN
```

Confusion Matrix table for xgboost model

```
#confusion matrix table for xgboost model
print(cm_xgb$table)
```

```
##                          Reference
## Prediction               0=Blood Donor 0s=suspect Blood Donor 1=Hepatitis
##   0=Blood Donor                    159                      1           2
##   0s=suspect Blood Donor             0                      1           0
##   1=Hepatitis                        0                      0           2
##   2=Fibrosis                         0                      0           3
##   3=Cirrhosis                        0                      0           0
##                          Reference
## Prediction               2=Fibrosis 3=Cirrhosis
##   0=Blood Donor                   1           0
##   0s=suspect Blood Donor          0           0
##   1=Hepatitis                     3           0
##   2=Fibrosis                      2           2
##   3=Cirrhosis                     0           7
```

```
print(cm_xgb$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.934426230    0.704441454    0.888257878    0.965660863    0.868852459
## AccuracyPValue  McnemarPValue
##    0.003381041            NaN
```

Base on what we aimed, we suggest to pick the Learning Vector Quantization (LVQ) model as the first choice as this only give 2 *"real"* false negative(i.e. putting the 2 Hepatitis patient into normal blood donor class) and the rest of the performance is reasonable (i.e. false positive is not high).
The second choice can possible consider is the xgBoost model, this model have very low false positive rate (i.e. it do not put any healthy people into the disease pool which can avoid wasting medical resources on rechecking). However, the *"real"* false negative is higher (1+2+1+0=4, instead of 2 from LVQ) than LVQ model. However, we hope with more samples in future or other facility can possible compensate this error and improve the system.

CONCLUSION

We think this system can give an extra level of supporting while performing the diagnosis. In the future, with more data sample and development, we hope this system can be help medical professional hand off from the routine diagnosis tasks and possible to put more focus on those advance medical development.

To enhance the system performance, beside hoping for a larger sample size, we can try feature engineering for better false negative and also using SMOTE(Synthetic Minority Oversampling Technique) for upsampling to gain a better imbalanced dataset enhancement.

MAJOR REFERENCE MATERIAL SOURCE

1. Introduction to Data Science by Rafael A. Irizarry
2. Hepatitis C (https://www.who.int/news-room/fact-sheets/detail/hepatitis-c)
3. 5 Steps to a Data Science Project Lifecycle(https://thelead.io/data-science/5-steps-to-a-data-science-project-lifecycle)
4. Understanding Lab Tests, Hepatitis C for Patients(https://www.hepatitis.va.gov/hcv/patient/diagnosis/labtests-index.asp)
5. Understanding Medical Words: A Tutorial, Appendix B: Some Common Abbreviations(https://medlineplus.gov/appendixb.html)
6. Various technical discussion forums, stackoverflow.com as an example.