Hepatitis C virus - Blood based Detection Project

Capstone project 2

HarvardX Data Science Professional Certificate: PH125.9x

jamal jouda

11/26/2021

Contents

1. Introduction

Hepatitis C virus (HCV) causes both acute and chronic infection. Acute HCV infections are usually asymptomatic and most do not lead to a life-threatening disease. Around 30% (15–45%) of infected persons spontaneously clear the virus within 6 months of infection without any treatment. The remaining 70% (55–85%) of persons will develop chronic HCV infection. Of those with chronic HCV infection, the risk of cirrhosis ranges from 15% to 30% within 20 years.

1.1 Objective

The main aim of this project is to analysis the Hepatitis C virus (HCV) dataset features and make a prediction for infected persons using multiple machine learning methods.

1.2 Dataset Overview

I use the dataset from Kaggel, https://www.kaggle.com/amritpal333/hepatitis-c-virus-blood-biomarkers. Dataset include data from 615 cases of laboratory values of blood donors and Hepatitis C patients and demographic values like age and sex.

To use dataset I download the dataset from Kaggel, extract it from archive, and save it to working directory then read the dataset.

```
# install required packages and libraries
if(!require(tidyverse)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(cowplot)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("readr", repos = "http://cran.us.r-project.org")
```

```
# load required packages and libraries
library(tidyverse)
library(readr)
library(corrplot)
library(scales)
library(ggplot2)
library(ggthemes)
library(cowplot)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
# read the dataset file and save it in hepa (hepatitis)
hepa <- read_csv("~/hepa.csv")
```

```
# show the first 6 rows in the dataset
head(hepa)
```

```
## # A tibble: 6 x 14
##    ...1 Category       Age Sex     ALB   ALP   ALT   AST   BIL   CHE  CHOL  CREA
##   <dbl> <chr>        <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1 0=Blood Don~    32 m      38.5  52.5   7.7  22.1   7.5  6.93  3.23   106
## 2     2 0=Blood Don~    32 m      38.5  70.3  18    24.7   3.9 11.2   4.8     74
## 3     3 0=Blood Don~    32 m      46.9  74.7  36.2  52.6   6.1  8.84  5.2     86
## 4     4 0=Blood Don~    32 m      43.2  52    30.6  22.6  18.9  7.33  4.74    80
## 5     5 0=Blood Don~    32 m      39.2  74.1  32.6  24.8   9.6  9.15  4.32    76
## 6     6 0=Blood Don~    32 m      41.6  43.3  18.5  19.7  12.3  9.92  6.05   111
## # ... with 2 more variables: GGT <dbl>, PROT <dbl>
```

The raw dataset contains 615 rows (observation) and 14 columns (features). The column "Category" we want to predict.

```
# show the structures of dataset
str(hepa)
```

```
## spec_tbl_df [615 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1    : num [1:615] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Category: chr [1:615] "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" ...
##  $ Age     : num [1:615] 32 32 32 32 32 32 32 32 32 32 ...
##  $ Sex     : chr [1:615] "m" "m" "m" "m" ...
##  $ ALB     : num [1:615] 38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
##  $ ALP     : num [1:615] 52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
##  $ ALT     : num [1:615] 7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
##  $ AST     : num [1:615] 22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
##  $ BIL     : num [1:615] 7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
##  $ CHE     : num [1:615] 6.93 11.17 8.84 7.33 9.15 ...
##  $ CHOL    : num [1:615] 3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
##  $ CREA    : num [1:615] 106 74 86 80 76 111 70 109 83 81 ...
##  $ GGT     : num [1:615] 12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
##  $ PROT    : num [1:615] 69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   Category = col_character(),
##   ..   Age = col_double(),
##   ..   Sex = col_character(),
##   ..   ALB = col_double(),
##   ..   ALP = col_double(),
##   ..   ALT = col_double(),
##   ..   AST = col_double(),
##   ..   BIL = col_double(),
##   ..   CHE = col_double(),
##   ..   CHOL = col_double(),
##   ..   CREA = col_double(),
##   ..   GGT = col_double(),
##   ..   PROT = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

All columns in dataset (except Category and Sex) are numerical. The laboratory data are the columns from 5-14. the columns of the dataset are as follows:

1. ...1: serial number.
2. Category: diagnosis (values: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis')
3. Age: (in years)
4. Sex: (vales: f for females, m for males)
5. ALB: albumin level in the blood (g/L)
6. ALP: alkaline phosphatase level in the blood (IU/L)
7. ALT: alanine aminotransferase level in the blood (IU/L)
8. AST: aspartate aminotransferase level in the blood (IU/L)
9. BIL: bilirubin level in the blood (mg/dL)
10. CHE: choline esterase level in the blood (IU/L)
11. CHOL: cholesterol level in the blood (mg/dL)
12. CREA: serum creatinine level in the blood (mg/dL)
13. GGT: gamma-glutamyl-transferase level in the blood (IU/L)
14. PROT: total protein level in the blood (g/L)

2. Data Preparing And Analysis

In this section I will prepare dataset to be ready to use in modeling, and make analysis for dataset features.

2.1 Dataset Preparing

The steps to preparing data for modeling are:

1. The first column in the dataset (called ..1) is serial number and we don't need it so I drop the column from the dataset.

```
# drop serial number column (..1)
hepa = select(hepa, -1)
# number of rows and columns after deleting serial no column
dim(hepa)
```

```
## [1] 615  13
```

2. There are 31 missing values in the dataset, one value in ALB column, 18 in ALP, one in ALT, 10 in CHOL and one in PROT. They are little according to dataset so I drop them from the dataset.

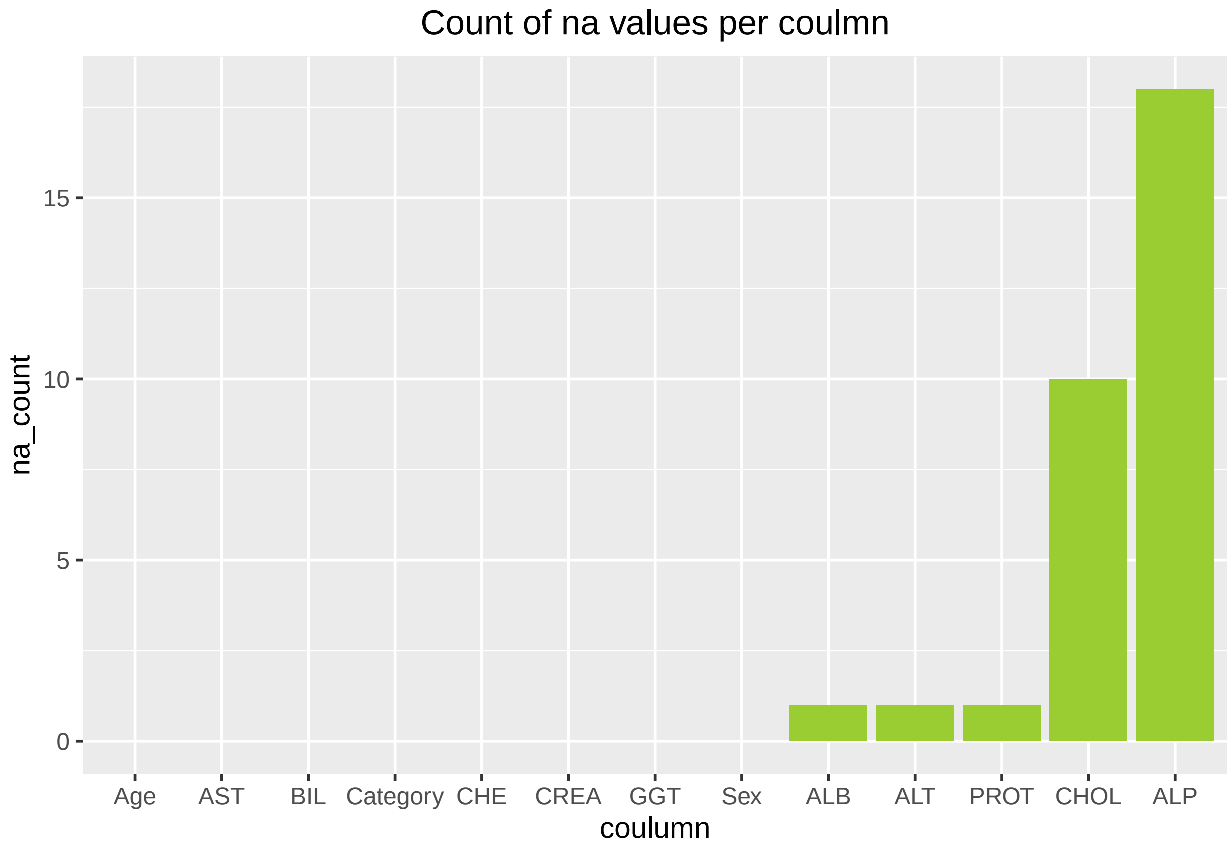
```
# find the sum of na values in the dataset
sum(is.na(hepa))
```

```
## [1] 31
```

```
# find the count of na values for each column
na_values <- hepa %>% summarise_all(funs(sum(is.na(.))))
na_values <- gather(na_values, key = "variables", value = "na_count")
na_values
```

```
## # A tibble: 13 x 2
##    variables na_count
##    <chr>        <int>
##  1 Category         0
##  2 Age              0
##  3 Sex              0
##  4 ALB              1
##  5 ALP             18
##  6 ALT              1
##  7 AST              0
##  8 BIL              0
##  9 CHE              0
## 10 CHOL            10
## 11 CREA             0
## 12 GGT              0
## 13 PROT             1
```

```
# plot the count of na values for each column
ggplot(na_values, aes(x = reorder(variables,na_count), y = na_count)) +
    geom_bar(stat = "identity", fill = "yellowgreen") +
    ggtitle("Count of na values per coulmn") +
    labs(x="coulumn", y="na_count") +
    theme(plot.title = element_text(hjust = 0.5))
```



```
# delete all na rows in the dataset
hepa <- drop_na(hepa)
# ensure that dataset has no na values
sum(is.na(hepa))
```

```
## [1] 0
```

```
# number of rows and columns after deleting na values
dim(hepa)
```

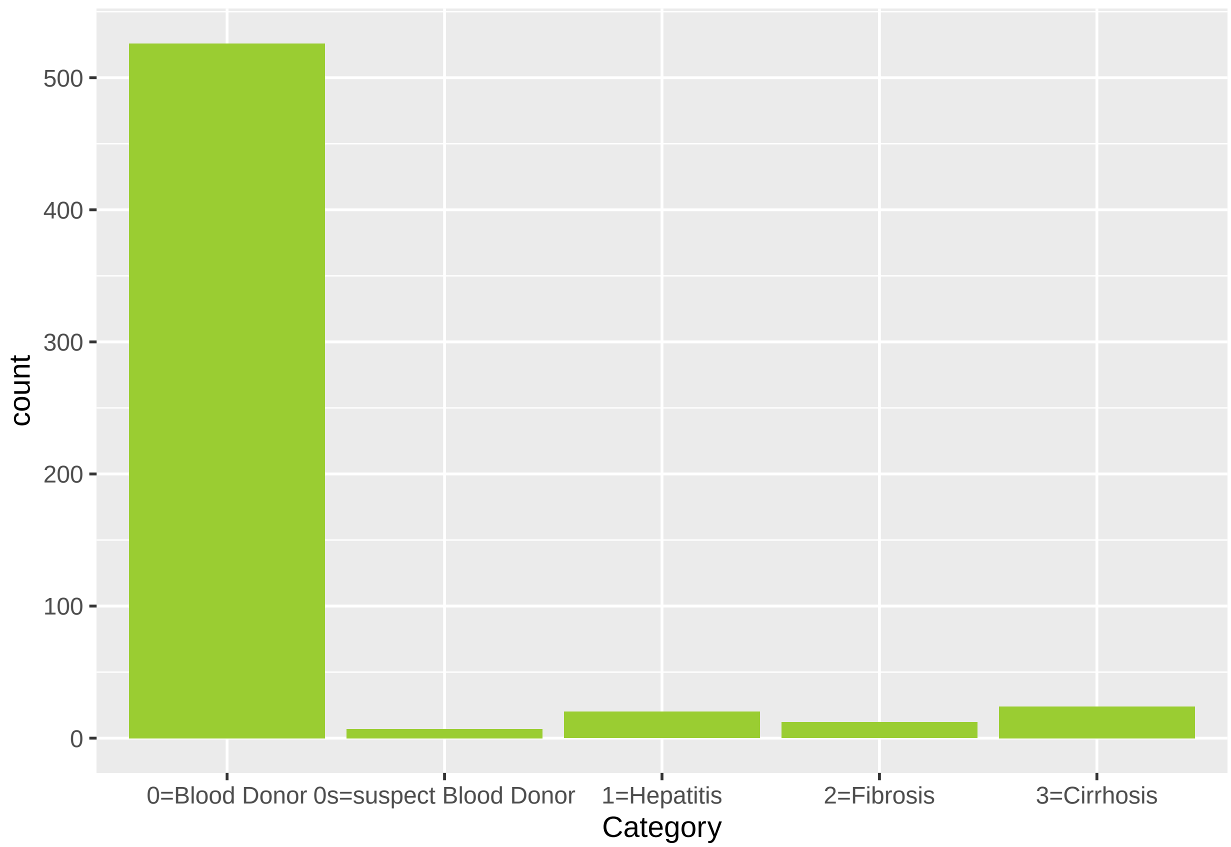
```
## [1] 589  13
```

3. The "Category" column which we want to predict have five values they are: 0=Blood Donor, 0s=suspect Blood Donor, 1=Hepatitis, 2=Fibrosis and 3=Cirrhosis.
   The first value (0=Blood Donor) belong to persons where can donate blood, and the remaining values (0s=suspect Blood Donor, 1=Hepatitis, 2=Fibrosis and 3=Cirrhosis) belong to persons where cant donate blood because Hepatitis or problems in liver, so I map the (Category) columns to two integer values (0 and 1). Then I factorized the values to simplify prediction.

```
# count the numbers of rows in each Category
hepa %>% group_by(Category) %>% summarise(count = n())
```

```
## # A tibble: 5 x 2
##   Category               count
##   <chr>                  <int>
## 1 0=Blood Donor            526
## 2 0s=suspect Blood Donor     7
## 3 1=Hepatitis               20
## 4 2=Fibrosis                12
## 5 3=Cirrhosis               24
```

```
# plot rows count in each Category
ggplot(hepa, aes(Category)) + geom_bar(fill = "yellowgreen")
```



```
# replace "0=Blood Donor" Category by 0
hepa$Category <- str_replace_all(hepa$Category, setNames(c("0"), c("0=Blood Donor")))
# replace other Categories by 1
hepa$Category <- str_replace_all(hepa$Category, setNames(c("1","1","1","1"),
  c("0s=suspect Blood Donor","1=Hepatitis","2=Fibrosis","3=Cirrhosis")))
# factorized Category column
hepa$Category <- as.factor(hepa$Category)
# count the numbers of rows in each Category
hepa %>% group_by(Category) %>% summarise(count = n())
```

```
## # A tibble: 2 x 2
##   Category count
##   <fct>    <int>
## 1 0          526
## 2 1           63
```

4. The “Sex” column has two values (m for males and f for females) I convert them to (0 for males and 1 for females) and factorized them to simplify prediction.

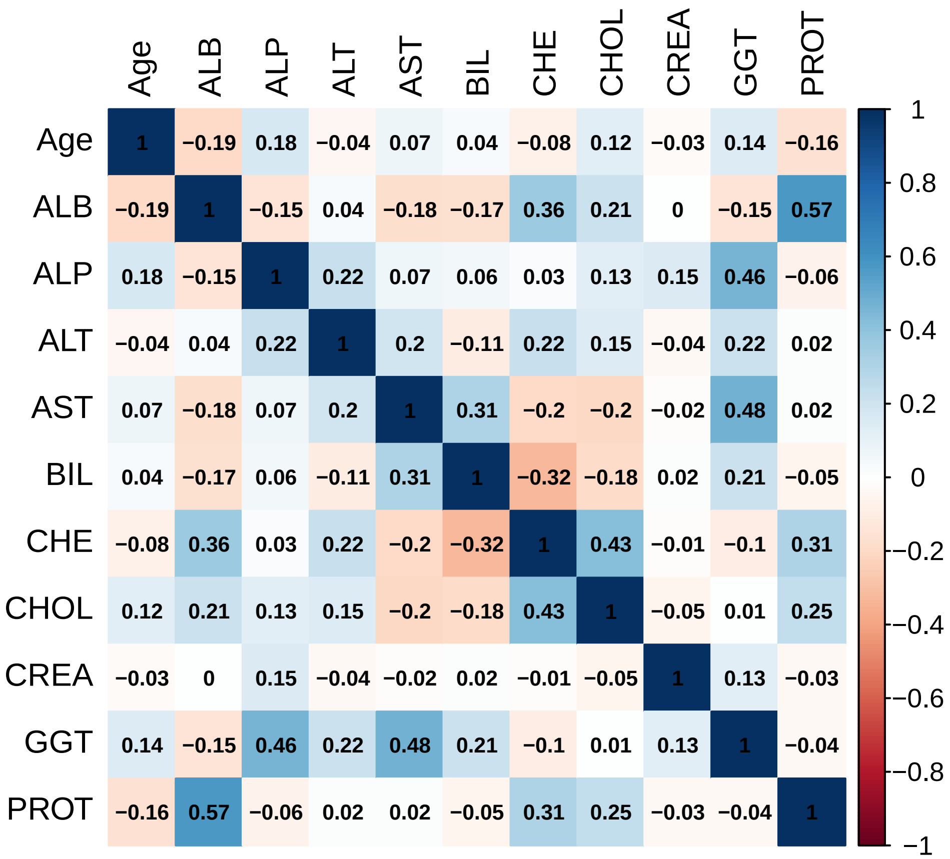
```
# replace m (male) with 0 and f (female) with 1 in sex column and factorized them
hepa$Sex <- str_replace_all(hepa$Sex, setNames(c("0"), c("m")))
hepa$Sex <- str_replace_all(hepa$Sex, setNames(c("1"), c("f")))
# factorized Sex column
hepa$Sex <- as.factor(hepa$Sex)
```

2.2 Data Analysis

2.2.1 Features Correlation

Most features in the dataset are numeric values (except Category and Sex) which represent laboratory data values so I plot the correlation matrix.

```r
# plot correlation matrix for dataset features using corrplot package
hepa_temp = select(hepa, -1, -3)
hepa_cor <- cor(hepa_temp)
corrplot(hepa_cor, method="color",addCoef.col = "black", tl.col="black", number.cex = 0.7)
```

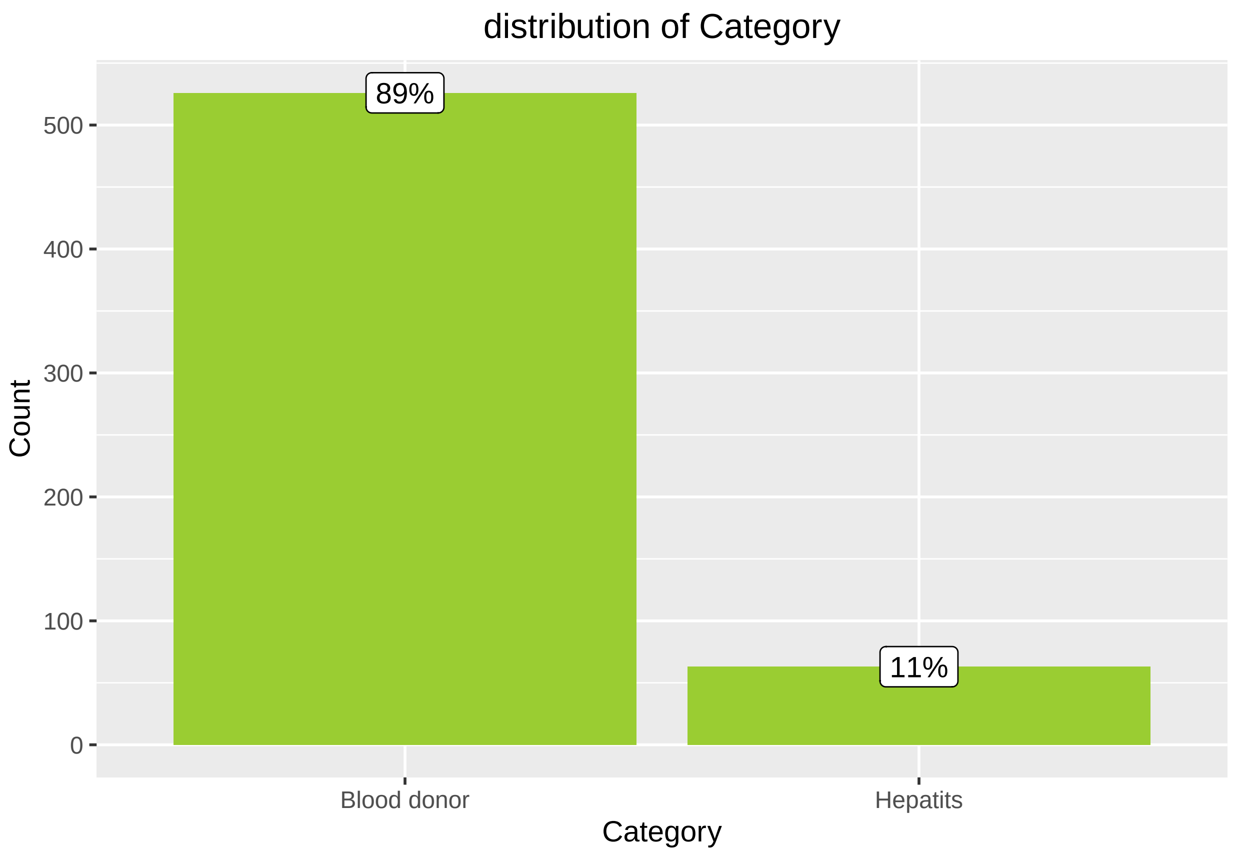


	Age	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
Age	1	−0.19	0.18	−0.04	0.07	0.04	−0.08	0.12	−0.03	0.14	−0.16
ALB	−0.19	1	−0.15	0.04	−0.18	−0.17	0.36	0.21	0	−0.15	0.57
ALP	0.18	−0.15	1	0.22	0.07	0.06	0.03	0.13	0.15	0.46	−0.06
ALT	−0.04	0.04	0.22	1	0.2	−0.11	0.22	0.15	−0.04	0.22	0.02
AST	0.07	−0.18	0.07	0.2	1	0.31	−0.2	−0.2	−0.02	0.48	0.02
BIL	0.04	−0.17	0.06	−0.11	0.31	1	−0.32	−0.18	0.02	0.21	−0.05
CHE	−0.08	0.36	0.03	0.22	−0.2	−0.32	1	0.43	−0.01	−0.1	0.31
CHOL	0.12	0.21	0.13	0.15	−0.2	−0.18	0.43	1	−0.05	0.01	0.25
CREA	−0.03	0	0.15	−0.04	−0.02	0.02	−0.01	−0.05	1	0.13	−0.03
GGT	0.14	−0.15	0.46	0.22	0.48	0.21	−0.1	0.01	0.13	1	−0.04
PROT	−0.16	0.57	−0.06	0.02	0.02	−0.05	0.31	0.25	−0.03	−0.04	1

2.2.2 Category Feature Distribution

From the distribution of category feature we found that 89% of the persons is blood donor (Category = 0), and 11% have hepatitis or problems in liver (Category = 1).

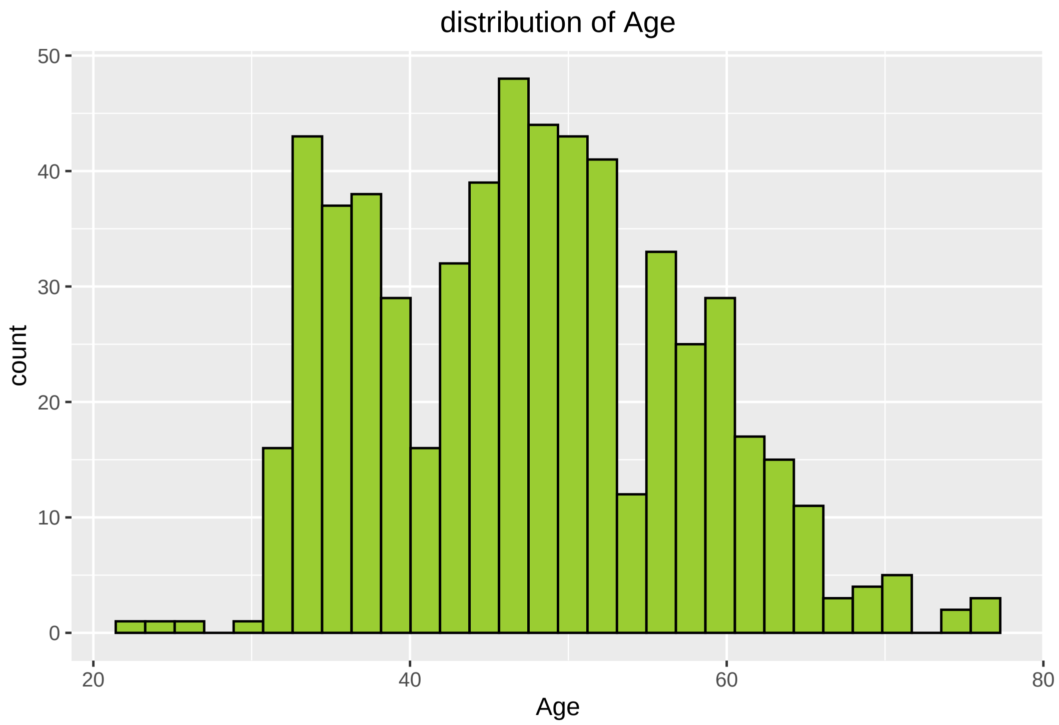
```r
# plot the distribution of Category (count and percentage)
hepa %>% group_by(Category) %>% summarise(count = n()) %>% mutate(prop =
round(count/sum(count),2)) %>% ungroup() %>% ggplot(aes(x = Category, y =
count)) + geom_bar(stat = "identity", fill = "yellowgreen") +
    labs(x = "Category",y = "Count", title = "distribution of Category") +
    geom_label(aes(label=percent(prop))) +
    scale_x_discrete(labels = c("Blood donor","Hepatits")) +
    theme(plot.title = element_text(hjust = 0.5))
```



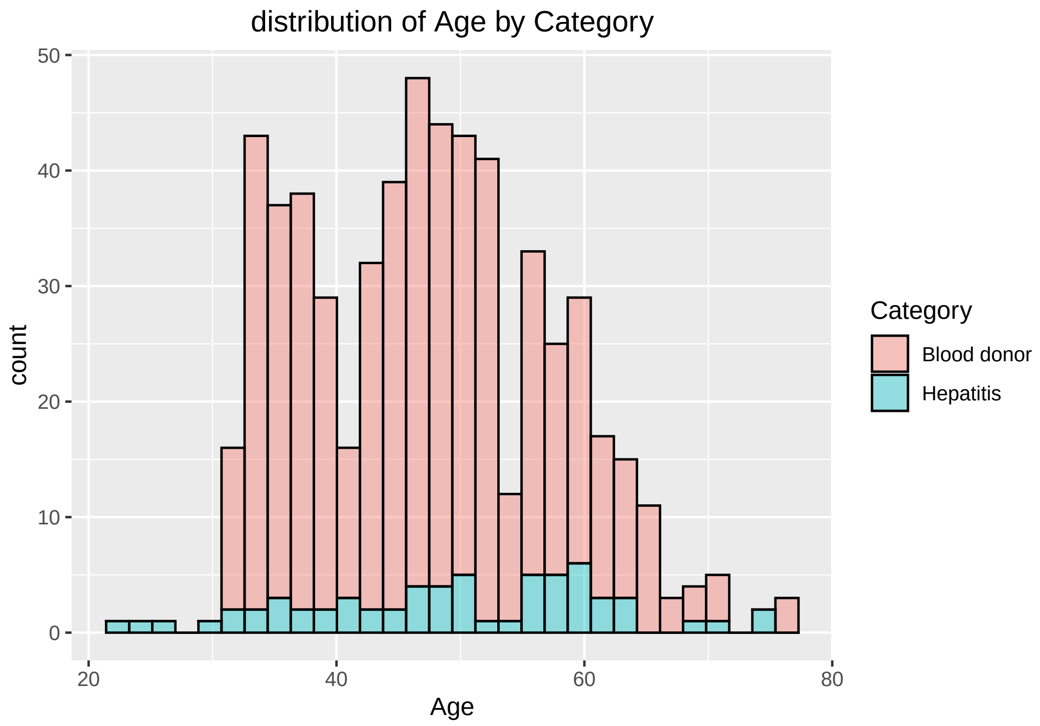
2.2.3 Age Distribution And Age by Category Distribution

The two histograms below show the distribution of Age values in the dataset, and the distribution of Age by Category.

```
# plot the distribution of Age
ggplot(hepa, aes(x = Age)) + geom_histogram(bins = 30, color = "black", fill = "yellowgreen") +
    ggtitle("distribution of Age") +
    theme(plot.title = element_text(hjust = 0.5))
```



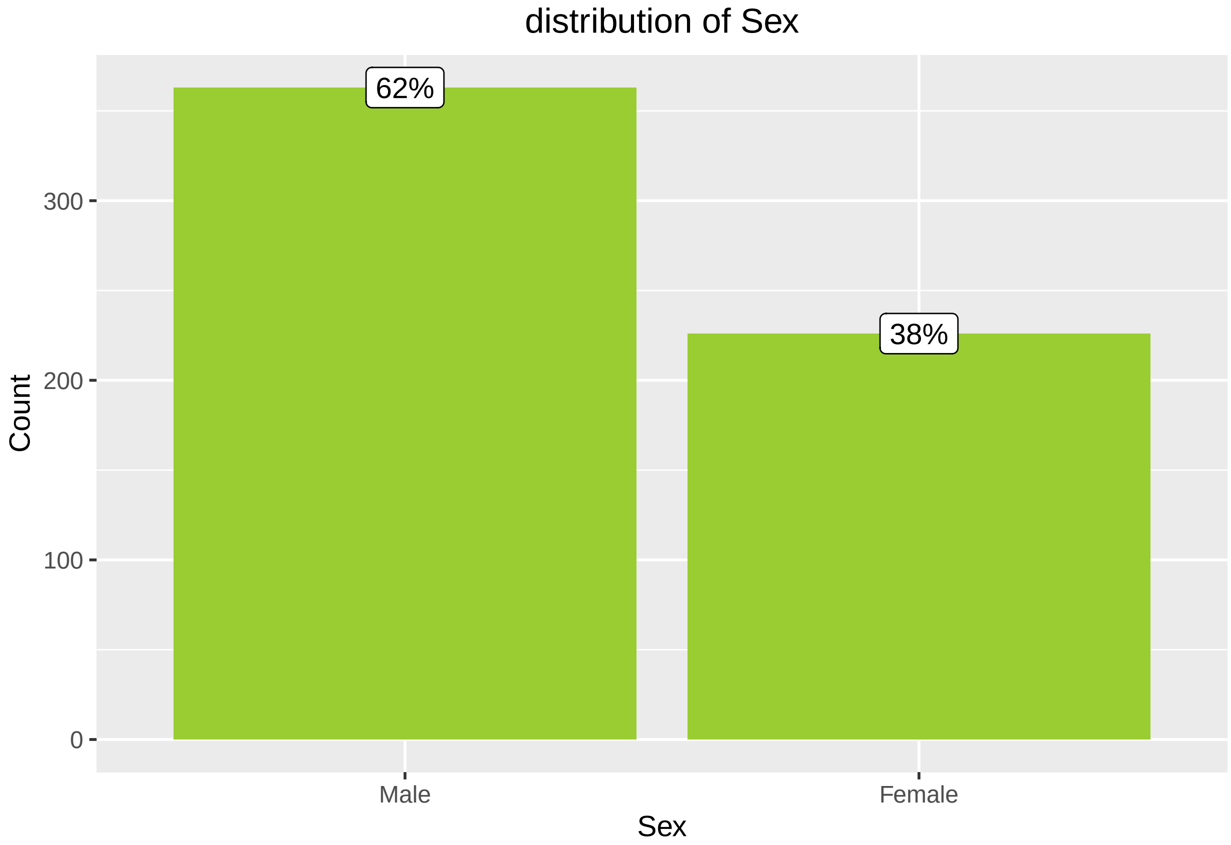
```
# plot the distribution of age by Category
ggplot(hepa, aes(x = Age, fill = Category)) + geom_histogram(color= "black", alpha = 0.4) +
    scale_fill_discrete(labels = c("Blood donor", "Hepatitis")) +
    ggtitle("distribution of Age by Category") +
    theme(plot.title = element_text(hjust = 0.5))
```



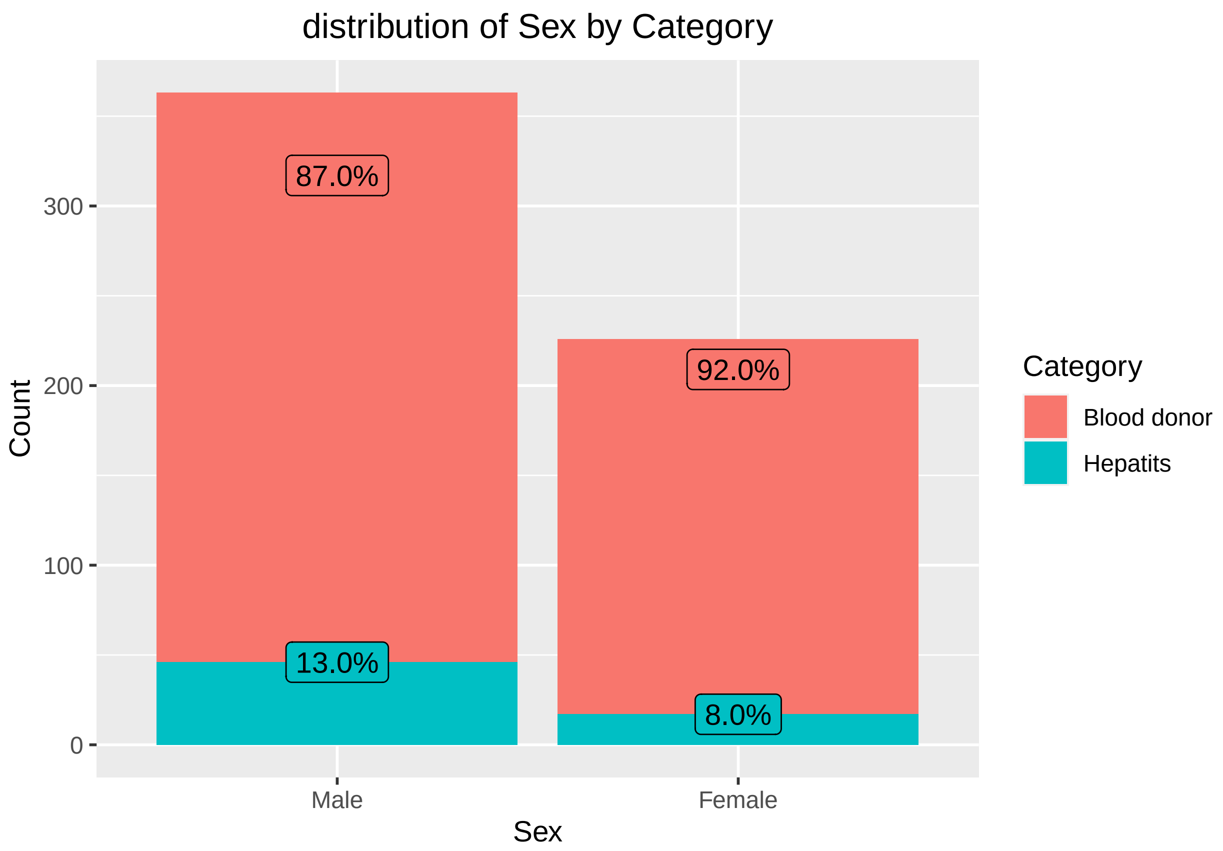
2.2.4 Sex distribution and Sex by Category distribution

From the distribution of sex feature we found that 62% of the persons are males, 13% of them have hepatitis or problems in liver, and 38% are females, 8% of them have hepatitis or problems in liver.

```
# plot the distribution of Sex
hepa %>% group_by(Sex) %>% summarise(count = n()) %>%
    mutate(prop = round(count/sum(count),2)) %>%
    ungroup() %>% ggplot(aes(x = Sex, y = count)) +
    geom_bar(stat = "identity", fill = "yellowgreen") +
    labs(x = "Sex",y = "Count", title = "distribution of Sex") +
    geom_label(aes(label=percent(prop))) + scale_x_discrete(labels = c("Male","Female")) +
    theme(plot.title = element_text(hjust = 0.5))
```



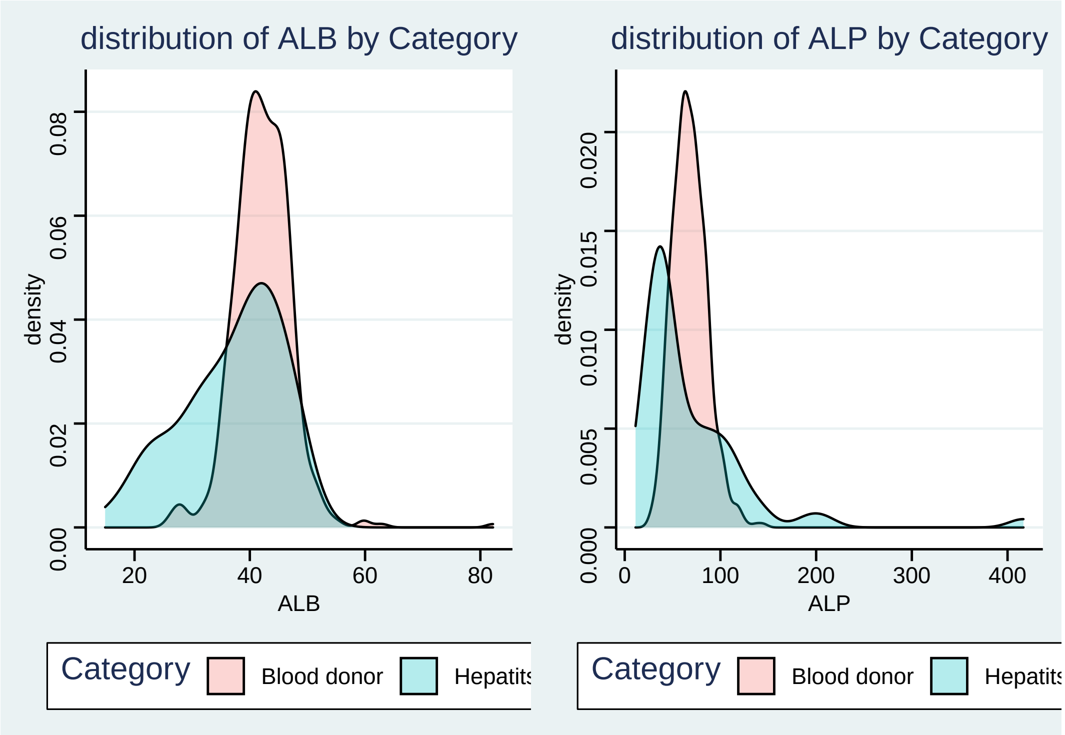
```
# plot the distribution of Sex by Category
hepa %>% group_by(Sex, Category) %>% summarise(count = n()) %>%
    mutate(prop = round(count/sum(count),2)) %>% ungroup() %>%
    ggplot(aes(x = Sex, y = count, fill = Category)) +
    geom_bar(stat = "identity") +
    labs(x = "Sex",y = "Count", title = "distribution of Sex by Category") +
geom_label(aes(label=percent(prop)), show.legend = FALSE) +
scale_x_discrete(labels = c("Male","Female")) +
scale_fill_discrete(labels = c("Blood donor", "Hepatits")) +
theme(plot.title = element_text(hjust = 0.5))
```



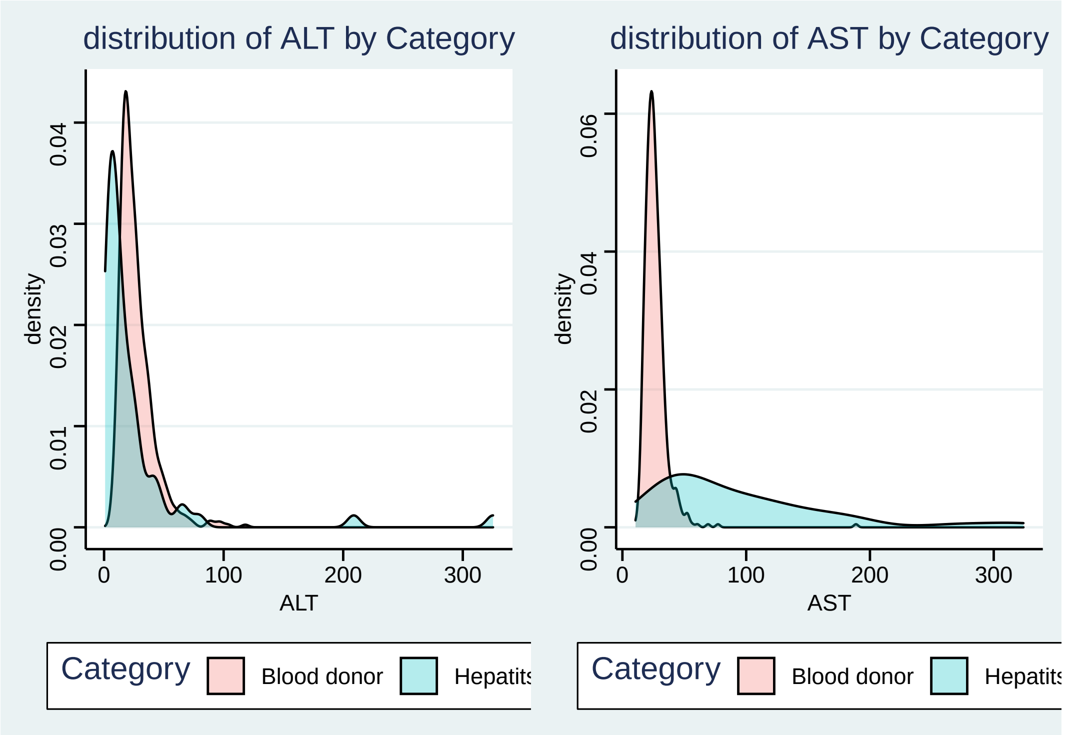
2.2.5 Laboratory Data Features

There are 10 features in the dataset belongs to laboratory blood test (ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT, PROT). I plot the density distribution by Category for these values. The results as shown below:

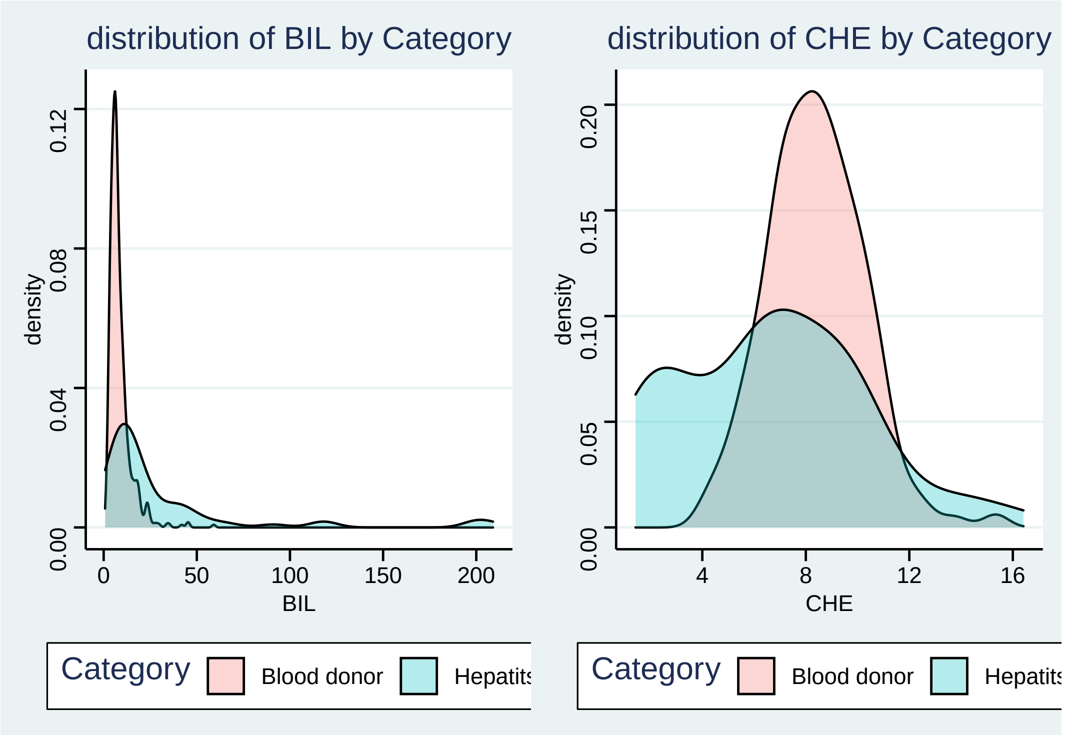
```
# plot the density distribution of laboratory blood test features
plot_blood_features <- function(feature){
    i <-1
    plots <- list()
    for(column in feature){
        col <- sym(column)
        plots[[i]]<-hepa %>% ggplot(aes(x=!!col, fill = Category)) +
            geom_density(alpha = 0.3) + ggtitle(paste("distribution of", col, "by Category")) +
            scale_fill_discrete(labels = c("Blood donor", "Hepatits")) + theme_stata()
        i<-i+1
    }
    plot_grid(plotlist = plots,ncol=2)
}
plot_blood_features(c("ALB","ALP"))
```



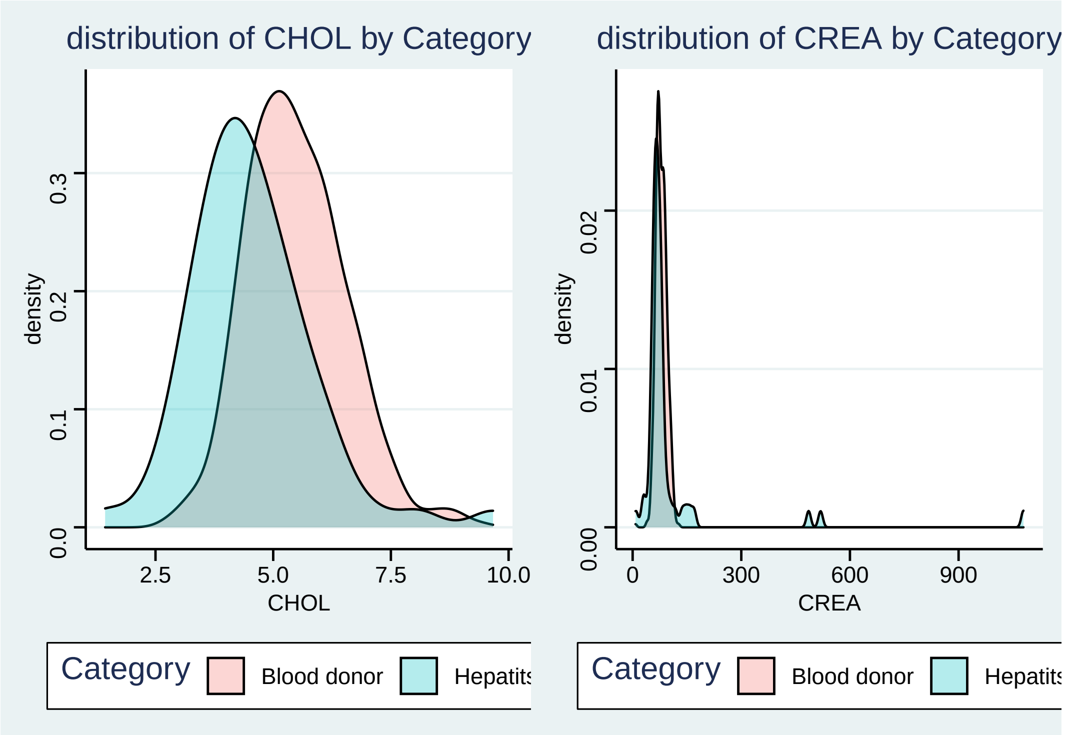
```
plot_blood_features(c("ALT","AST"))
```



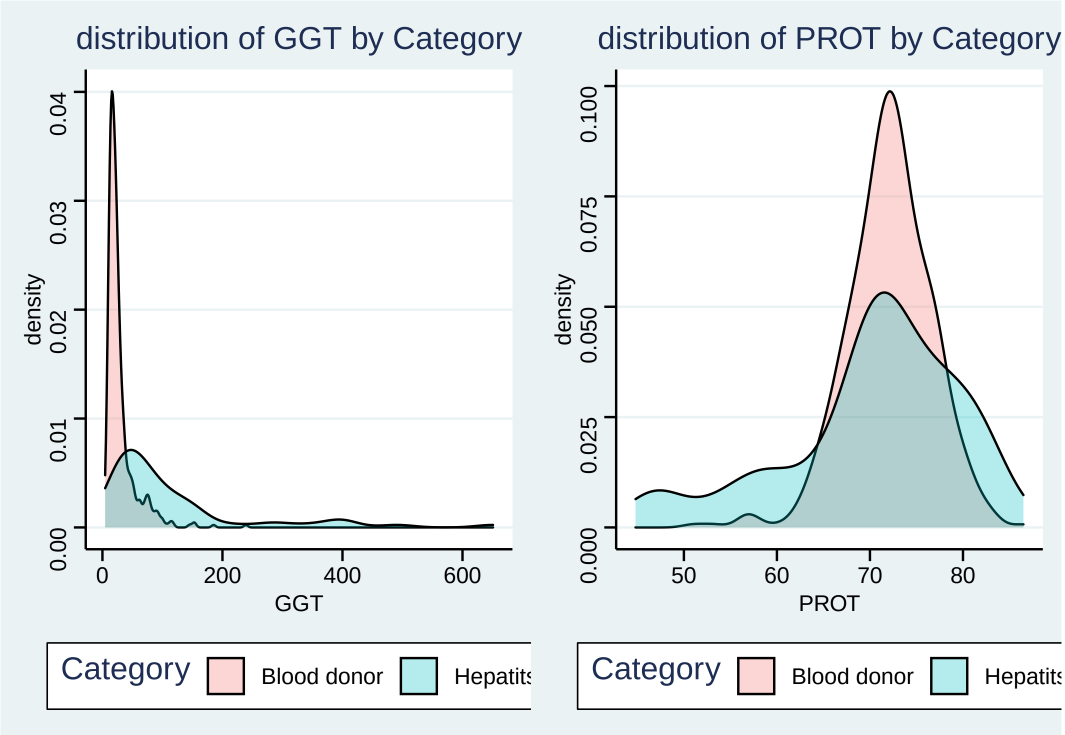
```
plot_blood_features(c("BIL","CHE"))
```



```
plot_blood_features(c("CHOL","CREA"))
```



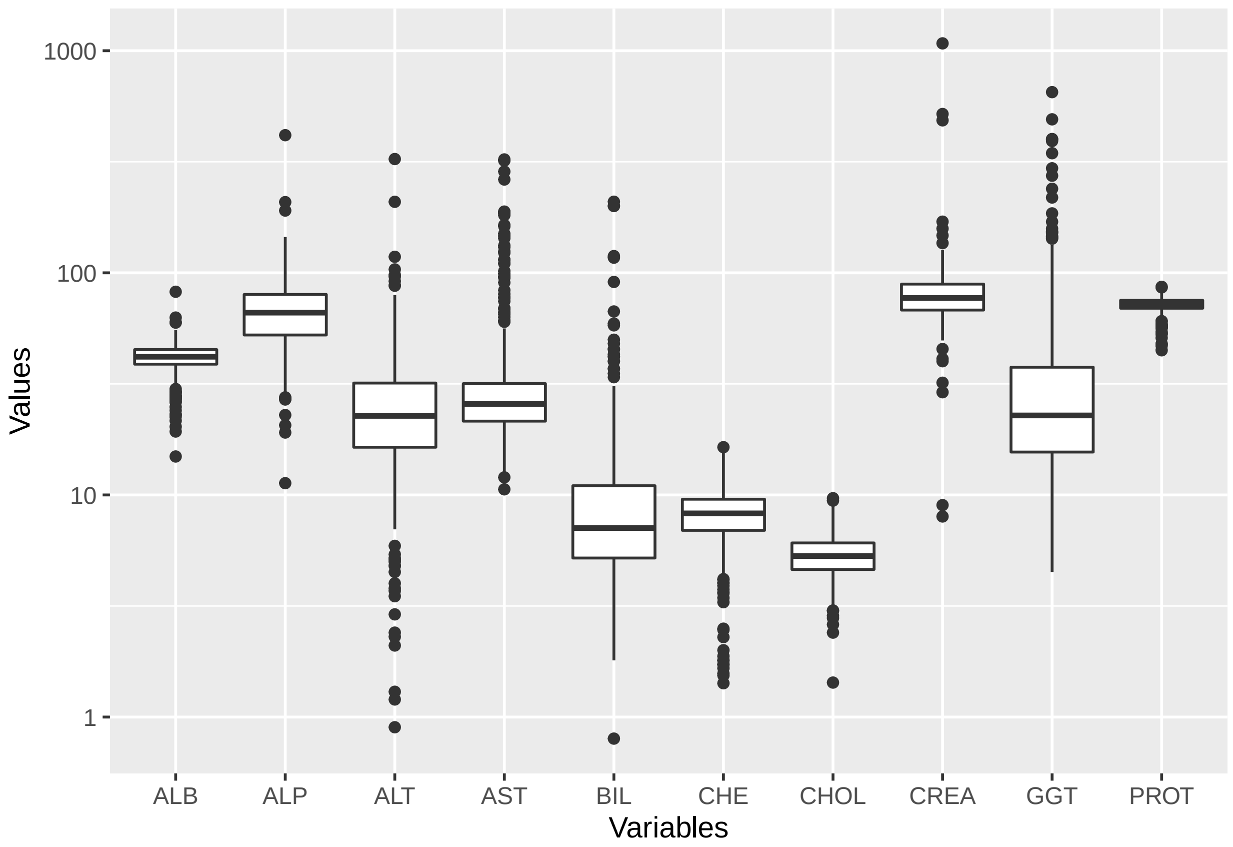
```
plot_blood_features(c("GGT","PROT"))
```



2.2.6 Mean Values

The plot below show the mean values for laboratory data using boxplots and log10 transformation:

```
# boxplot for dataset features
lab_values <- select(hepa, -1, -2, -3)
ggplot(stack(lab_values), aes(x = ind, y = values)) +
geom_boxplot() + scale_y_log10() + labs(x="Variables", y= "Values")
```



3. Modeling Approach

After completing data wrangling and analysis for the dataset, I will now use machine learning algorithms to predict the Category using other database features. In this project I will use five models of machine learning to predict the Category:

1- Logistic Regression Model
2- K-Nearest Neighbors (KNN) Model 3- Decision Tree Model 4- Random Forest Model 5- Ensemble Model

The machine learning models will use the dataset features to predict the Category feature (if the person is blood donor, or the person has hepatitis or have problems in liver). We will find confusion matrix (TP, FP, FN, TN) for each model, accuracy, sensitivity, specificity, and balance accuracy of the prediction model.

- TP (True Positive): count of values actually positive and predicted positive.
- FP (False Positive): count of values actually negative and predicted positive.
- FN (False Negative): count of values actually positive and predicted negative.
- TN (True Negative): count of values actually negative and predicted negative.
- Accuracy: The proportion of cases that were correctly predicted in the test set. • Sensitivity: known as the true positive rate (TPR) or recall, the proportion of actual positive outcomes correctly identified as such.
- Specificity: known as the true negative rate (TNR), the proportion of actual negative outcomes that are correctly identified as such.
- Balanced Accuracy: overall accuracy or the average of specificity and sensitivity.

The first step I split the dataset into two parts (training and testing) using caret library, the testing will be 20% of dataset.

```
# split the dataset into training and testing using caret package
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = hepa$Category, times = 1, p = 0.2, list = FALSE)
training <- hepa[-test_index,]
testing <- hepa[test_index,]
# find number of columns and rows in training dataset
dim(training)
```

```
## [1] 470  13
```

```
# find number of columns and rows in testing dataset
dim(testing)
```

```
## [1] 119  13
```

3.1 Logistic Regression Model

Logistic regression is a predictive modeling algorithm that is used when the Y variable is binary categorical. That is, it can take only two values like 1 or 0. The goal is to determine a mathematical equation that can be used to predict the probability of event 1. Once the equation is established, it can be used to predict the Y when only the Xs are known.
In this project I use the logistic regression model not linear regression because the target feature (Category) is categorical variable with two values (0,1) which 0 indicate the blood donor and 1 indicate the hepatitis or problems in liver.

We will use the base function (glm) in R to fit the model using training dataset, setting the family argument to “binomial” and set the type=“response” to predict probability.

The goal here is to model and predict if a given Category is 0 (Blood donor) or 1 (Hepatitis or problems in liver), based on the 12 other features in the dataset.

```
# build logestic regression model usin training dataset
glm_model <- glm(Category ~ ., family = "binomial", data=training)
summary(glm_model)
```

```
##
## Call:
## glm(formula = Category ~ ., family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7659  -0.2083  -0.1172  -0.0641   3.1199
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.215633   3.732456   1.397 0.162301
## Age         -0.015264   0.027713  -0.551 0.581784
## Sex1         0.154047   0.735802   0.209 0.834168
## ALB         -0.137453   0.079539  -1.728 0.083968 .
## ALP         -0.056797   0.017223  -3.298 0.000975 ***
## ALT         -0.019710   0.012927  -1.525 0.127324
## AST          0.067832   0.019546   3.470 0.000520 ***
## BIL          0.058606   0.037051   1.582 0.113698
## CHE          0.161560   0.160581   1.006 0.314368
## CHOL        -0.607725   0.336633  -1.805 0.071028 .
## CREA         0.007353   0.005453   1.348 0.177567
## GGT          0.034555   0.008232   4.198  2.7e-05 ***
## PROT        -0.014479   0.068377  -0.212 0.832295
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 318.55  on 469  degrees of freedom
## Residual deviance: 100.64  on 457  degrees of freedom
## AIC: 126.64
##
## Number of Fisher Scoring iterations: 8
```

The logistic regression model (glm_model) is now built. we can now use it to predict the response on testing dataset, take the probability cutoff as 0.5. If the probability of Category is > 0.5, then it can be classified 0 (Blood donor). And find accuracy using mean function.

```
# predict Category on testing using glm_model
preds_glm <- predict(glm_model, newdata = testing, type = "response")
y_preds <- ifelse(preds_glm > 0.5, 1, 0) %>% factor(levels=c(1, 0))
# find accuracy of linear regression model using mean function
mean(y_preds == testing$Category)
```

```
## [1] 0.9747899
```

```
# Confusion Matrix for logistic regression model
glm_cm <- confusionMatrix(y_preds , testing$Category)
glm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106   3
##          1   0  10
##
##                Accuracy : 0.9748
##                  95% CI : (0.9281, 0.9948)
##     No Information Rate : 0.8908
##     P-Value [Acc > NIR] : 0.0006578
##
##                   Kappa : 0.8559
##
##  Mcnemar's Test P-Value : 0.2482131
##
##             Sensitivity : 1.0000
##             Specificity : 0.7692
##          Pos Pred Value : 0.9725
##          Neg Pred Value : 1.0000
##               Prevalence : 0.8908
##           Detection Rate : 0.8908
##     Detection Prevalence : 0.9160
##        Balanced Accuracy : 0.8846
##
##         ’Positive’ Class : 0
##
```

```
# summary table for evaluation metrics for logistic regression model
glm_results <- data.frame(TP=glm_cm$table[1,1], FP=glm_cm$table[1,2],
TN=glm_cm$table[2,2], FN=glm_cm$table[2,1],
Accuracy=round(glm_cm$overall["Accuracy"]*100,2),
Sensitivity=round(glm_cm$byClass["Sensitivity"]*100, 2),
Specificity=round(glm_cm$byClass["Specificity"]*100, 2),
Balanced_Accuracy=round(glm_cm$byClass["Balanced Accuracy"]*100, 2),
row.names = "Logistic Regression Model")
glm_results %>% knitr::kable(align = "cccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
Logistic Regression Model	106	3	10	0	97.48	100	76.92	88.46

3.2 K-Nearest Neighbors (KNN) Model

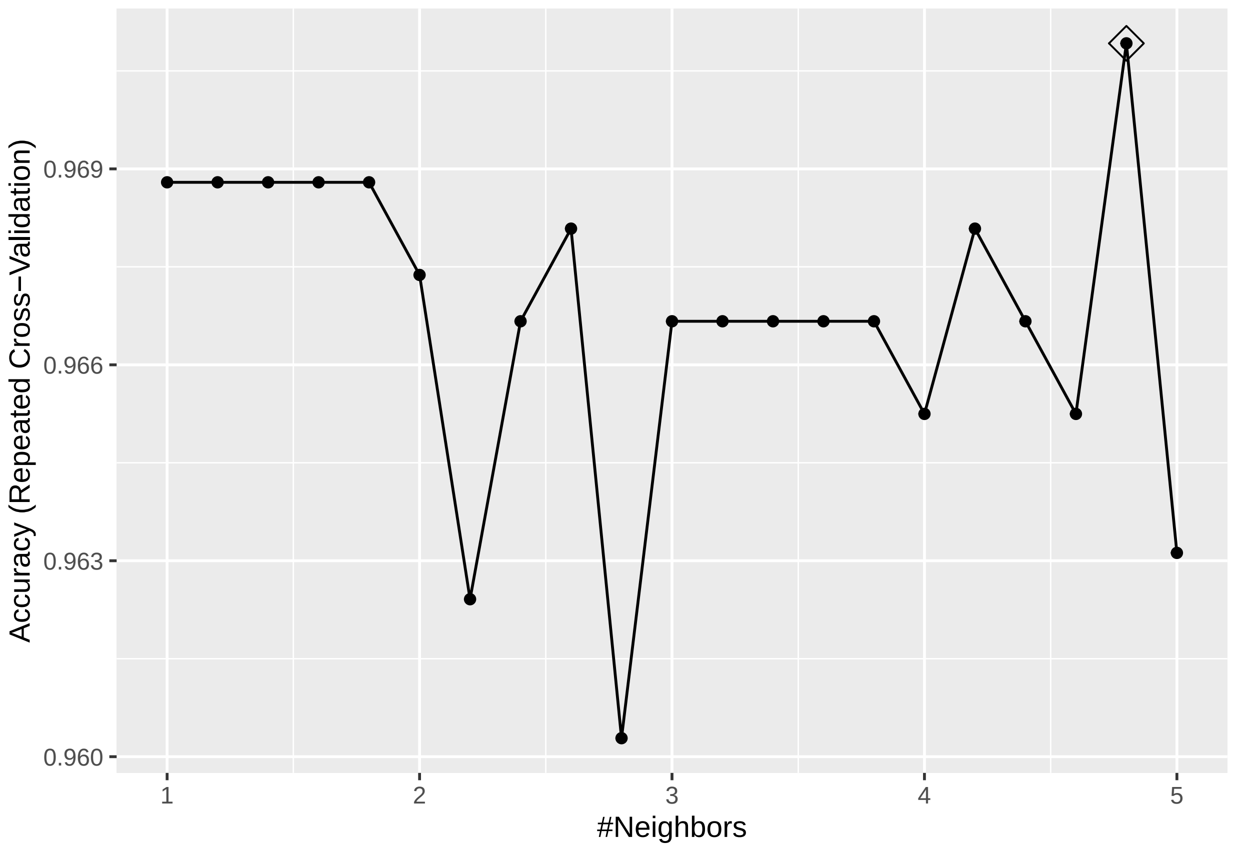
KNN is learning technique in which we try to classify the data point to a given category with the help of training set. In simple words, it captures information of all training cases and classifies new cases based on a similarity.

There are many distance functions but Euclidean is the most commonly used measure. It calculate distance of all the training cases with new case and calculates the rank in terms of distance. The smallest distance value will be ranked 1 and considered as nearest neighbor.

Cross-validation is a smart way to find out the optimal K value. It estimates the validation error rate by holding out a subset of the training set from the model building process. Here we are using cross validation method using trainControl method with number of folds equals 10 and repeats = 3.

Using train function we run our knn model, with Category as target variable and other features is independent variables. In ‘data=’ we pass our training set, method=knn and multiple K values (tuneGrid) to find optimal K values.

```
# use train function to find optimal K using 10-fold cross validation
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "repeatedcv", number = 10, repeats=3, p = .9)
train_knn <- train(Category ~ ., method = "knn", data = training,
tuneGrid = data.frame(k = seq(1, 5, 0.2)), trControl = control)
# plot accuracy for each k value and select optimal k value
ggplot(train_knn, highlight = TRUE)
```



```
# print optimal k value
train_knn$bestTune
```

```
##      k
## 20 4.8
```

Now I build the KNN model using knn3 method with optimal K value is 4.8:

```
# build final knn model with optimal k = 4.8
knn_model <- knn3(Category ~ ., data = training, k = 4.8)
```

we can now use KNN model to predict Category on testing dataset.

```
# predict Category in testing using final knn model
preds_knn <- predict(knn_model, newdata = testing, type = "class")
# Confusion Matrix for KNN model
knn_cm <- confusionMatrix(preds_knn , testing$Category)
knn_cm
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0   1
##            0 106   4
##          1   0   9
##
##                Accuracy : 0.9664
##                  95% CI : (0.9162, 0.9908)
##     No Information Rate : 0.8908
##     P-Value [Acc > NIR] : 0.002545
##
##                   Kappa : 0.8003
##
##  Mcnemar’s Test P-Value : 0.133614
##
##             Sensitivity : 1.0000
##             Specificity : 0.6923
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8908
##          Detection Rate : 0.8908
##    Detection Prevalence : 0.9244
##       Balanced Accuracy : 0.8462
##
##        ’Positive’ Class : 0
##
```

```r
# summary table for evaluation metrics for knn model
knn_results <- data.frame(TP=knn_cm$table[1,1], FP=knn_cm$table[1,2],
TN=knn_cm$table[2,2], FN=knn_cm$table[2,1],
Accuracy=round(knn_cm$overall["Accuracy"]*100,2),
Sensitivity=round(knn_cm$byClass["Sensitivity"]*100, 2),
Specificity=round(knn_cm$byClass["Specificity"]*100, 2),
Balanced_Accuracy=round(knn_cm$byClass["Balanced Accuracy"]*100, 2),
row.names = "K-Nearest Neighbors (KNN) model")
knn_results %>% knitr::kable(align = "cccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
K-Nearest Neighbors (KNN) model	106	4	9	0	96.64	100	69.23	84.62

3.3 Decision Tree Model

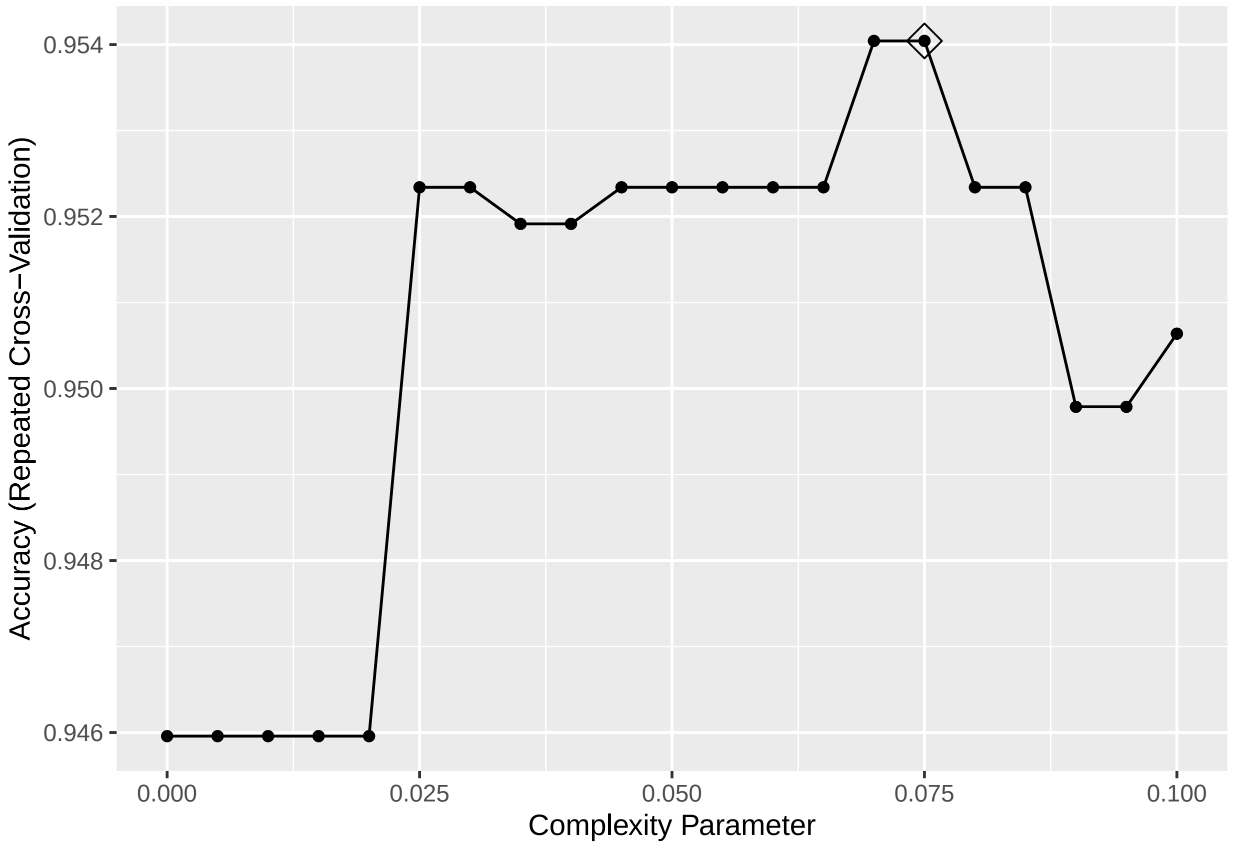
Decision Tree Analysis is a general, predictive modeling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

I will use the rpart library. And plot important variables related to Category in a hierarchical format using rpart.plot library. before the model is constructed, an optimal complexity parameter is chosen (cp). We will use trainControl with replacement and 10-fold cross validation and repeats = 5 to select the optimal complexity parameter (cp).

Using train function we run our model, with Category as target variable and other features is independent variables. In ‘data=’ we pass our training set, method=rpart

```
# use train function to find optimal cp using 10-fold cross validation
set.seed(1, sample.kind = "Rounding")
control = trainControl(method="repeatedcv", number=10, repeats=5, p = .9)
train_dt <- train(Category ~ ., method = "rpart",
tuneGrid = data.frame(cp = seq(0, 0.1, 0.005)), trControl = control, data = training)
# plot accuracy for each cp value and select optimal cp value
ggplot(train_dt, highlight = TRUE)
```



```
# print optimal cp value
train_dt$bestTune
```

```
##       cp
## 16 0.075
```

Now I build the decision tree model using rpart method with optimal cp value is 0.075:

```
# build final decision tree model with optimal cp = 0.075
dt_model <- rpart(Category~., cp = 0.075, data = training)
```

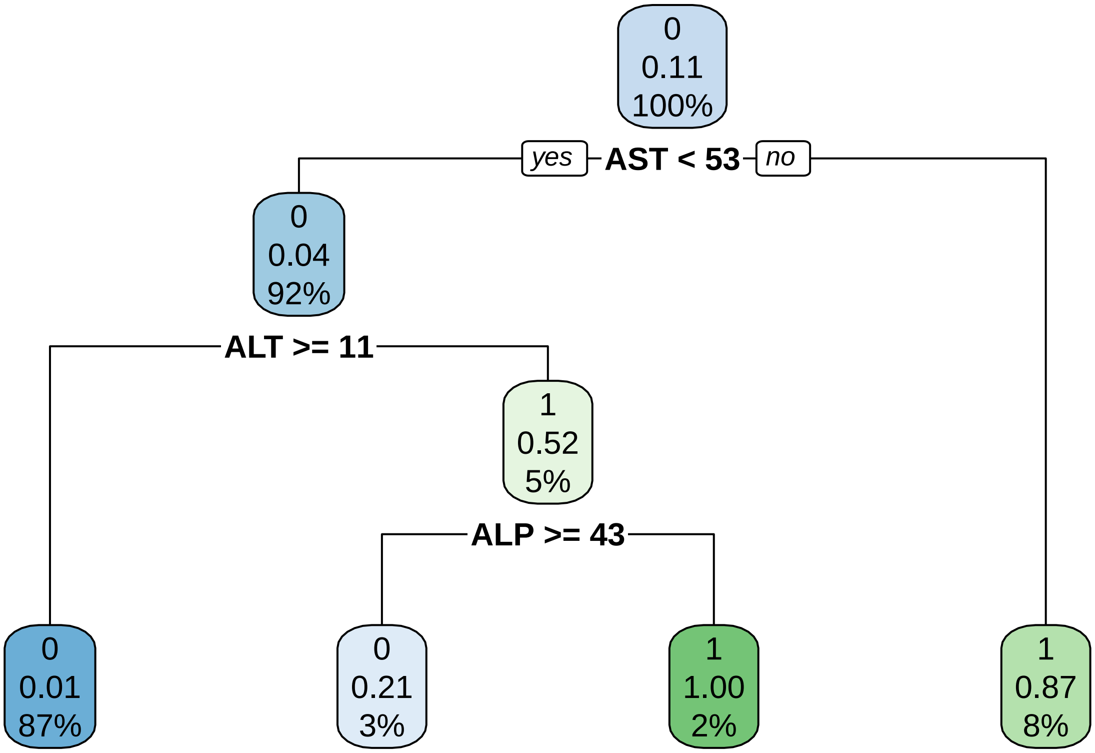
we can now use decision tree model to predict Category on testing dataset.

```
# predict Category in testing using final decision tree model
preds_dt <- predict(dt_model, newdata = testing, type = "class")
# Confusion Matrix for decision tree model
dt_cm <- confusionMatrix(preds_dt , testing$Category)
dt_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106   4
##          1   0   9
##
##                Accuracy : 0.9664
##                  95% CI : (0.9162, 0.9908)
##     No Information Rate : 0.8908
##     P-Value [Acc > NIR] : 0.002545
##
##                   Kappa : 0.8003
##
##  Mcnemar's Test P-Value : 0.133614
##
##             Sensitivity : 1.0000
##             Specificity : 0.6923
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8908
##          Detection Rate : 0.8908
##    Detection Prevalence : 0.9244
##       Balanced Accuracy : 0.8462
##
##        'Positive' Class : 0
##
```

plot decision tree in a hierarchical format using rpart.plot library

```
# plot decision tree
rpart.plot(dt_model)
```



```
# summary table for evaluation metrics for decision tree model
dt_results <- data.frame(TP=dt_cm$table[1,1], FP=dt_cm$table[1,2],
TN=dt_cm$table[2,2], FN=dt_cm$table[2,1],
Accuracy=round(dt_cm$overall["Accuracy"]*100,2),
Sensitivity=round(dt_cm$byClass["Sensitivity"]*100, 2),
Specificity=round(dt_cm$byClass["Specificity"]*100, 2),
Balanced_Accuracy=round(dt_cm$byClass["Balanced Accuracy"]*100, 2),
row.names = "Decision Tree model")
dt_results %>% knitr::kable(align = "cccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
Decision Tree model	106	4	9	0	96.64	100	69.23	84.62

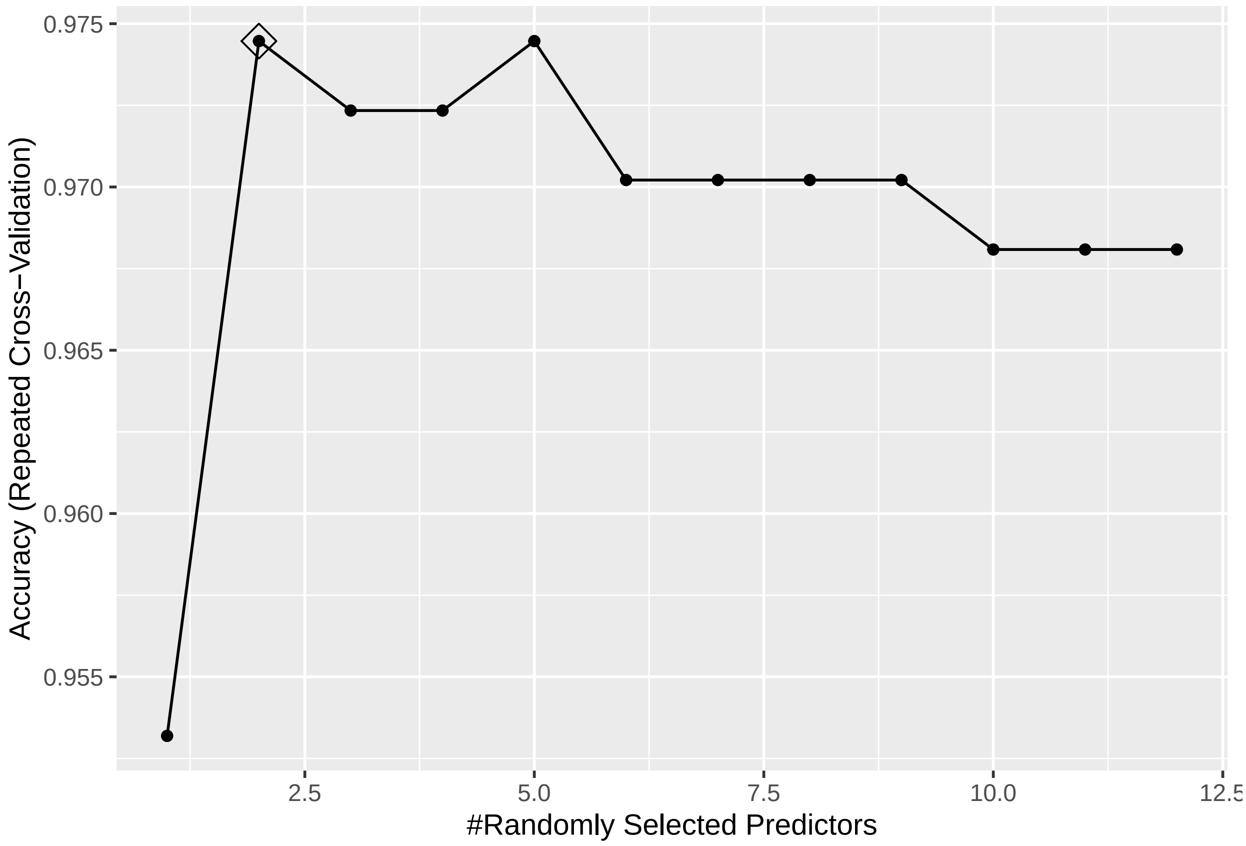
3.4 Random Forest Model

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

As we do in decision trees we use train function to run our model, with Category as target variable and other features is independent variables. In 'data=' we pass our training set, method=rf

```
# use train function to find optimal mtry using 10-fold cross validation
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "repeatedcv", number = 10, p = .9)
train_rf <- train(Category ~ . ,method = "rf",
tuneGrid = data.frame(mtry = 1:12), data = training, trControl = control)
# plot accuracy for each mtry value and select optimal mtry value
ggplot(train_rf, highlight = TRUE)
```



```
# print optimal mtry value
train_rf$bestTune
```

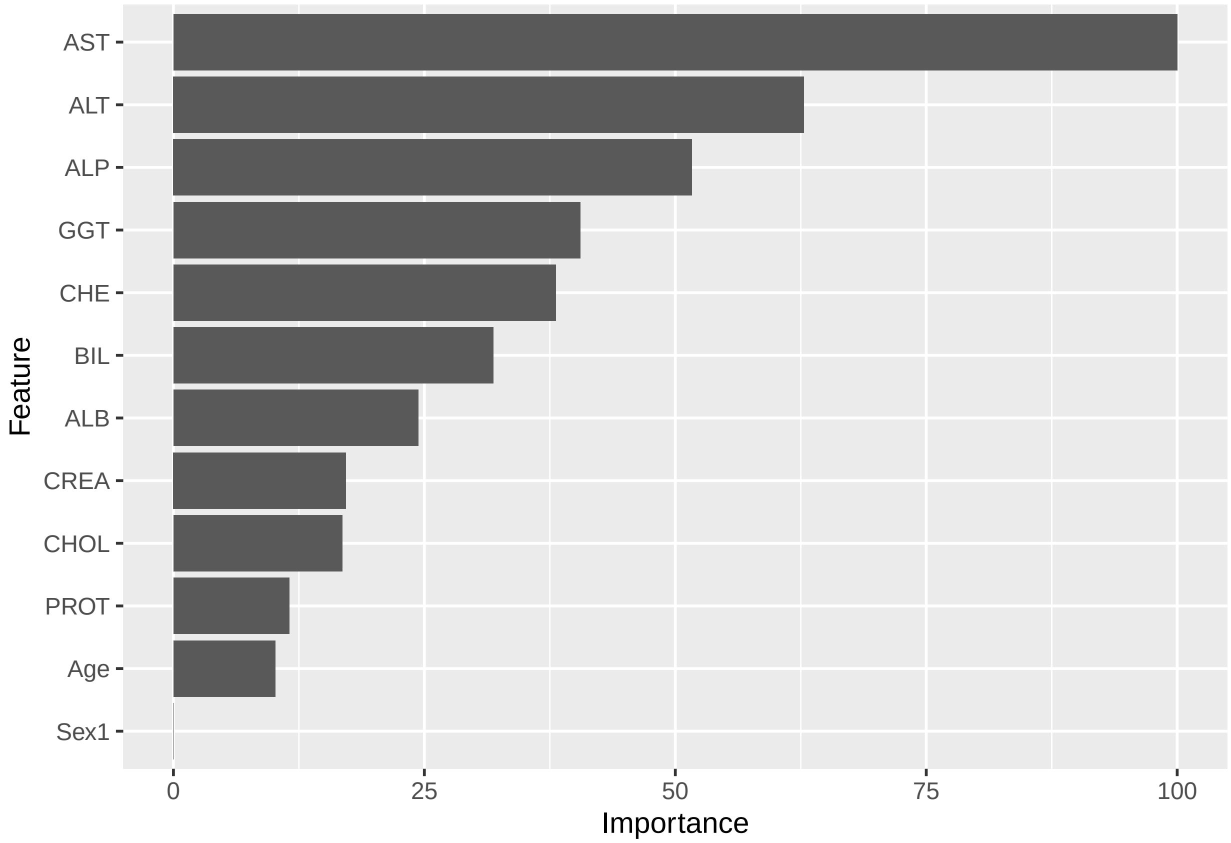
```
##   mtry
## 2    2
```

Now I build the rf model using randomForest method with optimal mtry value is 2:

```
# build final rf model with optimal mtry = 2
rf_model <- randomForest(Category ~., data = training, mtry = 2, importance = TRUE)
# list importance variable
varImp(train_rf)
```

```
## rf variable importance
##
##       Overall
## AST   100.00
## ALT    62.81
## ALP    51.66
## GGT    40.54
## CHE    38.07
## BIL    31.87
## ALB    24.40
## CREA   17.19
## CHOL   16.80
## PROT   11.55
## Age    10.14
## Sex1    0.00
```

```
# plot importance variable
ggplot(varImp(train_rf))
```



we can now use random forst model (rf_model) to predict Category on testing dataset.

```
# predict Category in testing using final random forest model
preds_rf <- predict(rf_model, newdata = testing, type = "class")
# Confusion Matrix for random forest model
rf_cm <- confusionMatrix(preds_rf , testing$Category)
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106   2
##          1   0  11
##
##                Accuracy : 0.9832
##                  95% CI : (0.9406, 0.998)
##     No Information Rate : 0.8908
##     P-Value [Acc > NIR] : 0.0001273
##
##                   Kappa : 0.9074
##
##  Mcnemar's Test P-Value : 0.4795001
##
##             Sensitivity : 1.0000
##             Specificity : 0.8462
##          Pos Pred Value : 0.9815
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8908
##          Detection Rate : 0.8908
##    Detection Prevalence : 0.9076
##       Balanced Accuracy : 0.9231
##
##        'Positive' Class : 0
##
```

```
# summary table for evaluation metrics for decision tree model
rf_results <- data.frame(TP=rf_cm$table[1,1], FP=rf_cm$table[1,2],
TN=rf_cm$table[2,2], FN=rf_cm$table[2,1],
Accuracy=round(rf_cm$overall["Accuracy"]*100,2),
Sensitivity=round(rf_cm$byClass["Sensitivity"]*100, 2),
Specificity=round(rf_cm$byClass["Specificity"]*100, 2),
Balanced_Accuracy=round(rf_cm$byClass["Balanced Accuracy"]*100, 2),
row.names = "Random Forest model")
rf_results %>% knitr::kable(align = "cccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
Random Forest model	106	2	11	0	98.32	100	84.62	92.31

3.5 Ensemble Method

Now I will combine the prediction data from the previous four machine learning (logistic regression, k-nearest neighbors, decision trees, random forest) to find if we can obtain better prediction.

```
# combine glm, knn, dt, rf ML methods
set.seed(1, sample.kind = "Rounding")
ensemble <- cbind(glm = preds_glm == 1, rf = preds_rf == 1, knn = preds_knn == 1, dt = preds_dt == 1)
preds_en <- ifelse(rowMeans(ensemble) > 0.5, 1, 0) %>% factor(levels=c(0, 1))
mean(preds_en == testing$Category)
```

```
## [1] 0.9579832
```

```
# Confusion Matrix for ensemble model
en_cm <- confusionMatrix(preds_en, testing$Category)
en_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106   5
##          1   0   8
##
##                Accuracy : 0.958
##                  95% CI : (0.9047, 0.9862)
##     No Information Rate : 0.8908
##     P-Value [Acc > NIR] : 0.007867
##
##                   Kappa : 0.7403
##
##  Mcnemar's Test P-Value : 0.073638
##
##             Sensitivity : 1.0000
##             Specificity : 0.6154
##          Pos Pred Value : 0.9550
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8908
##          Detection Rate : 0.8908
##    Detection Prevalence : 0.9328
##       Balanced Accuracy : 0.8077
##
##        'Positive' Class : 0
##
```

```
# summary table for evaluation metrics for ensemble model
en_results <- data.frame(TP=en_cm$table[1,1], FP=en_cm$table[1,2],
TN=en_cm$table[2,2], FN=en_cm$table[2,1],
Accuracy=round(en_cm$overall["Accuracy"]*100,2),
Sensitivity=round(en_cm$byClass["Sensitivity"]*100, 2),
Specificity=round(en_cm$byClass["Specificity"]*100, 2),
Balanced_Accuracy=round(en_cm$byClass["Balanced Accuracy"]*100, 2),
row.names = "Ensemble model")
en_results %>% knitr::kable(align = "cccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
Ensemble model	106	5	8	0	95.8	100	61.54	80.77

4. Results

The results of the five machine learning used in this report (logistic regression, k-nearest neighbors, decision trees, random forest, ensemble method) are summarized in the following table:

```r
# final results for all 5 machine learning methods
all_results <- rbind("linear regression" = glm_results,
"k-nearest neighbors"=  knn_results, "decision tree"= dt_results,
"random forest"= rf_results, ensemble = en_results)
all_results %>% knitr::kable(align = "lcccccccc")
```

	TP	FP	TN	FN	Accuracy	Sensitivity	Specificity	Balanced_Accuracy
linear regression	106	3	10	0	97.48	100	76.92	88.46
k-nearest neighbors	106	4	9	0	96.64	100	69.23	84.62
decision tree	106	4	9	0	96.64	100	69.23	84.62
random forest	106	2	11	0	98.32	100	84.62	92.31
ensemble	106	5	8	0	95.80	100	61.54	80.77

5. Conclusion

In this report I analysis the Hepatitis C virus dataset for group of patients and use five methods of machine learning to predict the persons who have Hepatitis or problems in liver.

I analysis the dataset like correlations between fetchers, Category (Hepatitis or no) distribution, age distribution, sex distribution, density distribution for Lab fetchers (ALB, ALP, AST, ...) and means values for all fetchers.

The machine learning methods I use are: logistic regression, k-nearest neighbors, decision trees, random forest, ensemble method. And the performance parameters used is accuracy, sensitivity, specificity, and balanced accuracy.

The best model was the random forest model in all performance parameters: accuracy, sensitivity, specificity, and balanced accuracy (shown in above table).

The dataset I use is relatively small, I think that if I use a dataset larger the results will be better especially in specificity.

6. References

1. Hepatitis C virus - Blood based Detection, Kaggle website, link
2. “Introduction to Data Science - Data Analysis and Prediction Algorithms with R”, Rafael A. Irizarry, link
3. “R Markdown Cookbook”, Yihui Xie, Christophe Dervieux, Emily Riederer, link
4. World Health Organization, link
5. ggplot2 package, link
6. machine learning plus website, link
7. listen data website, link
8. hack earth website, link
9. java point website, link