# BalgobinS.DA5030.Project.Rmd

2023-08-08

## Predicting HCV Infection and Progrssion Using Machine Learning

**This data set contains the lab values of both blood donors and Hep C patients including demographic attributes like age and sex. It is taken from the UC Irvine repository. The target variable is categorical, and we are planning to predict whether an individual has HCV or progressed HCV (fibrosis/cirrhosis). We will be employing multi-class classification and grouping patients of progressed stages of hepatitis into one group as there is extreme class imbalance and trying to predict fibrosis and cirrhosis separately would not yield significant statistical power. Hepatitis C is a a viral infection that causes liver inflammation. Fibrosis occurs when where is a limited accumulation of scar tissue, and cirrhosis occurs when there is extensive fibrosis. Among those with a chronic HCV infection, 15-20% progress to end-stage liver disease. HCV remains a significant public health challenge, and in order to reap the benefits of novel therapies, we need a reduction in the undiagnosed population coupled with early diagnosis so that patients can be treated before experiencing the long term ramifications of HCV.**

### Load data

```
# Load data from google drive URL
hcv_data <- read.csv("https://drive.google.com/uc?export=download&id=1IBnIVbW_uSiDxp_kGwkmhk2D3GVEXaQB")
```

### Explore data

```
dim(hcv_data)
```

```
## [1] 615  14
```

```
glimpse(hcv_data)
```

```
## Rows: 615
## Columns: 14
## $ X        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
## $ Category <chr> "0=Blood Donor", "0=Blood Donor", "0=Blood Donor", "0=Blood D~
## $ Age      <int> 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 33, 3~
## $ Sex      <chr> "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "~
## $ ALB      <dbl> 38.5, 38.5, 46.9, 43.2, 39.2, 41.6, 46.3, 42.2, 50.9, 42.4, 4~
## $ ALP      <dbl> 52.5, 70.3, 74.7, 52.0, 74.1, 43.3, 41.3, 41.9, 65.5, 86.3, 5~
## $ ALT      <dbl> 7.7, 18.0, 36.2, 30.6, 32.6, 18.5, 17.5, 35.8, 23.2, 20.3, 21~
## $ AST      <dbl> 22.1, 24.7, 52.6, 22.6, 24.8, 19.7, 17.8, 31.1, 21.2, 20.0, 2~
```

```
## $ BIL      <dbl> 7.5, 3.9, 6.1, 18.9, 9.6, 12.3, 8.5, 16.1, 6.9, 35.2, 17.2, 5~
## $ CHE      <dbl> 6.93, 11.17, 8.84, 7.33, 9.15, 9.92, 7.01, 5.82, 8.69, 5.46, ~
## $ CHOL     <dbl> 3.23, 4.80, 5.20, 4.74, 4.32, 6.05, 4.79, 4.60, 4.10, 4.45, 3~
## $ CREA     <dbl> 106, 74, 86, 80, 76, 111, 70, 109, 83, 81, 78, 79, 78, 65, 63~
## $ GGT      <dbl> 12.1, 15.6, 33.2, 33.8, 29.9, 91.0, 16.9, 21.5, 13.7, 15.9, 2~
## $ PROT     <dbl> 69.0, 76.5, 79.3, 75.7, 68.7, 74.0, 74.5, 67.1, 71.3, 69.9, 7~
```

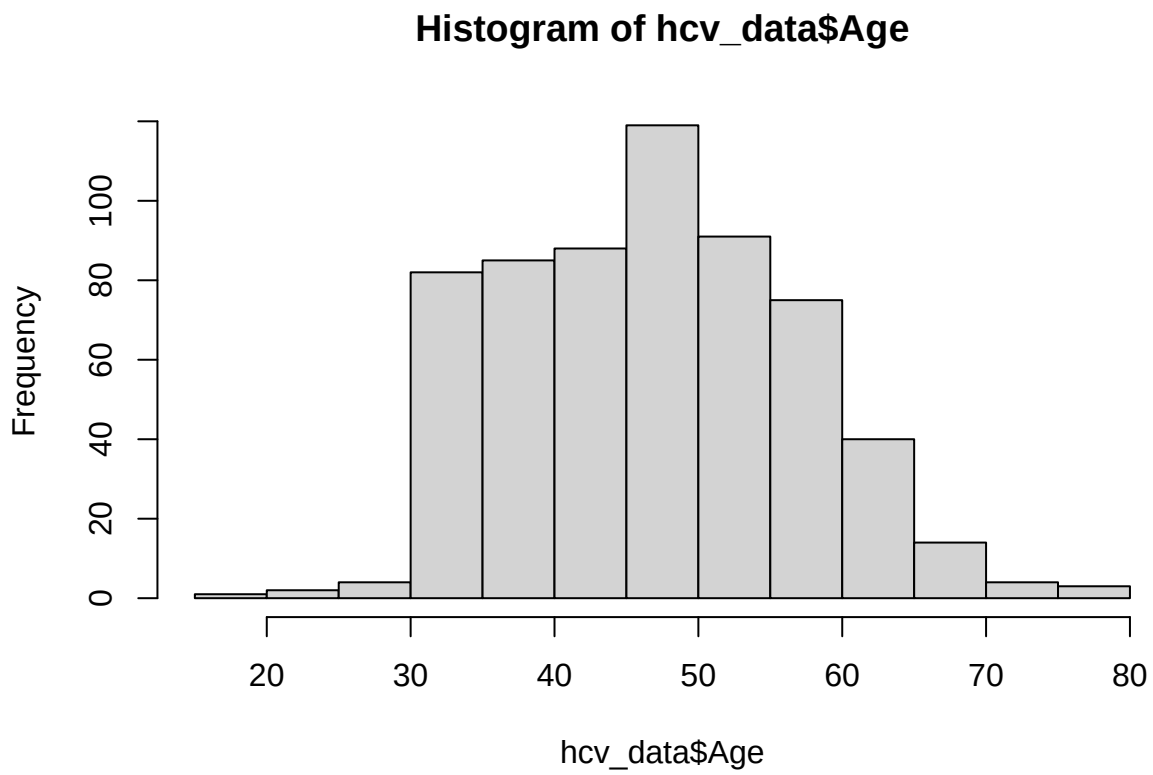
```
summary(hcv_data)
```

```
##        X           Category              Age            Sex
##  Min.   :  1.0   Length:615         Min.   :19.00   Length:615
##  1st Qu.:154.5   Class :character   1st Qu.:39.00   Class :character
##  Median :308.0   Mode  :character   Median :47.00   Mode  :character
##  Mean   :308.0                      Mean   :47.41
##  3rd Qu.:461.5                      3rd Qu.:54.00
##  Max.   :615.0                      Max.   :77.00
##
##       ALB              ALP              ALT              AST
##  Min.   :14.90   Min.   : 11.30   Min.   :  0.90   Min.   : 10.60
##  1st Qu.:38.80   1st Qu.: 52.50   1st Qu.: 16.40   1st Qu.: 21.60
##  Median :41.95   Median : 66.20   Median : 23.00   Median : 25.90
##  Mean   :41.62   Mean   : 68.28   Mean   : 28.45   Mean   : 34.79
##  3rd Qu.:45.20   3rd Qu.: 80.10   3rd Qu.: 33.08   3rd Qu.: 32.90
##  Max.   :82.20   Max.   :416.60   Max.   :325.30   Max.   :324.00
##  NA's   :1       NA's   :18       NA's   :1
##       BIL              CHE              CHOL             CREA
##  Min.   :  0.8   Min.   : 1.420   Min.   :1.430   Min.   :   8.00
##  1st Qu.:  5.3   1st Qu.: 6.935   1st Qu.:4.610   1st Qu.:  67.00
##  Median :  7.3   Median : 8.260   Median :5.300   Median :  77.00
##  Mean   : 11.4   Mean   : 8.197   Mean   :5.368   Mean   :  81.29
##  3rd Qu.: 11.2   3rd Qu.: 9.590   3rd Qu.:6.060   3rd Qu.:  88.00
##  Max.   :254.0   Max.   :16.410   Max.   :9.670   Max.   :1079.10
##                                   NA's   :10
##       GGT              PROT
##  Min.   :  4.50   Min.   :44.80
##  1st Qu.: 15.70   1st Qu.:69.30
##  Median : 23.30   Median :72.20
##  Mean   : 39.53   Mean   :72.04
##  3rd Qu.: 40.20   3rd Qu.:75.40
##  Max.   :650.90   Max.   :90.00
##                   NA's   :1
```

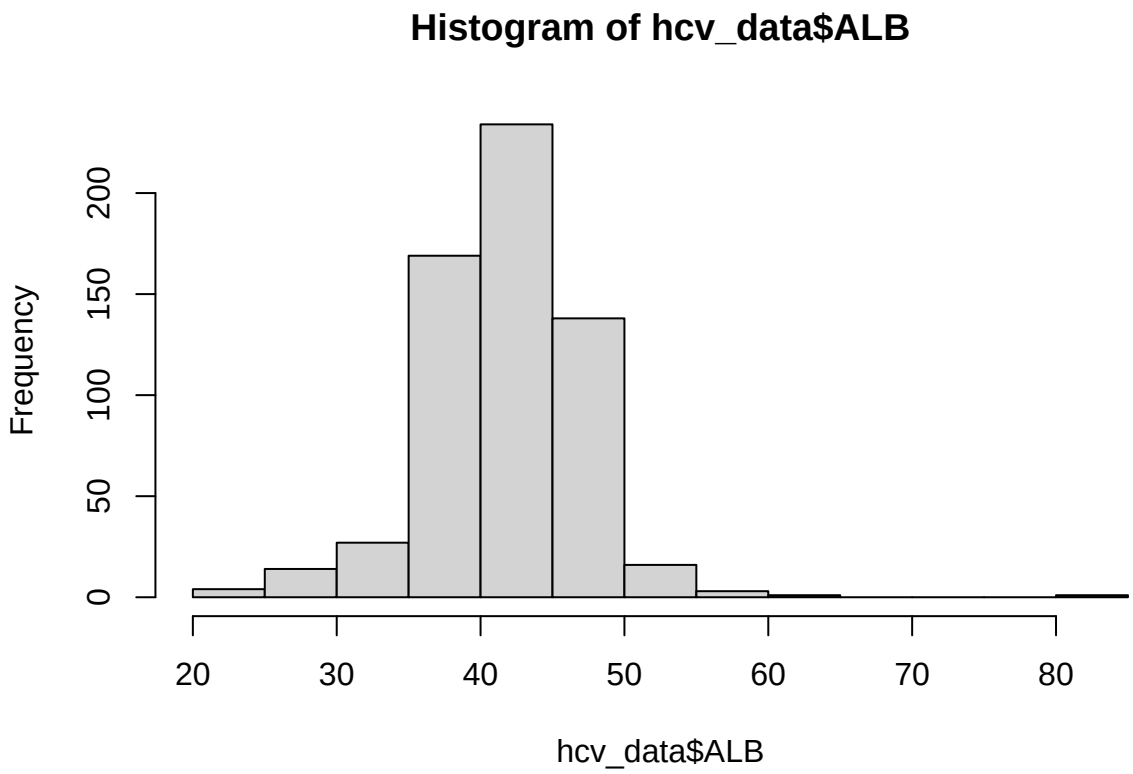
From the initial data exploration, we see that the target variable is "Category" and has 5 classes. There are 2 categorical variables, age and sex, and 10 continuous variables which represent the different lab tests and their values. The first column seems to be the patient ID, which we can drop. The mean and median are close for several variables, however the max values are quite far off, indicating a skewed distribution with outliers for some of the variables. ALB, CHOL, and PROT might be normally distributed. We will remove variable X as it is patient ID and not important, and also remove the rows that have 0s= suspected Blood Donor. These likely indicate patients who are suspected to have HCV infection and could likely be in any stage of HCV. This just adds noise to the data so we will remove those patients. Given the extreme class imbalance, we will group patients who have fibrosis and cirrhosis into one category called "Progressed HCV".

```r
# Remove ID column
hcv_data <- subset(hcv_data, select = -X)

# Remove rows with "0s = suspected Blood Donor in category
hcv_data <- hcv_data %>% filter(Category != "0s=suspect Blood Donor")

# Combine Hep groups into one for binary classification
vals_to_replace <- c("2=Fibrosis", "3=Cirrhosis")
replacement_val <- c("Progressed HCV")
hcv_data <- hcv_data %>%
  mutate(
    Category = ifelse(Category %in% vals_to_replace, replacement_val, Category)
  )
```

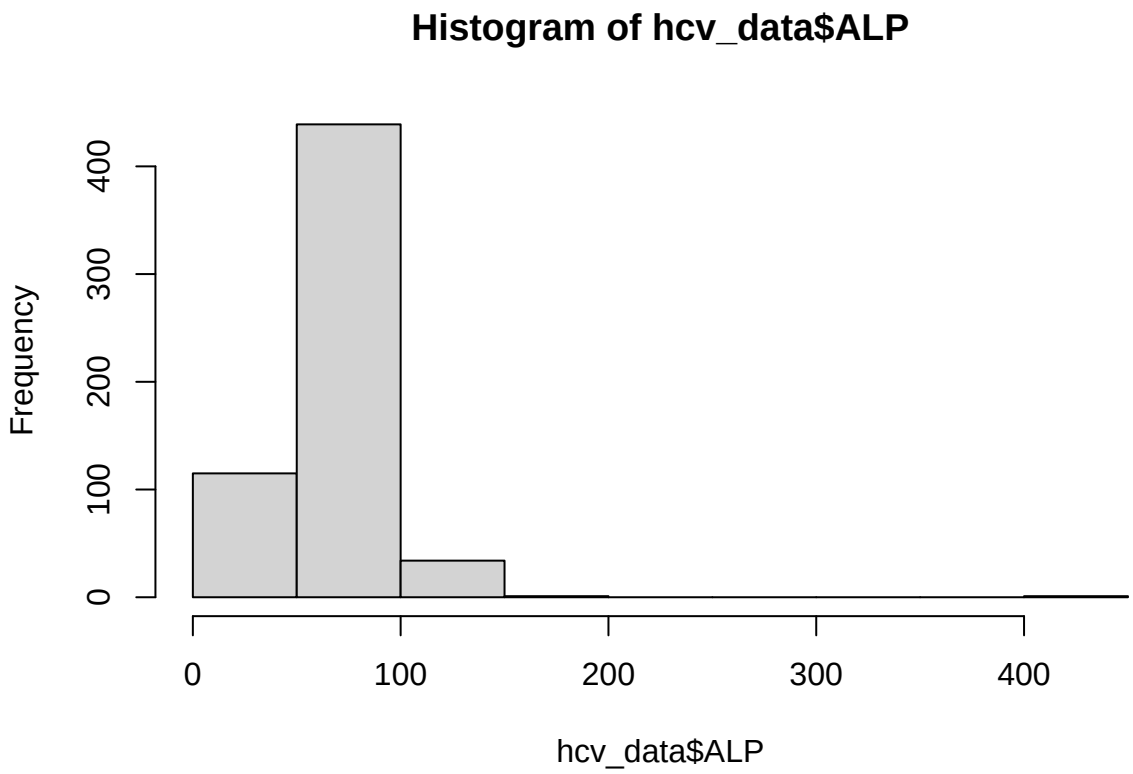
```r
# Plot density for each variable
hist(hcv_data$Age)
```



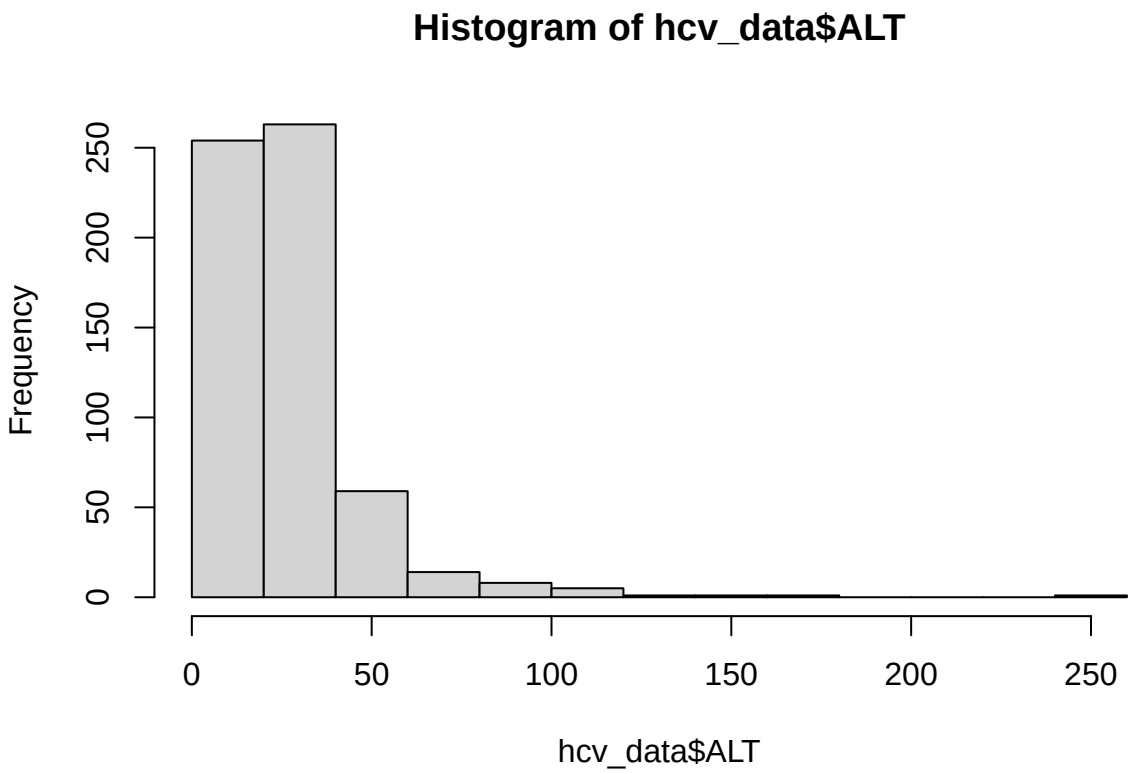

```r
hist(hcv_data$ALB)
```



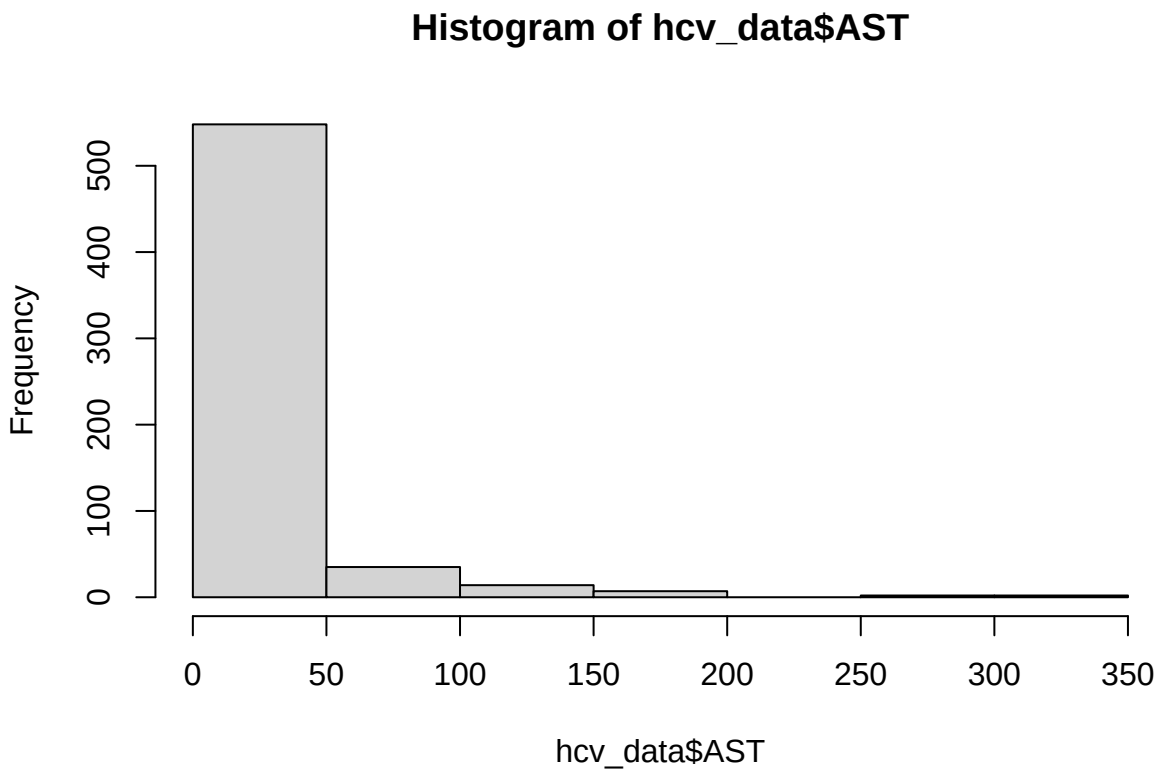

```
hist(hcv_data$ALP)
```



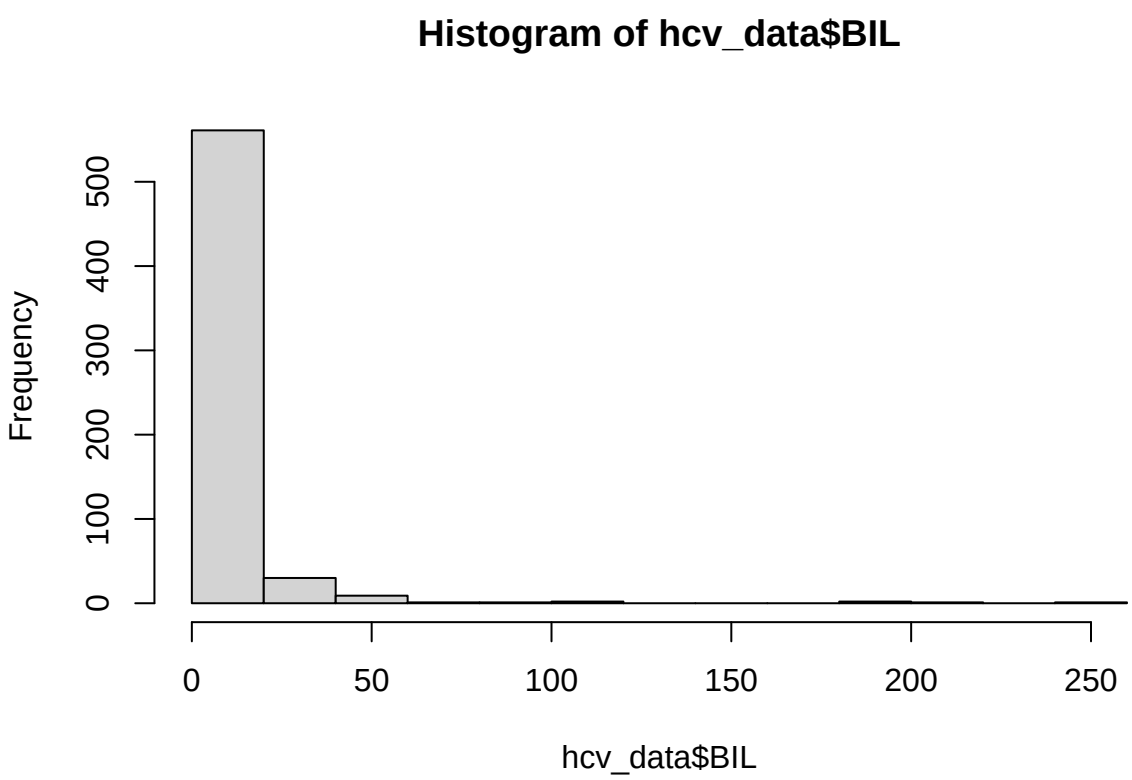

```
hist(hcv_data$ALT)
```



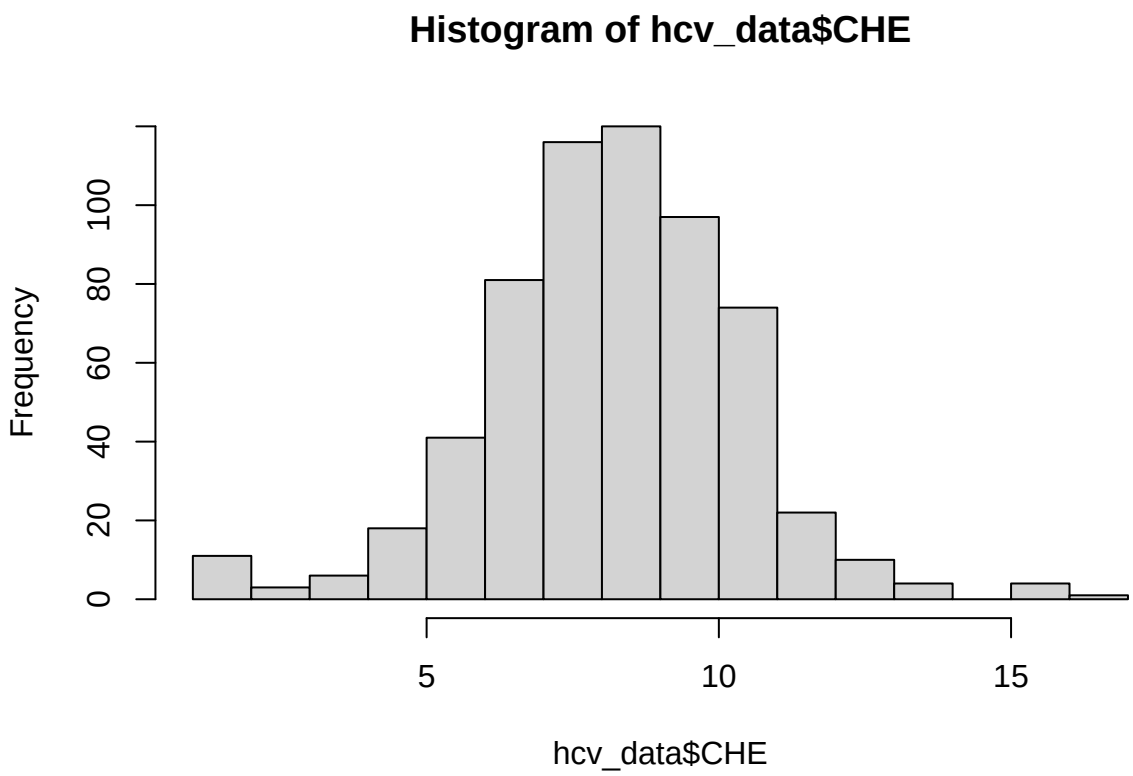

```
hist(hcv_data$AST)
```



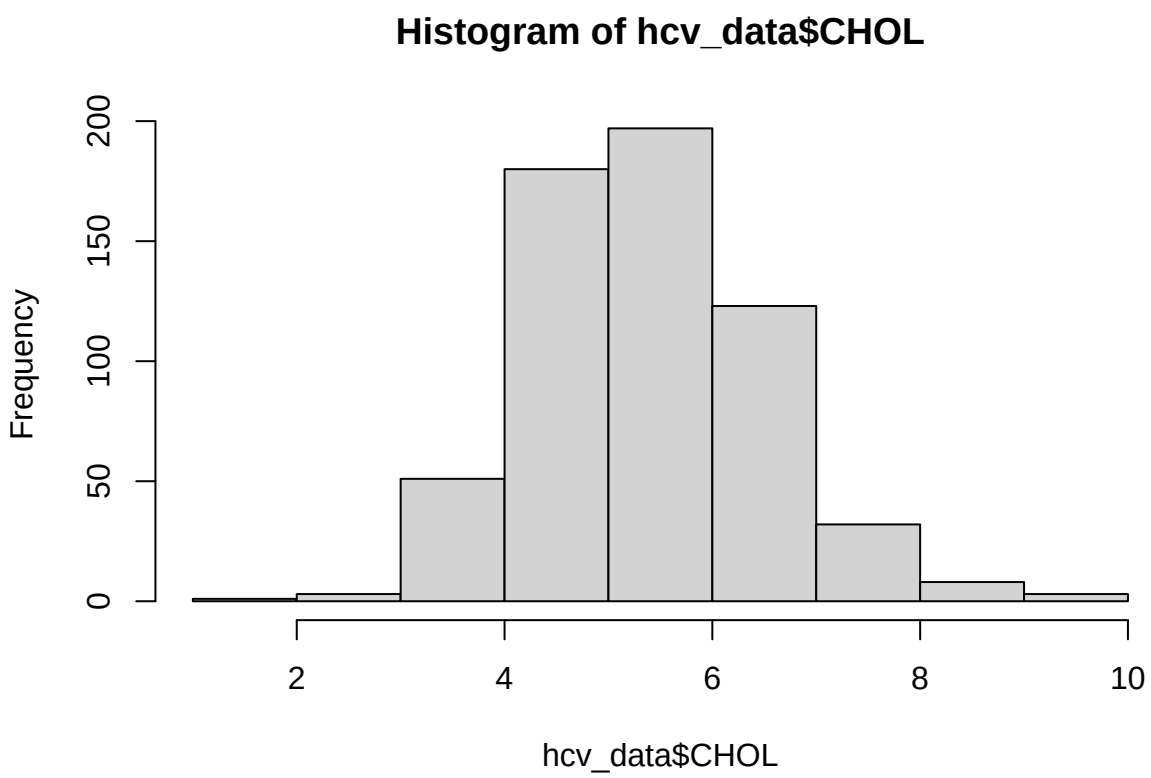

```
hist(hcv_data$BIL)
```



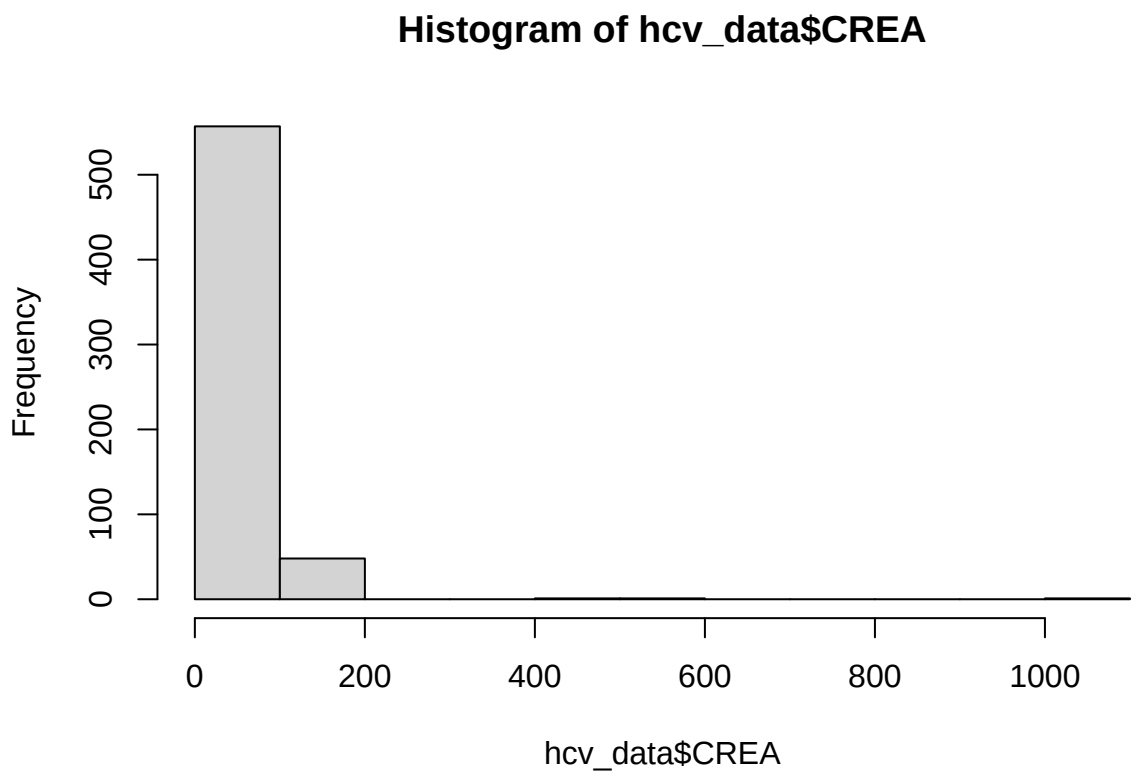

```
hist(hcv_data$CHE)
```



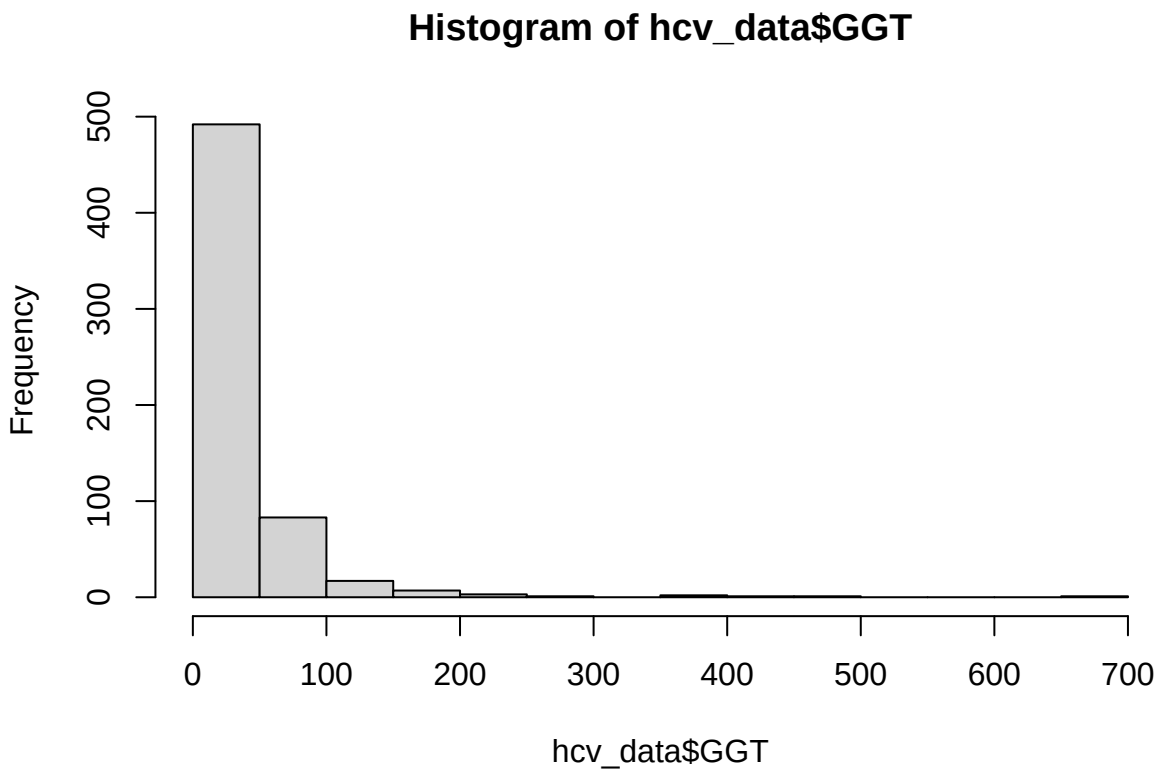

```
hist(hcv_data$CHOL)
```



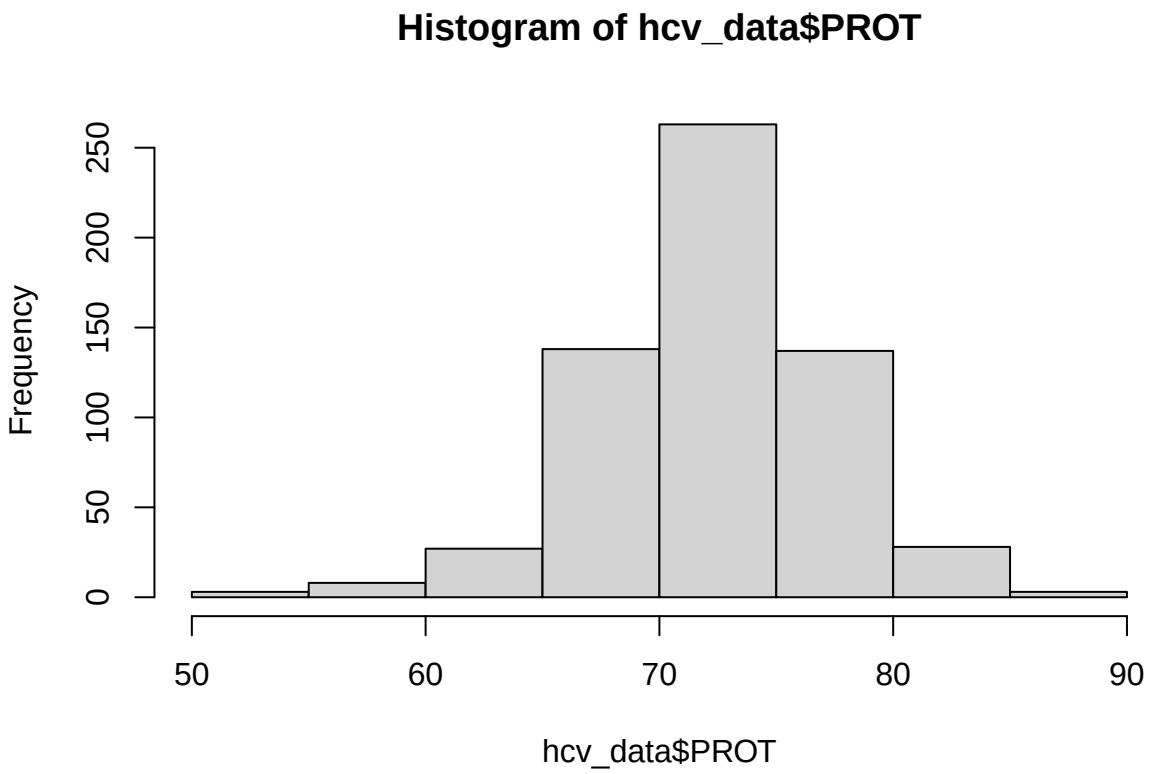

```
hist(hcv_data$CREA)
```




```
hist(hcv_data$GGT)
```




```
hist(hcv_data$PROT)
```




Age is fairly normal with a slight right skew. ALB seems normally distributed. Most of ALP's values fall within a small range, however there is a small proportion of outliers. ALT, AST, BIL has a significant right skew. CHE has a normal distribution. CHOL has a fairly normal distribution. CREA has a significant right skew. GGT is right skewed. PROT is slightly left skewed.

### Identifying Outliers

```
# Function to calculate z score
calc_z_score <- function(x) {
  mu <- mean(x, na.rm = TRUE)
  sd <- sd(x, na.rm = TRUE)
  (x - mu) / sd
}

# Calculate z_scores and find outliers for each column

# Age
z_scores <- calc_z_score(hcv_data$Age)
range(z_scores, na.rm=TRUE)
```

```
## [1] -2.831177  2.973057
```

```r
age_out <- which(abs(z_scores) > 2.5, arr.ind = TRUE)
length(age_out)
```

```
## [1] 6
```

```r
# ALB
z_scores <- calc_z_score(hcv_data$ALB)
range(z_scores, na.rm=TRUE)
```

```
## [1] -4.035496  7.468714
```

```r
median(z_scores, na.rm=TRUE)
```

```
## [1] 0.03351741
```

```r
alb_out <- which(abs(z_scores) > 4, arr.ind = TRUE)
length(alb_out)
```

```
## [1] 2
```

```r
# ALP
z_scores <- calc_z_score(hcv_data$ALP)
range(z_scores, na.rm=TRUE)
```

```
## [1] -2.236293 13.799681
```

```r
alp_out <- which(abs(z_scores) > 3, arr.ind = TRUE)
length(alp_out)
```

```
## [1] 3
```

```r
# ALT
z_scores <- calc_z_score(hcv_data$ALT)
range(z_scores, na.rm=TRUE)
```

```
## [1] -1.257862 10.853763
```

```r
alt_out <- which(abs(z_scores) > 4, arr.ind = TRUE)
length(alt_out)
```

```
## [1] 8
```

```r
# AST
z_scores <- calc_z_score(hcv_data$AST)
range(z_scores, na.rm=TRUE)
```

```
## [1] -0.6857061  8.8782622
```

```r
ast_out <- which(abs(z_scores) > 5, arr.ind = TRUE)
length(ast_out)
```

```
## [1] 4
```

```r
# BIL
z_scores <- calc_z_score(hcv_data$BIL)
range(z_scores, na.rm=TRUE)
```

```
## [1] -0.4893141 12.2670277
```

```r
bil_out <- which(abs(z_scores) > 2.5, arr.ind = TRUE)
length(bil_out)
```

```
## [1] 8
```

```r
# CHE
z_scores <- calc_z_score(hcv_data$CHE)
range(z_scores, na.rm=TRUE)
```

```
## [1] -3.128983  3.783950
```

```r
che_out <- which(abs(z_scores) > 3, arr.ind = TRUE)
length(che_out)
```

```
## [1] 10
```

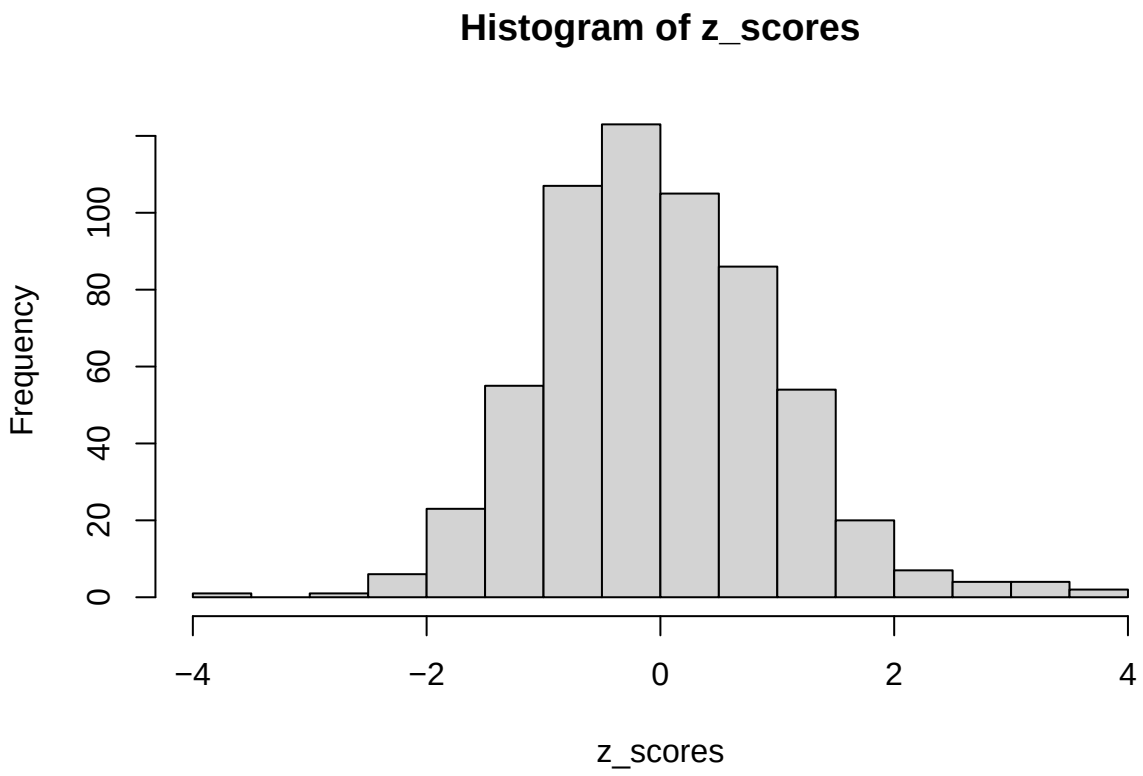
```r
# CHOL
z_scores <- calc_z_score(hcv_data$CHOL)
range(z_scores, na.rm=TRUE)
```

```
## [1] -3.527649  3.833477
```

```r
chol_out <- which(abs(z_scores) > 3.5, arr.ind = TRUE)
length(chol_out)
```

```
## [1] 3
```

```r
hist(z_scores)
```



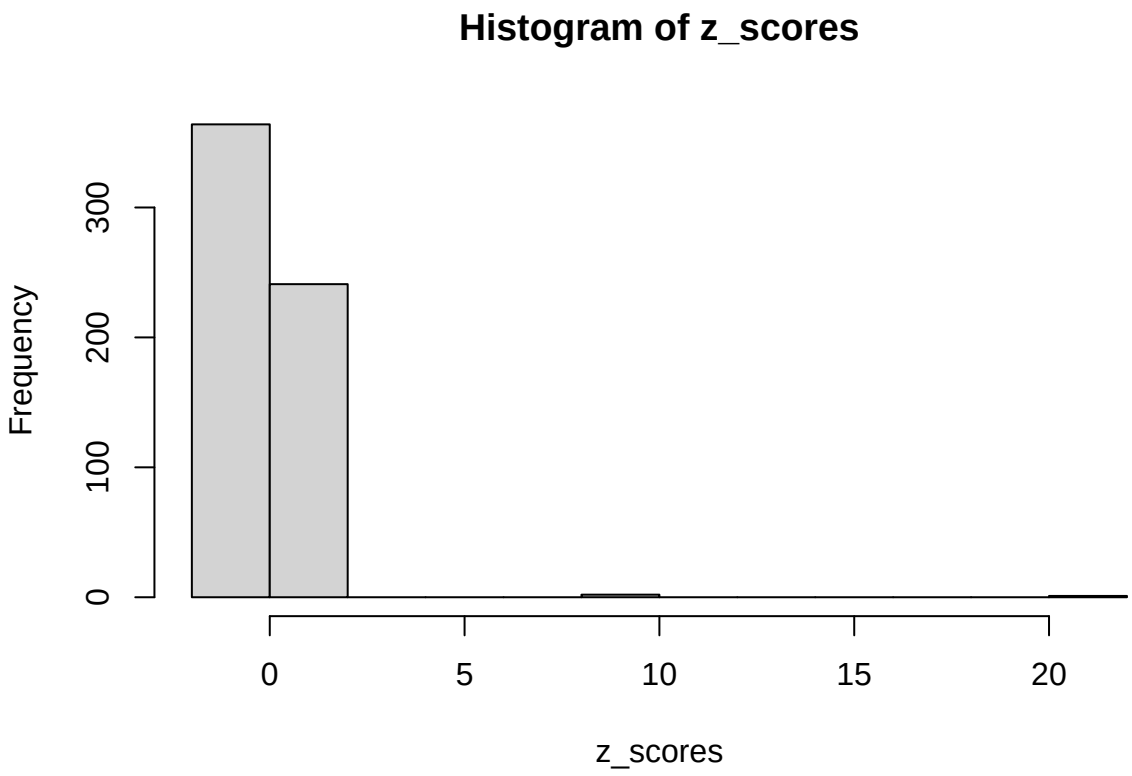

```
# CREA
z_scores <- calc_z_score(hcv_data$CREA)
range(z_scores, na.rm=TRUE)
```

```
## [1] -1.478524 20.063833
```

```
crea_out <- which(abs(z_scores) > 2.5, arr.ind = TRUE)
length(crea_out)
```

```
## [1] 3
```

```
hist(z_scores)
```



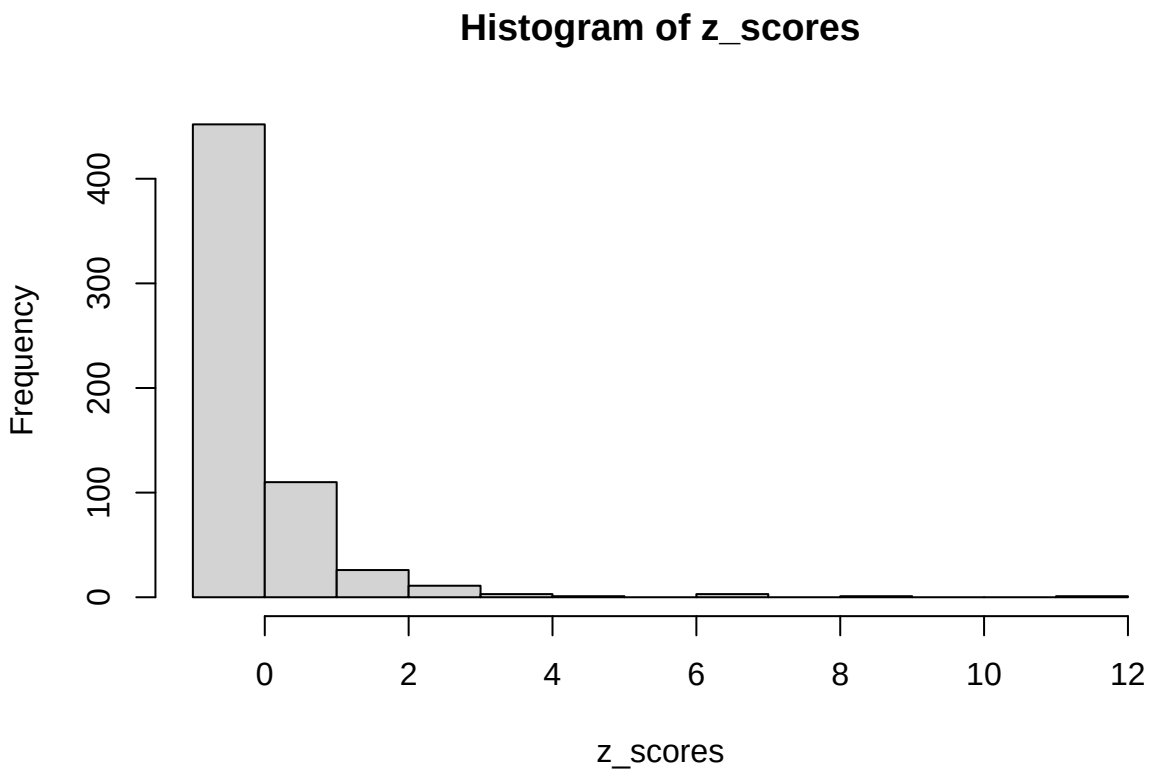

```
# GGT
z_scores <- calc_z_score(hcv_data$GGT)
range(z_scores, na.rm=TRUE)
```

```
## [1] -0.6495057 11.7924565
```

```
ggt_out <- which(abs(z_scores) > 4, arr.ind = TRUE)
length(ggt_out)
```

```
## [1] 6
```

```
hist(z_scores)
```



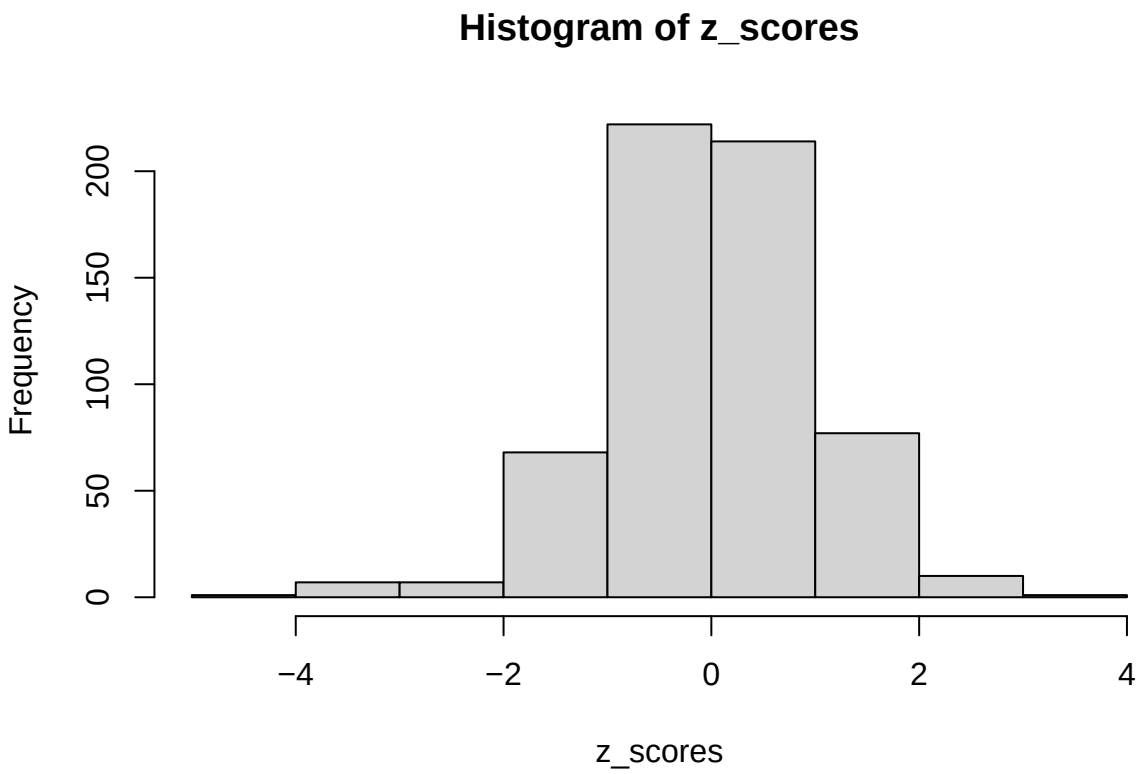

```
# PROT
z_scores <- calc_z_score(hcv_data$PROT)
range(z_scores, na.rm=TRUE)
```

```
## [1] -4.309028  3.598110
```

```
prot_out <- which(abs(z_scores) > 3.5, arr.ind = TRUE)
length(prot_out)
```

```
## [1] 4
```

```
hist(z_scores)
```




We looked at the z-scores, range of z-scores, and distribution of both the data and z_scores to identify outliers. We also used domain knowledge to assess whether we should remove certain outliers and which z-score threshold to set. For age, there are only 7 outliers with a z score threshold of 2.5. We chose not to remove these data points as they represent those who have an age a few years above 70 and this is important information as those who are older are more likely to be in later stages of HCV infection. There are only 2 outliers for ALB using a z score threshold of 4. Those represent one high value and one low value, which isn't necessarily correlated with HCV infection and could just indicate dehydration or some other issue. Studies have shown that AST, ALT, and ALP levels are significantly correlated with viral load of HCV. There are 3 outliers using a threshold of 3 for ALP. We won't remove these because these are for the patients with HCV and are important information. Using a threshold of 3 for ALT, we have 8 outliers with most of them for infected patients. To not lose valuable data, we won't remove these outliers. Same applies to AST. Using a threshold of 2.5 for BIL, we see there are 8 outliers. Since these represent high billirubin levels for cirrhosis patients, we will not remove these. High billirubin is usually linked to jaundice which can indicate severe liver disease/cirrhosis. There are not many cirrhosis data points so we don't want to lose data. Using a z score threshold of 3 for CHE, there are 9 outliers. 4 of these represent values in the normal range for CHE, and 5 of those represent abnormal values and are found in patients with HCV infection. Decreased CHE can be due to liver damage. The mean z score for CHE is 2.3. We will keep these outliers as they are important information, aren't too far from the mean z_score, and come from a normal distribution. For CHOL, its unclear what the values represent as there are different types of cholesterol. We won't remove outliers for CHOL given the lack of clarity around its relevance. The range of z scores for CREA has extreme variance. The outliers seem to be for patients who have cirrhosis, which makes sense. At later stages of liver disease, there may also be the co-morbidity of failure of kidneys to remove creatinine, causing increased levels. There are outliers for GGT as well as we get to the hepatitis and cirrhosis patients. We won't remove these few outliers either because literature suggests that the variance in GGT values can help differentiate between fibrosis and more extensive scarring. There are only 4 protein outliers using a z score threshold of 3.5. Not worth removing, especially since the data is normally distributed and we have a small dataset to begin with.

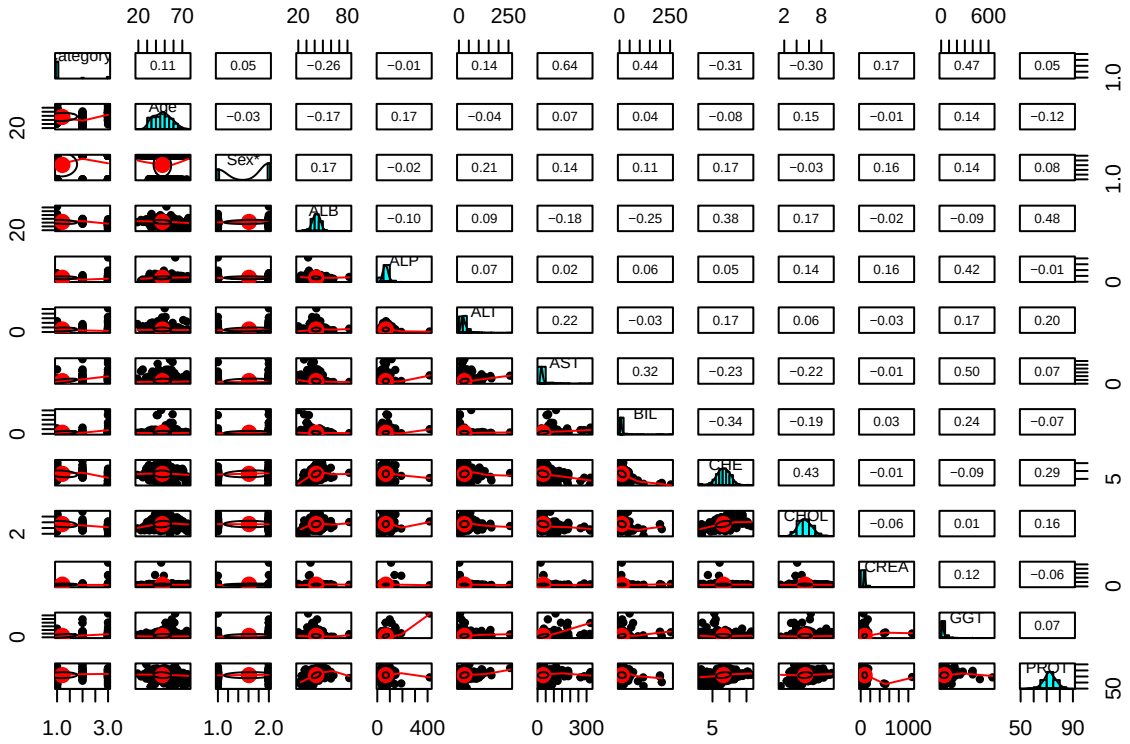
In summary, there aren't enough outliers from each variable to warrant removing, some of the outliers are relevant to particular classes, and our data has a class imbalance issue. Therefore we will keep the outliers in the data.

## Correlation and Collinearity

```
# Get numerical columns
numerical_cols <- hcv_data[, sapply(hcv_data, is.numeric)]
# Correlation matrix
cor(numerical_cols, use = "pairwise.complete.obs")
```

```
##              Age         ALB          ALP         ALT         AST         BIL
## Age   1.00000000 -0.17013809  0.165691235 -0.04210156  0.07311133  0.03575808
## ALB  -0.17013809  1.00000000 -0.099209390  0.09261131 -0.18077421 -0.24896048
## ALP   0.16569124 -0.09920939  1.000000000  0.07110051  0.02450357  0.06434890
## ALT  -0.04210156  0.09261131  0.071100515  1.00000000  0.21516086 -0.03203252
## AST   0.07311133 -0.18077421  0.024503570  0.21516086  1.00000000  0.32262769
## BIL   0.03575808 -0.24896048  0.064348899 -0.03203252  0.32262769  1.00000000
## CHE  -0.08078676  0.38100624  0.048188073  0.17336094 -0.23470505 -0.34254608
## CHOL  0.14667028  0.16666883  0.135029955  0.06269707 -0.22018647 -0.18558380
## CREA -0.01207865 -0.02120562  0.162796411 -0.03486221 -0.01316502  0.03101676
## GGT   0.14108112 -0.08775050  0.421882473  0.16991911  0.49652909  0.23878057
## PROT -0.12200214  0.48060159 -0.006811518  0.20428930  0.07127657 -0.06581359
##              CHE        CHOL        CREA         GGT         PROT
## Age  -0.08078676  0.14667028 -0.01207865  0.14108112 -0.122002142
## ALB   0.38100624  0.16666883 -0.02120562 -0.08775050  0.480601586
## ALP   0.04818807  0.13502995  0.16279641  0.42188247 -0.006811518
## ALT   0.17336094  0.06269707 -0.03486221  0.16991911  0.204289300
## AST  -0.23470505 -0.22018647 -0.01316502  0.49652909  0.071276572
## BIL  -0.34254608 -0.18558380  0.03101676  0.23878057 -0.065813594
## CHE   1.00000000  0.42564835 -0.01208672 -0.09164602  0.285723009
## CHOL  0.42564835  1.00000000 -0.05595637  0.01339899  0.161667717
## CREA -0.01208672 -0.05595637  1.00000000  0.12313322 -0.061982202
## GGT  -0.09164602  0.01339899  0.12313322  1.00000000  0.073540445
## PROT  0.28572301  0.16166772 -0.06198220  0.07354045  1.000000000
```

```
# Distribution and collinearity/correlation
pairs.panels(hcv_data)
```




We used a pairwise correlation test on the numerical columns to check for collinearity. There seems to be a weak to moderate correlation between GGT and AST and between ALB and PROT. PROT has a weak correlation with the target variable as well as the other variables aside from ALB. Based on domain knowledge, it seems to be far less important to liver function than the other lab values. We will remove PROT from the data. Age and sex also have weak correlation with the target variable, so we will remove those as well. We will not be using PCA as there is a non-linear relationship between features.

## Cleaning and Shaping Data

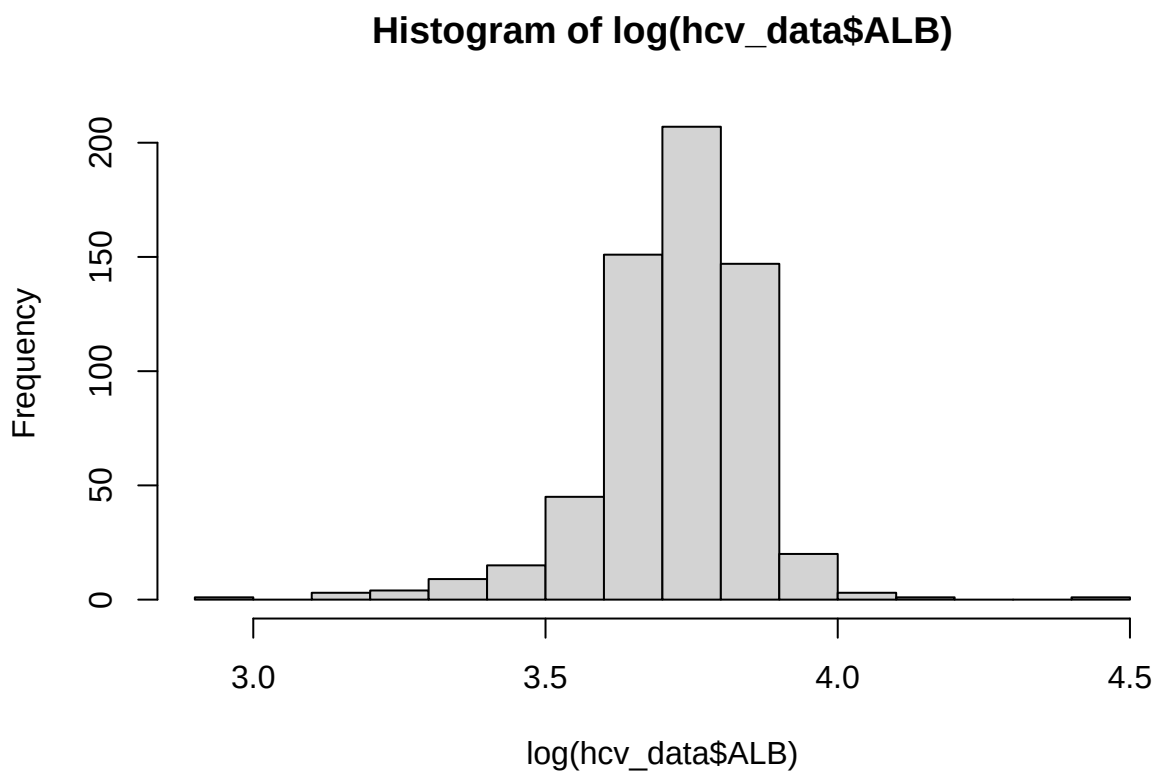
```
# Remove PROT
hcv_data <- subset(hcv_data, select = -PROT)
hcv_data <- subset(hcv_data, select = -Age)
hcv_data <- subset(hcv_data, select = -Sex)

# Find the number of missing values in each column
missing_counts <- colSums(is.na(hcv_data))
missing_counts
```

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

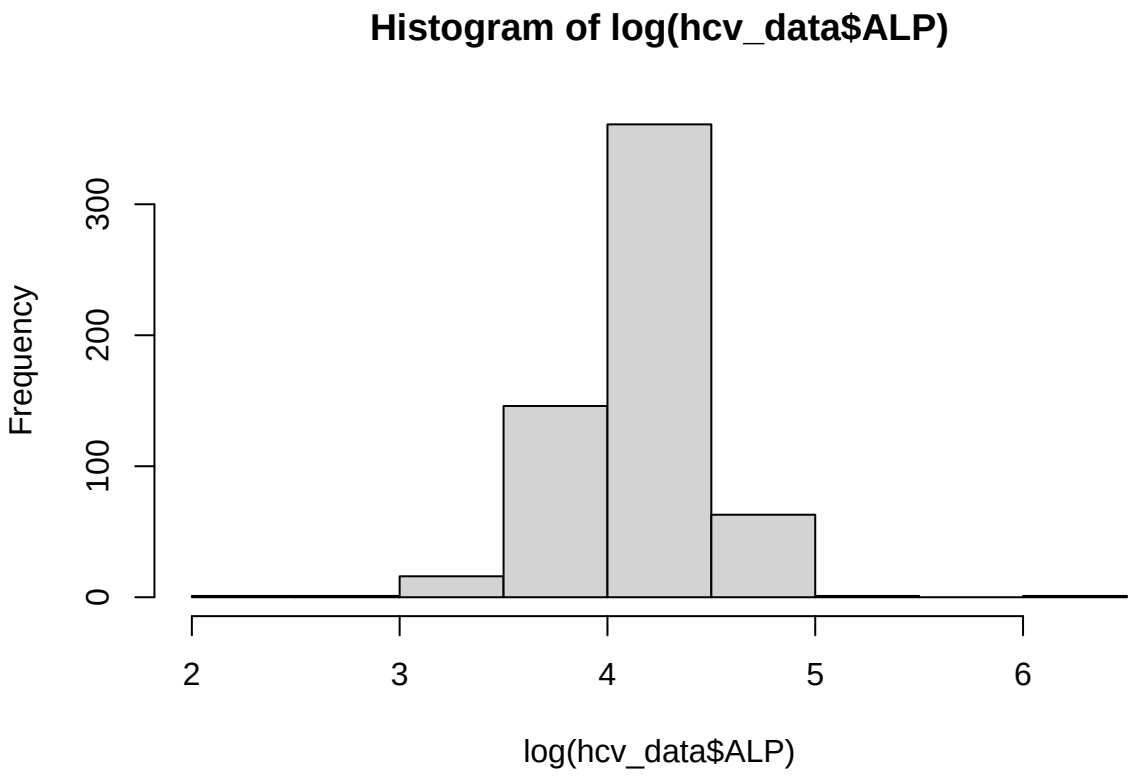
We have some missing data for ALB, ALP, ALT, and CHOL. I don't want to lose information, and given

that the data is very skewed, we don't want to take the mean/median for imputation, so we will make the features more normal through transformation before imputation.

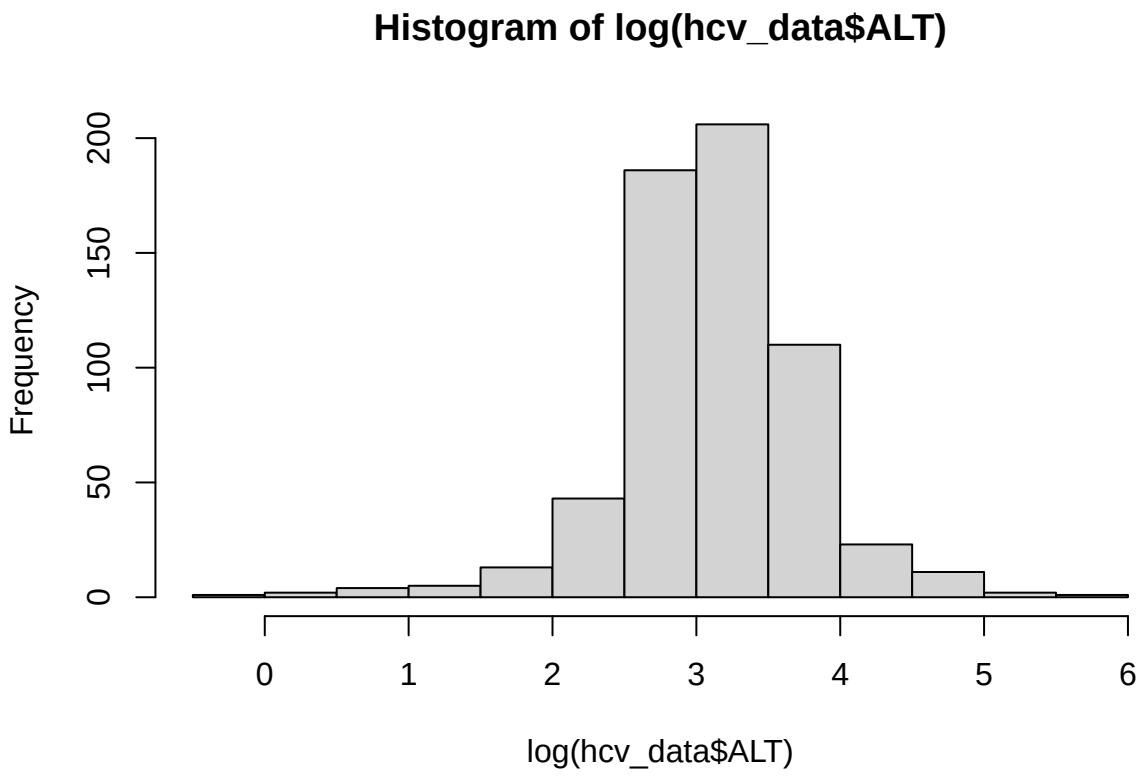
```
# Let's see if we can transform features to look more normal before imputation
# ALB
hist(log(hcv_data$ALB))
```



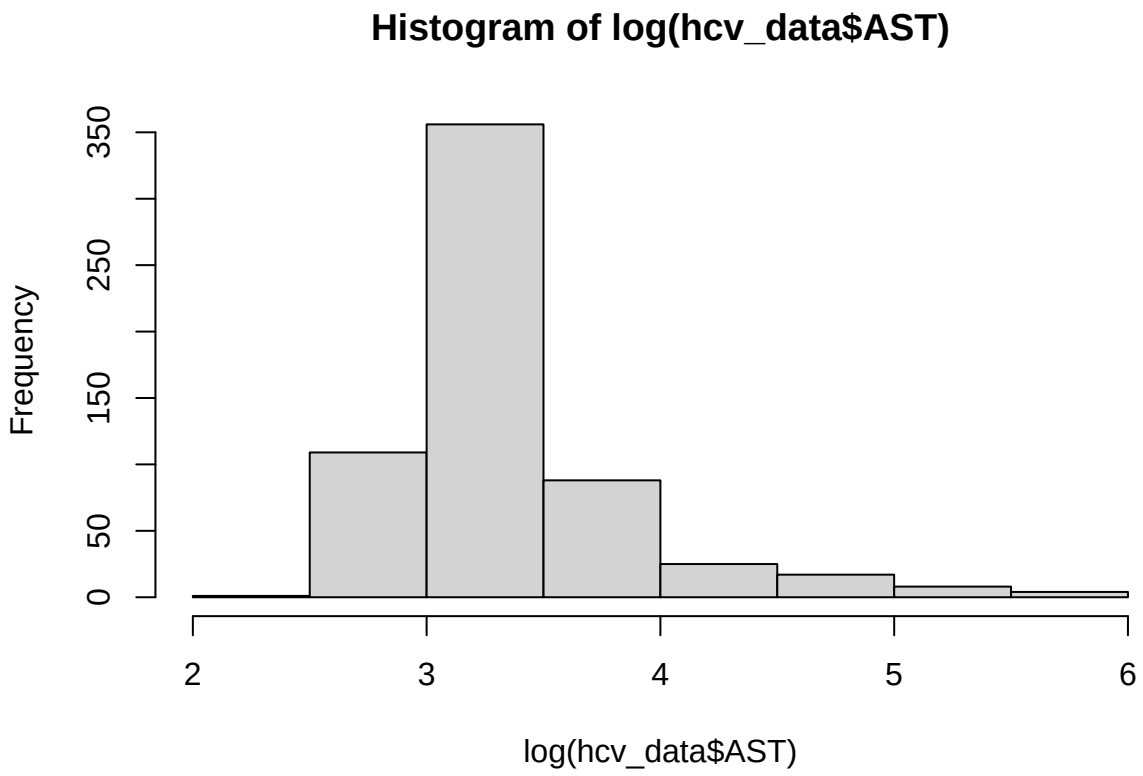

```
hcv_data$ALB <- log(hcv_data$ALB)

# ALP
hist(log(hcv_data$ALP))
```



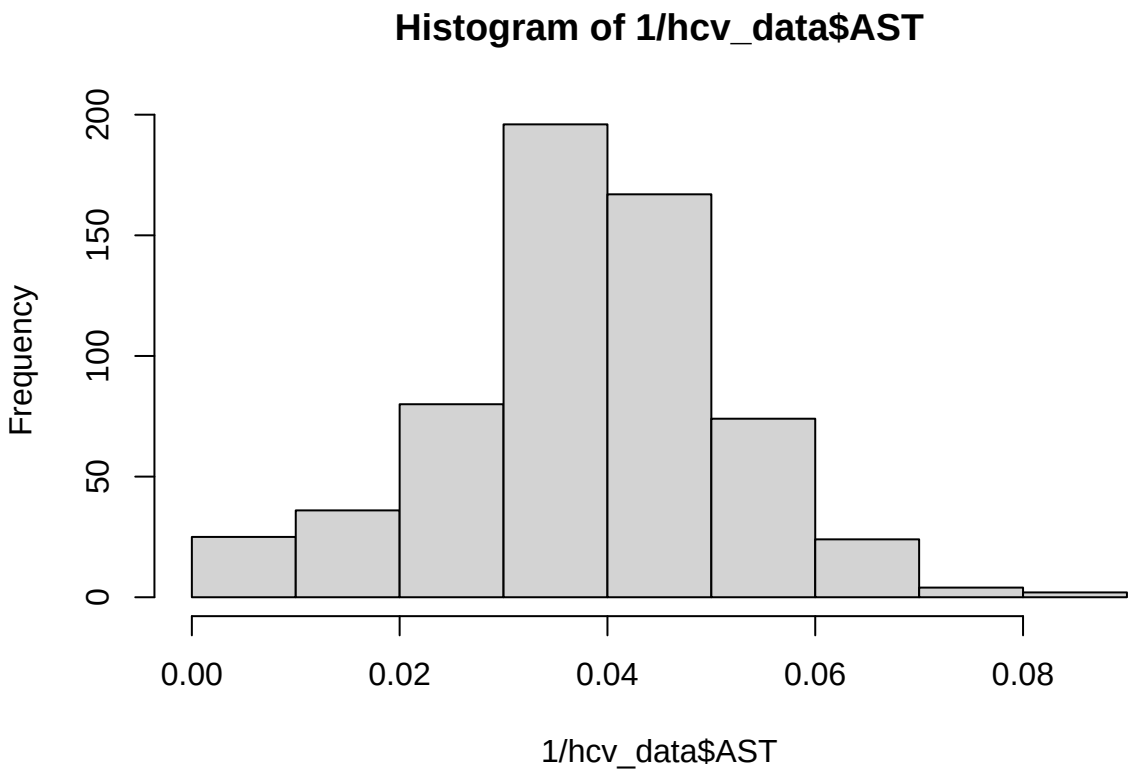

```r
hcv_data$ALP <- log(hcv_data$ALP)

# ALT
hist(log(hcv_data$ALT))
```



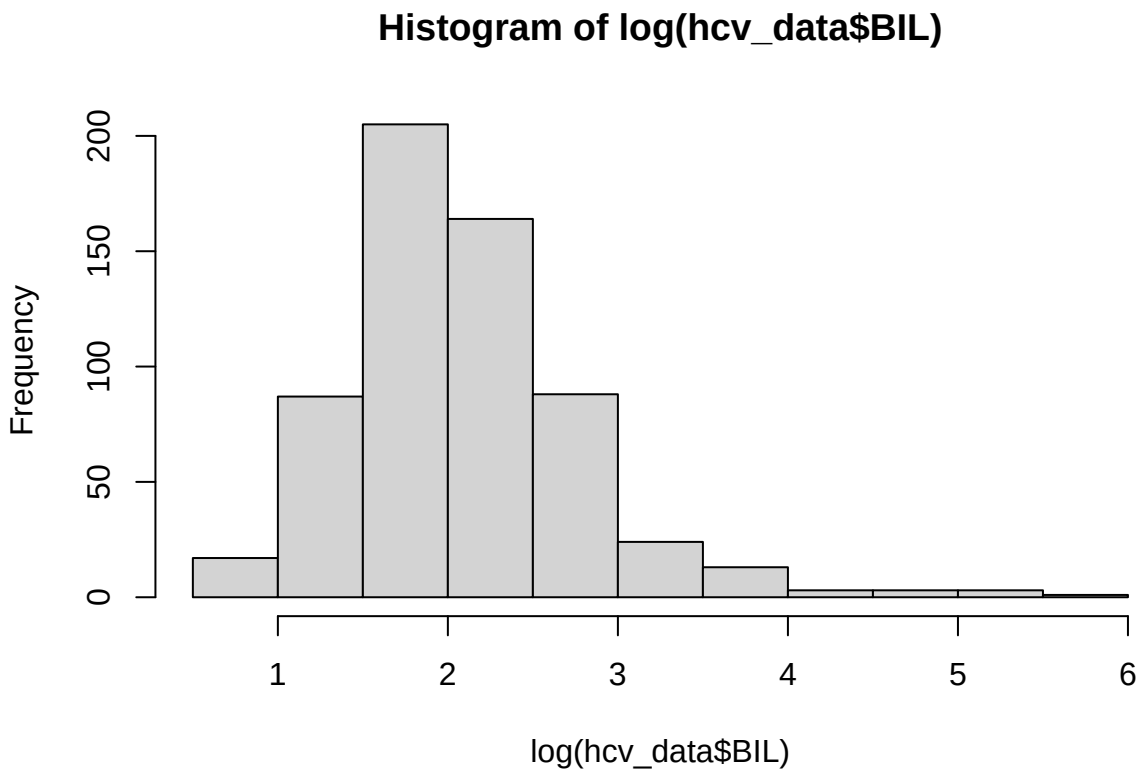

```
hcv_data$ALT <- log(hcv_data$ALT)

# AST
hist(log(hcv_data$AST))
```



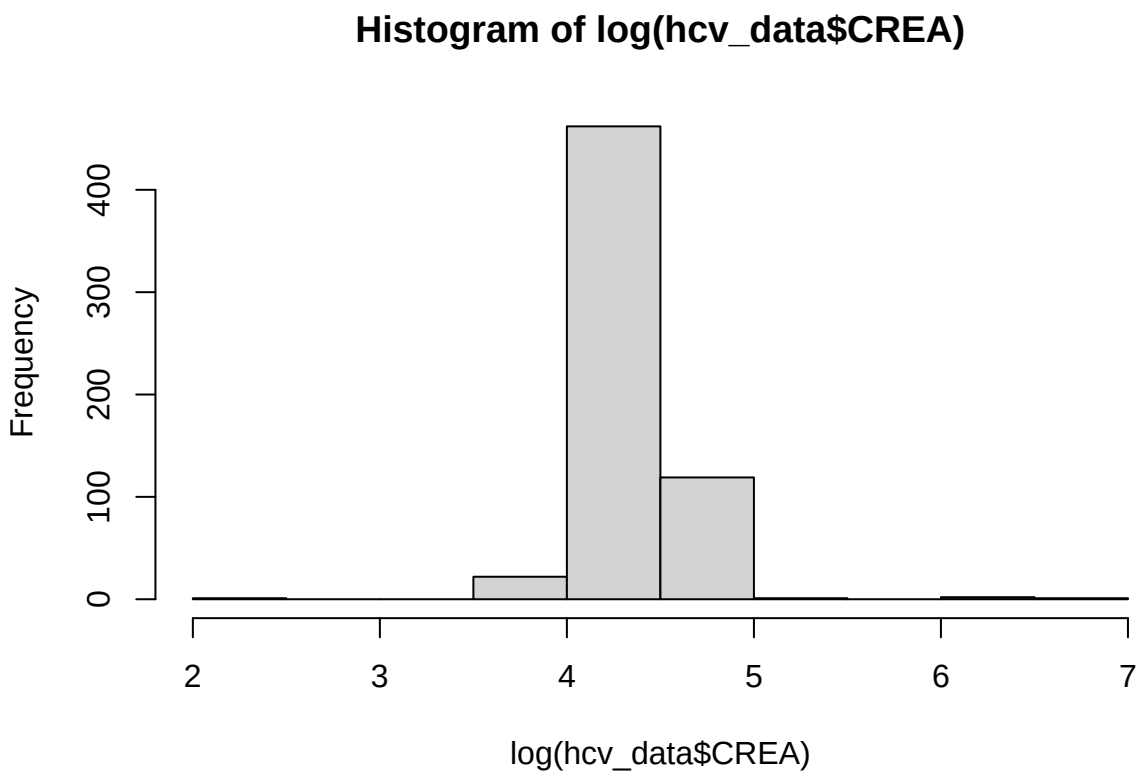

```r
hist(1/hcv_data$AST)
```



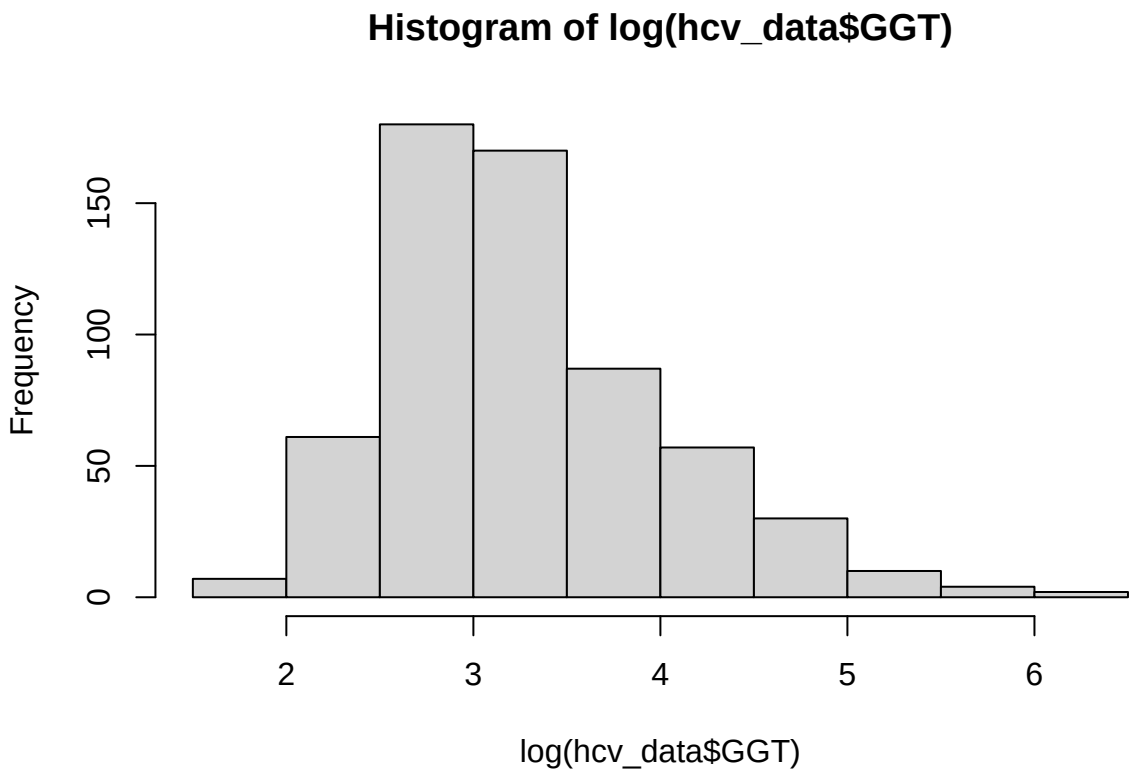

```
hcv_data$AST <- 1 / hcv_data$AST

# BIL
hist(log(hcv_data$BIL))
```




```
hcv_data$BIL <- log(hcv_data$BIL)

# CREA
hist(log(hcv_data$CREA))
```

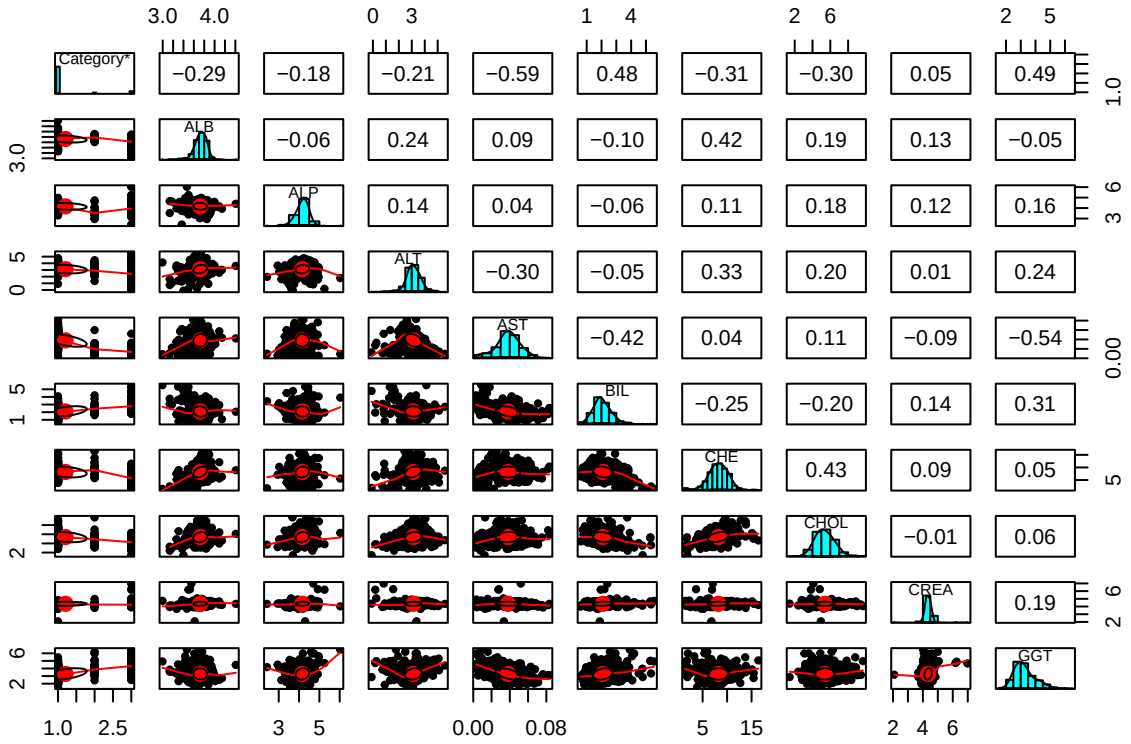



```
hcv_data$CREA <- log(hcv_data$CREA)

# GGT
hist(log(hcv_data$GGT))
```




```
hcv_data$GGT <- log(hcv_data$GGT)
```

```
pairs.panels(hcv_data)
```




We transformed the features that were not normally distributed using log transformation or inverse transformation, depending on which transformation produced a more normal distribution.

```
# Get numeric columns
numerical_cols <- sapply(hcv_data, is.numeric)

# Impute NAs with median
hcv_data[numerical_cols] <- apply(hcv_data[numerical_cols], 2, function(x) {
  x[is.na(x)] <- median(x, na.rm=TRUE)
  x
})
```

Given that the feature distributions were originally skewed and we did not want to lose data given the small sample size and class imbalance, we imputed missing values with the median.

```
# Factoring
hcv_data$Category <- as.factor(hcv_data$Category)
```

## Model Construction and Evaluation

### Downsampling and oversampling

```
# Split the data into subsets
majority_data <- hcv_data[hcv_data$Category == '0=Blood Donor', ]
```

```r
hepatitis_data <- hcv_data[hcv_data$Category == '1=Hepatitis', ]
progressed_hcv_data <- hcv_data[hcv_data$Category == 'Progressed HCV', ]

# Convert 'Category' to factor in minority datasets
hepatitis_data$Category <- as.factor(hepatitis_data$Category)
progressed_hcv_data$Category <- as.factor(progressed_hcv_data$Category)

# Downsample the majority class
set.seed(123)
num_to_sample <- 5 * (nrow(hepatitis_data) + nrow(progressed_hcv_data))
num_to_sample <- min(num_to_sample, nrow(majority_data))
downsampled_majority_data <- majority_data[sample(nrow(majority_data), num_to_sample), ]

set.seed(123)

# Manually oversample '1=Hepatitis'
oversample_size_hepatitis <- 5 * nrow(hepatitis_data)  # Adjust the multiplier as needed
oversampled_hepatitis <- hepatitis_data[sample(1:nrow(hepatitis_data), size = oversample_size_hepatitis

# Manually oversample 'Progressed HCV'
#oversample_size_progressed_hcv <- 5 * nrow(progressed_hcv_data)  # Adjust the multiplier as needed
#oversampled_progressed_hcv <- progressed_hcv_data[sample(1:nrow(progressed_hcv_data), size = oversampl

# Combine the downsampled majority class with the oversampled minority class
balanced_data <- rbind(downsampled_majority_data, oversampled_hepatitis, progressed_hcv_data)
```

### Create training and validation sets

```r
# Fixed seed for reproducibility
set.seed(123)

# Randomize data
balanced_data <- balanced_data[sample(nrow(balanced_data)), ]

# Create a stratified random train-validation split
train_index <- createDataPartition(y = balanced_data$Category,
                                   p = 0.8,
                                   list = FALSE,
                                   times = 1 )


# Training data
train_data <- balanced_data[train_index, ]

# Validation data
validation_data <- balanced_data[-train_index, ]
```

The data has a significant class imbalance. About 88% of the data falls under "Blood Donor" with only ~ 4% falling under "Hepatitis", and ~8% falling under "Progressed HCV. We randomize the dataset and use stratified sampling to split the data 80/20 while preserving class distribution in both the training and validation sets. We also downsample the majority class using a 5:1 ratio to reduce bias in overfitting the

model to the majority class, and randomly oversample the hepatitis class due to the very small sample size. Because the progressed class has more variation, I chose not to oversample this class. SMOTE would be a preferable alternative to manual oversampling giving it's advanced statistical properties, but the hepatitis sample size was too small to use the popular SMOTE package.

### XGBoost

```
# Train XGBoost model
# Set target variable
target_column <- "Category"

# Encoding for xgboost - convert to factor
train_data_xg <- train_data
validation_data_xg <- validation_data
#train_data_xg$Category <- as.numeric(as.factor(train_data_xg$Category)) - 1
#validation_data_xg$Category <- as.numeric(as.factor(validation_data_xg$Category)) -1
train_labels_numeric <- as.numeric(as.factor(train_data_xg$Category)) - 1
validation_labels_factor <- factor(validation_data_xg$Category, levels = c("0=Blood Donor", "1=Hepatitis

# Remove target variable from data to get features
features <- names(train_data_xg)[names(train_data_xg) != target_column]

# Convert data to matrix format
train_matrix <- as.matrix(train_data_xg[, features])
validation_matrix <- as.matrix(validation_data_xg[, features])

# xgboost parameters for multi-class classification
params <- list(
  objective = "multi:softprob",
  num_class = length(unique(train_data_xg$Category)),
  max_depth = 3
)

# Train the model
xgb_model <- xgboost(data = train_matrix,
                     label = train_labels_numeric,
                     params = params,
                     nrounds = 50)
```

```
## [1]  train-mlogloss:0.766602
## [2]  train-mlogloss:0.566046
## [3]  train-mlogloss:0.432994
## [4]  train-mlogloss:0.342085
## [5]  train-mlogloss:0.274754
## [6]  train-mlogloss:0.222778
## [7]  train-mlogloss:0.186333
## [8]  train-mlogloss:0.157970
## [9]  train-mlogloss:0.135680
## [10] train-mlogloss:0.118718
## [11] train-mlogloss:0.103516
## [12] train-mlogloss:0.090994
## [13] train-mlogloss:0.080155
```

```
## [14] train-mlogloss:0.069959
## [15] train-mlogloss:0.062073
## [16] train-mlogloss:0.055600
## [17] train-mlogloss:0.049365
## [18] train-mlogloss:0.044210
## [19] train-mlogloss:0.039764
## [20] train-mlogloss:0.036271
## [21] train-mlogloss:0.032699
## [22] train-mlogloss:0.030653
## [23] train-mlogloss:0.028673
## [24] train-mlogloss:0.026529
## [25] train-mlogloss:0.024632
## [26] train-mlogloss:0.022783
## [27] train-mlogloss:0.021303
## [28] train-mlogloss:0.020275
## [29] train-mlogloss:0.019175
## [30] train-mlogloss:0.018199
## [31] train-mlogloss:0.017208
## [32] train-mlogloss:0.016486
## [33] train-mlogloss:0.015627
## [34] train-mlogloss:0.014822
## [35] train-mlogloss:0.014160
## [36] train-mlogloss:0.013610
## [37] train-mlogloss:0.013139
## [38] train-mlogloss:0.012596
## [39] train-mlogloss:0.012057
## [40] train-mlogloss:0.011626
## [41] train-mlogloss:0.011219
## [42] train-mlogloss:0.010876
## [43] train-mlogloss:0.010534
## [44] train-mlogloss:0.010164
## [45] train-mlogloss:0.009888
## [46] train-mlogloss:0.009644
## [47] train-mlogloss:0.009418
## [48] train-mlogloss:0.009165
## [49] train-mlogloss:0.008957
## [50] train-mlogloss:0.008741
```

```r
# Make predictions on the validation data
xgb_predictions <- predict(xgb_model, newdata = validation_matrix)

# Convert predictions to class labels
max_prob_index <- matrix(xgb_predictions, ncol = length(unique(train_data_xg$Category)), byrow = TRUE)
xgb_class_labels <- apply(max_prob_index, 1, which.max) - 1
xgb_class_labels <- factor(xgb_class_labels, levels = 0:2, labels = c("0=Blood Donor", "1=Hepatitis", "

# Confustion Matrix
confusion_matrix <- confusionMatrix(xgb_class_labels, validation_labels_factor)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       0=Blood Donor 1=Hepatitis Progressed HCV
```

```
##   0=Blood Donor             75          0             1
##   1=Hepatitis                0         24             2
##   Progressed HCV             0          0             7
##
## Overall Statistics
##
##                Accuracy : 0.9725
##                  95% CI : (0.9217, 0.9943)
##     No Information Rate : 0.6881
##     P-Value [Acc > NIR] : 4.175e-14
##
##                   Kappa : 0.9404
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0=Blood Donor Class: 1=Hepatitis
## Sensitivity                        1.0000             1.0000
## Specificity                        0.9706             0.9765
## Pos Pred Value                     0.9868             0.9231
## Neg Pred Value                     1.0000             1.0000
## Prevalence                         0.6881             0.2202
## Detection Rate                     0.6881             0.2202
## Detection Prevalence               0.6972             0.2385
## Balanced Accuracy                  0.9853             0.9882
##                      Class: Progressed HCV
## Sensitivity                        0.70000
## Specificity                        1.00000
## Pos Pred Value                     1.00000
## Neg Pred Value                     0.97059
## Prevalence                         0.09174
## Detection Rate                     0.06422
## Detection Prevalence               0.06422
## Balanced Accuracy                  0.85000
```

```r
# Save metrics
# Extract the table from the confusion matrix
cm_table <- confusion_matrix$table

# Initialize vectors to store the metrics for each class
precision <- numeric(length = ncol(cm_table))
recall <- numeric(length = ncol(cm_table))
f1_score <- numeric(length = ncol(cm_table))

# Calculate metrics for each class
for (i in 1:ncol(cm_table)) {
  tp <- cm_table[i, i]
  fp <- sum(cm_table[, i]) - tp
  fn <- sum(cm_table[i, ]) - tp
  tn <- sum(cm_table) - tp - fp - fn

  precision[i] <- tp / (tp + fp)
  recall[i] <- tp / (tp + fn)
```

```
  f1_score[i] <- 2 * (precision[i] * recall[i]) / (precision[i] + recall[i])
}

# Output the results
xgb_metrics <- data.frame(
  Class = colnames(cm_table),
  Precision = precision,
  Recall = recall,
  F1_Score = f1_score
)

xgb_metrics
```

```
##            Class Precision    Recall  F1_Score
## 1  0=Blood Donor       1.0 0.9868421 0.9933775
## 2    1=Hepatitis       1.0 0.9230769 0.9600000
## 3 Progressed HCV       0.7 1.0000000 0.8235294
```

XGBoost falls under the category of gradient boosting. It can handle both regression and classification problems and is known for providing high predictive accuracy and handling complex relationships in data. It uses gradient boosting, essentially it is an ensemble learning technique that combines multiple weak learners to create a strong predictive model. It primarily uses decision trees as its base learners. It utilizes boosting by sequentially adding trees to the model. Each tree focuses on correcting the errors made by the previous model. We used a smaller number for the rounds and depth due to the small dataset and class imbalance. We used the holdout method to test this model. We use kappa, precision, recall, and F1 score to evaluate the model as these metrics are useful for imbalanced datasets.

A kappa value of 0.94 indicates a very good agreement between the model's predictions and the actual classes. The positive predictive value (PPV) and negative predictive value (NVP) are very high for for all classes, indicating the model's effectiveness in class prediction. There is excellent performance in classifying Blood Donor and Hepatitis cases, with perfect precision (100%) and very high recall. However, the model shows a lower precision for the Progressed HCV class, at 70%, although it achieves perfect recall. This suggests the model is particularly effective in identifying true Blood Donor and Hepatitis cases but prone to some false positives in detecting Progressed HCV. However, it is better to falsely classify a patient as progressed HCV and run additional labs than to miss a diagnosis.The F1 score which is the harmonic mean of the precision and recall is high for Blood Donor and Hepatitis, and moderate for Progressed HCV due to the lower precision. Overall, this is a strong model whose performance could potentially be improved by focusing on the precision for the Progressed HCV class, possibly through resampling or collecting more data.

**RandomForest**

```
# Train a bagged model using randomForest
set.seed(123)
bagged_model <- randomForest(Category ~ ., data = train_data)
predictions <- predict(bagged_model, newdata = validation_data)

# Confusion matrix
confusion_matrix <- confusionMatrix(predictions, validation_data$Category)
confusion_matrix
```

```
## Confusion Matrix and Statistics
```

```
##
##                 Reference
## Prediction       0=Blood Donor 1=Hepatitis Progressed HCV
##   0=Blood Donor             73           0               1
##   1=Hepatitis                0          24               2
##   Progressed HCV             2           0               7
##
## Overall Statistics
##
##                Accuracy : 0.9541
##                  95% CI : (0.8962, 0.9849)
##     No Information Rate : 0.6881
##     P-Value [Acc > NIR] : 4.999e-12
##
##                   Kappa : 0.903
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0=Blood Donor Class: 1=Hepatitis
## Sensitivity                        0.9733             1.0000
## Specificity                        0.9706             0.9765
## Pos Pred Value                     0.9865             0.9231
## Neg Pred Value                     0.9429             1.0000
## Prevalence                         0.6881             0.2202
## Detection Rate                     0.6697             0.2202
## Detection Prevalence               0.6789             0.2385
## Balanced Accuracy                  0.9720             0.9882
##                      Class: Progressed HCV
## Sensitivity                        0.70000
## Specificity                        0.97980
## Pos Pred Value                     0.77778
## Neg Pred Value                     0.97000
## Prevalence                         0.09174
## Detection Rate                     0.06422
## Detection Prevalence               0.08257
## Balanced Accuracy                  0.83990
```

```r
# Save metrics
# Extract the table from the confusion matrix
cm_table <- confusion_matrix$table

# Initialize vectors to store the metrics for each class
precision <- numeric(length = ncol(cm_table))
recall <- numeric(length = ncol(cm_table))
f1_score <- numeric(length = ncol(cm_table))

# Calculate metrics for each class
for (i in 1:ncol(cm_table)) {
  tp <- cm_table[i, i]
  fp <- sum(cm_table[, i]) - tp
  fn <- sum(cm_table[i, ]) - tp
  tn <- sum(cm_table) - tp - fp - fn
```

```
  precision[i] <- tp / (tp + fp)
  recall[i] <- tp / (tp + fn)
  f1_score[i] <- 2 * (precision[i] * recall[i]) / (precision[i] + recall[i])
}

# Output the results
rf_metrics <- data.frame(
  Class = colnames(cm_table),
  Precision = precision,
  Recall = recall,
  F1_Score = f1_score
)

rf_metrics
```

```
##            Class Precision    Recall  F1_Score
## 1  0=Blood Donor 0.9733333 0.9864865 0.9798658
## 2    1=Hepatitis 1.0000000 0.9230769 0.9600000
## 3 Progressed HCV 0.7000000 0.7777778 0.7368421
```

The RandomForest model is versatile and robust. It can handle class imbalance and non-linearity effectively. It's ensemble nature mitigates overfitting, and works fairly well without extensive parameter tuning. We use the holdout method for testing and we use kappa, precision, recall, and F1 score to evaluate the model as these metrics are useful for imbalanced datasets.

The RandomForest model demonstrates strong performance in identifying Blood Donor and Hepatitis classes, as evidenced by the F1 score and kappa (0.93). In the Hepatitis class, there is very strong precision coupled with a high recall, suggesting every prediction made for Hepatitis is correct. The F1 score of 96% indicates efficient identification of Hepatitis cases with minimal false negatives. The precision, recall and F1 score are moderate for the Progressed HCV class, and have a few more occurrences of false positives and negatives, with 2 out of 10 predictions misclassified as HCV and 1 misclassified as Blood Donor. We may be able to improve this model by assigning class weights and assigning a larger weight to the minority class.

### SVM

```
# Train SVM model
svm_model <- svm(Category ~ ., data = train_data, kernel = "radial")

# Make predictions on validation data
svm_predictions <- predict(svm_model, newdata = validation_data)

# Confusion matrix
confusion_matrix <- confusionMatrix(svm_predictions, validation_data$Category)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       0=Blood Donor 1=Hepatitis Progressed HCV
##   0=Blood Donor             74           2              1
##   1=Hepatitis                0          21              2
```

```
##   Progressed HCV             1           1              7
##
## Overall Statistics
##
##                Accuracy : 0.9358
##                  95% CI : (0.8722, 0.9738)
##     No Information Rate : 0.6881
##     P-Value [Acc > NIR] : 2.754e-10
##
##                   Kappa : 0.8604
##
##  Mcnemar's Test P-Value : 0.5062
##
## Statistics by Class:
##
##                      Class: 0=Blood Donor Class: 1=Hepatitis
## Sensitivity                        0.9867             0.8750
## Specificity                        0.9118             0.9765
## Pos Pred Value                     0.9610             0.9130
## Neg Pred Value                     0.9688             0.9651
## Prevalence                         0.6881             0.2202
## Detection Rate                     0.6789             0.1927
## Detection Prevalence               0.7064             0.2110
## Balanced Accuracy                  0.9492             0.9257
##                      Class: Progressed HCV
## Sensitivity                        0.70000
## Specificity                        0.97980
## Pos Pred Value                     0.77778
## Neg Pred Value                     0.97000
## Prevalence                         0.09174
## Detection Rate                     0.06422
## Detection Prevalence               0.08257
## Balanced Accuracy                  0.83990
```

```r
# Save metrics
# Extract the table from the confusion matrix
cm_table <- confusion_matrix$table

# Initialize vectors to store the metrics for each class
precision <- numeric(length = ncol(cm_table))
recall <- numeric(length = ncol(cm_table))
f1_score <- numeric(length = ncol(cm_table))

# Calculate metrics for each class
for (i in 1:ncol(cm_table)) {
  tp <- cm_table[i, i]
  fp <- sum(cm_table[, i]) - tp
  fn <- sum(cm_table[i, ]) - tp
  tn <- sum(cm_table) - tp - fp - fn

  precision[i] <- tp / (tp + fp)
  recall[i] <- tp / (tp + fn)
  f1_score[i] <- 2 * (precision[i] * recall[i]) / (precision[i] + recall[i])
}
```

```r
# Output the results
svm_metrics <- data.frame(
  Class = colnames(cm_table),
  Precision = precision,
  Recall = recall,
  F1_Score = f1_score
)

svm_metrics
```

```
##            Class Precision    Recall  F1_Score
## 1  0=Blood Donor 0.9866667 0.9610390 0.9736842
## 2    1=Hepatitis 0.8750000 0.9130435 0.8936170
## 3 Progressed HCV 0.7000000 0.7777778 0.7368421
```

SVMs with appropriate kernel functions can be effective in capturing non-linearity. We can address class imbalances by tuning kernel parameters. Although the math is complicated, SVMs have a reliable theoretical foundation and perform well in complex scenarios. The different kernel functions help to transform the input data into a higher-dimensional space, which can help capture more complex relationships. Here we use a RBF kernel which is highly versatile and effective for non-linear data. The RBF kernel can handle cases where the relationship between class labels and features is more complex. We use confusion matrix, kappa, precision, recall, and F1 score to evaluate the model as these metrics are useful for imbalanced datasets.

The SVM model has high precision, recall, and F1 scores reflecting the model's effectiveness in classifying Blood Donor cases. There is good model performance for the Hepatitis class. It is fairly accurate in predictions but has a slightly higher rate of false positives and negatives. It misclassifies 2 of out 24 as Blood Donor and 1 as Progressed HCV. There model misclassifies 2 out of 10 Progressed HCV as Hepatitis and 1 as Blood Donor. Although this could use some improvement by adding more data, it's better to have more false positives than to have false negatives. The strong kappa of 0.86 indicates a strong agreement between the model's predictions and the actual class labels.

## Model Comparison

```r
xgb_df <- data.frame(Value = xgb_metrics)
rf_df <- data.frame(Value = rf_metrics)
svm_df <- data.frame(Value = svm_metrics)

# Add a new column 'Model' to each dataframe
xgb_df$Model <- "XGBoost"
rf_df$Model <- "randomForest"
svm_df$Model <- "SVM"

# Combine all three dataframes into one
combined_df <- rbind(xgb_df, rf_df, svm_df)
combined_df <- combined_df[, c("Model", "Value.Class", "Value.Precision", "Value.Recall", "Value.F1_Sco
combined_df
```

```
##          Model    Value.Class Value.Precision Value.Recall Value.F1_Score
## 1      XGBoost  0=Blood Donor       1.0000000    0.9868421      0.9933775
## 2      XGBoost    1=Hepatitis       1.0000000    0.9230769      0.9600000
```

```
## 3      XGBoost Progressed HCV       0.7000000    1.0000000      0.8235294
## 4 randomForest  0=Blood Donor       0.9733333    0.9864865      0.9798658
## 5 randomForest    1=Hepatitis       1.0000000    0.9230769      0.9600000
## 6 randomForest Progressed HCV       0.7000000    0.7777778      0.7368421
## 7          SVM  0=Blood Donor       0.9866667    0.9610390      0.9736842
## 8          SVM    1=Hepatitis       0.8750000    0.9130435      0.8936170
## 9          SVM Progressed HCV       0.7000000    0.7777778      0.7368421
```

In comparing the performance of the XGBoost, RandomForest, and SVM models across three classes, Blood Donor, Hepatitis, and Progressed HCV, some distinct patterns emerge. For the Blood Donor class, XGBoost shows exceptional performance with perfect precision and the highest F1 score (0.993), closely followed by RandomForest and SVM, both exhibiting high precision and F1 scores, though slightly lower. In the Hepatitis class, while all three models achieve the same precision, XGBoost and RandomForest outperform SVM in terms of recall and F1 score, both achieving an F1 score of 0.960 compared to SVM's 0.894. The most notable differences are observed in the Progressed HCV class, where XGBoost excels with a perfect recall and the highest F1 score (0.824), significantly outperforming both RandomForest and SVM, which show identical precision, recall, and F1 scores (0.700, 0.778, and 0.737, respectively). This indicates XGBoost's superior ability in correctly identifying all cases of Progressed HCV, a class where the other two models demonstrate comparatively weaker performance.

## Ensemble

```
predictCategory <- function(new_data) {
  # XGBoost Model (Multi-class)
  # Train the model
  xgb_model <- xgboost(data = train_matrix,
                       label = train_labels_numeric,
                       params = params,
                       nrounds = 50)

  # Format validation data
  validation_data_xg <- new_data
  validation_labels_factor <- factor(validation_data_xg$Category, levels = c("0=Blood Donor", "1=Hepati
  validation_matrix <- as.matrix(validation_data_xg[, features])


  # Make predictions on new data
  xgb_predictions <- predict(xgb_model, newdata = as.matrix(validation_data_xg[, -which(names(validatio

  # Convert predictions to class labels
  max_prob_index <- matrix(xgb_predictions, ncol = length(unique(train_data_xg$Category)), byrow = TRUE
  xgb_class_labels <- apply(max_prob_index, 1, which.max) - 1
  xgb_class_labels <- factor(xgb_class_labels, levels = 0:2, labels = c("0=Blood Donor", "1=Hepatitis",

  # Random Forest Model (Multi-class)
  set.seed(123)
  bagged_model <- randomForest(Category ~ ., data = train_data)
  rf_predictions <- predict(bagged_model, newdata = new_data)

  # SVM model (Multi-class)
  svm_model <- svm(Category ~ ., data = train_data, kernel = "radial")
  svm_predictions <- predict(svm_model, newdata = new_data)
```

```
  combined_predictions <- cbind(xgb_class_labels, rf_predictions, svm_predictions)

   # Create a function to calculate majority vote
  majority_vote <- function(row) {
    # Count the occurrences of each class label in the row
    counts <- table(row)

    # Get the class label with the maximum count
    majority_label <- as.character(as.numeric(names(counts)[which.max(counts)]))

    return(majority_label)
  }

  # Apply the majority vote function to each row to get the final predictions
  ensemble_predictions <- apply(combined_predictions, 1, majority_vote)

  # Return the final prediction
  return(ensemble_predictions)
}
```

Here we create an ensemble is the aggregate of the 3 models and uses majority vote to make predictions.

### Comparison of Ensemble to Individual Models

```
predictions <- predictCategory(validation_data)
mapping <- c("0=Blood Donor", "1=Hepatitis", "Progressed HCV")
predictions <- mapping[as.integer(predictions)]
predictions <- as.factor(predictions)
```

```
confusion_matrix <- confusionMatrix(predictions, validation_data$Category)
confusion_matrix$table
```

```
##                 Reference
## Prediction       0=Blood Donor 1=Hepatitis Progressed HCV
##   0=Blood Donor             74           0              1
##   1=Hepatitis                0          24              1
##   Progressed HCV             1           0              8
```

```
confusion_matrix$byClass
```

```
##                      Sensitivity Specificity Pos Pred Value Neg Pred Value
## Class: 0=Blood Donor   0.9866667   0.9705882      0.9866667      0.9705882
## Class: 1=Hepatitis     1.0000000   0.9882353      0.9600000      1.0000000
## Class: Progressed HCV  0.8000000   0.9898990      0.8888889      0.9800000
##                      Precision    Recall        F1 Prevalence Detection Rate
## Class: 0=Blood Donor  0.9866667 0.9866667 0.9866667 0.68807339      0.6788991
## Class: 1=Hepatitis    0.9600000 1.0000000 0.9795918 0.22018349      0.2201835
## Class: Progressed HCV 0.8888889 0.8000000 0.8421053 0.09174312      0.0733945
##                      Detection Prevalence Balanced Accuracy
```

```
## Class: 0=Blood Donor            0.68807339         0.9786275
## Class: 1=Hepatitis              0.22935780         0.9941176
## Class: Progressed HCV           0.08256881         0.8949495
```

```
confusion_matrix$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.724771e-01   9.412504e-01   9.216699e-01   9.942877e-01   6.880734e-01
## AccuracyPValue  McnemarPValue
##   4.174843e-14            NaN
```

The ensemble model demonstrates a robust performance across the three classes, Blood Donor, Hepatitis, and Progressed HCV, as indicated by various metrics such as sensitivity, specificity, precision, NPV, recall, and F1 score. The model shows high precision and perfect recall for the Hepatitis class, indicating a high accuracy in the model's prediciton of Hepatitis with very few false positives. The F1 score is also high (97.96%) achieving a high blance in classification accuracy. The ensemble also performs better compared to the individual models in predicting Progressed HCV, with a higher sensitivity, specificity, precision, recall, and F1 score than the individual models. Most of the model's predictions for Progressed HCV are reliable with 8/10 correctly classified as Progressed HCV, 1/10 misclassified as Blood Donor and 1/10 misclassified as Hepatitis. The kappa of the overall model is high (94.13%) indicating a strong agreement between the model's predictions and the actual classes. We could boost sensitivity and recall in the Progressed HCV class by obtaining more data for Fibrosis and Cirrhosis patients, using class weights, and oversampling the Progressed HCV class. Overall, the ensemble model is highly effective in identifity Blood Donor and Hepatitis classes, with slightly less but strong performance in the Progressed HCV class. The model's ability to maintain high precision and recall across classes is indicative of its robustness in this multi-class classification task.

### K-Cross Validation

```
balanced_data$Category <- factor(balanced_data$Category, levels = c("0=Blood Donor", "1=Hepatitis", "Pr
features <- setdiff(names(balanced_data), "Category")

# Number of folds
k <- 3
set.seed(123)
folds <- createFolds(balanced_data$Category, k = k, list = TRUE)

# List and matrices for metrics
results <- list()
precision_sum <- matrix(0, nrow = length(levels(balanced_data$Category)), ncol = k)
recall_sum <- matrix(0, nrow = length(levels(balanced_data$Category)), ncol = k)
f1_sum <- matrix(0, nrow = length(levels(balanced_data$Category)), ncol = k)



for(i in seq_along(folds)) {
    # Split the data using stratified folds
    trainingSet <- balanced_data[folds[[i]], ]
    validationSet <- balanced_data[-folds[[i]], ]

    # Train the ensemble model and make predictions
    train_labels_numeric <- as.numeric(as.factor(trainingSet$Category)) - 1
```

```r
  train_matrix <- as.matrix(trainingSet[, features])
  validation_matrix <- as.matrix(validationSet[, features])

  # Train XGBoost Model
  xgb_model <- xgboost(data = train_matrix,
                       label = train_labels_numeric,
                       params = params,
                       nrounds = 50)

  # Make predictions on new data
  xgb_predictions <- predict(xgb_model, newdata = validation_matrix)

  # Convert predictions to class labels
  max_prob_index <- matrix(xgb_predictions, ncol = length(unique(trainingSet$Category)), byrow = TRUE)
  xgb_class_labels <- apply(max_prob_index, 1, which.max) - 1
  xgb_class_labels <- factor(xgb_class_labels, levels = 0:2, labels = c("0=Blood Donor", "1=Hepatitis

  # Random Forest Model (Multi-class)
  set.seed(123)
  bagged_model <- randomForest(Category ~ ., data = trainingSet)
  rf_predictions <- predict(bagged_model, newdata = validationSet)

  # SVM model (Multi-class)
  svm_model <- svm(Category ~ ., data = trainingSet, kernel = "radial")
  svm_predictions <- predict(svm_model, newdata = validationSet)

  # Combined predictions
  combined_predictions <- cbind(xgb_class_labels, rf_predictions, svm_predictions)

  majority_vote <- function(row) {
  # Count the occurrences of each class label in the row
  counts <- table(row)

  # Get the class label with the maximum count
  majority_label <- as.character(as.numeric(names(counts)[which.max(counts)]))

  return(majority_label)
}

  # Majority Vote
  ensemble_predictions <- apply(combined_predictions, 1, majority_vote)
  ensemble_predictions <- factor(ensemble_predictions, levels = c("1", "2", "3"), labels = c("0=Blood

  # Confusion Matrix
  cm <- confusionMatrix(ensemble_predictions, validationSet$Category)
  results[[i]] <- cm

  ## Extracting per-class metrics
  fold_metrics <- cm$byClass
  class_levels <- levels(validationSet$Category)

  for (j in 1:length(class_levels)) {
      class_name <- paste("Class:", class_levels[j])
```

```
        precision_sum[j, i] <- fold_metrics[class_name, "Precision"]
        recall_sum[j, i] <- fold_metrics[class_name, "Recall"]
        f1_sum[j, i] <- fold_metrics[class_name, "F1"]
    }
}
```

```
## [1]  train-mlogloss:0.759821
## [2]  train-mlogloss:0.562813
## [3]  train-mlogloss:0.427044
## [4]  train-mlogloss:0.331612
## [5]  train-mlogloss:0.261834
## [6]  train-mlogloss:0.210893
## [7]  train-mlogloss:0.171530
## [8]  train-mlogloss:0.137419
## [9]  train-mlogloss:0.111293
## [10] train-mlogloss:0.092366
## [11] train-mlogloss:0.077301
## [12] train-mlogloss:0.066217
## [13] train-mlogloss:0.056628
## [14] train-mlogloss:0.049355
## [15] train-mlogloss:0.043429
## [16] train-mlogloss:0.039173
## [17] train-mlogloss:0.035333
## [18] train-mlogloss:0.032415
## [19] train-mlogloss:0.029927
## [20] train-mlogloss:0.027597
## [21] train-mlogloss:0.025516
## [22] train-mlogloss:0.024072
## [23] train-mlogloss:0.022577
## [24] train-mlogloss:0.021253
## [25] train-mlogloss:0.020192
## [26] train-mlogloss:0.019216
## [27] train-mlogloss:0.018397
## [28] train-mlogloss:0.017544
## [29] train-mlogloss:0.017037
## [30] train-mlogloss:0.016558
## [31] train-mlogloss:0.015931
## [32] train-mlogloss:0.015551
## [33] train-mlogloss:0.015211
## [34] train-mlogloss:0.014888
## [35] train-mlogloss:0.014605
## [36] train-mlogloss:0.014319
## [37] train-mlogloss:0.014024
## [38] train-mlogloss:0.013811
## [39] train-mlogloss:0.013506
## [40] train-mlogloss:0.013325
## [41] train-mlogloss:0.013091
## [42] train-mlogloss:0.012945
## [43] train-mlogloss:0.012789
## [44] train-mlogloss:0.012659
## [45] train-mlogloss:0.012538
## [46] train-mlogloss:0.012416
## [47] train-mlogloss:0.012305
```

```
## [48] train-mlogloss:0.012198
## [49] train-mlogloss:0.012097
## [50] train-mlogloss:0.011996
## [1]  train-mlogloss:0.760625
## [2]  train-mlogloss:0.555487
## [3]  train-mlogloss:0.418218
## [4]  train-mlogloss:0.325195
## [5]  train-mlogloss:0.254105
## [6]  train-mlogloss:0.203952
## [7]  train-mlogloss:0.164472
## [8]  train-mlogloss:0.133376
## [9]  train-mlogloss:0.109846
## [10] train-mlogloss:0.090824
## [11] train-mlogloss:0.077223
## [12] train-mlogloss:0.065893
## [13] train-mlogloss:0.057415
## [14] train-mlogloss:0.049949
## [15] train-mlogloss:0.044079
## [16] train-mlogloss:0.038877
## [17] train-mlogloss:0.035452
## [18] train-mlogloss:0.032306
## [19] train-mlogloss:0.029774
## [20] train-mlogloss:0.027900
## [21] train-mlogloss:0.026009
## [22] train-mlogloss:0.024180
## [23] train-mlogloss:0.022769
## [24] train-mlogloss:0.021391
## [25] train-mlogloss:0.019968
## [26] train-mlogloss:0.019083
## [27] train-mlogloss:0.018011
## [28] train-mlogloss:0.017294
## [29] train-mlogloss:0.016600
## [30] train-mlogloss:0.016019
## [31] train-mlogloss:0.015433
## [32] train-mlogloss:0.015030
## [33] train-mlogloss:0.014622
## [34] train-mlogloss:0.014330
## [35] train-mlogloss:0.014002
## [36] train-mlogloss:0.013687
## [37] train-mlogloss:0.013419
## [38] train-mlogloss:0.013184
## [39] train-mlogloss:0.012968
## [40] train-mlogloss:0.012752
## [41] train-mlogloss:0.012461
## [42] train-mlogloss:0.012271
## [43] train-mlogloss:0.012086
## [44] train-mlogloss:0.011904
## [45] train-mlogloss:0.011736
## [46] train-mlogloss:0.011575
## [47] train-mlogloss:0.011421
## [48] train-mlogloss:0.011273
## [49] train-mlogloss:0.011136
## [50] train-mlogloss:0.011005
## [1]  train-mlogloss:0.771033
```

```
## [2]  train-mlogloss:0.579663
## [3]  train-mlogloss:0.449843
## [4]  train-mlogloss:0.361301
## [5]  train-mlogloss:0.296297
## [6]  train-mlogloss:0.241813
## [7]  train-mlogloss:0.199639
## [8]  train-mlogloss:0.169042
## [9]  train-mlogloss:0.142868
## [10] train-mlogloss:0.120611
## [11] train-mlogloss:0.103764
## [12] train-mlogloss:0.089417
## [13] train-mlogloss:0.077350
## [14] train-mlogloss:0.069284
## [15] train-mlogloss:0.061745
## [16] train-mlogloss:0.055595
## [17] train-mlogloss:0.050299
## [18] train-mlogloss:0.045791
## [19] train-mlogloss:0.041044
## [20] train-mlogloss:0.037921
## [21] train-mlogloss:0.034511
## [22] train-mlogloss:0.032264
## [23] train-mlogloss:0.030115
## [24] train-mlogloss:0.027997
## [25] train-mlogloss:0.026037
## [26] train-mlogloss:0.024426
## [27] train-mlogloss:0.023370
## [28] train-mlogloss:0.022216
## [29] train-mlogloss:0.021150
## [30] train-mlogloss:0.020475
## [31] train-mlogloss:0.019580
## [32] train-mlogloss:0.018951
## [33] train-mlogloss:0.018343
## [34] train-mlogloss:0.017793
## [35] train-mlogloss:0.017388
## [36] train-mlogloss:0.017065
## [37] train-mlogloss:0.016789
## [38] train-mlogloss:0.016409
## [39] train-mlogloss:0.016111
## [40] train-mlogloss:0.015875
## [41] train-mlogloss:0.015647
## [42] train-mlogloss:0.015411
## [43] train-mlogloss:0.015091
## [44] train-mlogloss:0.014873
## [45] train-mlogloss:0.014573
## [46] train-mlogloss:0.014377
## [47] train-mlogloss:0.014116
## [48] train-mlogloss:0.013887
## [49] train-mlogloss:0.013683
## [50] train-mlogloss:0.013527
```

```r
# Calculate average performance across all folds
avg_performance <- lapply(results, function(x) x$overall)
avg_performance <- do.call("rbind", avg_performance)
```

```r
# Calculate average metrics for each class
average_precision <- rowMeans(precision_sum, na.rm = TRUE)
average_recall <- rowMeans(recall_sum, na.rm = TRUE)
average_f1 <- rowMeans(f1_sum, na.rm = TRUE)

# Combine the averages into a data frame
avg_metrics <- data.frame(
    Class = levels(balanced_data$Category),
    Precision = average_precision,
    Recall = average_recall,
    F1_Score = average_f1
)

avg_metrics
```

```
##            Class Precision    Recall  F1_Score
## 1  0=Blood Donor 0.9587283 0.9880000 0.9731185
## 2    1=Hepatitis 0.8603557 0.8708333 0.8651548
## 3 Progressed HCV 0.7885595 0.5784314 0.6610869
```

```r
colMeans(avg_performance)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.239927e-01   8.340619e-01   8.918157e-01   9.490184e-01   6.868132e-01
## AccuracyPValue  McnemarPValue
##   2.893736e-27   1.540543e-01
```

Here we use stratified k-cross validation to further test our ensemble. K-cross validation partitions the dataset in k equally sized folds and for each iteration, the model is trained on "k-1" folds of the data. The remaining 1 fold is used as a test set to evaluate the model. This means that every data point gets to be the test set exactly once and in the training set "k-1" times. This reduces the bias that the model's performance estimate is dependent on the specific way the data is split. This is useful for imbalanced classes. The results show us a similar conclusion to the aforementioned evaluation, however, the recall and F1 score for Progressed HCV is lower, which we hypothesize is due to the class imbalance. More data for Fibrosis and Cirrhosis patients can improve this model.

## Using Ensemble to Predict New Data

```r
# Assuming new data has been transformed and cleaned
new_data <- data.frame(ALB = 3.8,
                       ALP = 3.5,
                       ALT = 3.2,
                       AST = 0.0007,
                       BIL = 1.8,
                       CHE = 10.12,
                       CHOL = 5.23,
                       CREA = 4.33,
                       GGT = 4.33,
                       Category = "")
# Prediction
```

```
prediction <- predictCategory(new_data)
prediction <- mapping[as.integer(prediction)]
```

```
prediction
```

```
## [1] "1=Hepatitis"
```

Assuming that the new input data has been cleaned and transformed, the ensemble model predicts the new data to be HCV. I used lab values that were similar to a patient with HCV and the model correctly predicted the patient to have HCV.