BalgobinS.DA5030.Project.Rmd

2023-08-08

Predicting HCV Infection 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. We will be using binary classification and grouping patients of all stages of hepatitis into one group as there is extreme class imbalance. 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~
```

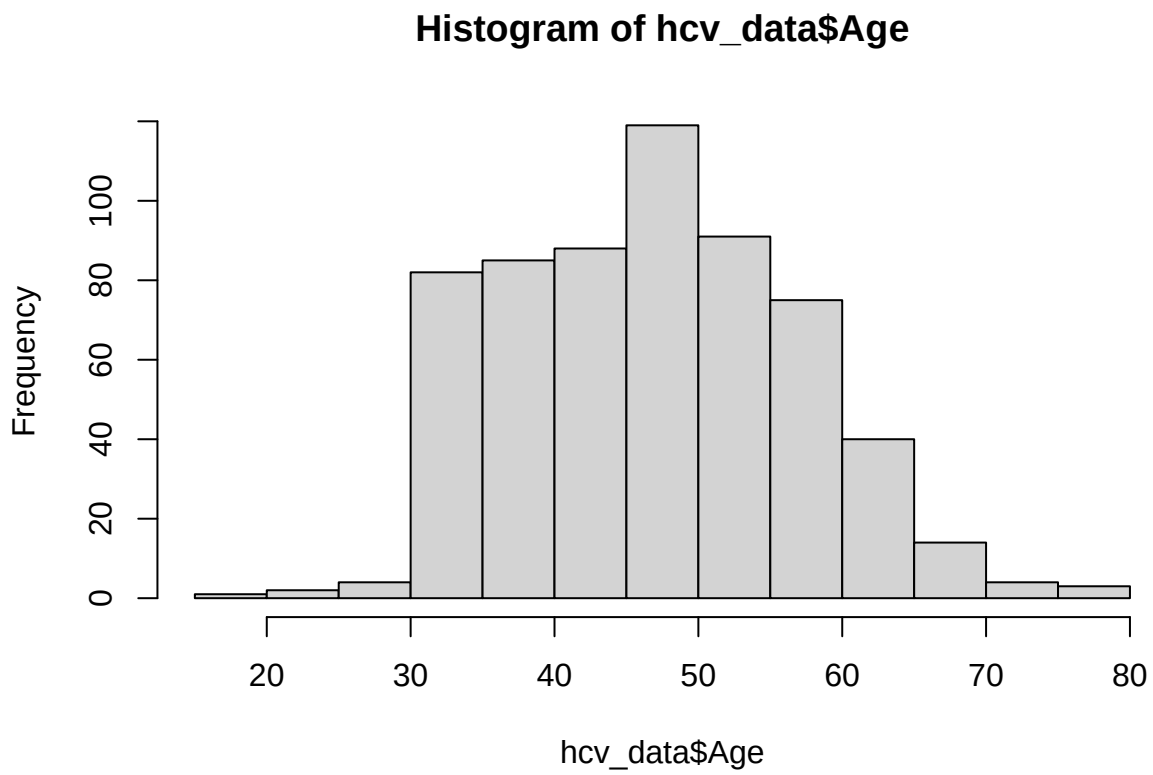
```
summary(hcv_data)
```

```
##        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
```

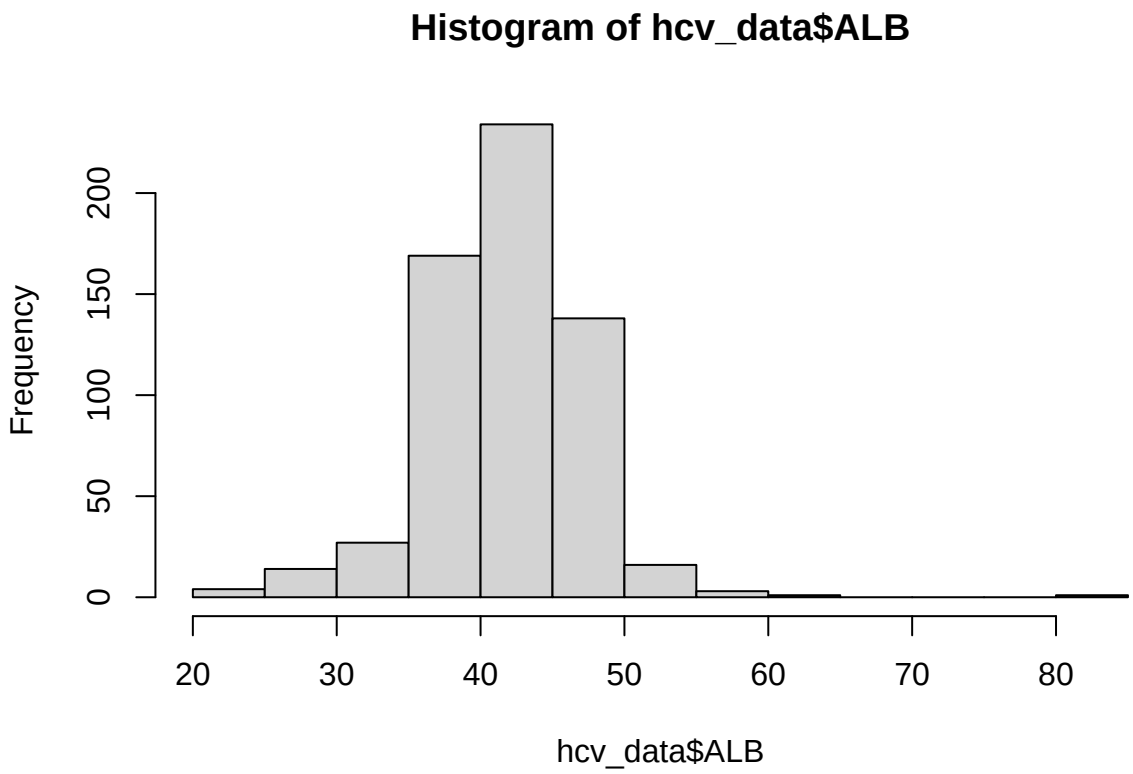
From the initial data exploration, we see that the target variable is "Category" and has 5 classes. There are 2 categorical variables, age and sex, and 10 continuous variables which represent the different lab tests and their values. The first column seems to be the patient ID, which we can drop. The 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 suspect that a multi-class classification model will have very low predictive power. We will do binary classification instead, and group the hepatitis, fibrosis, and cirrhosis patients into one category.

```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("1=Hepatitis", "2=Fibrosis", "3=Cirrhosis")
replacement_val <- c("HCV")
hcv_data <- hcv_data %>%
  mutate(
    Category = ifelse(Category %in% vals_to_replace, replacement_val, Category)
  )
```

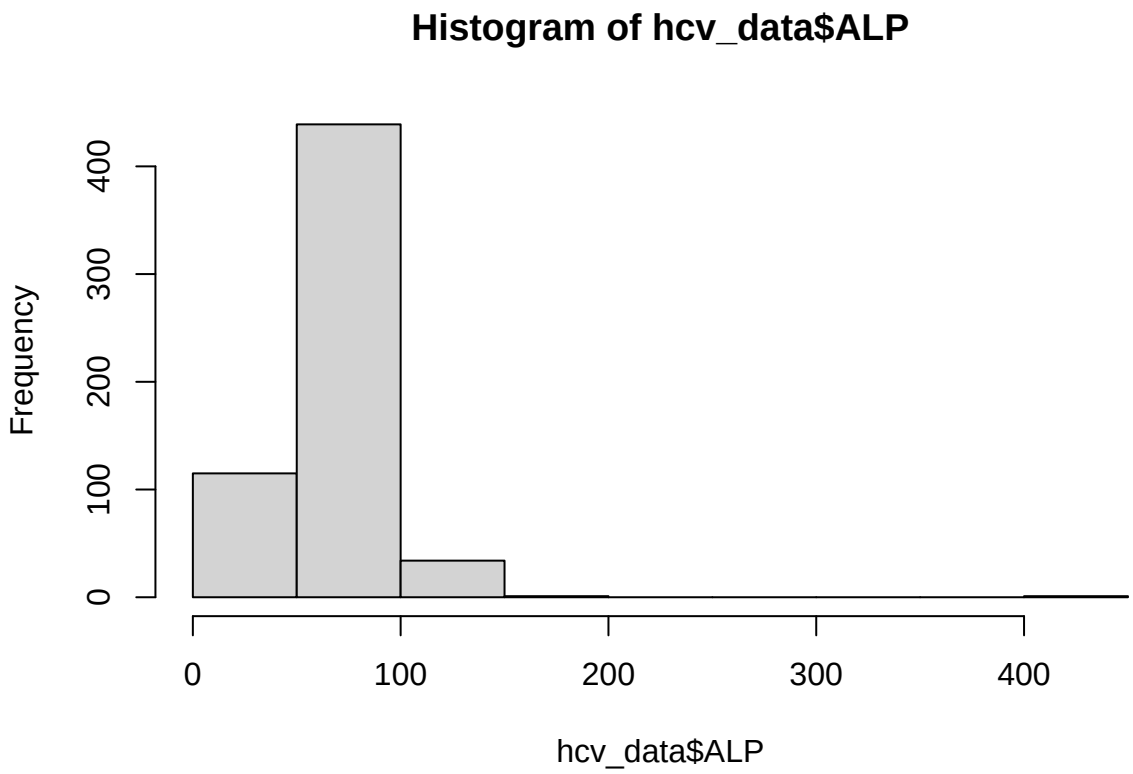
```r
# Plot density for each variable
hist(hcv_data$Age)
```



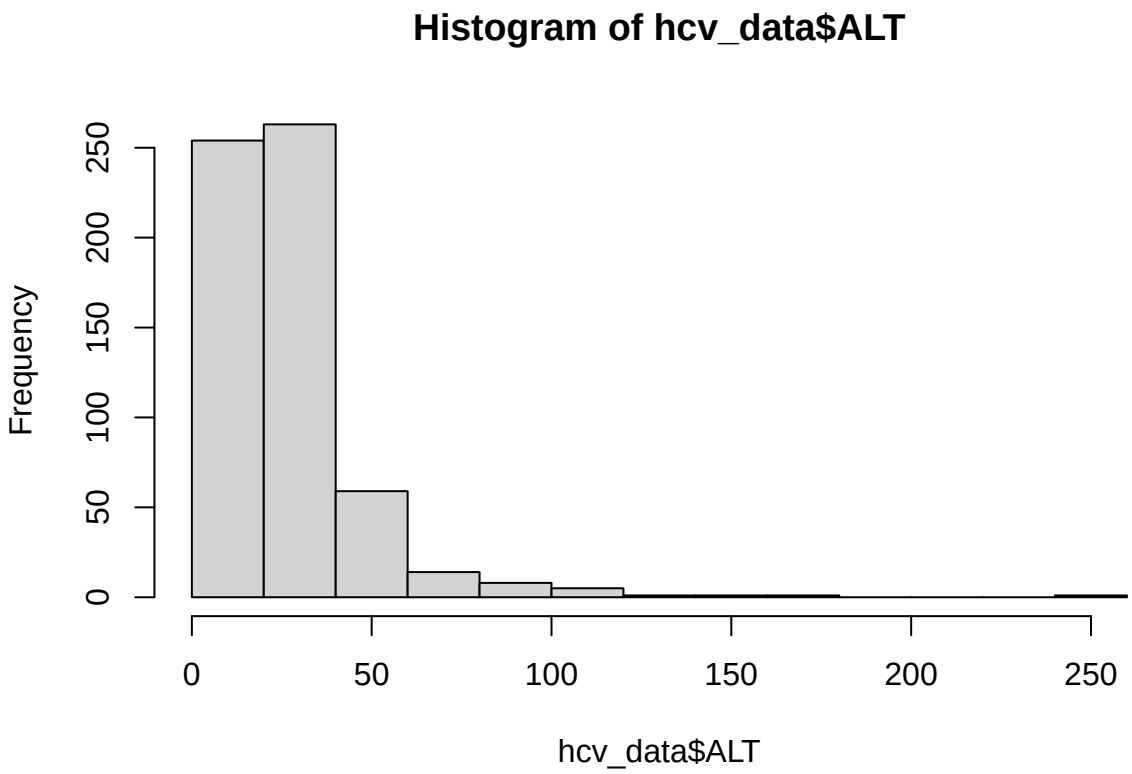

```r
hist(hcv_data$ALB)
```



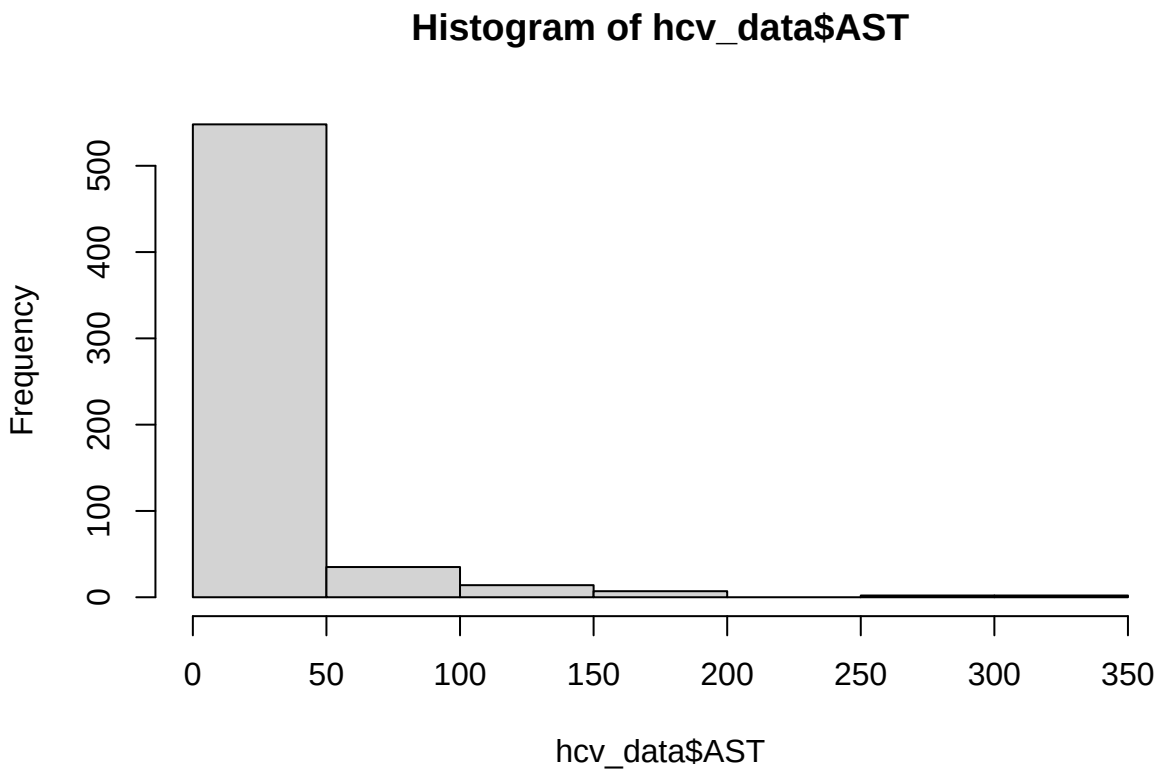

```
hist(hcv_data$ALP)
```



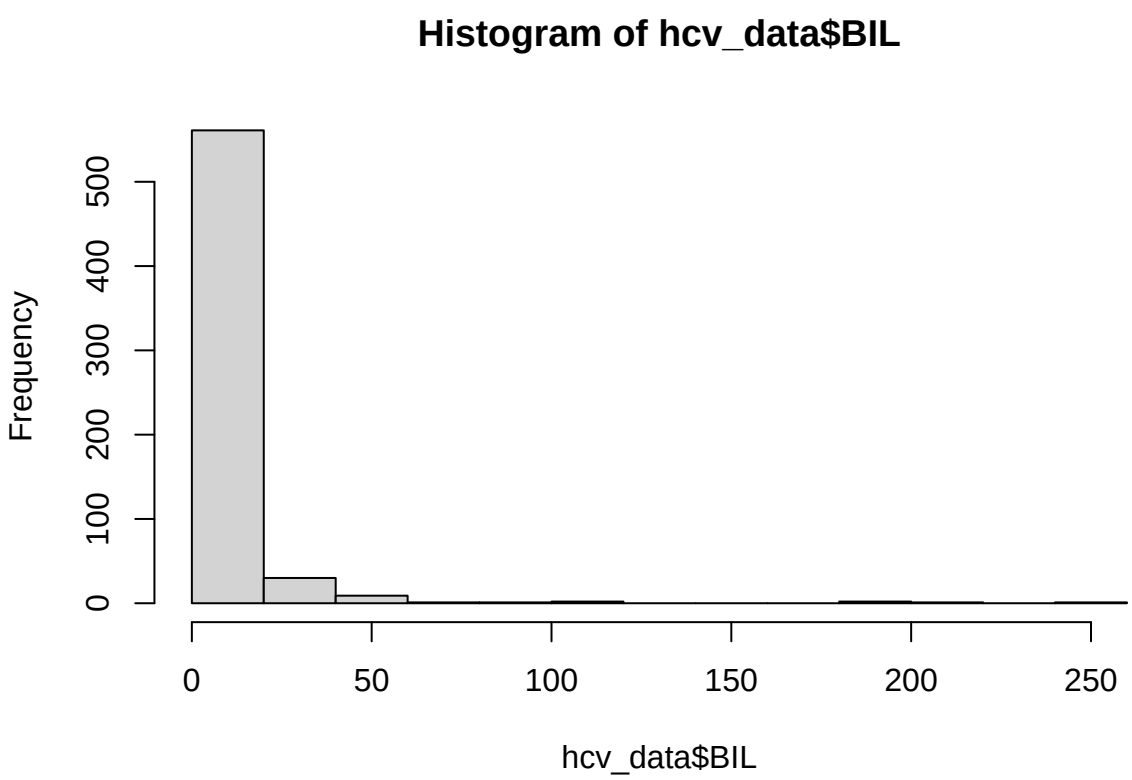

```
hist(hcv_data$ALT)
```



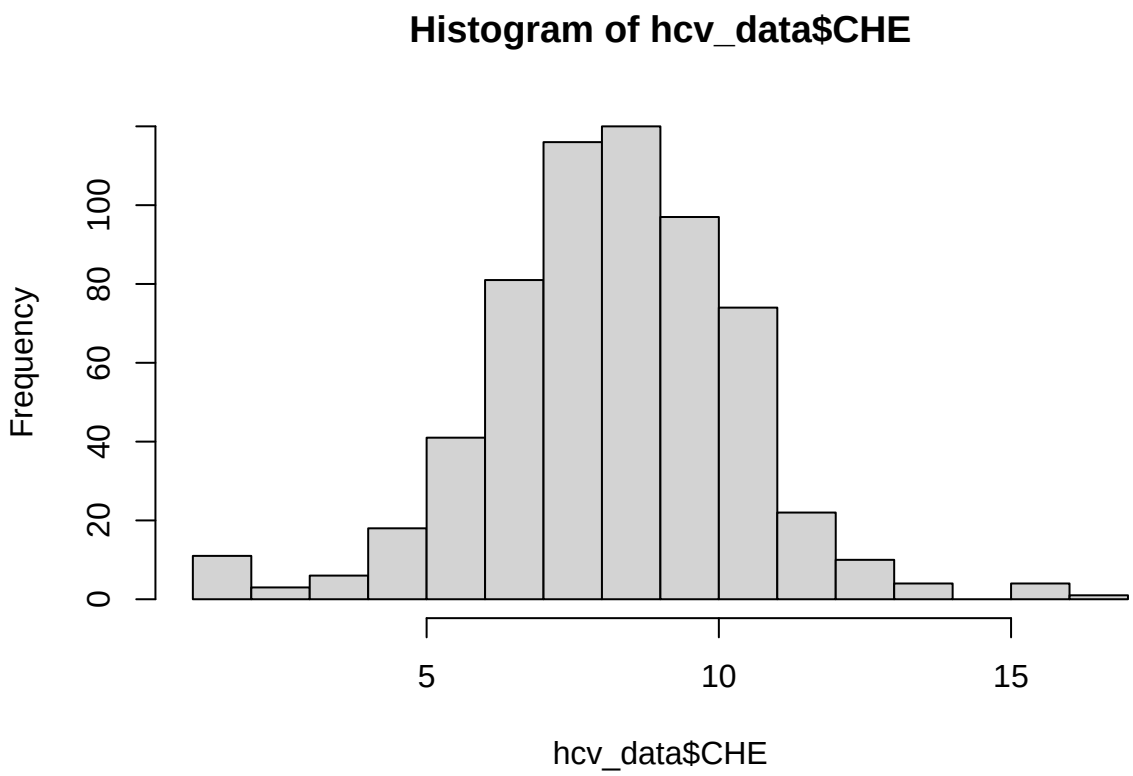

```
hist(hcv_data$AST)
```



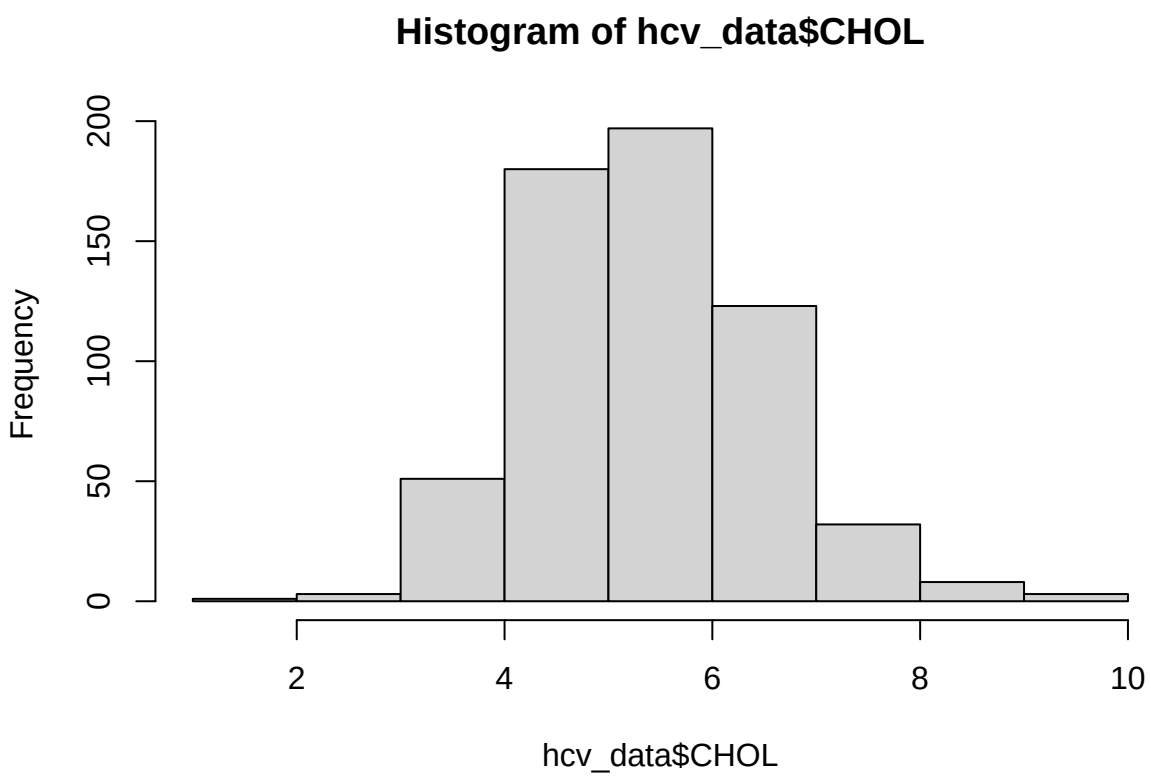

```
hist(hcv_data$BIL)
```



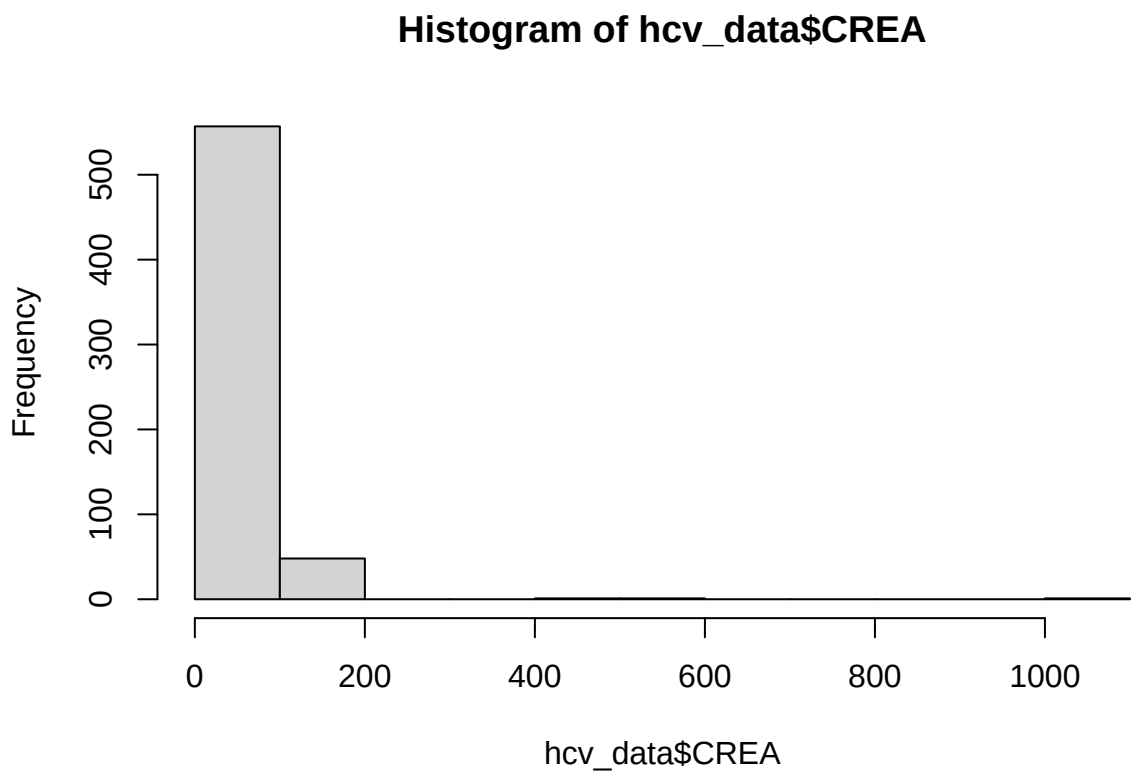

```
hist(hcv_data$CHE)
```



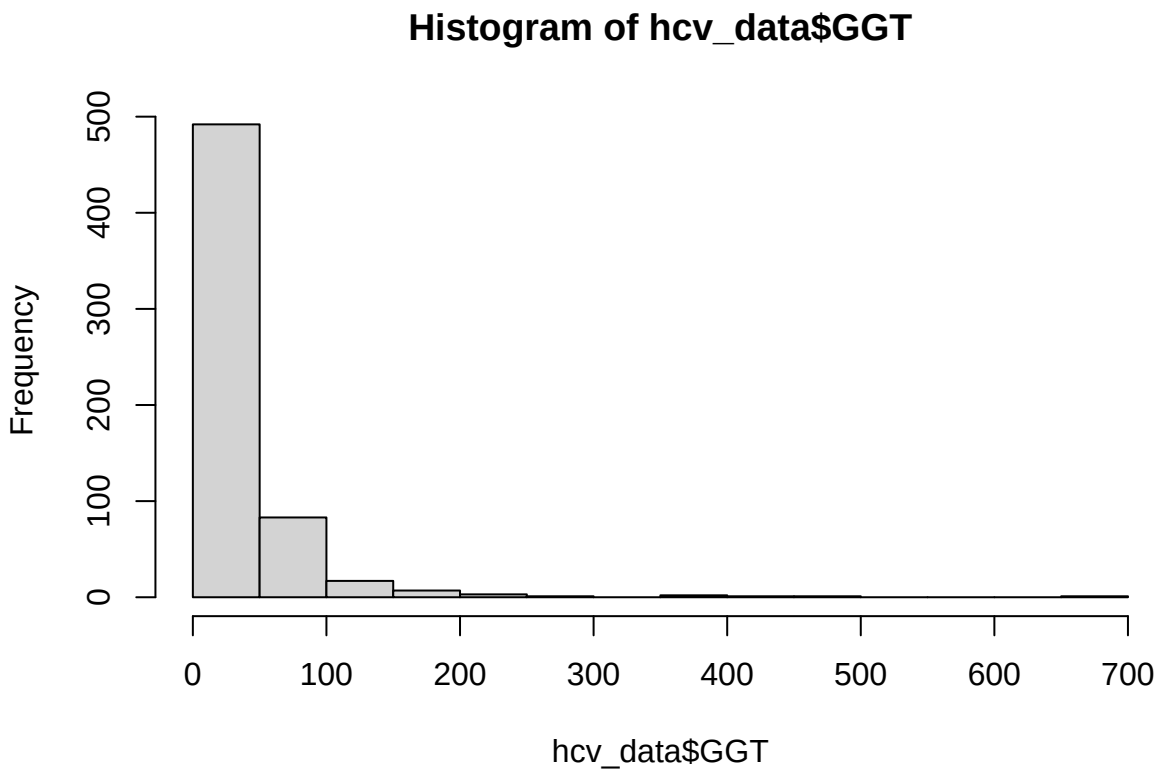

```
hist(hcv_data$CHOL)
```



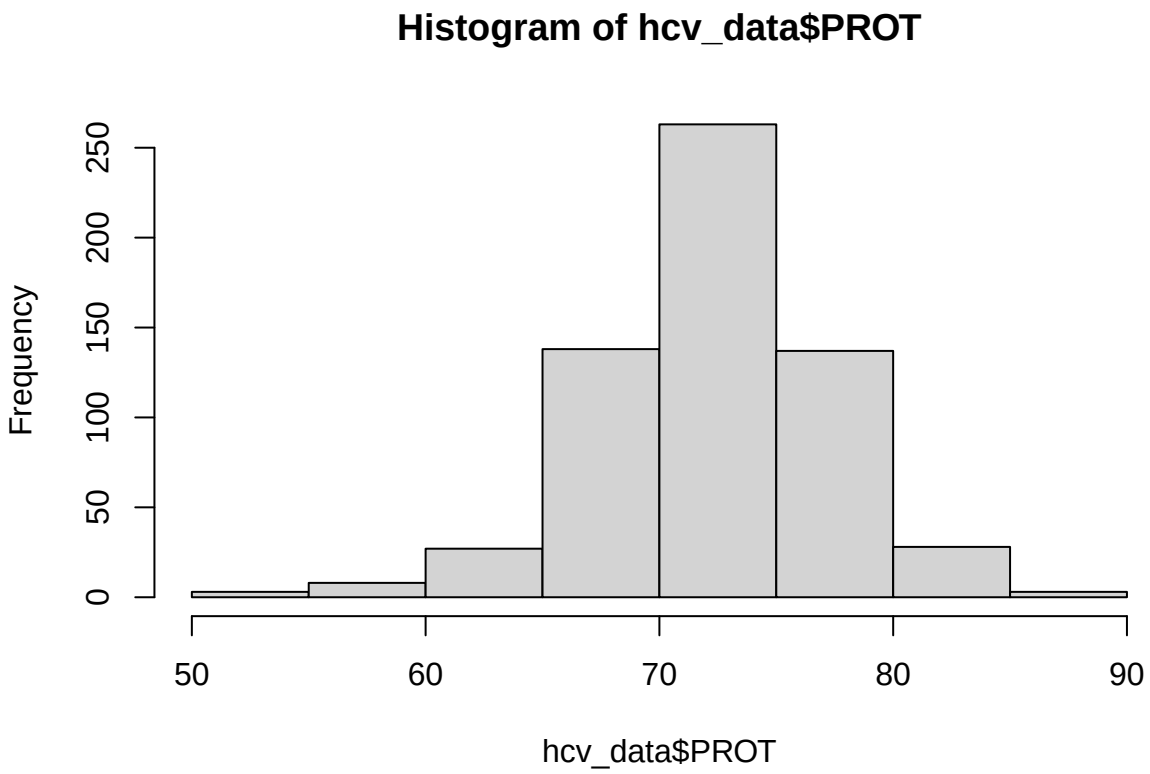

```
hist(hcv_data$CREA)
```




```
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

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

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

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

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

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

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

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

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

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

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

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

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

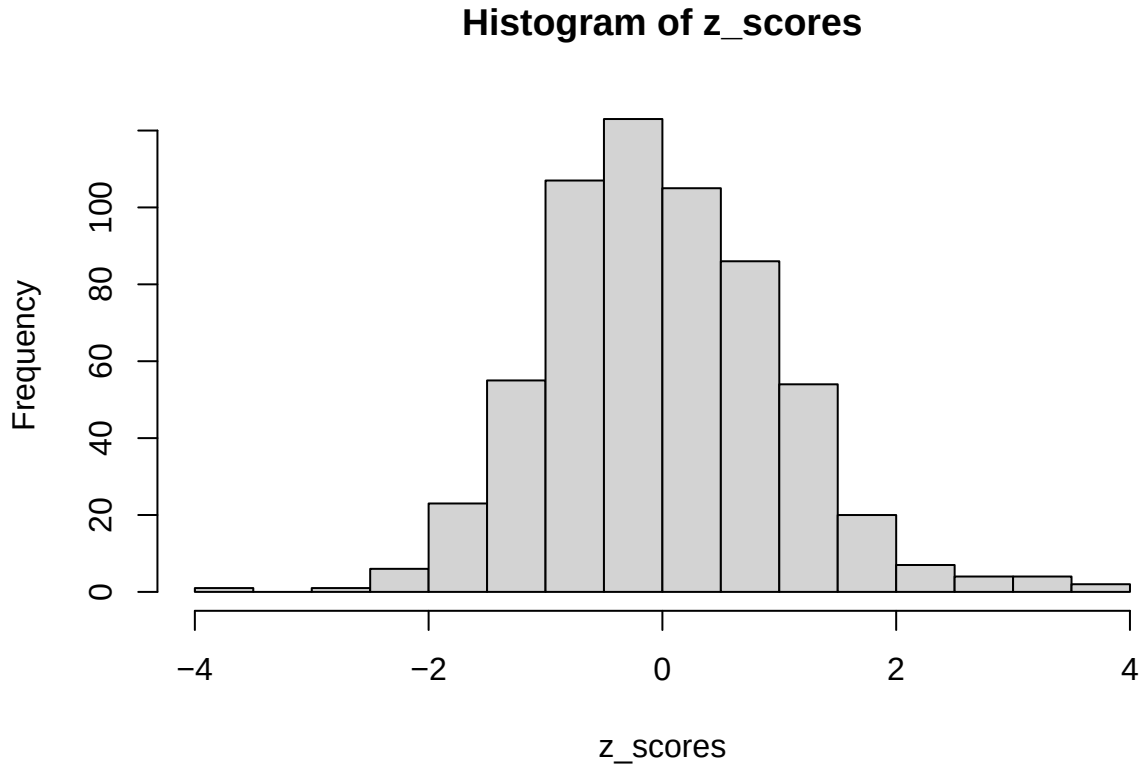
```
## [1] 8
```

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

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

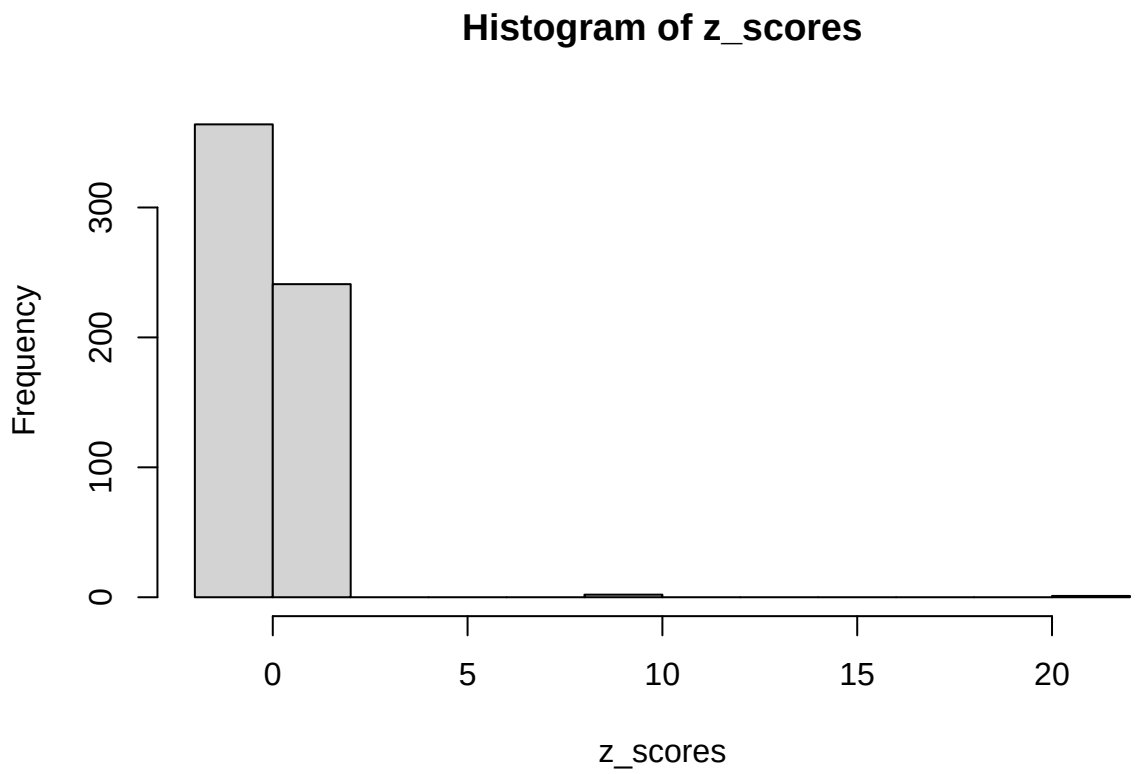
```
## [1] -3.527649  3.833477
```

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



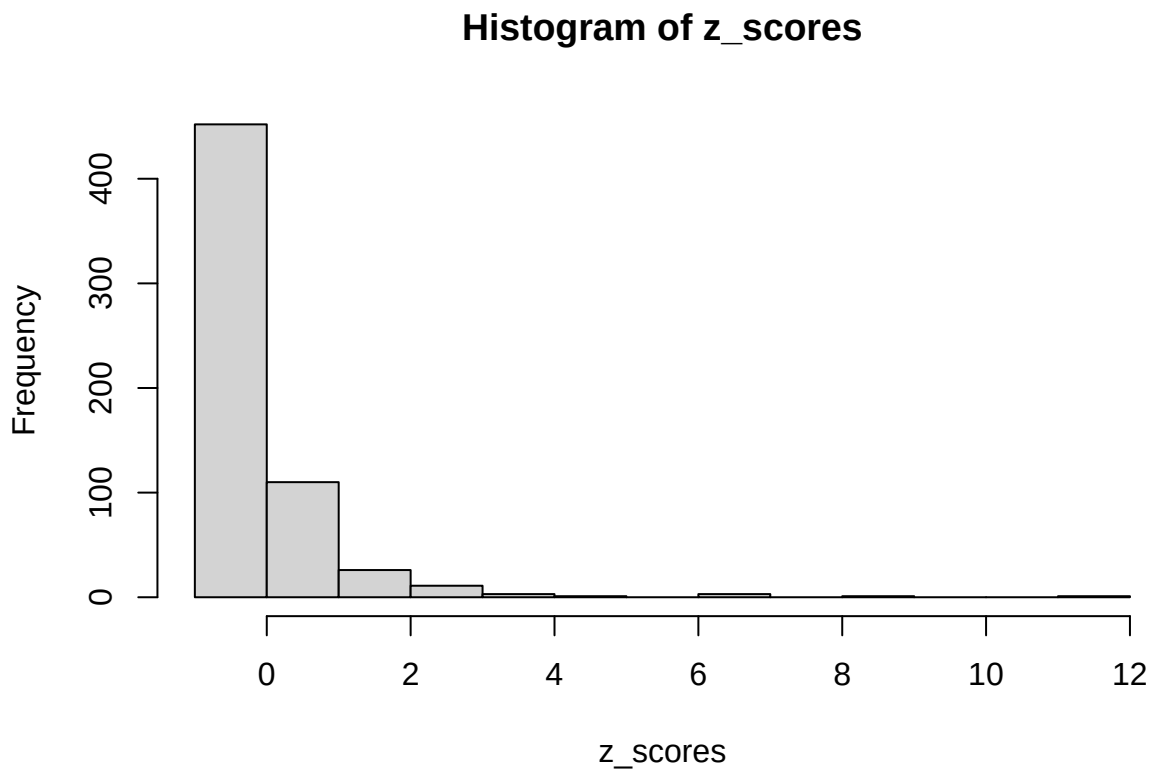

```
## [1] -1.478524 20.063833
```

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



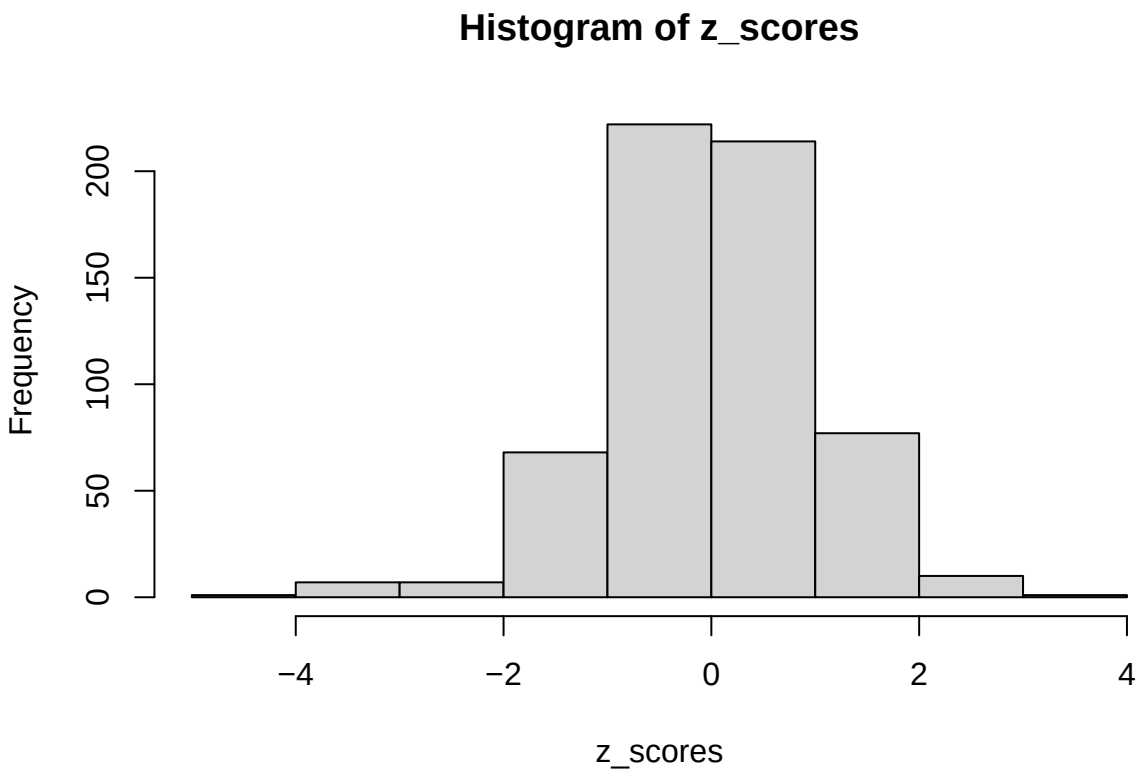

```
## [1] -0.6495057 11.7924565
```

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




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

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



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.

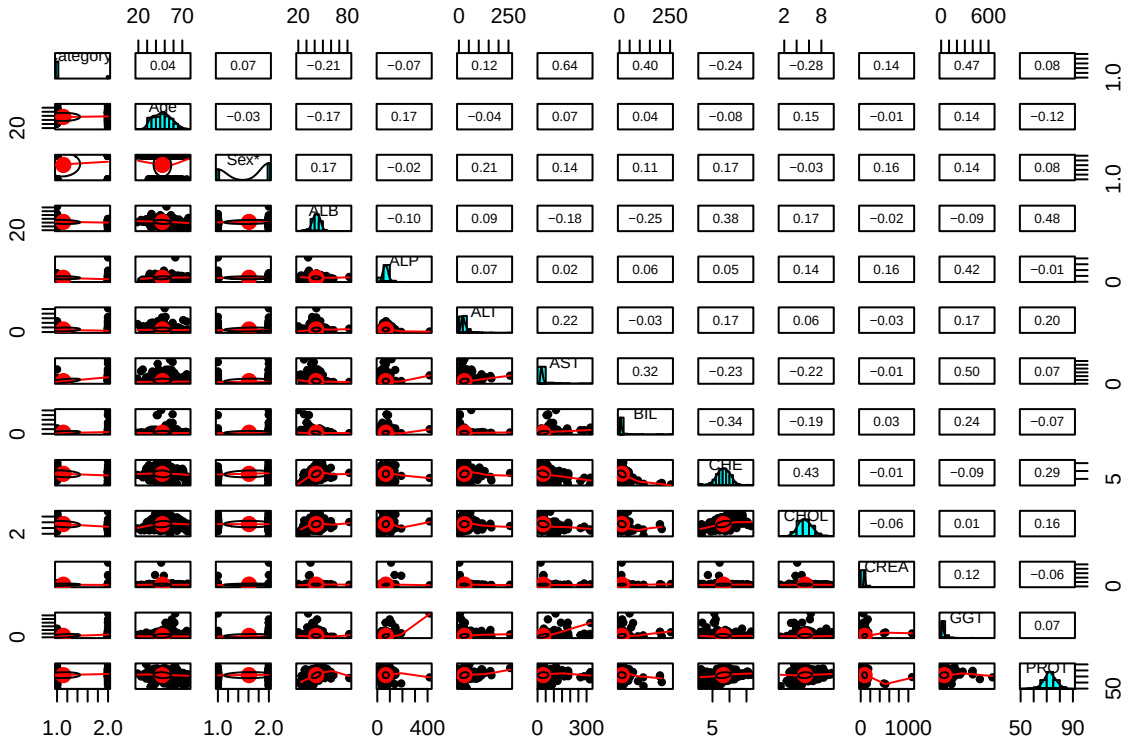
In summary, there aren't enough outliers from each variable to warrant removing, some of the outliers are relevant to particular classes, and our data has a class imbalance issue. Therefore we will keep the outliers in the data.

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)
```



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.

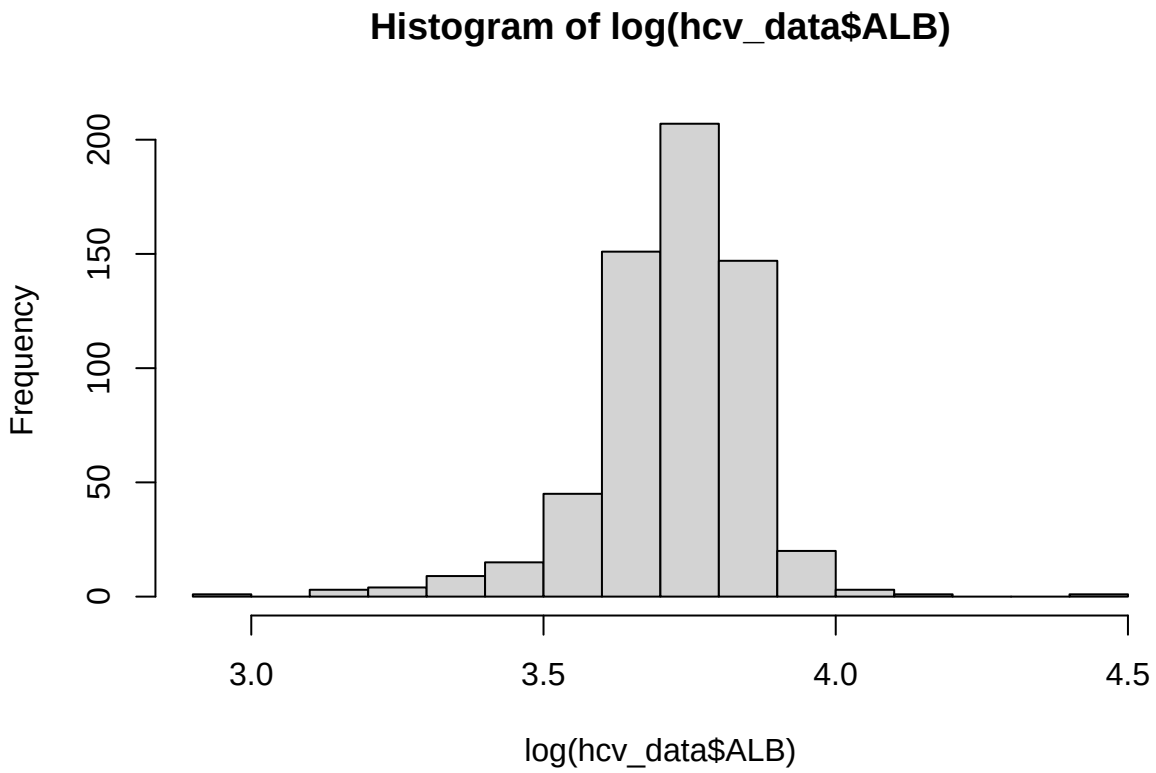
Cleaning and Shaping Data

```
# 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
```

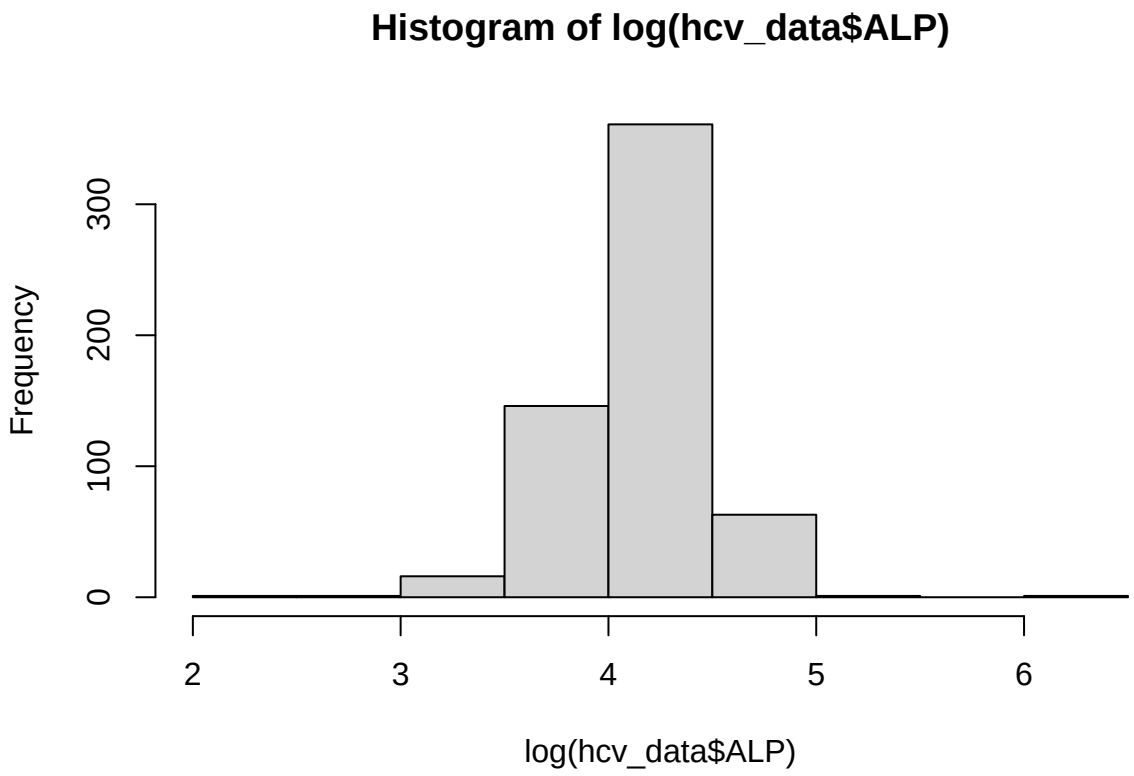
```
## Category      ALB      ALP      ALT      AST      BIL      CHE     CHOL
##        0        1       18        1        0        0        0       10
##     CREA      GGT
##        0        0
```

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.

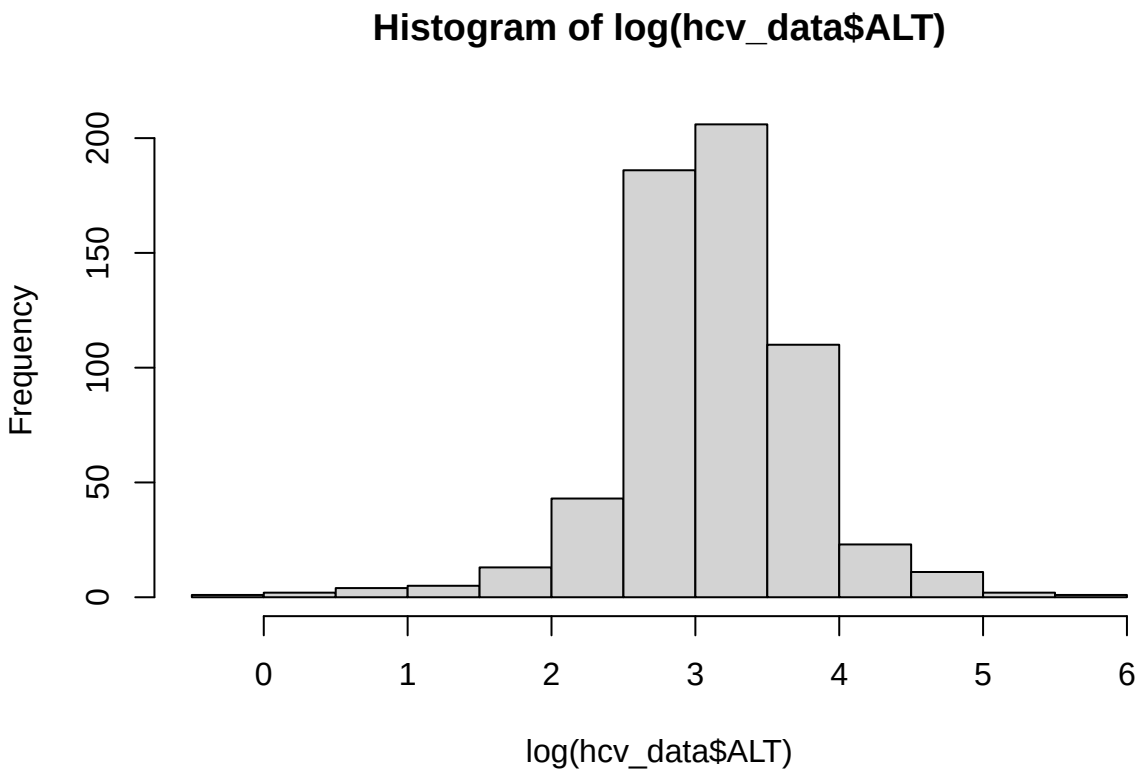
```r
# Let's see if we can transform features to look more normal before imputation
# ALB
hist(log(hcv_data$ALB))
```



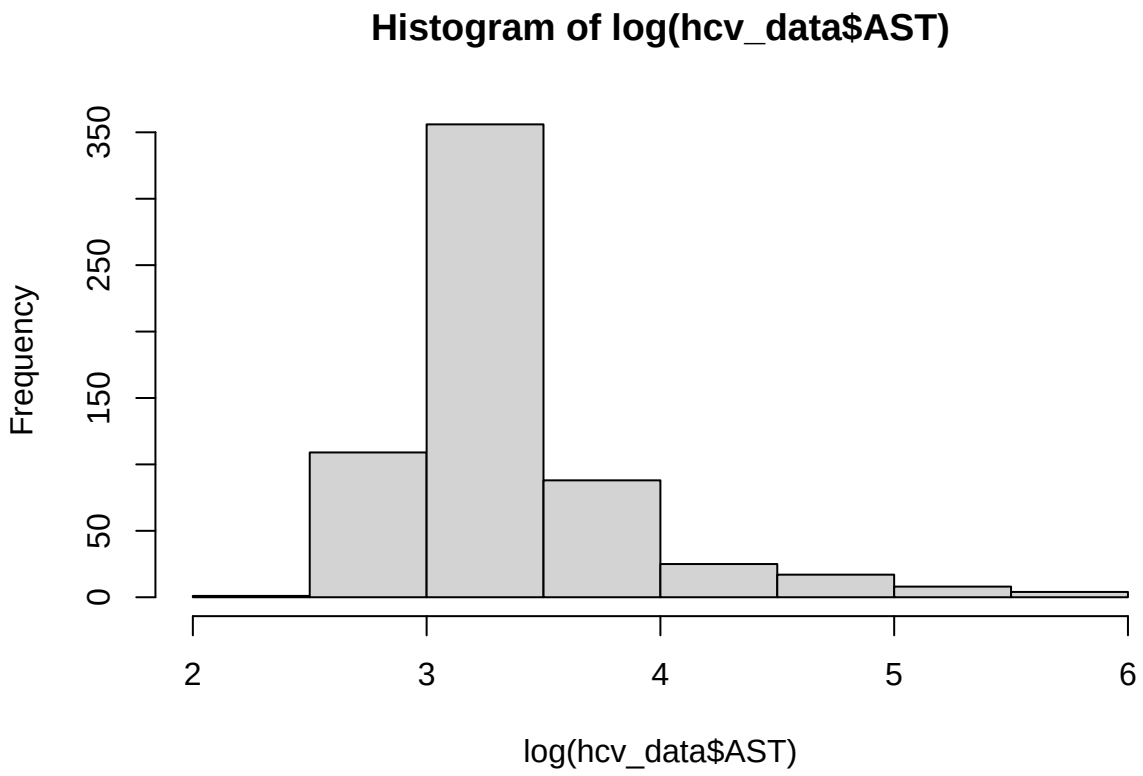

```r
hcv_data$ALB <- log(hcv_data$ALB)

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



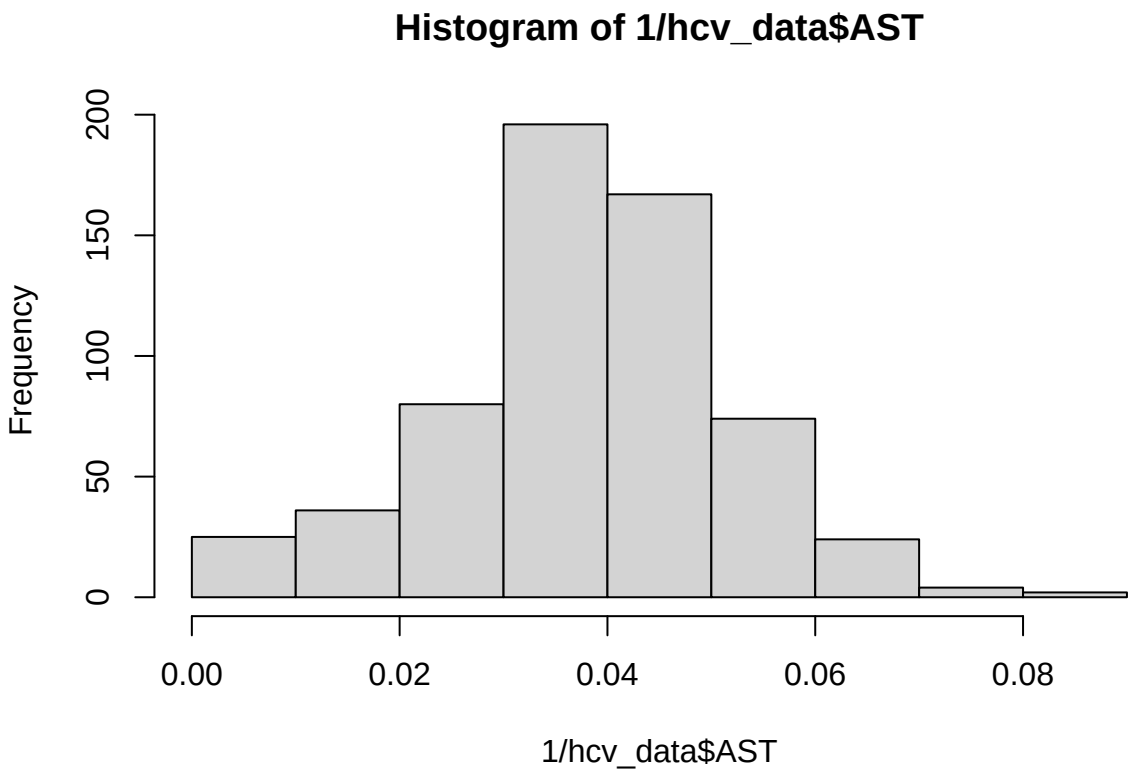

```r
hcv_data$ALP <- log(hcv_data$ALP)

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



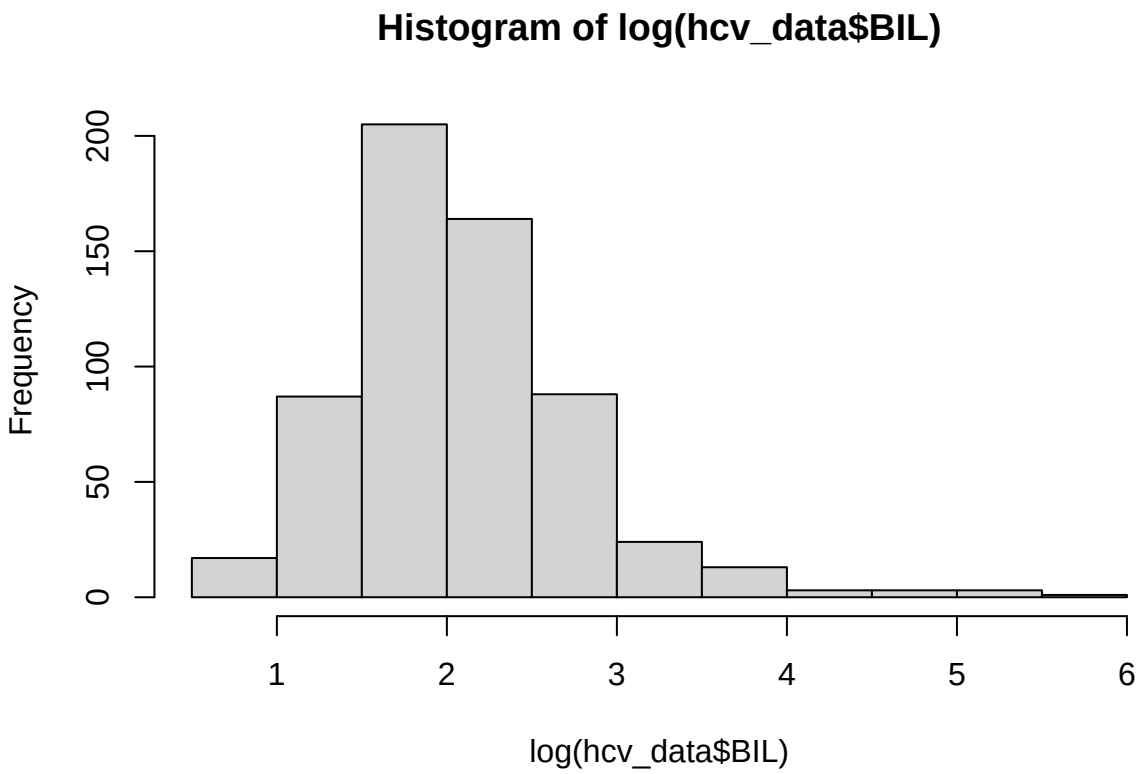

```r
hcv_data$ALT <- log(hcv_data$ALT)

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



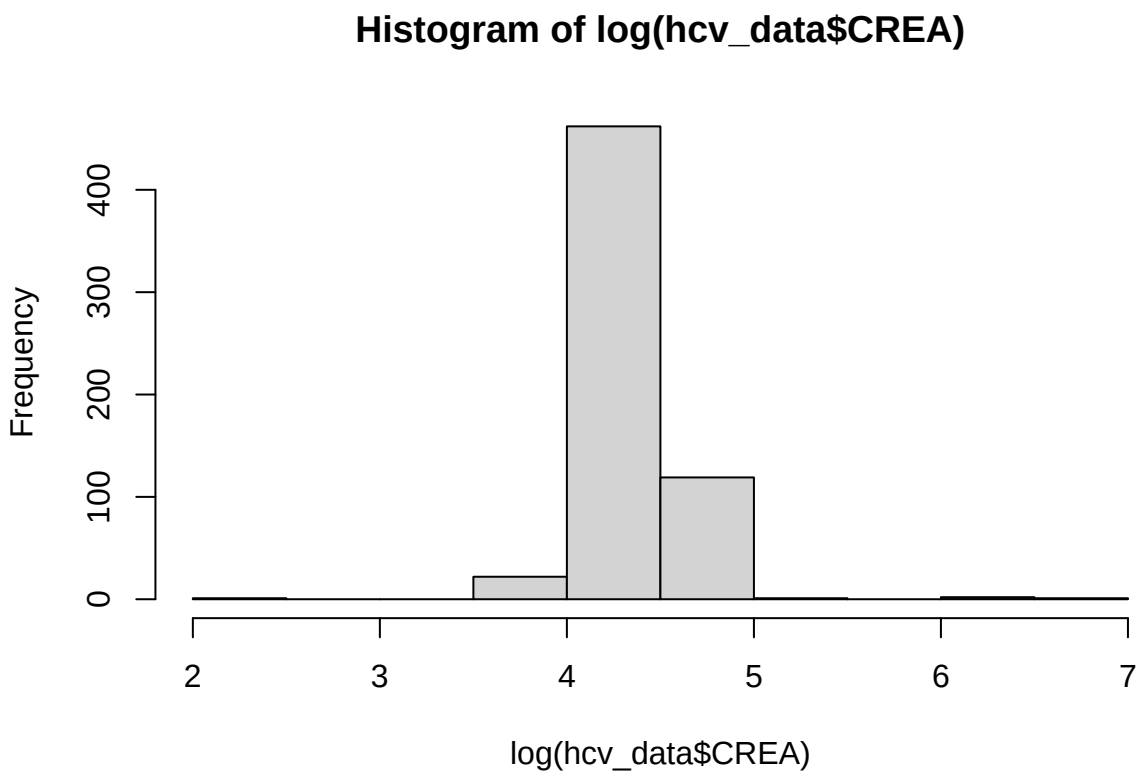

```
hist(1/hcv_data$AST)
```



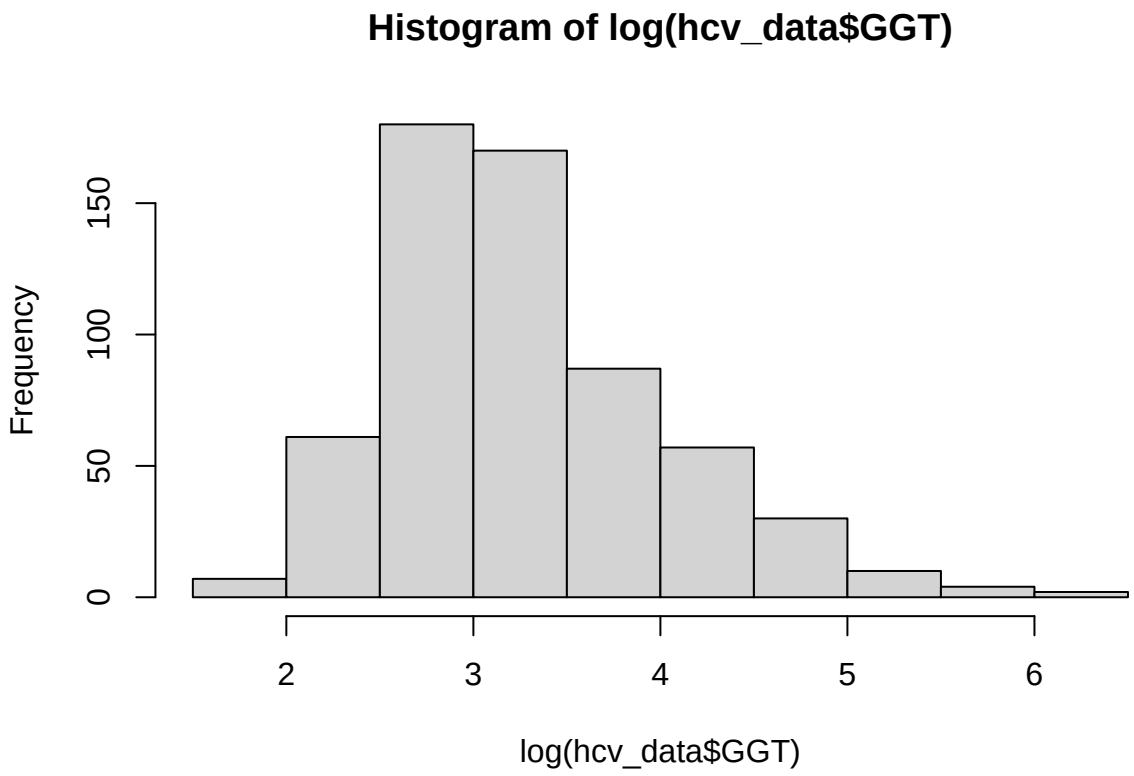

```
hcv_data$AST <- 1 / hcv_data$AST

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




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

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

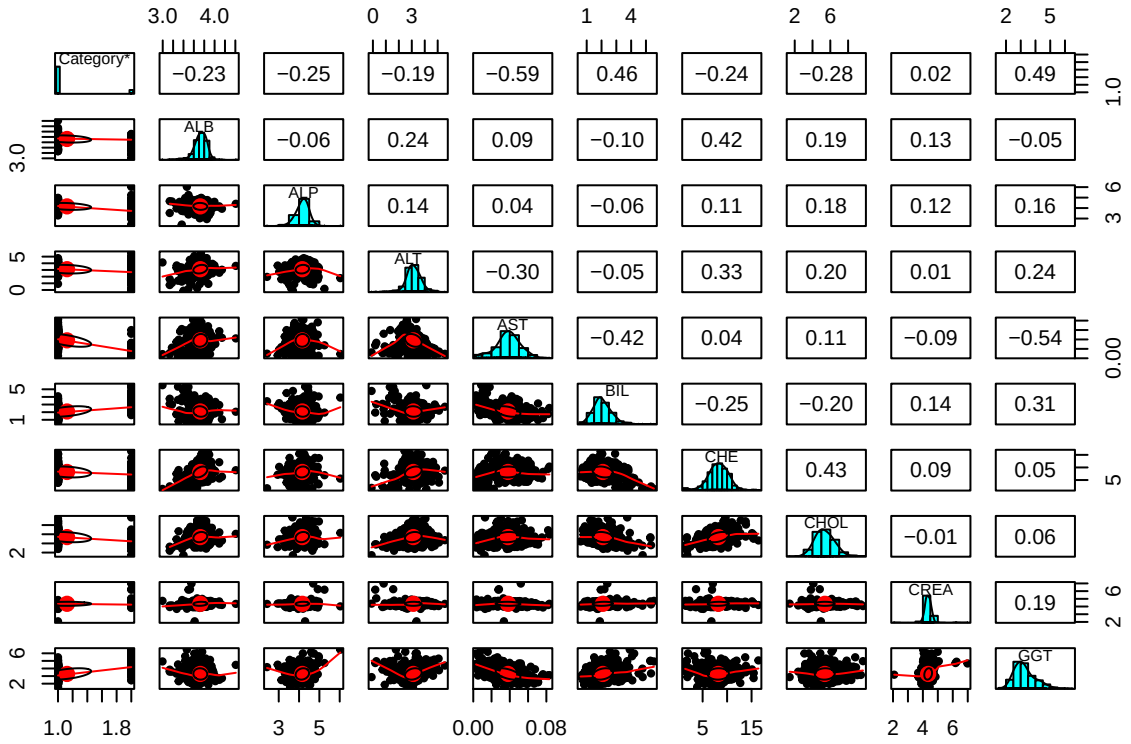



```r
hcv_data$CREA <- log(hcv_data$CREA)

# 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

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

# Randomize data
```

```r
hcv_data <- hcv_data[sample(nrow(hcv_data)), ]

# Create a stratified random train-validation split
split_indices <- createDataPartition(y = hcv_data$Category,
                                     p = 0.2,
                                     list = FALSE,
                                     times = 1 )


# Training data
train_data <- hcv_data[-split_indices, ]

# Validation data
validation_data <- hcv_data[split_indices, ]
```

The data has a significant class imbalance. About 90% of the data falls under "Blood Donor" with only ~ 10% falling under "Hepatitis". 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.

XGBoost

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

# Encoding for xgboost
train_data_xg <- train_data
validation_data_xg <- validation_data

train_data_xg$Category <- ifelse(train_data_xg$Category == "0=Blood Donor", 1, 0)
train_data_xg$Category <- as.numeric(as.factor(train_data_xg$Category)) - 1
validation_data_xg$Category <- ifelse(validation_data_xg$Category == "0=Blood Donor", 1, 0)
validation_data_xg$Category <- as.numeric(as.factor(validation_data_xg$Category)) -1

# Remove target variable from data to get features
features <- hcv_data[, !(colnames(data) %in% target_column)]

xgb_model <- xgboost(data = as.matrix(train_data_xg[, -which(names(train_data_xg) == target_column)]),
                     label = train_data_xg[[target_column]],
                     nrounds = 50,
                     max_depth = 3,
                     objective = "binary:logistic")
```

```
## [1]  train-logloss:0.467445
## [2]  train-logloss:0.338849
## [3]  train-logloss:0.256056
## [4]  train-logloss:0.198283
## [5]  train-logloss:0.156059
## [6]  train-logloss:0.122861
## [7]  train-logloss:0.100635
## [8]  train-logloss:0.082760
```

```
## [9]  train-logloss:0.069342
## [10] train-logloss:0.059716
## [11] train-logloss:0.052475
## [12] train-logloss:0.045708
## [13] train-logloss:0.041050
## [14] train-logloss:0.037014
## [15] train-logloss:0.033916
## [16] train-logloss:0.031320
## [17] train-logloss:0.028209
## [18] train-logloss:0.025876
## [19] train-logloss:0.024190
## [20] train-logloss:0.022871
## [21] train-logloss:0.021572
## [22] train-logloss:0.020294
## [23] train-logloss:0.019164
## [24] train-logloss:0.018057
## [25] train-logloss:0.017066
## [26] train-logloss:0.016138
## [27] train-logloss:0.015598
## [28] train-logloss:0.014778
## [29] train-logloss:0.014255
## [30] train-logloss:0.013872
## [31] train-logloss:0.013211
## [32] train-logloss:0.012685
## [33] train-logloss:0.012260
## [34] train-logloss:0.011854
## [35] train-logloss:0.011544
## [36] train-logloss:0.011320
## [37] train-logloss:0.011051
## [38] train-logloss:0.010843
## [39] train-logloss:0.010673
## [40] train-logloss:0.010460
## [41] train-logloss:0.010247
## [42] train-logloss:0.010095
## [43] train-logloss:0.009964
## [44] train-logloss:0.009807
## [45] train-logloss:0.009641
## [46] train-logloss:0.009484
## [47] train-logloss:0.009354
## [48] train-logloss:0.009209
## [49] train-logloss:0.009048
## [50] train-logloss:0.008920
```

```r
# Make predictions on the test data
xgb_predictions <- predict(xgb_model, newdata = as.matrix(validation_data_xg[, -which(names(validation_

# Convert probabilities to class labels based on the threshold
threshold <- 0.5
xgb_class_labels <- ifelse(xgb_predictions > threshold, "0=Blood Donor", "HCV")
xgb_class_labels <- as.factor(xgb_class_labels)

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

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      0=Blood Donor HCV
##   0=Blood Donor           106   1
##   HCV                       1  14
##
##                Accuracy : 0.9836
##                  95% CI : (0.942, 0.998)
##     No Information Rate : 0.877
##     P-Value [Acc > NIR] : 1.826e-05
##
##                   Kappa : 0.924
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9907
##             Specificity : 0.9333
##          Pos Pred Value : 0.9907
##          Neg Pred Value : 0.9333
##              Prevalence : 0.8770
##          Detection Rate : 0.8689
##    Detection Prevalence : 0.8770
##       Balanced Accuracy : 0.9620
##
##        'Positive' Class : 0=Blood Donor
##
```

```
# Save metrics
xg_eval1 <- confusion_matrix$byClass
xg_eval2 <- confusion_matrix$overall
xg_combined <- c(xg_eval1, xg_eval2)
names(xg_combined) <- c(names(xg_eval1), names(xg_eval2))

# ROC-AUC
roc_curve <- roc(validation_data$Category, xgb_predictions)
```

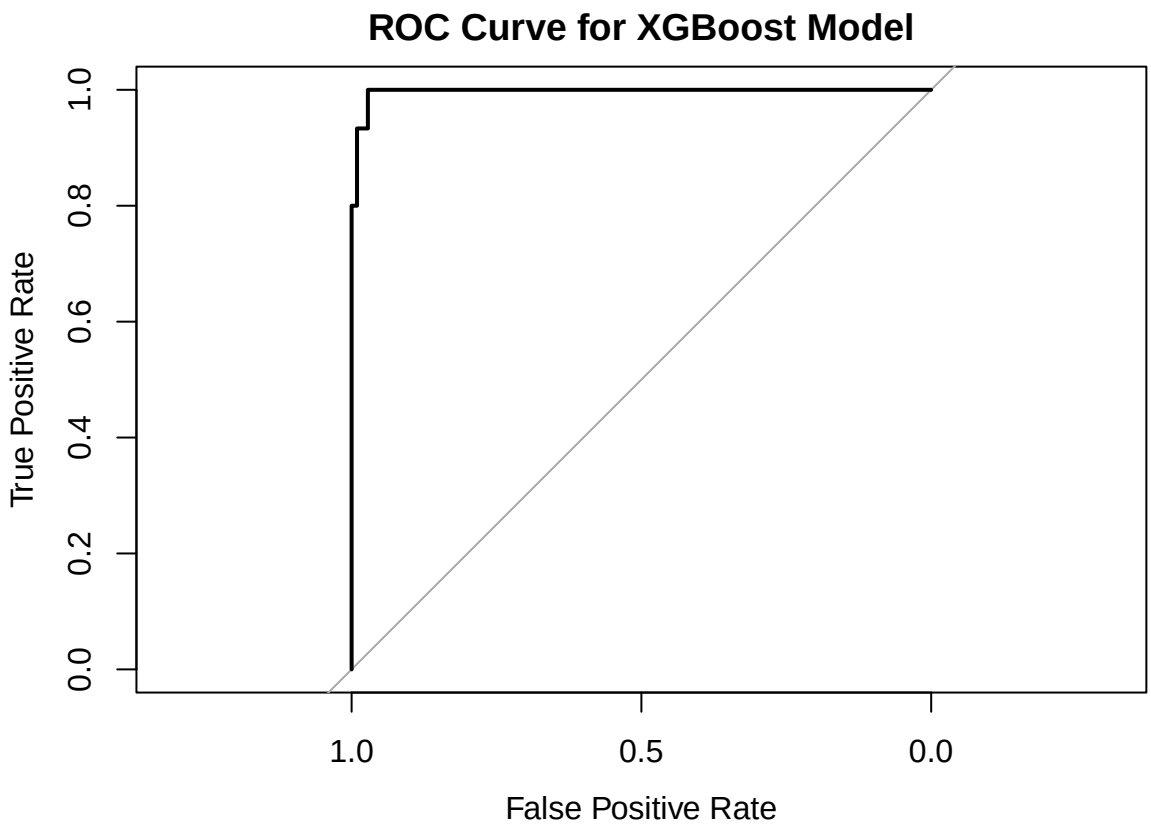
```
## Setting levels: control = 0=Blood Donor, case = HCV
```

```
## Setting direction: controls > cases
```

```
roc_auc <- auc(roc_curve)

cat("ROC-AUC:", roc_auc, "\n")
```

```
## ROC-AUC: 0.9968847
```

```
# Visualize ROC-AUC
plot(roc_curve, main = "ROC Curve for XGBoost Model",
     xlab = "False Positive Rate", ylab = "True Positive Rate")
```



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 use ROC-AUC, kappa, precision, recall, and F1 score to evaluate the model as these metrics are useful for imbalanced datasets. The model does well, with a kappa of 0.925 indicating a strong agreement between the predicted values and actual data. There is a 0.98 accuracy. Precision of 0.99, pos pred value of 0.99, and negative pred value of 0.93. This model does well at predicting the negative class which happens to be the smallest class. It correctly predicted 106 Blood Donors and incorrectly classified 1 as HCV. It correctly predicted 14 HCV and incorrectly classified 1 as blood donor. The model also has a ROC AUC value of 0.99 which means that it has very high predictive power. We care about this value because we want to ensure that the model has good discriminative power and can distinguish between positive and negative classes, especially given there is a class imbalance.

Random Forest

```
# 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
```

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

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      0=Blood Donor HCV
##   0=Blood Donor           105   4
##   HCV                       2  11
##
##                Accuracy : 0.9508
##                  95% CI : (0.896, 0.9817)
##     No Information Rate : 0.877
##     P-Value [Acc > NIR] : 0.005179
##
##                   Kappa : 0.7581
##
##  Mcnemar's Test P-Value : 0.683091
##
##             Sensitivity : 0.9813
##             Specificity : 0.7333
##          Pos Pred Value : 0.9633
##          Neg Pred Value : 0.8462
##              Prevalence : 0.8770
##          Detection Rate : 0.8607
##    Detection Prevalence : 0.8934
##       Balanced Accuracy : 0.8573
##
##        'Positive' Class : 0=Blood Donor
##
```

```r
# Save metrics
rf_eval1 <- confusion_matrix$byClass
rf_eval2 <- confusion_matrix$overall
rf_combined <- c(rf_eval1, rf_eval2)
names(rf_combined) <- c(names(rf_eval1), names(rf_eval2))

# Calculate ROC-AUC
roc_curve <- roc(validation_data$Category, as.numeric(predictions))
```

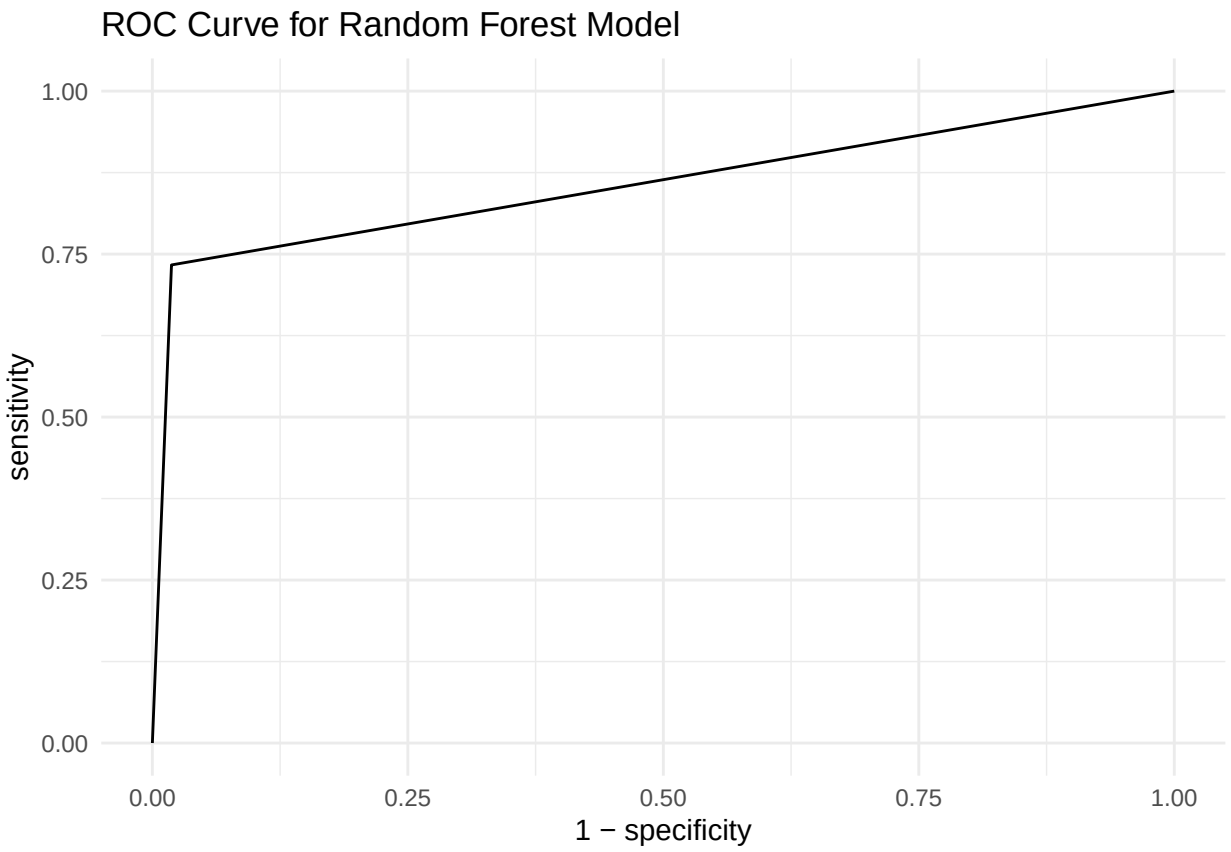
```
## Setting levels: control = 0=Blood Donor, case = HCV
```

```
## Setting direction: controls < cases
```

```r
roc_auc <- auc(roc_curve)

# Print ROC-AUC
cat("ROC-AUC:", roc_auc, "\n")
```

```
## ROC-AUC: 0.8573209
```

```r
# Visualize ROC curve
roc_plot <- ggroc(roc_curve, legacy.axes = TRUE)
roc_plot <- roc_plot + ggtitle("ROC Curve for Random Forest Model")
roc_plot <- roc_plot + theme_minimal()
roc_plot
```



The Random Forest 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 roc-auc, kappa, precision, recall, and F1 score to evaluate the model as these metrics are useful for imbalanced datasets. Using the holdout method, the accuracy of this model is 0.95, which indicates that the model might be performing well. We want to check other metrics however, given that most of the predictions were for one class. Looking at the confusion matrix, the model correctly predicted 105 Blood Donor patients and misclassified 4 as HCV. It also correctly predicted 11 HCV patients and misclassified 2 as Blood Donor. We chose to play around with the number of trees from the default value of 500 to 50, 100, 200, 300, and 400 given the small sample size. Too many trees could lead to high variance in predictions on a small dataset, and the model could start memorizing the training data. The accuracy of the model seems to stay the same at the tested number of trees. The precision which measures how many of the predicted positive instances were actually positive is 0.96. The recall, which measures how many actual positive instances were correctly predicted is 0.98. We have a kappa of 0.76 which is a good to substantial agreement between the predictions made by the model and the actual truth in the data. It's important to also consider the Neg Prediction Value or the True Negative Rate as the negative class has far fewer observations. We have a Neg Pred Value of 0.85. The roc-auc value of 0.85 is lower compared to the xgboost model indicating that it has less distinguishing power between positive and negative classes, although its still a fairly decent value.

SVM

```r
# 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 HCV
##   0=Blood Donor           107   1
##   HCV                       0  14
##
##                Accuracy : 0.9918
##                  95% CI : (0.9552, 0.9998)
##     No Information Rate : 0.877
##     P-Value [Acc > NIR] : 2.026e-06
##
##                   Kappa : 0.9609
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9333
##          Pos Pred Value : 0.9907
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8770
##          Detection Rate : 0.8770
##    Detection Prevalence : 0.8852
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : 0=Blood Donor
##
```

```r
# Save metrics
svm_eval1 <- confusion_matrix$byClass
svm_eval2 <- confusion_matrix$overall
svm_combined <- c(svm_eval1, svm_eval2)
names(svm_combined) <- c(names(svm_eval1), names(svm_eval2))
```

SVMs with appropriate kernel functions can be effective in capturing non-linearity. We can address class imbalances by adjusting class weights and 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. 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 performs well with an accuracy of

0.99, precision of 0.99, F1 of 0.99, recall of 1, positive pred value of 0.99 and neg pred value of 1. SVM predicts more accurately for the "Hepatitis" class than the other models. It also has the highest kappa of 0.96, indicating that there is excellent agreement between the model's predictions and the actual data. The model correctly classified 107 Blood Donor patients and misclassified 1 as HCV, and correctly predicted all 14 Hepatitis patients. Given that there is severe class imbalance, there still could be some overfitting of the data. Given that the SVM is performing well on its own, we do not see a need for bagging of the model. There was difficulty trying to obtain the ROC-AUC value for this model so it was not calculated.

Model Comparison

```
xg_df <- data.frame(Value = xg_combined)
dt_df <- data.frame(Value = rf_combined)
svm_df <- data.frame(Value = svm_combined)

# Combine evaluation data
combined_matrix <- cbind(xg_df, dt_df, svm_df)
colnames(combined_matrix) <- c("XGBoost", "Random Forest", "SVM")

combined_matrix
```

```
##                          XGBoost Random Forest          SVM
## Sensitivity          9.906542e-01   0.981308411 1.000000e+00
## Specificity          9.333333e-01   0.733333333 9.333333e-01
## Pos Pred Value       9.906542e-01   0.963302752 9.907407e-01
## Neg Pred Value       9.333333e-01   0.846153846 1.000000e+00
## Precision            9.906542e-01   0.963302752 9.907407e-01
## Recall               9.906542e-01   0.981308411 1.000000e+00
## F1                   9.906542e-01   0.972222222 9.953488e-01
## Prevalence           8.770492e-01   0.877049180 8.770492e-01
## Detection Rate       8.688525e-01   0.860655738 8.770492e-01
## Detection Prevalence 8.770492e-01   0.893442623 8.852459e-01
## Balanced Accuracy    9.619938e-01   0.857320872 9.666667e-01
## Accuracy             9.836066e-01   0.950819672 9.918033e-01
## Kappa                9.239875e-01   0.758096497 9.608724e-01
## AccuracyLower        9.420358e-01   0.896027384 9.551773e-01
## AccuracyUpper        9.980085e-01   0.981740397 9.997925e-01
## AccuracyNull         8.770492e-01   0.877049180 8.770492e-01
## AccuracyPValue       1.826111e-05   0.005178902 2.026130e-06
## McnemarPValue        1.000000e+00   0.683091398 1.000000e+00
```

The SVM model had a higher kappa of the three at 0.96, while XGBoost had 0.92 and Random Forest had 0.75. Although XGBoost has the highest accuracy, kappa is more important here as it is useful when dealing with imbalanced datasets. I was mostly concerned with the true negative prediction value as the negative class is much smaller, and SVM has a higher neg pred value. The F1, precision and recall scores for all three models were around the same. Although Random Forest had a high accuracy rate, it had a much lower kappa than the other two models, which could be due to overfitting of the data or the model memorizing the training data.

Ensemble

```
predictCategory <- function(new_data) {
  # XGBoost Model
  xgb_model <- xgboost(data = as.matrix(train_data_xg[, -which(names(train_data_xg) == target_column)])
                       label = train_data_xg[[target_column]],
                       nrounds = 50,
                       max_depth = 3,
                       objective = "binary:logistic")

  # Format validation data
  validation_data_xg <- new_data
  validation_data_xg$Category <- ifelse(validation_data_xg$Category == "0=Blood Donor", 1, 0)
  validation_data_xg$Category <- as.numeric(as.factor(validation_data_xg$Category)) -1

  # Make predictions on new data
  xgb_predictions <- predict(xgb_model, newdata = as.matrix(validation_data_xg[, -which(names(validatio
  # Convert probabilities to class labels based on the threshold
  threshold <- 0.5
  xgb_class_labels <- ifelse(xgb_predictions > threshold, "0=Blood Donor", "HCV")
  xgb_class_labels <- as.factor(xgb_class_labels)

  # Random Forest Model
  bagged_model <- randomForest(Category ~ ., data = train_data)
  # Make predictions on new data
  rf_predictions <- predict(bagged_model, newdata = new_data)

  # SVM model
  svm_model <- svm(Category ~ ., data = train_data, kernel = "radial")
  # Make predictions on validation data
  svm_predictions <- predict(svm_model, newdata = new_data)

  # Majority vote function
  majority_vote <- function(list1, list2, list3) {
    # Combine lists of predictions into a matrix
    all_predictions <- cbind(list1, list2, list3)

    # Calculate the mode for each row across the three lists
    majority_predictions <- apply(all_predictions, 1, function(row) {
      mode_value <- as.character(as.numeric(names(table(row)))[which.max(table(row))])
      return(mode_value)
    })

    return(majority_predictions)
  }

  ensemble_predictions <- majority_vote(xgb_class_labels, rf_predictions, svm_predictions)

  # 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)
```

```
## [1]  train-logloss:0.467445
## [2]  train-logloss:0.338849
## [3]  train-logloss:0.256056
## [4]  train-logloss:0.198283
## [5]  train-logloss:0.156059
## [6]  train-logloss:0.122861
## [7]  train-logloss:0.100635
## [8]  train-logloss:0.082760
## [9]  train-logloss:0.069342
## [10] train-logloss:0.059716
## [11] train-logloss:0.052475
## [12] train-logloss:0.045708
## [13] train-logloss:0.041050
## [14] train-logloss:0.037014
## [15] train-logloss:0.033916
## [16] train-logloss:0.031320
## [17] train-logloss:0.028209
## [18] train-logloss:0.025876
## [19] train-logloss:0.024190
## [20] train-logloss:0.022871
## [21] train-logloss:0.021572
## [22] train-logloss:0.020294
## [23] train-logloss:0.019164
## [24] train-logloss:0.018057
## [25] train-logloss:0.017066
## [26] train-logloss:0.016138
## [27] train-logloss:0.015598
## [28] train-logloss:0.014778
## [29] train-logloss:0.014255
## [30] train-logloss:0.013872
## [31] train-logloss:0.013211
## [32] train-logloss:0.012685
## [33] train-logloss:0.012260
## [34] train-logloss:0.011854
## [35] train-logloss:0.011544
## [36] train-logloss:0.011320
## [37] train-logloss:0.011051
## [38] train-logloss:0.010843
## [39] train-logloss:0.010673
## [40] train-logloss:0.010460
## [41] train-logloss:0.010247
## [42] train-logloss:0.010095
## [43] train-logloss:0.009964
## [44] train-logloss:0.009807
## [45] train-logloss:0.009641
## [46] train-logloss:0.009484
```

```
## [47] train-logloss:0.009354
## [48] train-logloss:0.009209
## [49] train-logloss:0.009048
## [50] train-logloss:0.008920
```

```
predictions <- ifelse(predictions == "1", "0=Blood Donor", "HCV")
predictions <- as.factor(predictions)
```

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

```
##                Reference
## Prediction      0=Blood Donor HCV
##   0=Blood Donor           107   2
##   HCV                       0  13
```

```
confusion_matrix$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            1.0000000            0.8666667            0.9816514
##       Neg Pred Value            Precision               Recall
##            1.0000000            0.9816514            1.0000000
##                   F1           Prevalence       Detection Rate
##            0.9907407            0.8770492            0.8770492
## Detection Prevalence    Balanced Accuracy
##            0.8934426            0.9333333
```

```
confusion_matrix$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.836066e-01   9.193655e-01   9.420358e-01   9.980085e-01   8.770492e-01
## AccuracyPValue  McnemarPValue
##   1.826111e-05   4.795001e-01
```

The ensemble has a kappa of 0.91, precision of 0.98, recall of 1, F1 of 0.99, and high pos pred values and neg pred values. It also correctly predicts 107 Blood Donor patients, and incorrectly predicts 2 to be HCV. It correctly predicts all HCV patients. The SVM model still performs better than the ensemble. The ensemble performs well, but it isn't necessary compared to using a well performing individual model like SVM.

Using Ensemble to Predict New Data

```
# Assuming new data has been transformed and cleaned
new_data <- data.frame(ALB = 3.1,
                       ALP = 3.6,
                       ALT = 0.91,
                       AST = 0.02,
                       BIL = 4.01,
                       CHE = 3.04,
                       CHOL = 5.00,
```

```
                          CREA = 5.64,
                          GGT = 4.78,
                          Category = "")
# Prediction
prediction <- predictCategory(new_data)
```

```
## [1]  train-logloss:0.467445
## [2]  train-logloss:0.338849
## [3]  train-logloss:0.256056
## [4]  train-logloss:0.198283
## [5]  train-logloss:0.156059
## [6]  train-logloss:0.122861
## [7]  train-logloss:0.100635
## [8]  train-logloss:0.082760
## [9]  train-logloss:0.069342
## [10] train-logloss:0.059716
## [11] train-logloss:0.052475
## [12] train-logloss:0.045708
## [13] train-logloss:0.041050
## [14] train-logloss:0.037014
## [15] train-logloss:0.033916
## [16] train-logloss:0.031320
## [17] train-logloss:0.028209
## [18] train-logloss:0.025876
## [19] train-logloss:0.024190
## [20] train-logloss:0.022871
## [21] train-logloss:0.021572
## [22] train-logloss:0.020294
## [23] train-logloss:0.019164
## [24] train-logloss:0.018057
## [25] train-logloss:0.017066
## [26] train-logloss:0.016138
## [27] train-logloss:0.015598
## [28] train-logloss:0.014778
## [29] train-logloss:0.014255
## [30] train-logloss:0.013872
## [31] train-logloss:0.013211
## [32] train-logloss:0.012685
## [33] train-logloss:0.012260
## [34] train-logloss:0.011854
## [35] train-logloss:0.011544
## [36] train-logloss:0.011320
## [37] train-logloss:0.011051
## [38] train-logloss:0.010843
## [39] train-logloss:0.010673
## [40] train-logloss:0.010460
## [41] train-logloss:0.010247
## [42] train-logloss:0.010095
## [43] train-logloss:0.009964
## [44] train-logloss:0.009807
## [45] train-logloss:0.009641
## [46] train-logloss:0.009484
## [47] train-logloss:0.009354
```

```
## [48] train-logloss:0.009209
## [49] train-logloss:0.009048
## [50] train-logloss:0.008920
```

```
prediction <- ifelse(prediction == "1", "0=Blood Donor", "HCV")
prediction
```

```
##     1
## "HCV"
```

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 predicted the patient to have HCV.