Survival Analysis of Cirrhosis Patients Using Classification Machine Learning Models: A Comprehensive Data Analysis

DA5030

Anusha Devi Doddi

Spring 2024

Business Understanding

Cirrhosis is a chronic liver disease often associated with alcoholism, viral hepatitis B and C, and fatty liver disease, among other causes. It is characterized by the replacement of liver tissue by fibrosis, scar tissue, and regenerative nodules, leading to a progressive loss of liver function. Early detection and accurate prognosis of survival rates are crucial for effective management and treatment planning. The healthcare industry requires robust tools for predicting the survival chances of cirrhosis patients. Accurate predictions can assist healthcare providers in making informed decisions about treatment options, prioritizing patients based on severity, and managing healthcare resources more effectively.

Aim of the Analysis

The aim of this analysis is to develop predictive models that can accurately predict the survival status of patients with cirrhosis based on a set of clinical and laboratory variables. The analysis will involve data exploration, preprocessing, model construction, and evaluation to identify the most effective predictive models for this task. The models will be evaluated based on their accuracy, precision, recall, F1 score, and other relevant metrics to assess their performance in predicting the survival status of cirrhosis patients.

Source of Data

The data used in this analysis is sourced from a Mayo Clinic study on primary biliary cirrhosis of the liver. The dataset contains information about patients with cirrhosis, including demographic details, clinical features, and laboratory test results. The dataset includes both numeric and categorical variables, making it suitable for developing classification models to predict the survival status of patients with cirrhosis. The link to the dataset is here.

Data Acquisition

Load necessary libraries

Load the Data from CSV File

```
##   ID N_Days Status           Drug   Age Sex Ascites Hepatomegaly Spiders Edema
## 1  1    400      D D-penicillamine 21464   F       Y            Y       Y     Y
## 2  2   4500      C D-penicillamine 20617   F       N            Y       Y     N
```

```
## 3  3   1012      D D-penicillamine 25594   M       N            N       N     S
## 4  4   1925      D D-penicillamine 19994   F       N            Y       Y     S
## 5  5   1504     CL         Placebo 13918   F       N            Y       Y     N
## 6  6   2503      D         Placebo 24201   F       N            Y       N     N
##   Bilirubin Cholesterol Albumin Copper Alk_Phos   SGOT Tryglicerides Platelets
## 1      14.5         261    2.60    156   1718.0 137.95           172       190
## 2       1.1         302    4.14     54   7394.8 113.52            88       221
## 3       1.4         176    3.48    210    516.0  96.10            55       151
## 4       1.8         244    2.54     64   6121.8  60.63            92       183
## 5       3.4         279    3.53    143    671.0 113.15            72       136
## 6       0.8         248    3.98     50    944.0  93.00            63        NA
##   Prothrombin Stage
## 1        12.2     4
## 2        10.6     3
## 3        12.0     4
## 4        10.3     4
## 5        10.9     3
## 6        11.0     3
```

- The data is about patients with cirrhosis of the liver. It is sourced from a Mayo Clinic study on primary biliary cirrhosis of the liver.
- It is loaded into a data frame named `cirrhosis`. It is a csv file with 418 rows and 20 columns.

Data Exploration

Summary and Structure of the Data

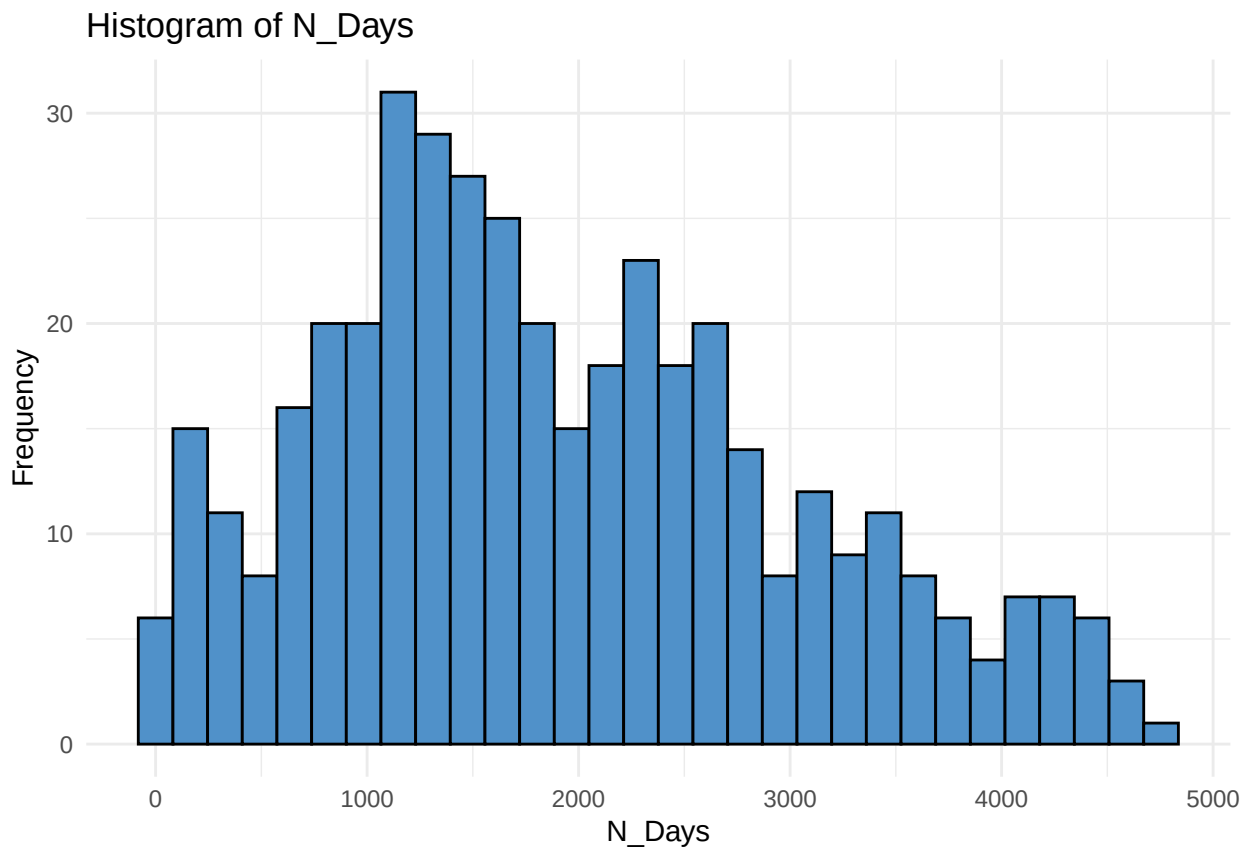
```
##        ID            N_Days        Status              Drug
##  Min.   :  1.0   Min.   :  41   Length:418         Length:418
##  1st Qu.:105.2   1st Qu.:1093   Class :character   Class :character
##  Median :209.5   Median :1730   Mode  :character   Mode  :character
##  Mean   :209.5   Mean   :1918
##  3rd Qu.:313.8   3rd Qu.:2614
##  Max.   :418.0   Max.   :4795
##
##       Age            Sex              Ascites          Hepatomegaly
##  Min.   : 9598   Length:418         Length:418         Length:418
##  1st Qu.:15644   Class :character   Class :character   Class :character
##  Median :18628   Mode  :character   Mode  :character   Mode  :character
##  Mean   :18533
##  3rd Qu.:21272
##  Max.   :28650
##
##    Spiders             Edema             Bilirubin       Cholesterol
##  Length:418         Length:418         Min.   : 0.300   Min.   : 120.0
##  Class :character   Class :character   1st Qu.: 0.800   1st Qu.: 249.5
##  Mode  :character   Mode  :character   Median : 1.400   Median : 309.5
##                                        Mean   : 3.221   Mean   : 369.5
##                                        3rd Qu.: 3.400   3rd Qu.: 400.0
##                                        Max.   :28.000   Max.   :1775.0
##                                                         NA's   :134
##     Albumin          Copper          Alk_Phos            SGOT
```

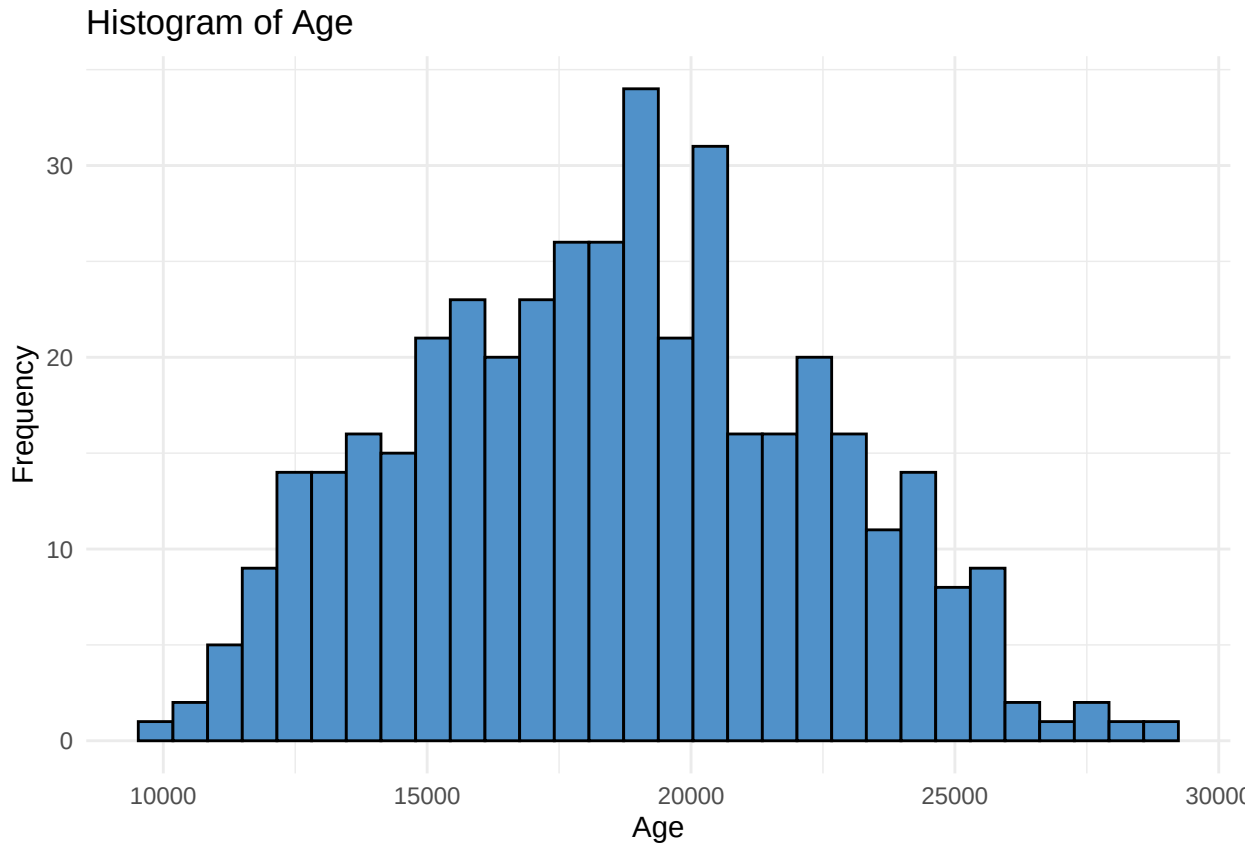
```
##  Min.   :1.960   Min.   :  4.00   Min.   :  289.0   Min.   : 26.35
##  1st Qu.:3.243   1st Qu.: 41.25   1st Qu.:  871.5   1st Qu.: 80.60
##  Median :3.530   Median : 73.00   Median : 1259.0   Median :114.70
##  Mean   :3.497   Mean   : 97.65   Mean   : 1982.7   Mean   :122.56
##  3rd Qu.:3.770   3rd Qu.:123.00   3rd Qu.: 1980.0   3rd Qu.:151.90
##  Max.   :4.640   Max.   :588.00   Max.   :13862.4   Max.   :457.25
##                  NA's   :108      NA's   :106       NA's   :106
##  Tryglicerides      Platelets      Prothrombin        Stage
##  Min.   : 33.00   Min.   : 62.0   Min.   : 9.00   Min.   :1.000
##  1st Qu.: 84.25   1st Qu.:188.5   1st Qu.:10.00   1st Qu.:2.000
##  Median :108.00   Median :251.0   Median :10.60   Median :3.000
##  Mean   :124.70   Mean   :257.0   Mean   :10.73   Mean   :3.024
##  3rd Qu.:151.00   3rd Qu.:318.0   3rd Qu.:11.10   3rd Qu.:4.000
##  Max.   :598.00   Max.   :721.0   Max.   :18.00   Max.   :4.000
##  NA's   :136      NA's   :11      NA's   :2       NA's   :6
```

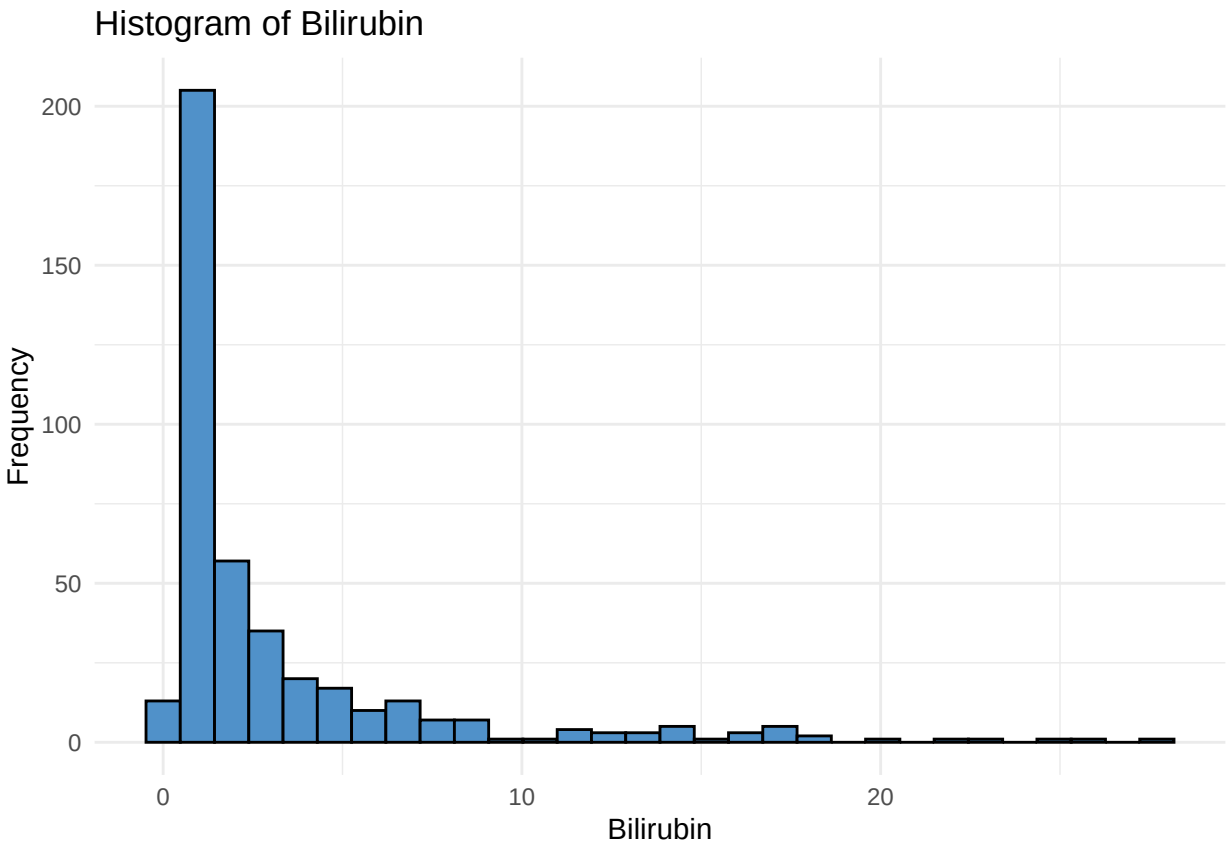
```
## 'data.frame':    418 obs. of  20 variables:
##  $ ID           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ N_Days       : int  400 4500 1012 1925 1504 2503 1832 2466 2400 51 ...
##  $ Status       : chr  "D" "C" "D" "D" ...
##  $ Drug         : chr  "D-penicillamine" "D-penicillamine" "D-penicillamine" "D-penicillamine" ...
##  $ Age          : int  21464 20617 25594 19994 13918 24201 20284 19379 15526 25772 ...
##  $ Sex          : chr  "F" "F" "M" "F" ...
##  $ Ascites      : chr  "Y" "N" "N" "N" ...
##  $ Hepatomegaly : chr  "Y" "Y" "N" "Y" ...
##  $ Spiders      : chr  "Y" "Y" "N" "Y" ...
##  $ Edema        : chr  "Y" "N" "S" "S" ...
##  $ Bilirubin    : num  14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
##  $ Cholesterol  : int  261 302 176 244 279 248 322 280 562 200 ...
##  $ Albumin      : num  2.6 4.14 3.48 2.54 3.53 3.98 4.09 4 3.08 2.74 ...
##  $ Copper       : int  156 54 210 64 143 50 52 52 79 140 ...
##  $ Alk_Phos     : num  1718 7395 516 6122 671 ...
##  $ SGOT         : num  137.9 113.5 96.1 60.6 113.2 ...
##  $ Tryglicerides: int  172 88 55 92 72 63 213 189 88 143 ...
##  $ Platelets    : int  190 221 151 183 136 NA 204 373 251 302 ...
##  $ Prothrombin  : num  12.2 10.6 12 10.3 10.9 11 9.7 11 11 11.5 ...
##  $ Stage        : int  4 3 4 4 3 3 3 3 2 4 ...
```

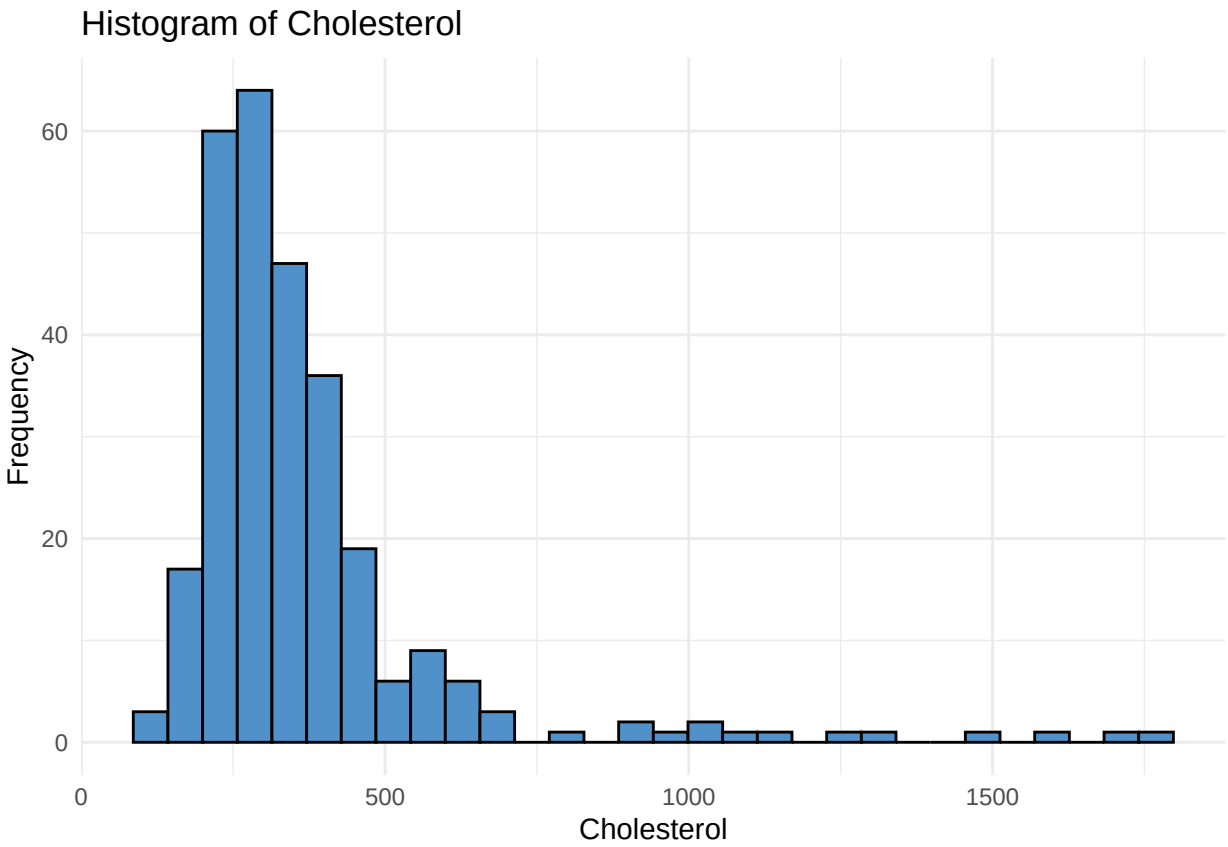
- The 'ID' column is removed as it is not relevant for analysis.
- The dataset contains 418 observations and 19 variables.
- The data has a mix of numeric and categorical variables.
- The 'Status' column is the target variable, indicating the survival status of the patients.

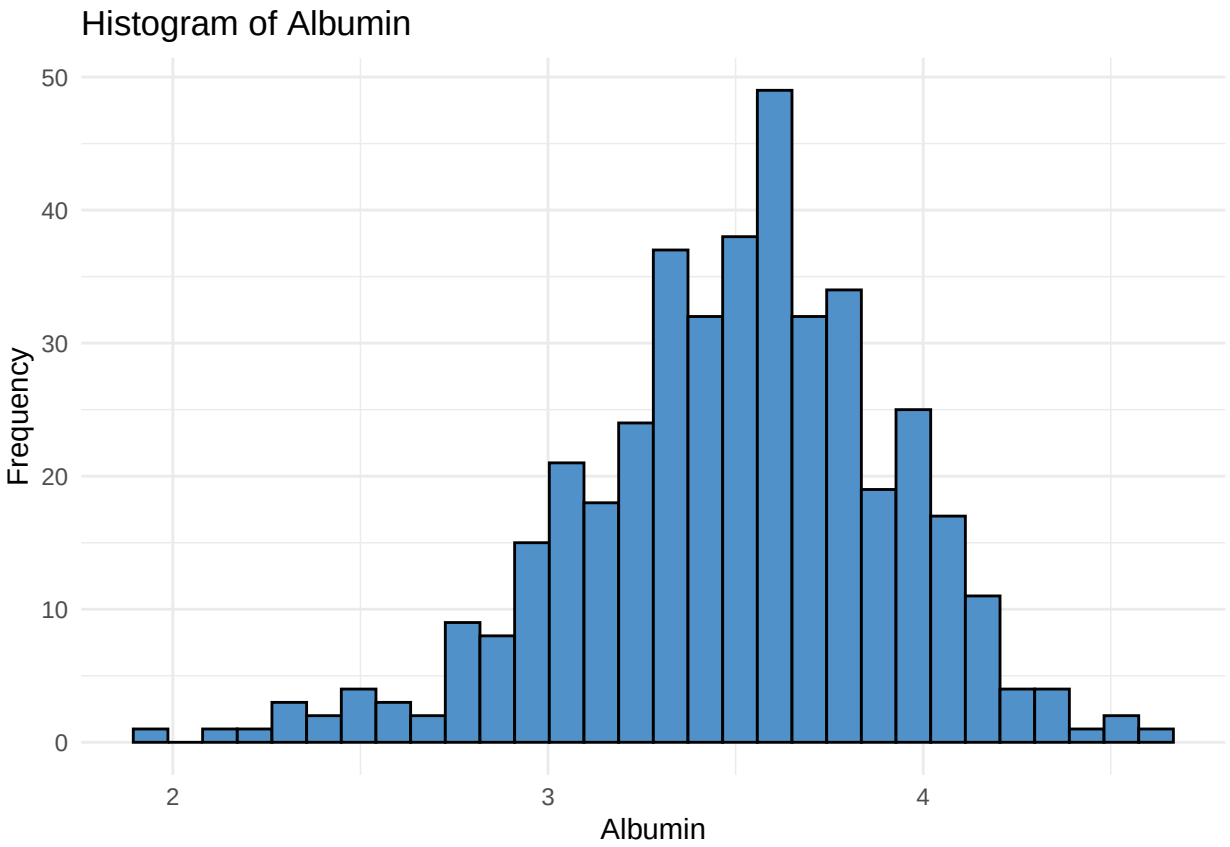
Data Visualization using Histograms

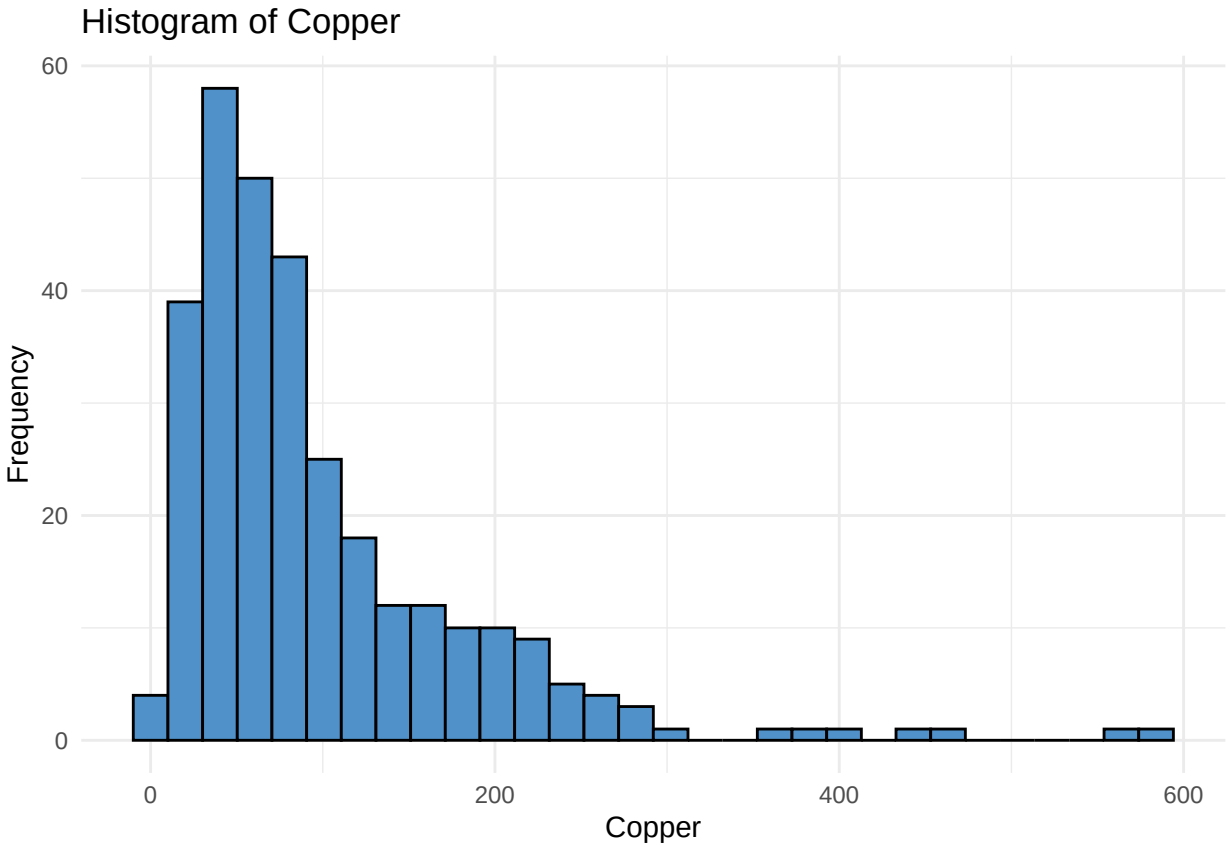


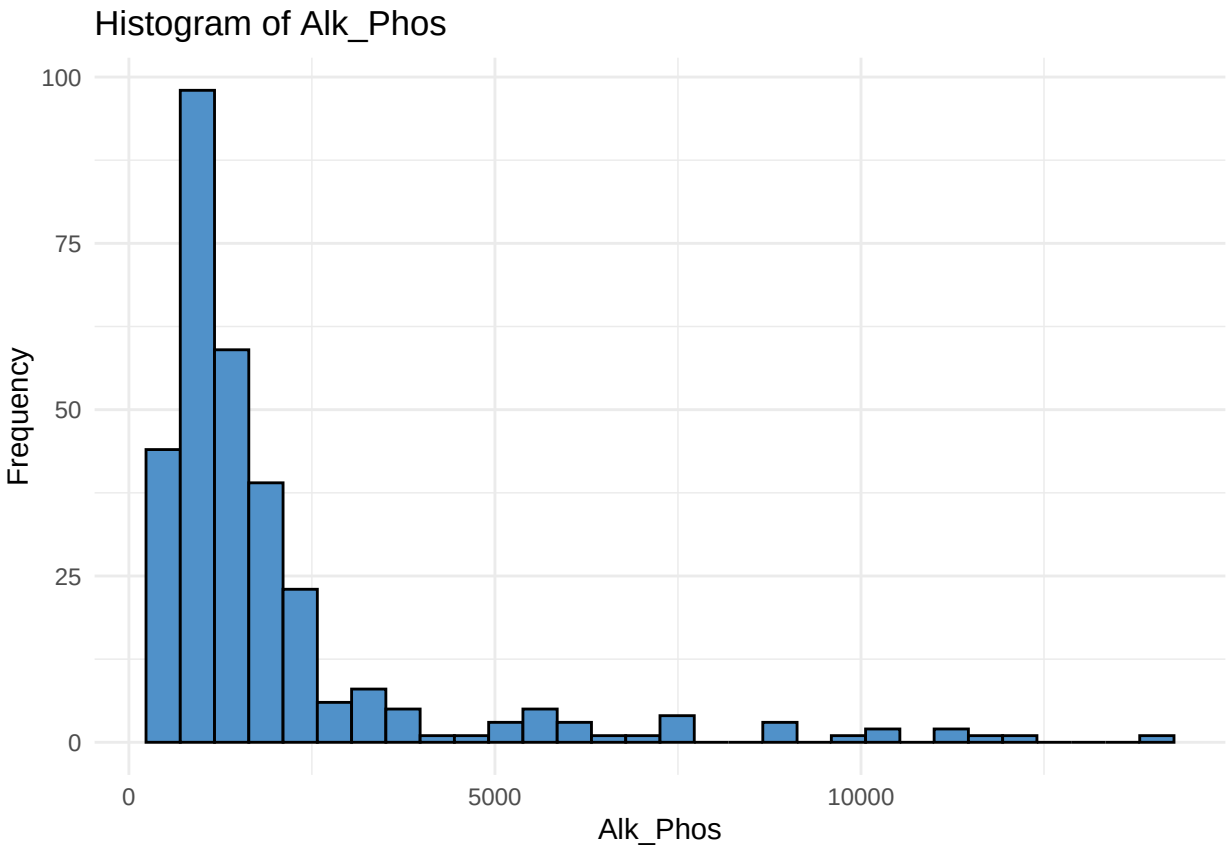


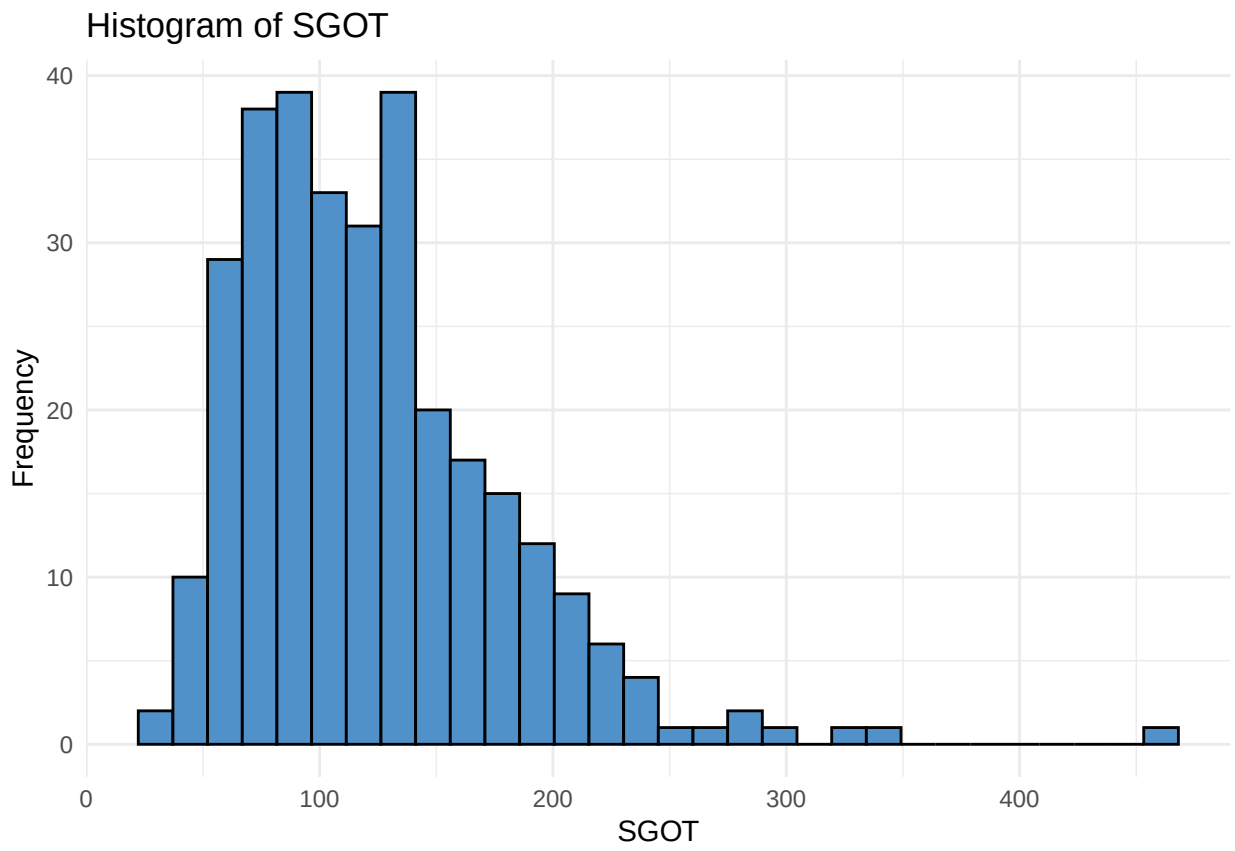

Histogram of Bilirubin
Frequency
200
150
100
50
0
0
10
20
Bilirubin

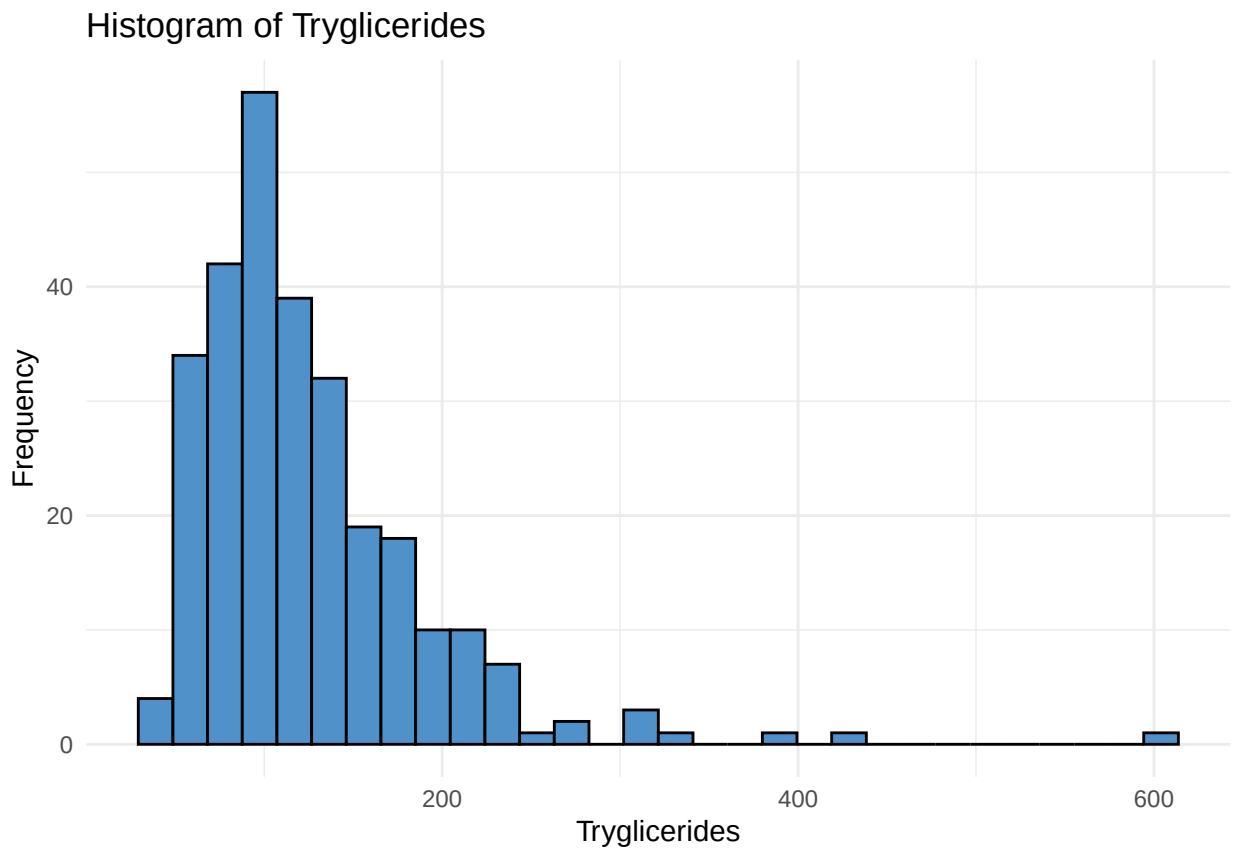

Histogram of Cholesterol
Frequency
60
40
20
0
0
500
1000
1500
Cholesterol

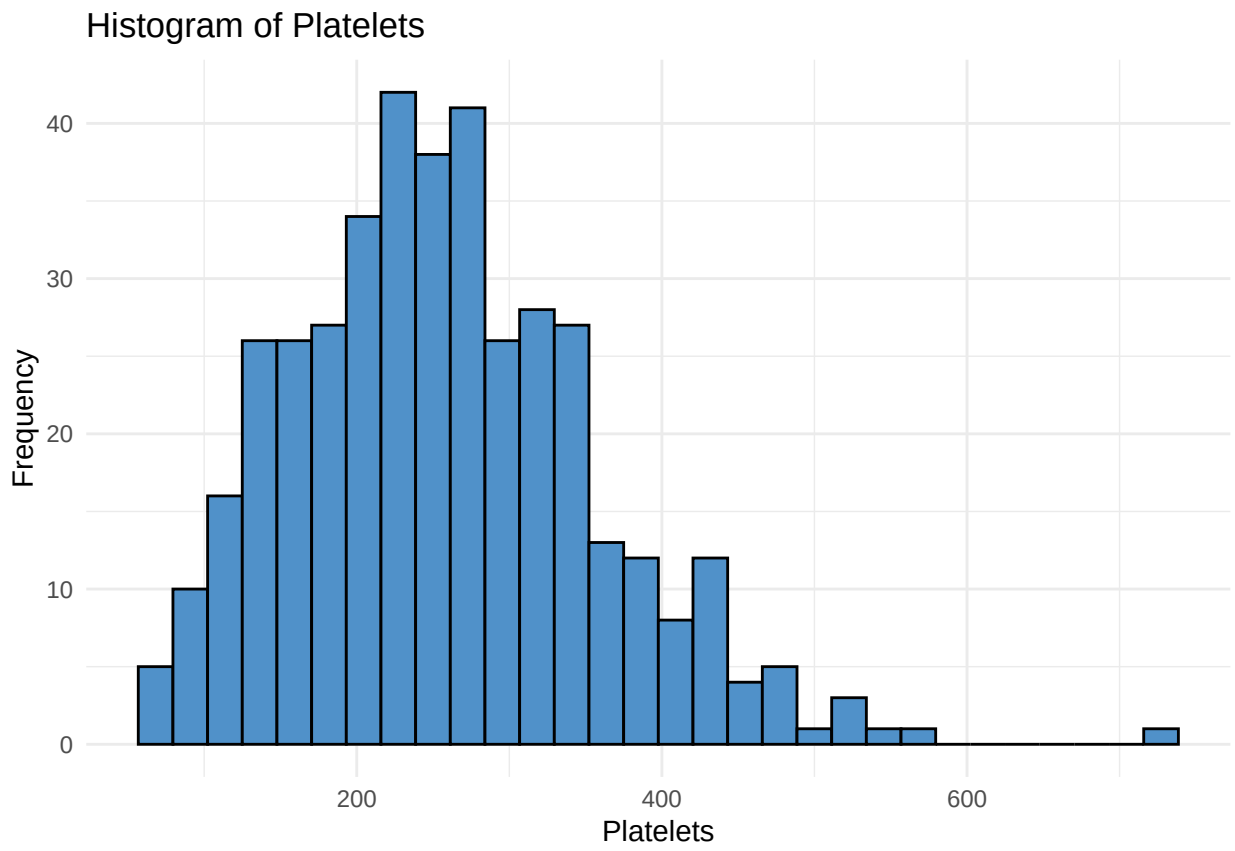

Histogram of Albumin
Frequency
50
40
30
20
10
0
2
3
4
Albumin

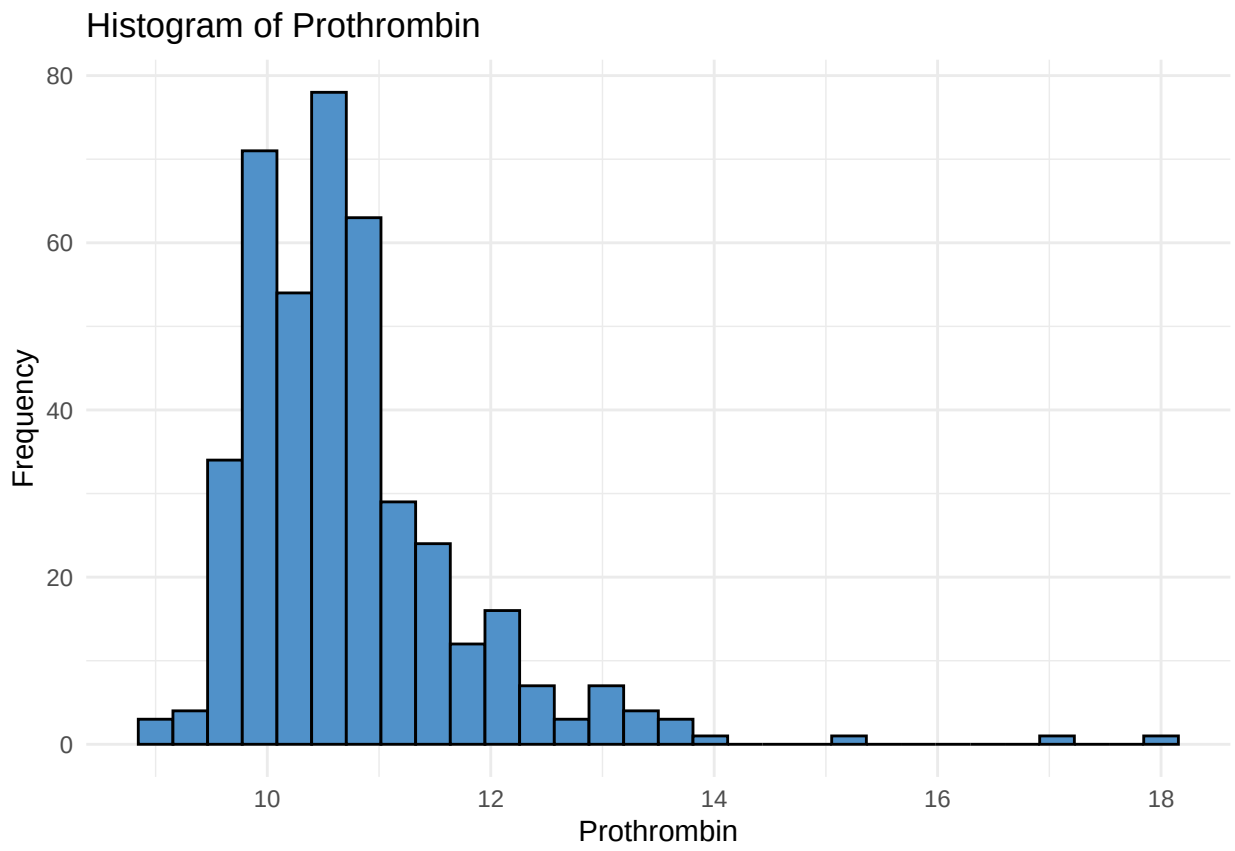

Histogram of Copper
Frequency
60
40
20
0
0
200
400
600
Copper

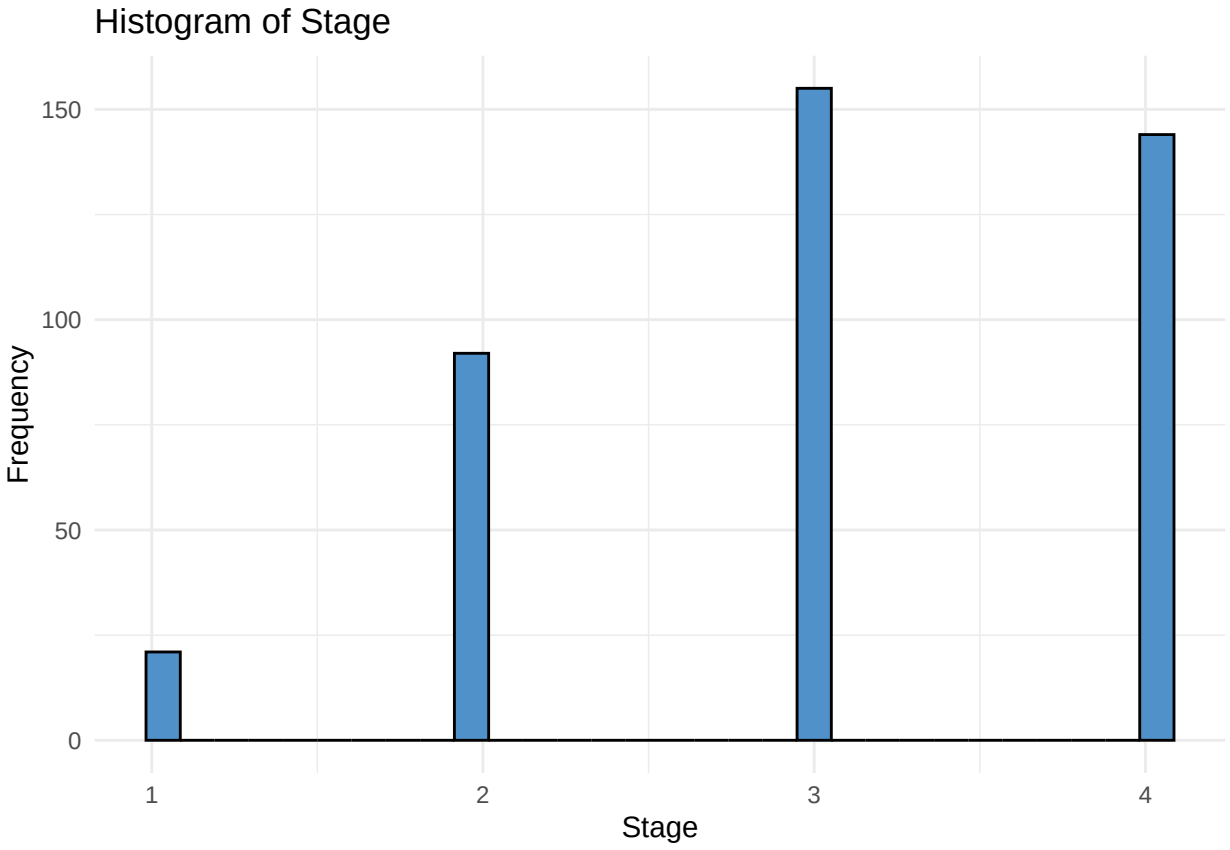

Histogram of Alk_Phos
Frequency
100
75
50
25
0
0
5000
10000
Alk_Phos


Histogram of SGOT
Frequency
SGOT
0
10
20
30
40
0
100
200
300
400


Histogram of Trygliceride
Frequency
40
20
0
200
400
600
Trygliceride

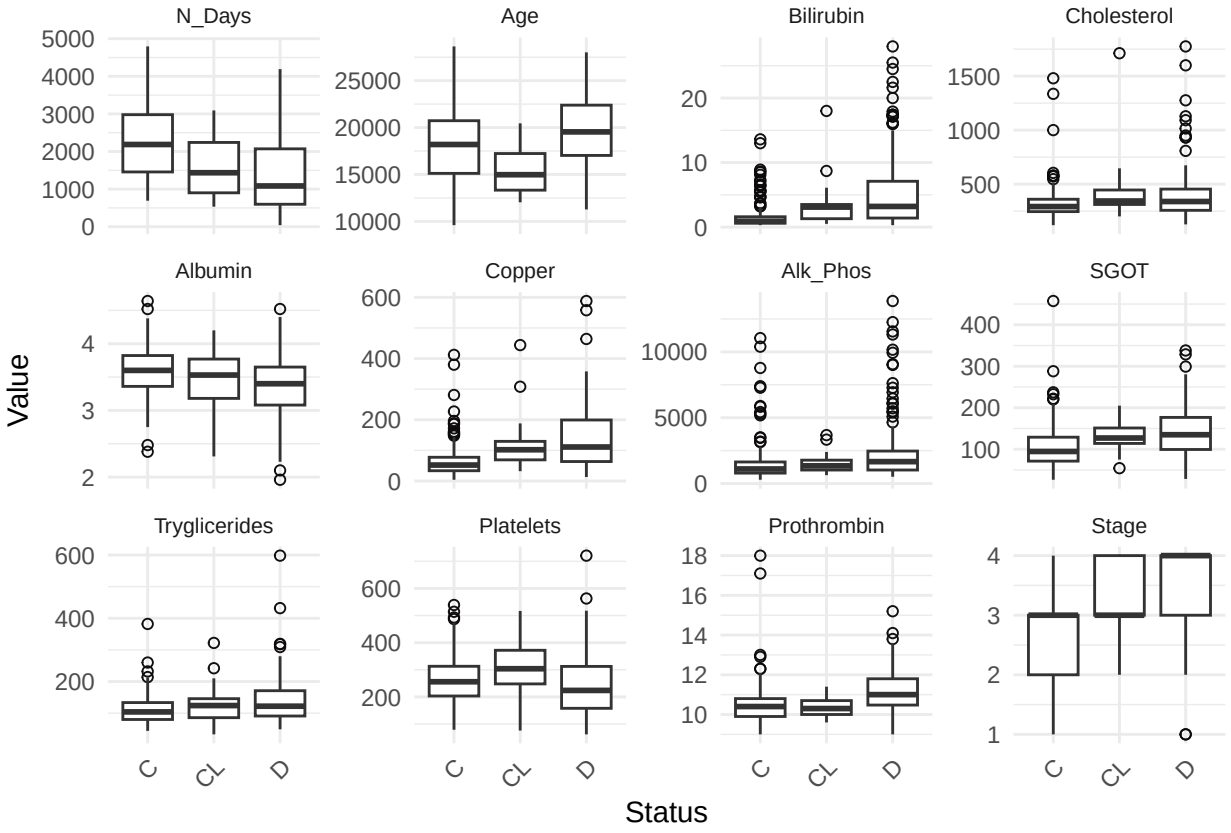

Histogram of Platelets
Frequency
40
30
20
10
0
200
400
600
Platelets


Histogram of Prothrombin
Frequency
80
60
40
20
0
10
12
14
16
18
Prothrombin



- The histograms display a range of distribution shapes for different variables, suggesting a mix of skewed and normally distributed data.
- The histograms show a variety of distribution shapes, indicating a mix of skewed and normally distributed data across the different variables.
- The spread of the histograms gives an indication of the range of each variable. Wide spreads can suggest high variability, while narrow spreads may indicate that values are concentrated around a particular number.
- The peak of each histogram shows the mode, providing insight into the most common values within a dataset.
- This suggests that normalization or standardization may be necessary to align them on a common scale for analytical purposes.
- Depending on the shape of the histograms, certain variables might benefit from transformations to normalize the data, such as logarithmic or square root transformations for right-skewed data.

Detecting Outliers using Boxplots for continuous variables



- The boxplots show the distribution of numeric features by the survival status of the patients.
- Several features display a significant number of outliers, suggesting variations in the dataset that may affect model accuracy. The boxplots show that the distributions of numeric variables vary greatly, indicating diverse patient characteristics.
- Many boxplots exhibit skewness either to the right or left, indicating that most numeric variables are not symmetrically distributed. The medians of the boxplots vary across different levels of the 'Status' target variable, highlighting their potential impact on patient survival outcomes.

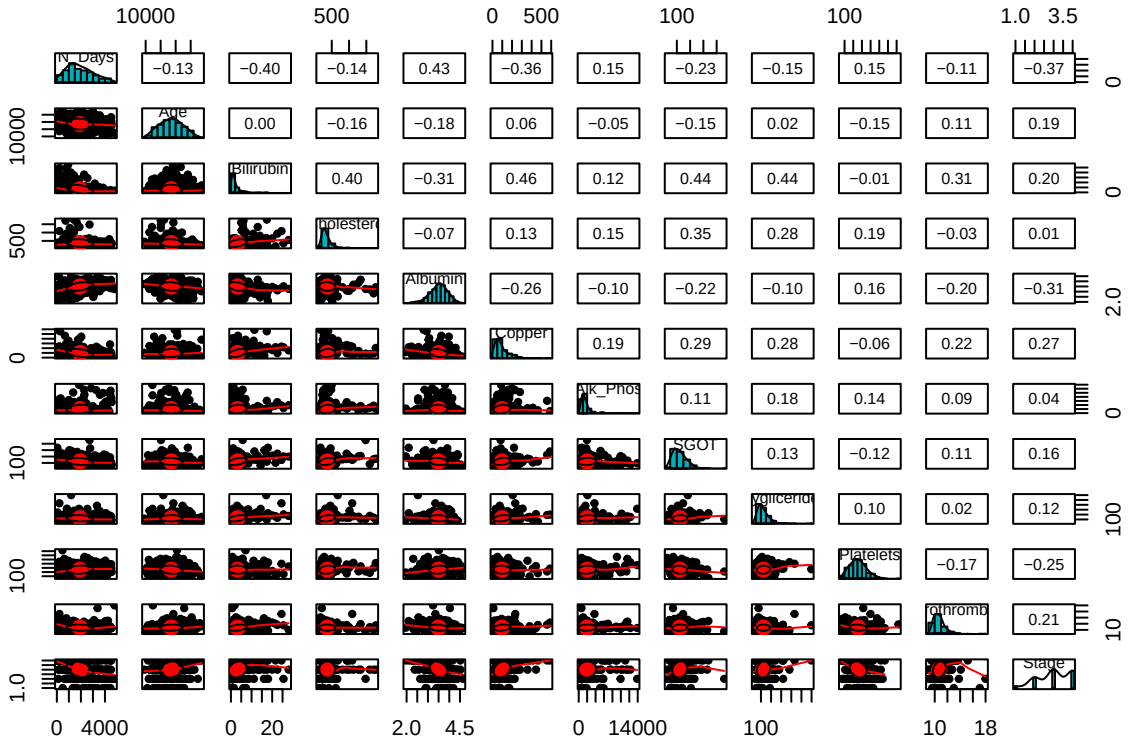
Detecting Outliers using Tukey's Method

```
##        N_Days           Age     Bilirubin   Cholesterol       Albumin
##             0             0            46           154             9
##       Copper      Alk_Phos          SGOT Tryglicerides     Platelets
##          125           141           113           146            17
##   Prothrombin         Stage
##           20             6
```

- The number of outliers detected for each numeric variable is displayed above using Tukey's method which is 1.5 times the Interquartile Range (IQR).
- IQR is a robust measure of spread that is less sensitive to outliers than the range.

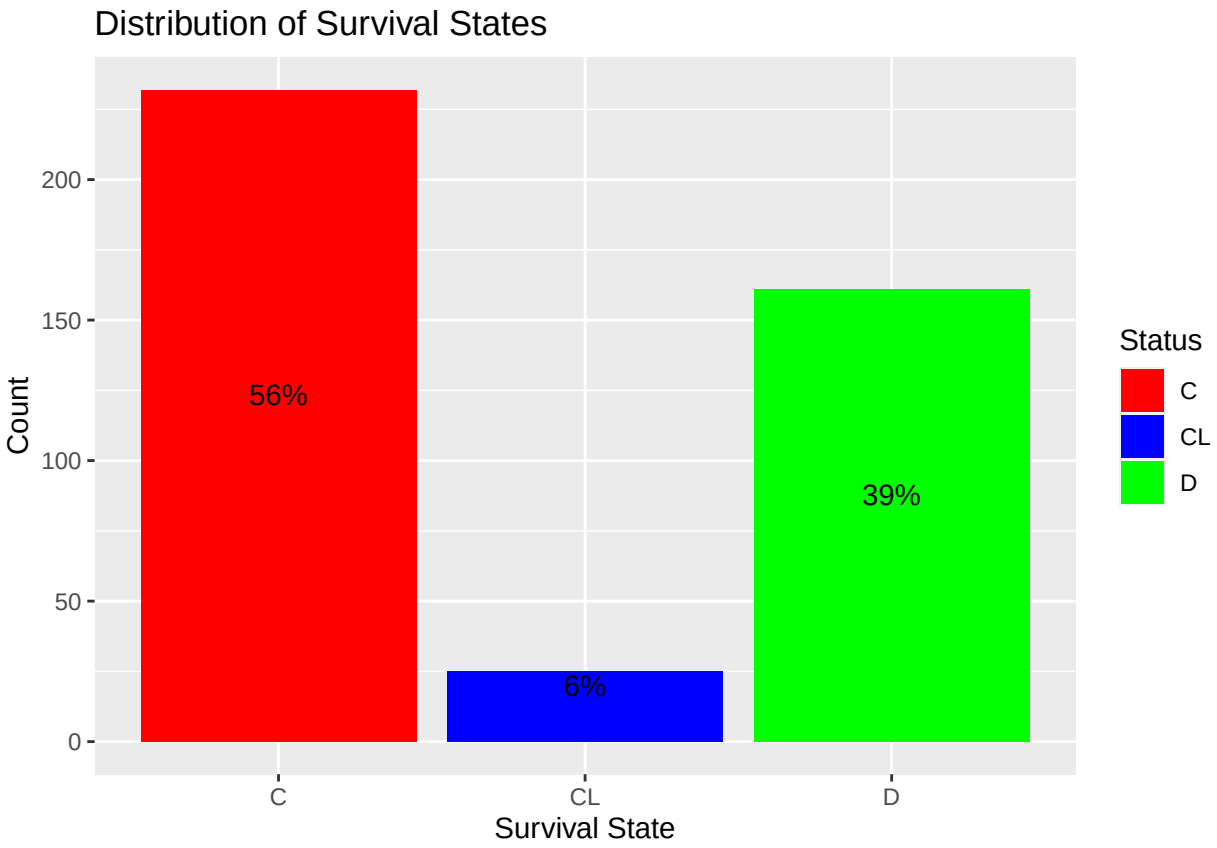
- It is important to consider these outliers during data preprocessing and model building to ensure robust and accurate predictions.

Pairs Panels Visualization



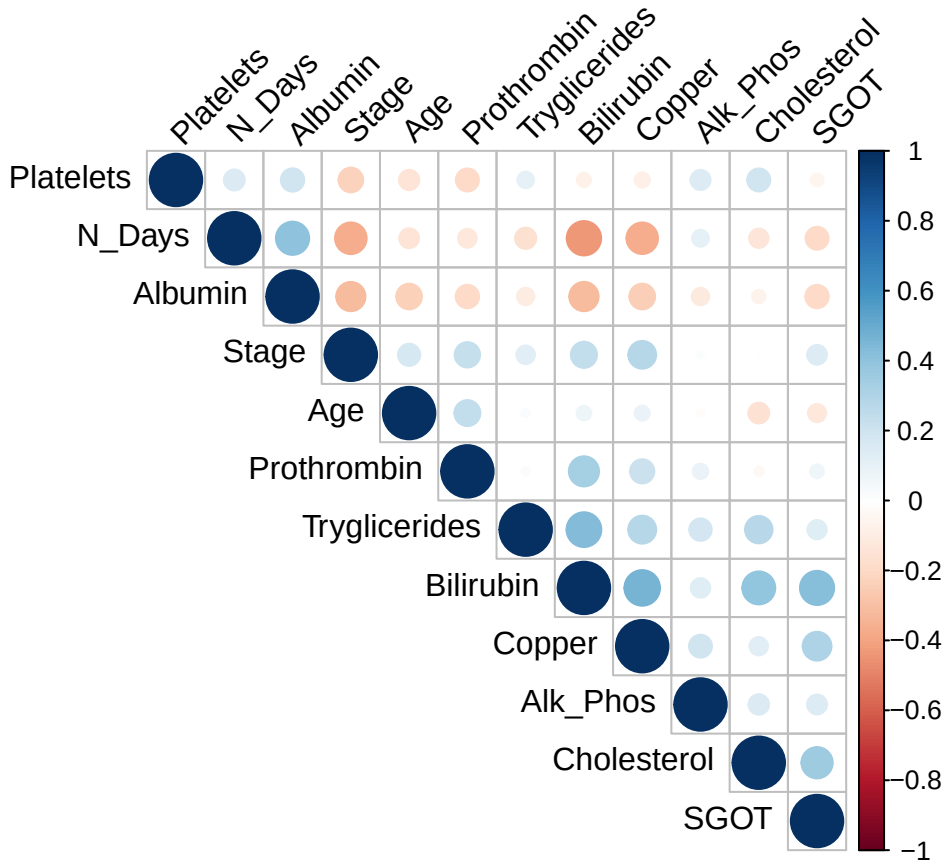
- The pairs panel visualization provides a comprehensive view of the relationships between numeric variables in the dataset. The histograms along the diagonal provide distributions for each variable, showing varied skewness and central tendencies which may suggest different underlying distributions for each variable.
- The scatter plots below the diagonal indicate the relationships between pairs of variables. Some pairs show positive correlations, negative correlations, or no discernible relationship. The correlation coefficients above the diagonal numerically summarize the degree of linear relationship between pairs of variables. These coefficients range from -1 to 1, indicating strong negative to strong positive correlations respectively.
- The use of ellipses in some plots visually emphasizes the strength and direction of the correlations, with tighter ellipses indicating stronger correlations. The color coding and density plots overlaying the scatter plots provide further depth, highlighting the density of data points and the gradient of relationships, which can be crucial for understanding complex interactions within the data.

Evaluation of Distribution of Survival States



- The bar plot above shows the distribution of survival states in the dataset. The survival states are labeled as 'C', 'CL', and 'D', representing different survival outcomes for patients with cirrhosis.
- The percentage of survival stage of C is 55.5%, CL is 5.98%, and D is 38.52%. The distribution of survival states is imbalanced, with the majority of patients in the 'C' category.

Correlation Analysis



- The correlation matrix provides insights into the relationships between numeric variables in the dataset. The correlation coefficients range from -1 to 1, indicating the strength and direction of the linear relationships between variables.
- The correlation matrix is visualized using a circular plot, with the intensity of the color and the size of the circle representing the magnitude of the correlation coefficients. The variables are ordered based on hierarchical clustering, highlighting groups of variables with similar correlation patterns.
- There is a strong positive correlation between some of the liver function tests, like "Bilirubin" and "Alk_Phos" (alkaline phosphatase), "SGOT" (serum glutamic-oxaloacetic transaminase), and "Copper". This is expected as these are indicators of liver function and typically move in the same direction in response to liver injury or disease.
- "Albumin" appears to be negatively correlated with "Bilirubin", "Copper", and "SGOT". A high level of albumin is often found in healthy individuals, while the other three are indicators that can signify liver damage when elevated.
- "Stage" of the disease shows correlations with multiple variables such as "Albumin", "Bilirubin", "Copper", and others. This implies that as the stage of liver disease progresses, these biochemical markers tend to change correspondingly.

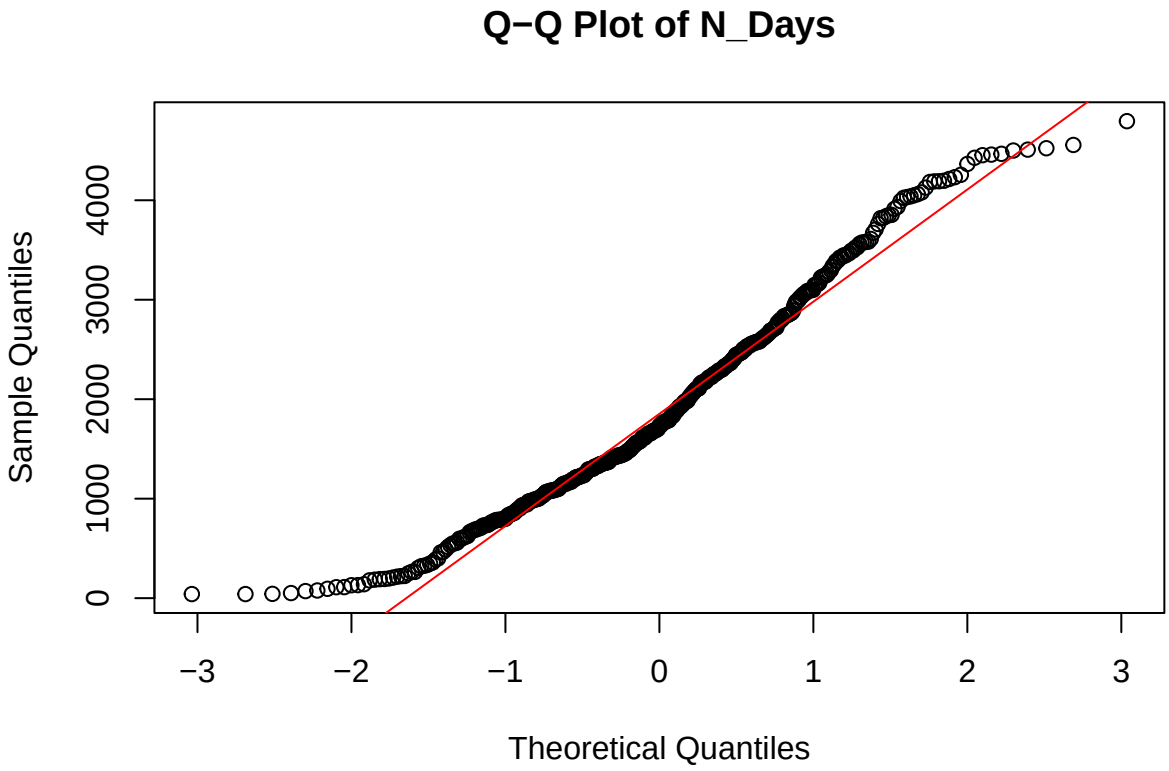
Shapiro-Wilk Normality Test for Normality Assessment

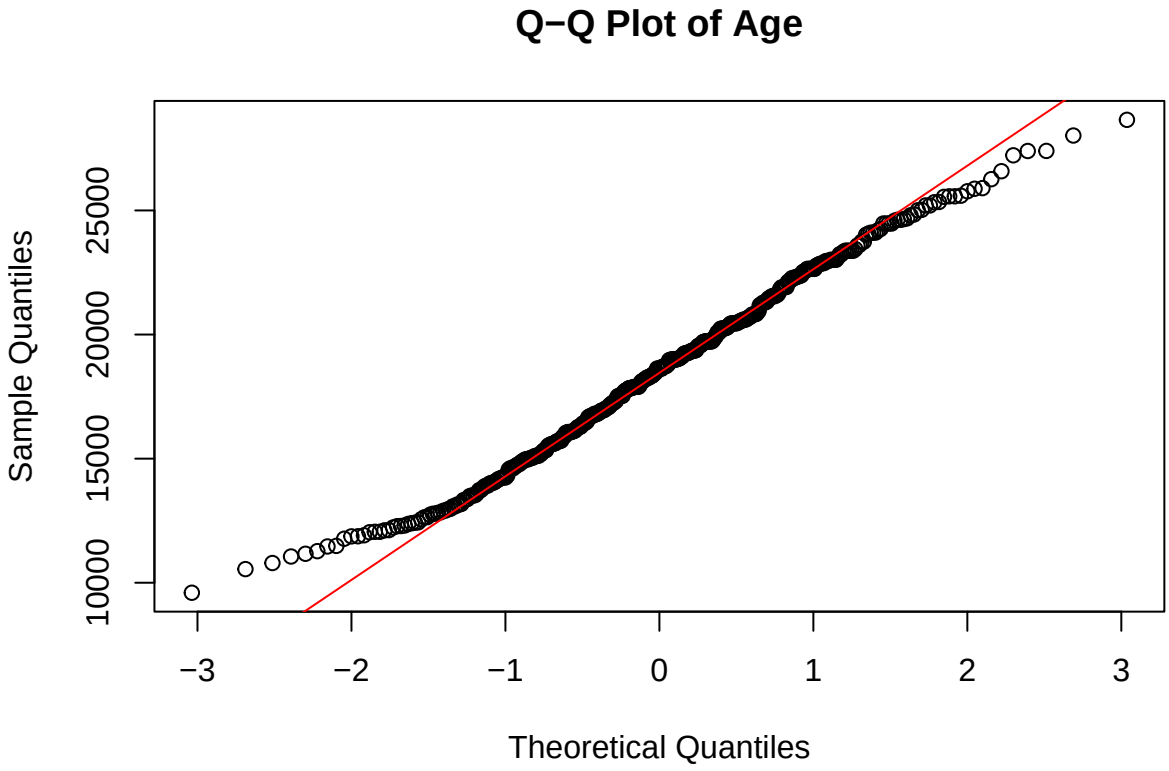
```
##        N_Days           Age     Bilirubin   Cholesterol       Albumin
```

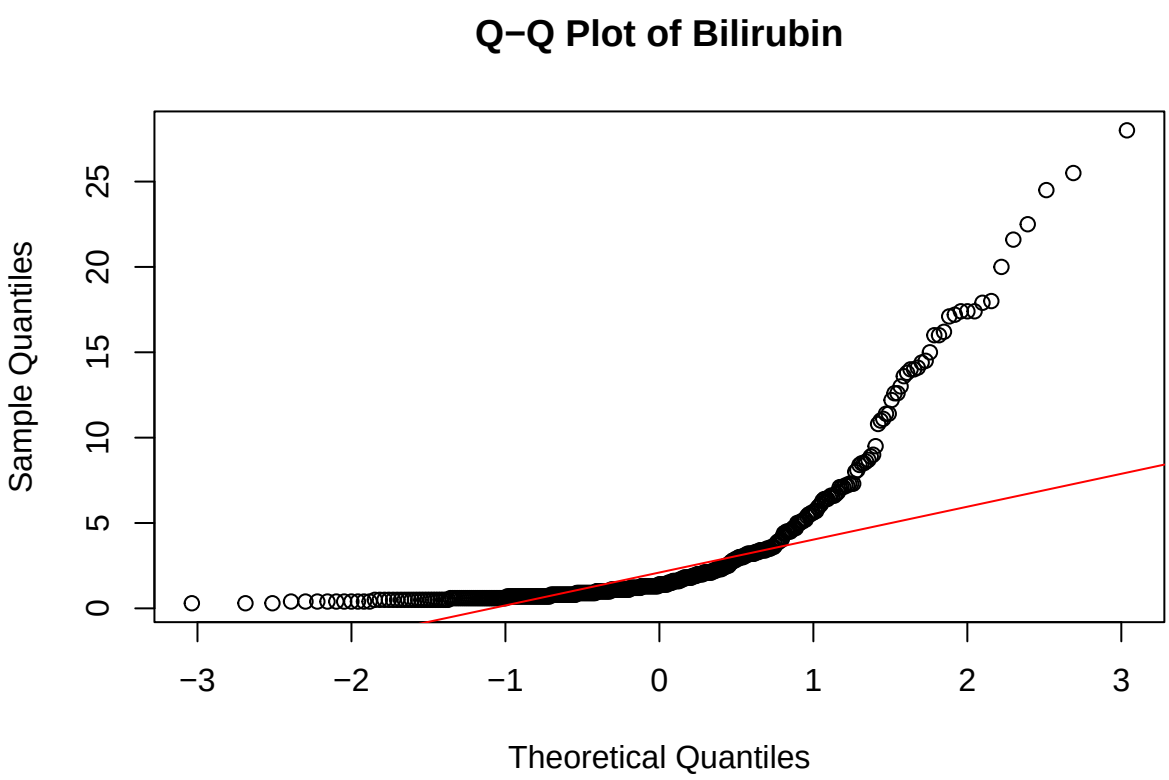
```
##  8.291488e-08  8.918234e-03  5.907597e-29  6.423407e-24  6.386612e-04
##        Copper      Alk_Phos          SGOT Tryglicerides     Platelets
##  8.351096e-20  6.849605e-26  1.467220e-12  1.221285e-17  4.207792e-06
##   Prothrombin         Stage
##  1.590303e-19  6.406701e-20
```

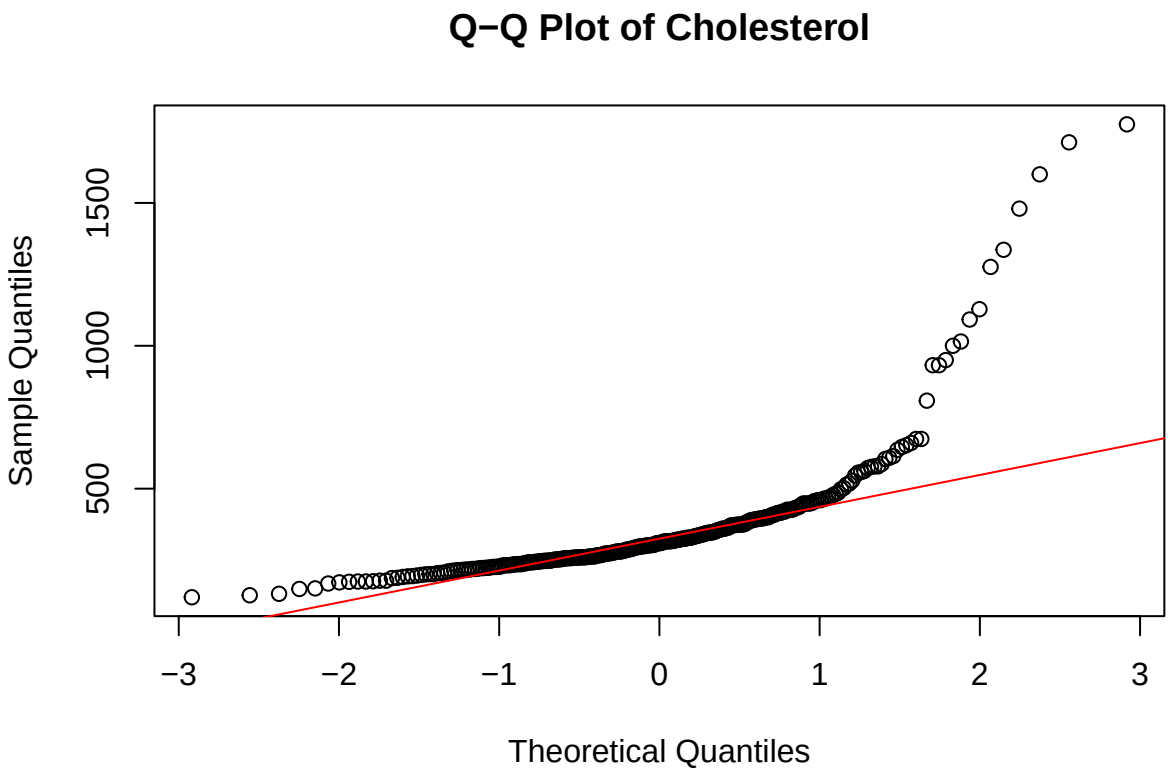
- The Shapiro-Wilk normality test is used to assess the normality of the distribution of numeric variables. The p-values indicate whether the data significantly deviates from a normal distribution.
- The p-values for the Shapiro-Wilk test are displayed above for each numeric variable. Low p-values suggest that the data may not be normally distributed, which can impact the choice of statistical tests and modeling techniques.
- A low p-value (typically less than 0.05) suggests that the distribution of the variable is not normal.

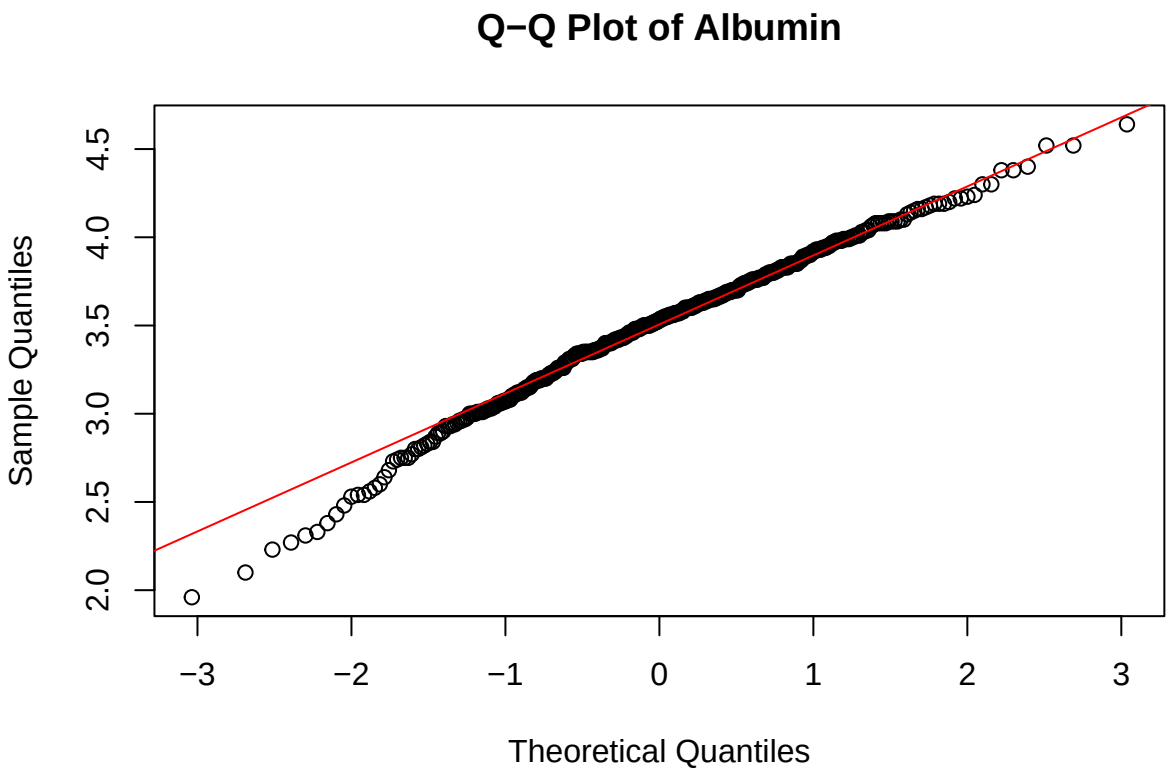
Q-Q Plots for Normality Assessment

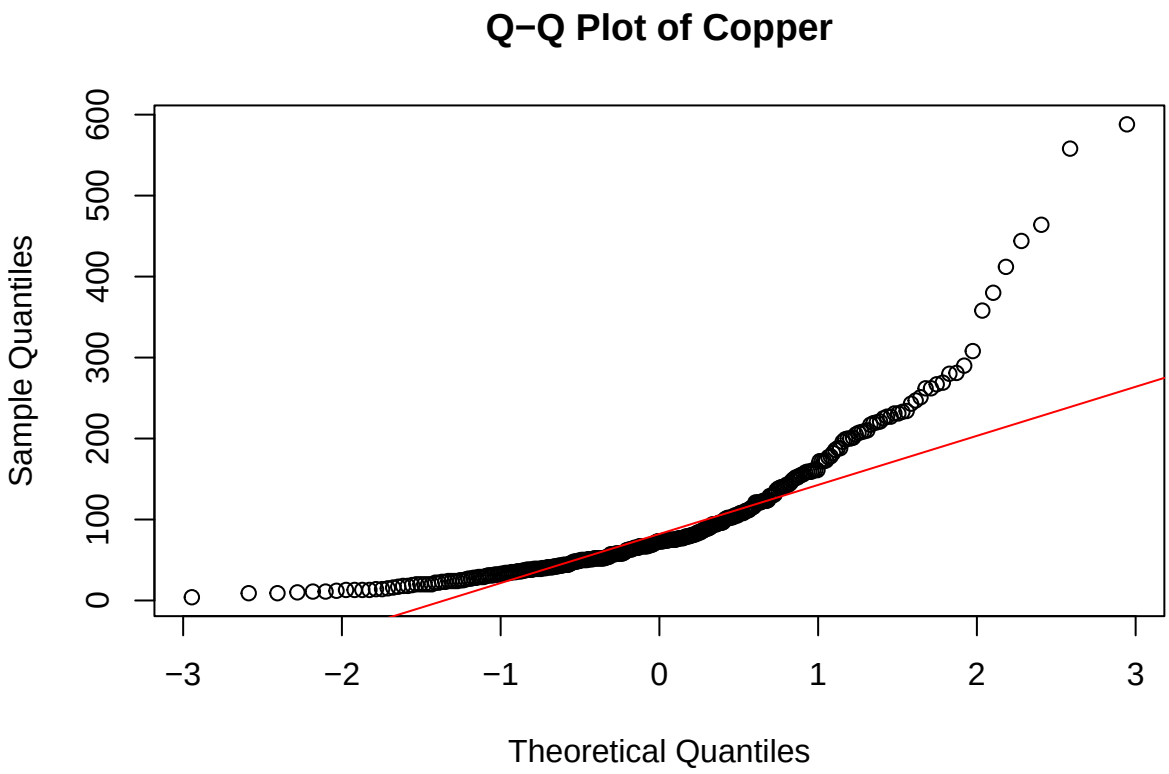


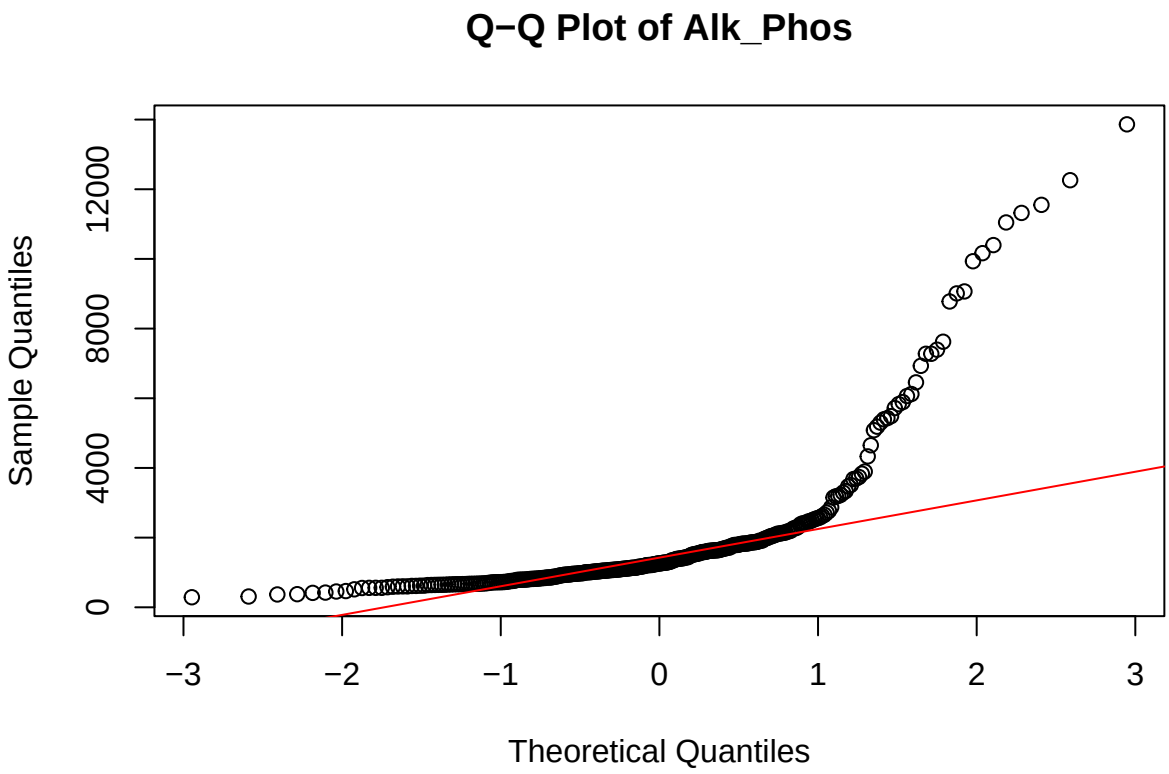


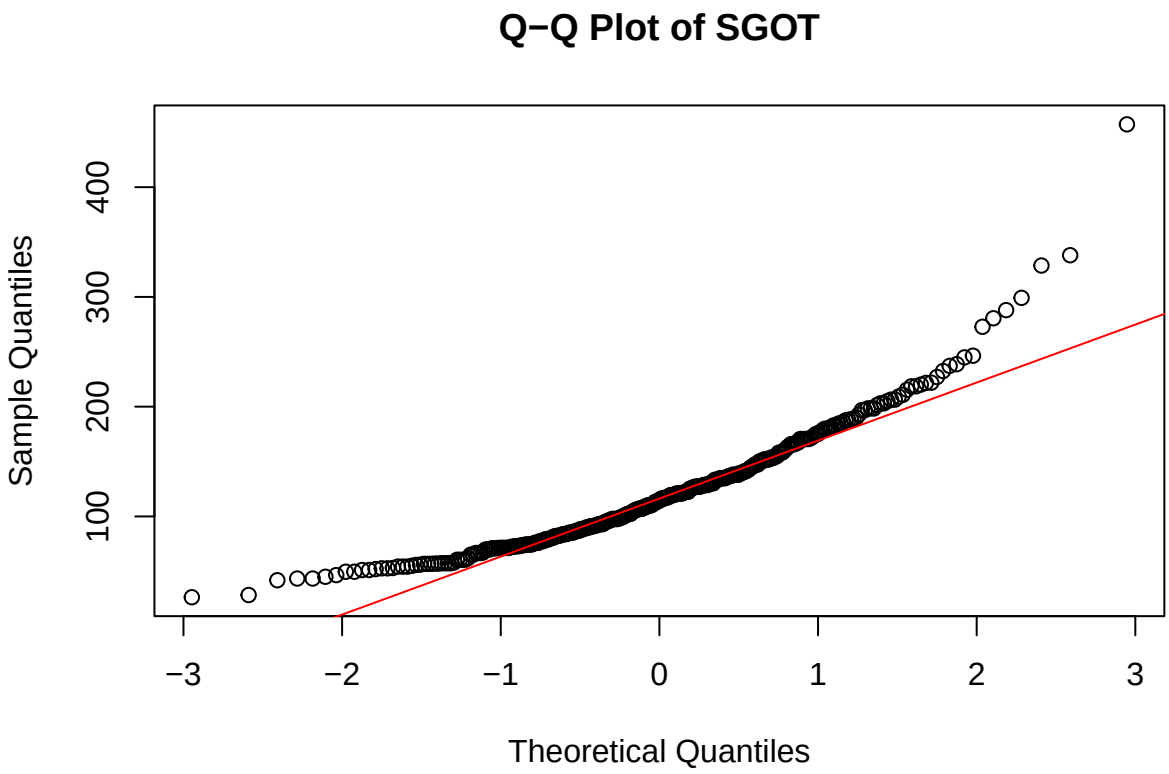

Q−Q Plot of Bilirubin
Sample Quantiles
0
5
10
15
20
25
−3
−2
−1
0
1
2
3
Theoretical Quantiles

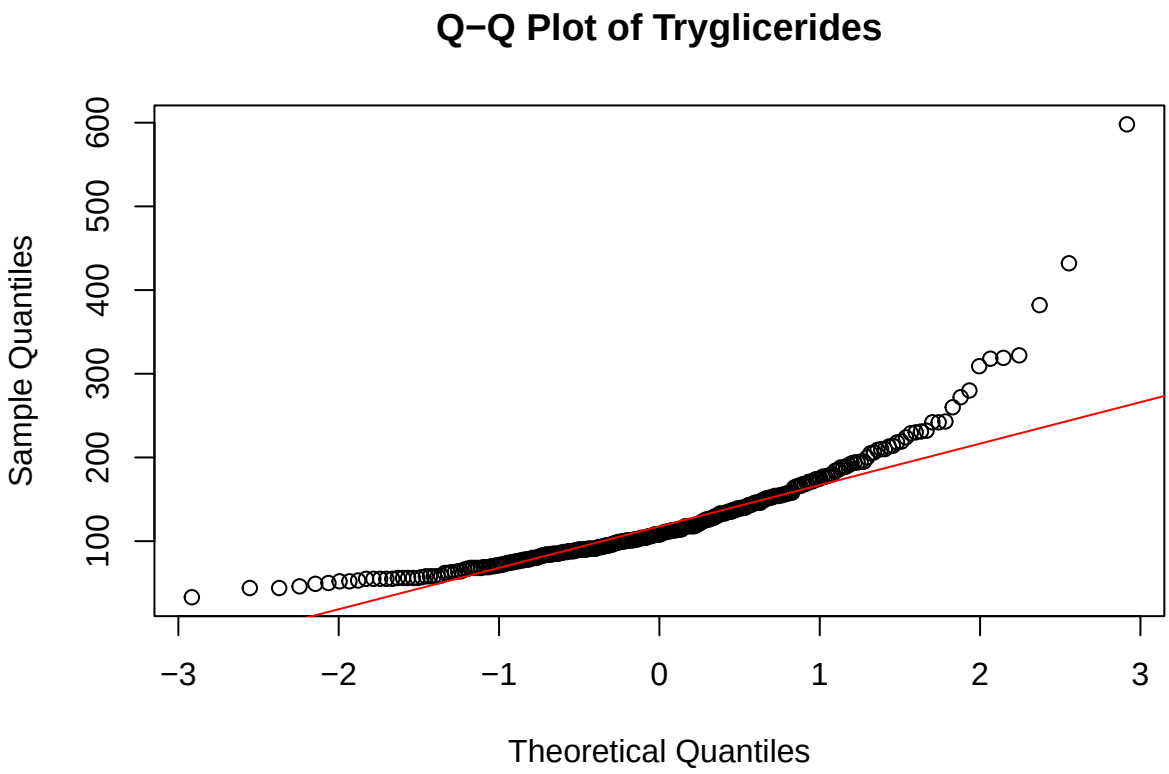

Q−Q Plot of Cholesterol
Sample Quantiles
500
1000
1500
−3
−2
−1
0
1
2
3
Theoretical Quantiles

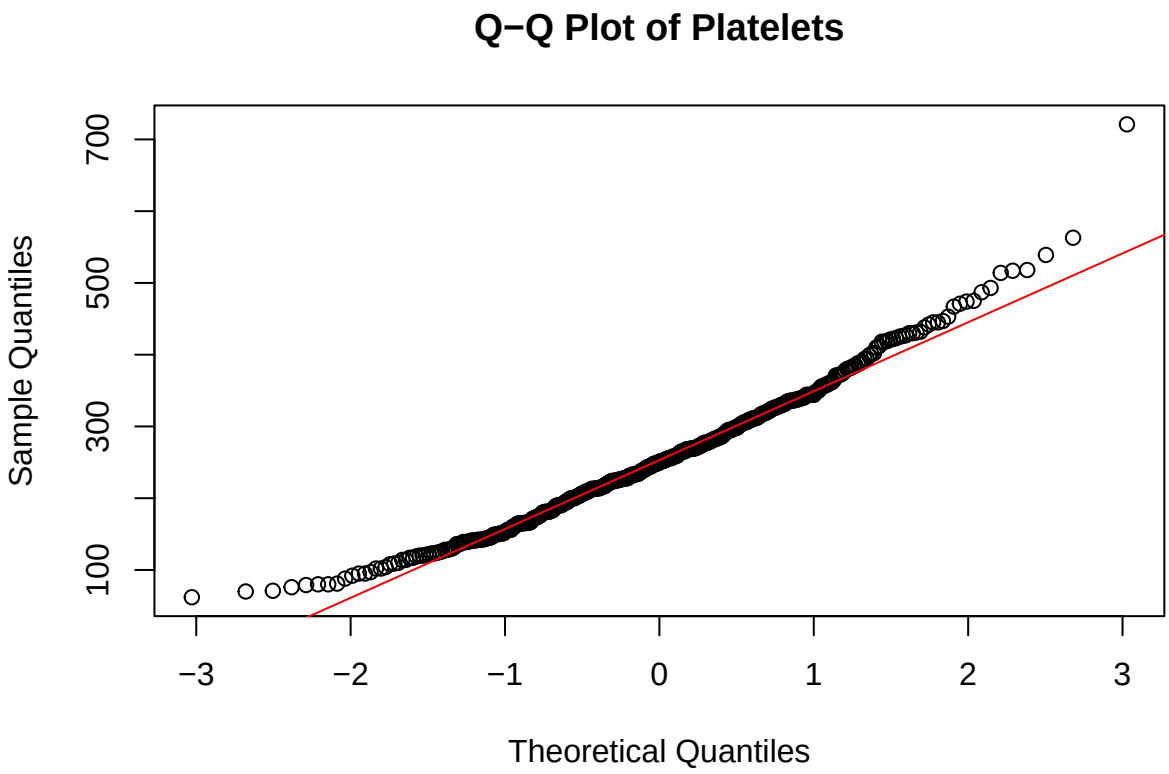

Q−Q Plot of Albumin
Sample Quantiles
Theoretical Quantiles
2.0
2.5
3.0
3.5
4.0
4.5
−3
−2
−1
0
1
2
3

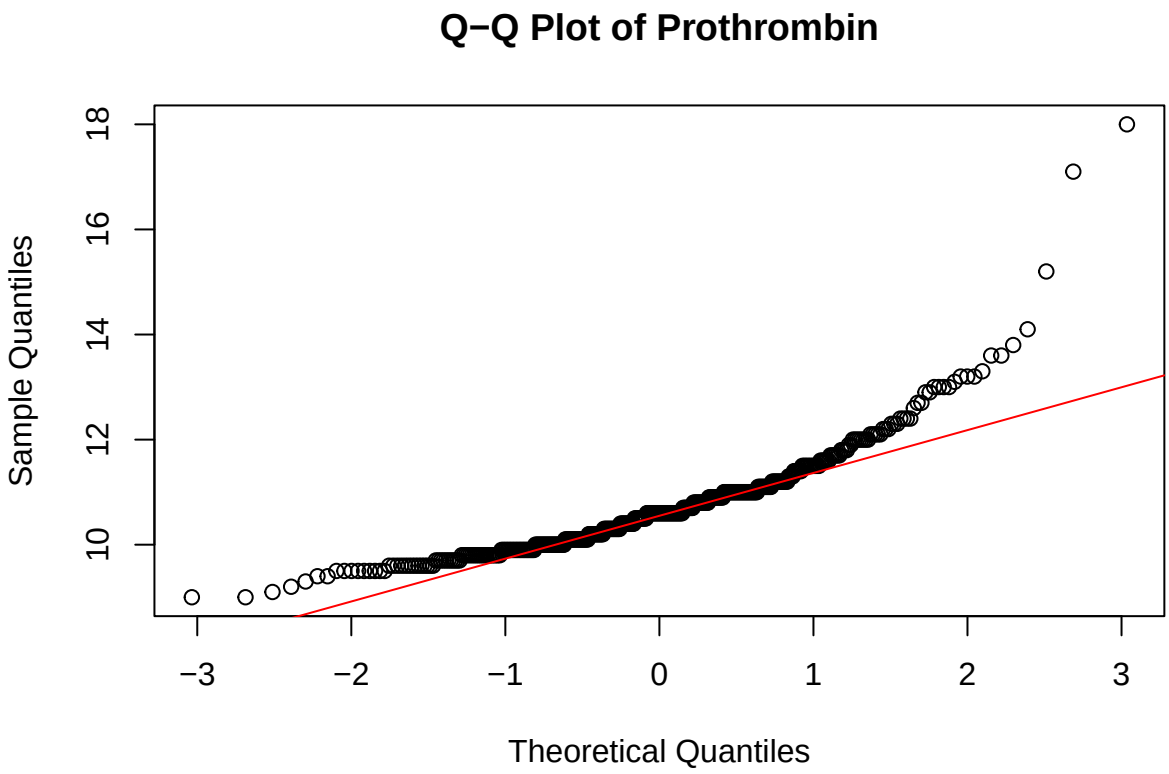

Q−Q Plot of Copper
Sample Quantiles
0
100
200
300
400
500
600
−3
−2
−1
0
1
2
3
Theoretical Quantiles

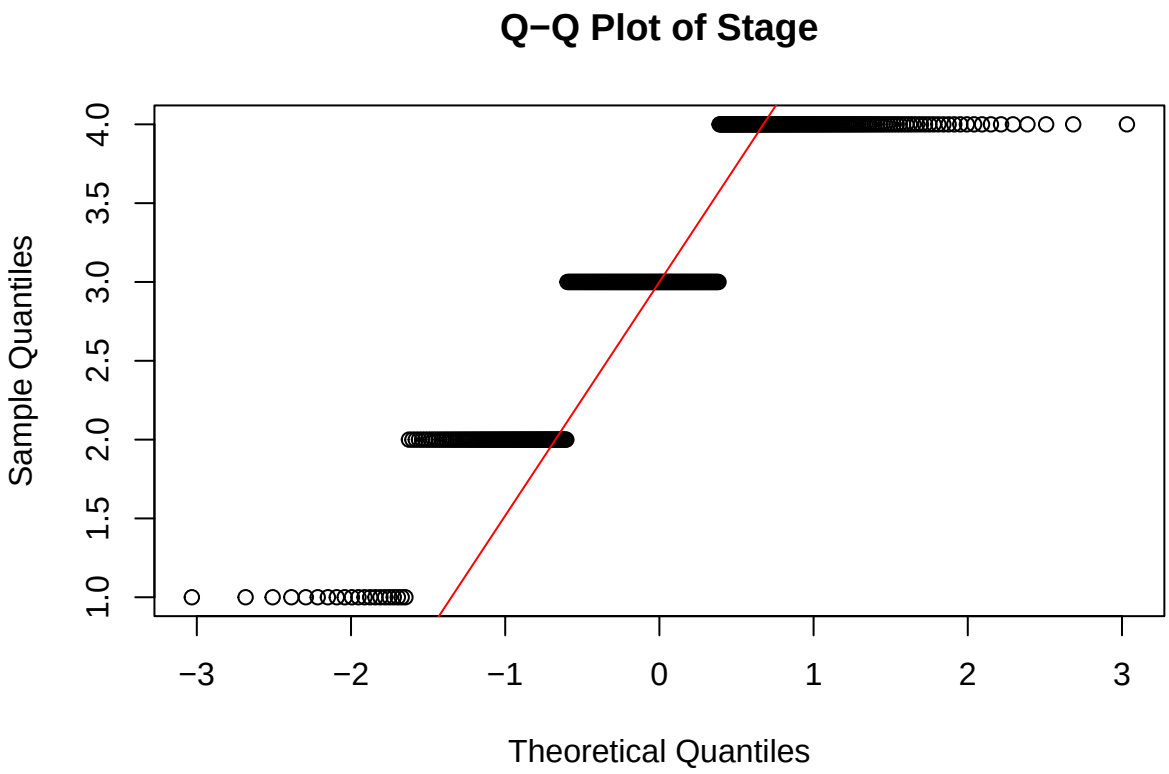

Q−Q Plot of Alk_Phos
Sample Quantiles
0
4000
8000
12000
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Q−Q Plot of SGOT
Sample Quantiles
100
200
300
400
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Q−Q Plot of Tryglicerides
Sample Quantiles
100
200
300
400
500
600
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Q−Q Plot of Platelets
Sample Quantiles
100
300
500
700
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Q−Q Plot of Prothrombin
Sample Quantiles
10
12
14
16
18
−3
−2
−1
0
1
2
3
Theoretical Quantiles



- The Q-Q plots above provide a visual assessment of the normality of the distribution of numeric variables. The points should ideally fall along the red line, indicating a normal distribution.
- Deviations from the red line suggest non-normality in the data. These plots can help identify variables that may require transformation or non-parametric analysis to address non-normality.
- The Q-Q plots are useful for assessing the distribution of numeric variables and identifying potential deviations from normality, which can inform data preprocessing and modeling decisions.
- The Q-Q plots provide a visual representation of the distribution of each numeric variable, allowing for a quick assessment of normality assumptions.
- It is important to consider the normality of data when selecting statistical tests and modeling techniques to ensure accurate and reliable results.
- For features like N_Days, Age, Bilirubin, and Cholesterol, the points deviate from the straight line, suggesting that these features do not follow a normal distribution.
- In some plots, such as for Bilirubin and Cholesterol, the points form a curve that tails off at the ends. This pattern suggests a heavy-tailed distribution, indicating the presence of outliers or extreme values in the data.

Data Cleaning and Preprocessing

Missing Values Identification

```
##         N_Days         Status           Drug            Age            Sex
```

```
##             0             0           106             0             0
##       Ascites  Hepatomegaly       Spiders         Edema     Bilirubin
##           106           106           106             0             0
##   Cholesterol       Albumin        Copper      Alk_Phos          SGOT
##           134             0           108           106           106
## Tryglicerides     Platelets   Prothrombin         Stage
##           136            11             2             6
```

- The missing values are identified in the dataset, with the number of missing values displayed for each column. The presence of missing values can impact the quality and reliability of the analysis and modeling.
- There are 1033 missing values in the dataset. It is important to address these missing values through imputation or removal to ensure the integrity of the data.

Imputation of Missing Values

```
## 'data.frame':    418 obs. of  19 variables:
##  $ N_Days       : num  400 4500 1012 1925 1504 ...
##  $ Status       : Factor w/ 3 levels "C","CL","D": 3 1 3 3 2 3 1 3 3 3 ...
##  $ Drug         : chr  "D-penicillamine" "D-penicillamine" "D-penicillamine" "D-penicillamine" ...
##  $ Age          : num  21464 20617 25594 19994 13918 ...
##  $ Sex          : chr  "F" "F" "M" "F" ...
##  $ Ascites      : chr  "Y" "N" "N" "N" ...
##  $ Hepatomegaly : chr  "Y" "Y" "N" "Y" ...
##  $ Spiders      : chr  "Y" "Y" "N" "Y" ...
##  $ Edema        : chr  "Y" "N" "S" "S" ...
##  $ Bilirubin    : num  14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
##  $ Cholesterol  : num  261 302 176 244 279 248 322 280 562 200 ...
##  $ Albumin      : num  2.6 4.14 3.48 2.54 3.53 3.98 4.09 4 3.08 2.74 ...
##  $ Copper       : num  156 54 210 64 143 50 52 52 79 140 ...
##  $ Alk_Phos     : num  1718 7395 516 6122 671 ...
##  $ SGOT         : num  137.9 113.5 96.1 60.6 113.2 ...
##  $ Tryglicerides: num  172 88 55 92 72 63 213 189 88 143 ...
##  $ Platelets    : int  190 221 151 183 136 251 204 373 251 302 ...
##  $ Prothrombin  : num  12.2 10.6 12 10.3 10.9 11 9.7 11 11 11.5 ...
##  $ Stage        : num  4 3 4 4 3 3 3 3 2 4 ...
```

- Missing values are imputed in the dataset using the median for numeric columns and the mode for categorical columns. Imputation helps maintain the integrity of the data and ensures that all variables have complete information for analysis and modeling.
- Mean imputation involves replacing missing values in numeric data with the mean value of the respective variable. This technique is widely used because it preserves the sample mean, making it a reasonable choice for maintaining the overall distribution and central tendency of the data.
- Mode imputation involves replacing missing values in categorical data with the most frequent value (mode) of the respective variable. This technique is suitable for categorical variables with a limited number of unique values and helps maintain the distribution of the data.
- After imputation, there are no missing values in the dataset. The structure of the data is displayed to confirm that all missing values have been addressed through imputation.

- I didn't convert the categorical columns to numeric before imputation because converting categorical variables into numeric formats before imputation (such as assigning numbers to categories) can introduce arbitrary ordinality or false numeric relationships, which might not exist naturally within the data.
- For instance, converting "red", "blue", and "green" into 1, 2, and 3 might imply that green is somehow "higher" or "more" than red, which is semantically incorrect and can mislead the analysis. By keeping the data categorical and using mode imputation, we can maintain the integrity and meaning of the data.

One-Hot Encoding for Categorical Variables

```
## 'data.frame':    418 obs. of  21 variables:
##  $ N_Days             : num  400 4500 1012 1925 1504 ...
##  $ Status             : Factor w/ 3 levels "C","CL","D": 3 1 3 3 2 3 1 3 3 3 ...
##  $ Age                : num  21464 20617 25594 19994 13918 ...
##  $ Bilirubin          : num  14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
##  $ Cholesterol        : num  261 302 176 244 279 248 322 280 562 200 ...
##  $ Albumin            : num  2.6 4.14 3.48 2.54 3.53 3.98 4.09 4 3.08 2.74 ...
##  $ Copper             : num  156 54 210 64 143 50 52 52 79 140 ...
##  $ Alk_Phos           : num  1718 7395 516 6122 671 ...
##  $ SGOT               : num  137.9 113.5 96.1 60.6 113.2 ...
##  $ Tryglicerides      : num  172 88 55 92 72 63 213 189 88 143 ...
##  $ Platelets          : int  190 221 151 183 136 251 204 373 251 302 ...
##  $ Prothrombin        : num  12.2 10.6 12 10.3 10.9 11 9.7 11 11 11.5 ...
##  $ Stage              : num  4 3 4 4 3 3 3 3 2 4 ...
##  $ DrugD-penicillamine: num  1 1 1 1 0 0 0 0 1 0 ...
##  $ DrugPlacebo        : num  0 0 0 0 1 1 1 1 0 1 ...
##  $ SexM               : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ AscitesY           : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ HepatomegalyY      : num  1 1 0 1 1 1 1 0 0 0 ...
##  $ SpidersY           : num  1 1 0 1 1 0 0 0 1 1 ...
##  $ EdemaS             : num  0 0 1 1 0 0 0 0 0 0 ...
##  $ EdemaY             : num  1 0 0 0 0 0 0 0 0 1 ...
```

- One-hot encoding is applied to the categorical variables in the dataset to convert them into a format suitable for modeling. This technique creates binary columns for each category within a categorical variable, allowing the model to interpret the categorical data effectively.
- It is aslo called dummy encoding or one-of-K encoding. It is a technique used to convert categorical variables into a format that can be provided to ML algorithms to improve model performance.
- For my dataset, I used one-hot encoding to convert the categorical variables into binary columns because it is a common method for handling categorical variables in machine learning models. By converting categorical variables into binary columns, we can represent each category as a separate feature, allowing the model to capture the information encoded in the categories.
- I didn't apply one-hot encoding to the 'Status' column because it is the target variable, and one-hot encoding would create unnecessary additional columns for each category, which is not required for the target variable in classification tasks.

Skewness Identification

```
##             N_Days                Age           Bilirubin         Cholesterol
```

```
##          0.46921558          0.08622782          2.69813743          4.25914487
##             Albumin              Copper            Alk_Phos                SGOT
##         -0.46417641          2.81638425          3.56976804          1.77173219
##       Tryglicerides           Platelets         Prothrombin               Stage
##          3.24232286          0.63569052          2.21418180         -0.49506749
## DrugD-penicillamine         DrugPlacebo                SexM           AscitesY
##         -0.54358820          0.54358820          2.56325292          3.79129565
##       HepatomegalyY            SpidersY              EdemaS              EdemaY
##         -0.56491343          1.38025218          2.56325292          4.22157905
```

- Skewness is a measure of the asymmetry of the distribution of a variable. Positive skewness indicates a right-skewed distribution, while negative skewness indicates a left-skewed distribution.
- Positive skewness means that the right tail of the distribution is longer or fatter than the left tail, while negative skewness means that the left tail is longer or fatter than the right tail.
- A value > 0 indicates positive skew, a value < 0 indicates negative skew, and values close to 0 suggest a symmetric distribution.
- The skewness of numeric columns in the dataset is displayed above. Skewness correction is necessary to ensure that the data is normally distributed, which is a common assumption in many statistical tests and machine learning algorithms.
- The skewness of the numeric columns in the dataset is assessed to identify variables that may require transformation to correct skewness. Skewness correction is important for ensuring that the data meets the assumptions of statistical tests and modeling techniques.

Transformation of Skewed Variables

```
##             N_Days                Age           Bilirubin         Cholesterol
##          0.46921558          0.08622782          1.12849331          1.58420364
##             Albumin              Copper            Alk_Phos                SGOT
##         -0.46417641         -0.18896324          1.19959440         -0.14232209
##       Tryglicerides           Platelets         Prothrombin               Stage
##          0.53624530          0.03373105          1.54625855         -0.49506749
## DrugD-penicillamine         DrugPlacebo                SexM           AscitesY
##         -0.54358820          0.54358820          2.56325292          3.79129565
##       HepatomegalyY            SpidersY              EdemaS              EdemaY
##         -0.56491343          1.38025218          2.56325292          4.22157905
```

- The log transformation is applied to the specified numeric columns to correct skewness and normalize the data. The log transformation is commonly used to stabilize variance and improve the normality of the data.
- The square root transformation is applied to the 'Platelets' column to address skewness and normalize the data. The square root transformation is useful for reducing the impact of extreme values and improving the distribution of the data.
- The skewness of the numeric columns is checked after the transformations to ensure that the data is closer to a normal distribution. Skewness correction is essential for preparing the data for statistical analysis and machine learning modeling.
- Binary variables are typically represented as 0 or 1, and they do not require transformation for skewness correction or standardization. But if applied to binary variables, the log transformation would not be meaningful as it would not change the distribution of the data and we should revert them back to their original scale.

Standardization of Numeric Data

```
##      N_Days        Status        Age             Bilirubin
##  Min.   :-1.6989   C :232   Min.   :-2.3416   Min.   :-1.2196
##  1st Qu.:-0.7469   CL: 25   1st Qu.:-0.7571   1st Qu.:-0.7600
##  Median :-0.1700   D :161   Median : 0.0248   Median :-0.3537
##  Mean   : 0.0000            Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6298            3rd Qu.: 0.7178   3rd Qu.: 0.5023
##  Max.   : 2.6046            Max.   : 2.6512   Max.   : 3.1654
##   Cholesterol         Albumin             Copper            Alk_Phos
##  Min.   :-2.7366   Min.   :-3.61775   Min.   :-3.84865   Min.   :-2.5064
##  1st Qu.:-0.4652   1st Qu.:-0.59990   1st Qu.:-0.47425   1st Qu.:-0.5018
##  Median :-0.1177   Median : 0.07662   Median : 0.02627   Median :-0.1599
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.2052   3rd Qu.: 0.64136   3rd Qu.: 0.48419   3rd Qu.: 0.3267
##  Max.   : 4.7288   Max.   : 2.68856   Max.   : 3.00923   Max.   : 3.6707
##       SGOT          Tryglicerides        Platelets         Prothrombin
##  Min.   :-3.70049   Min.   :-3.26892   Min.   :-2.58249   Min.   :-1.9297
##  1st Qu.:-0.53672   1st Qu.:-0.42036   1st Qu.:-0.64116   1st Qu.:-0.7525
##  Median : 0.06108   Median :-0.07184   Median : 0.03516   Median :-0.0966
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.49702   3rd Qu.: 0.38514   3rd Qu.: 0.66561   3rd Qu.: 0.4246
##  Max.   : 3.65086   Max.   : 4.60422   Max.   : 3.65121   Max.   : 5.9976
##      Stage          DrugD-penicillamine  DrugPlacebo           SexM
##  Min.   :-2.31126   Min.   :-1.3077     Min.   :-0.7628   Min.   :-0.3426
##  1st Qu.:-1.16929   1st Qu.:-1.3077     1st Qu.:-0.7628   1st Qu.:-0.3426
##  Median :-0.02732   Median : 0.7628     Median :-0.7628   Median :-0.3426
##  Mean   : 0.00000   Mean   : 0.0000     Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.11465   3rd Qu.: 0.7628     3rd Qu.: 1.3077   3rd Qu.:-0.3426
##  Max.   : 1.11465   Max.   : 0.7628     Max.   : 1.3077   Max.   : 2.9120
##     AscitesY       HepatomegalyY       SpidersY           EdemaS
##  Min.   :-0.2465   Min.   :-1.321   Min.   :-0.5232   Min.   :-0.3426
##  1st Qu.:-0.2465   1st Qu.:-1.321   1st Qu.:-0.5232   1st Qu.:-0.3426
##  Median :-0.2465   Median : 0.755   Median :-0.5232   Median :-0.3426
##  Mean   : 0.0000   Mean   : 0.000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.2465   3rd Qu.: 0.755   3rd Qu.:-0.5232   3rd Qu.:-0.3426
##  Max.   : 4.0469   Max.   : 0.755   Max.   : 1.9068   Max.   : 2.9120
##      EdemaY
##  Min.   :-0.2239
##  1st Qu.:-0.2239
##  Median :-0.2239
##  Mean   : 0.0000
##  3rd Qu.:-0.2239
##  Max.   : 4.4556

## 'data.frame':    418 obs. of  21 variables:
##  $ N_Days             : num  -1.37397 2.33754 -0.81996 0.00653 -0.37457 ...
##  $ Status             : Factor w/ 3 levels "C","CL","D": 3 1 3 3 2 3 1 3 3 3 ...
##  $ Age                : num  0.768 0.546 1.85 0.383 -1.21 ...
##  $ Bilirubin          : num  2.281 -0.542 -0.354 -0.136 0.502 ...
##  $ Cholesterol        : num  -0.59 -0.186 -1.68 -0.776 -0.405 ...
##  $ Albumin            : num  -2.1118 1.512 -0.041 -2.253 0.0766 ...
##  $ Copper             : num  1.108 -0.4 1.533 -0.16 0.984 ...
##  $ Alk_Phos           : num  0.336 2.667 -1.583 2.365 -1.164 ...
```

```
##  $ SGOT               : num  0.5387 0.0343 -0.396 -1.5816 0.0259 ...
##  $ Tryglicerides      : num  1.196 -0.628 -1.9 -0.507 -1.172 ...
##  $ Platelets          : num  -0.641 -0.286 -1.133 -0.725 -1.338 ...
##  $ Prothrombin        : num  1.4992 -0.0966 1.3107 -0.4202 0.2187 ...
##  $ Stage              : num  1.1147 -0.0273 1.1147 1.1147 -0.0273 ...
##  $ DrugD-penicillamine: num  0.763 0.763 0.763 0.763 -1.308 ...
##  $ DrugPlacebo        : num  -0.763 -0.763 -0.763 -0.763 1.308 ...
##  $ SexM               : num  -0.343 -0.343 2.912 -0.343 -0.343 ...
##  $ AscitesY           : num  4.047 -0.247 -0.247 -0.247 -0.247 ...
##  $ HepatomegalyY      : num  0.755 0.755 -1.321 0.755 0.755 ...
##  $ SpidersY           : num  1.907 1.907 -0.523 1.907 1.907 ...
##  $ EdemaS             : num  -0.343 -0.343 2.912 2.912 -0.343 ...
##  $ EdemaY             : num  4.456 -0.224 -0.224 -0.224 -0.224 ...
```

- Standardization is applied to the numeric columns in the dataset to ensure that all variables are on a common scale. Standardization is important for many machine learning algorithms that are sensitive to the scale of the input features.
- Standardization involves transforming the data so that it has a mean of 0 and a standard deviation of 1. This process helps to align the variables on a common scale, making it easier to compare and interpret their values.
- The summary of the dataset is rechecked to confirm that the standardization process has been applied successfully. The mean and standard deviation of the numeric variables should be close to 0 and 1, respectively, after standardization.
- Logistic Regression and SVM: These models often benefit from standardization, especially SVM, which is sensitive to the magnitude of input features and can behave unpredictably if features are not on the same scale. Standardization helps these algorithms converge faster and perform better by transforming the data into a scale where the features contribute equally.
- Random Forest: This model is generally less sensitive to the scale of the features because it uses rule-based splitting. Thus, normalization or standardization is not typically necessary for tree-based models like random forests.

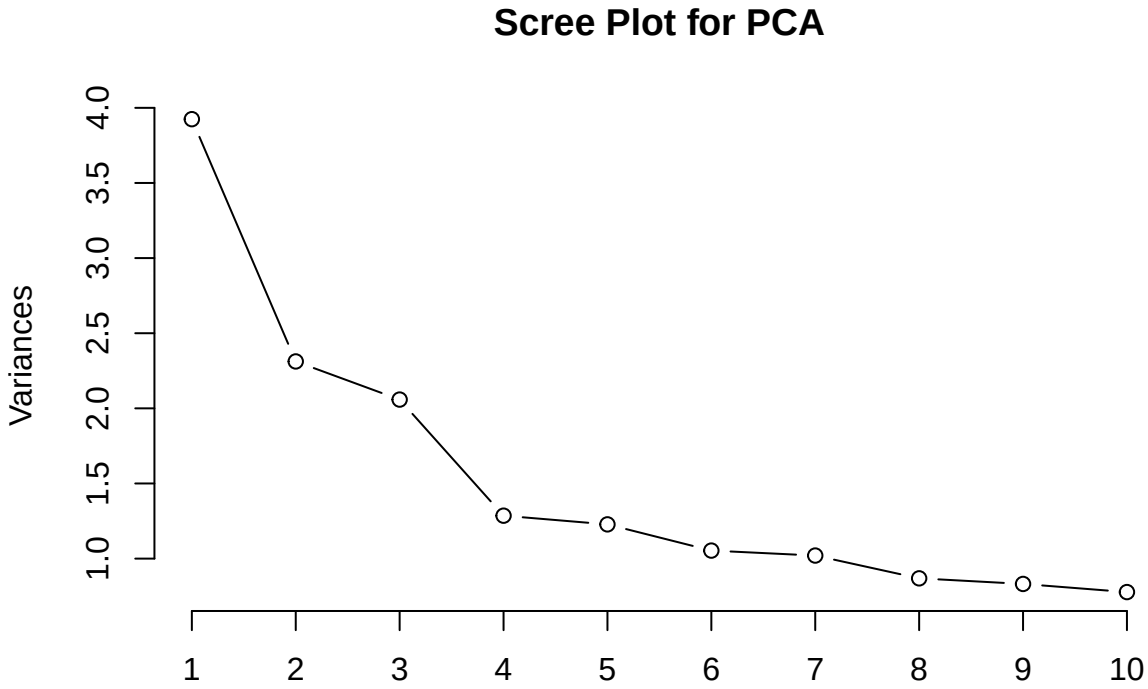
Reverting Binary Variables to Original Scale

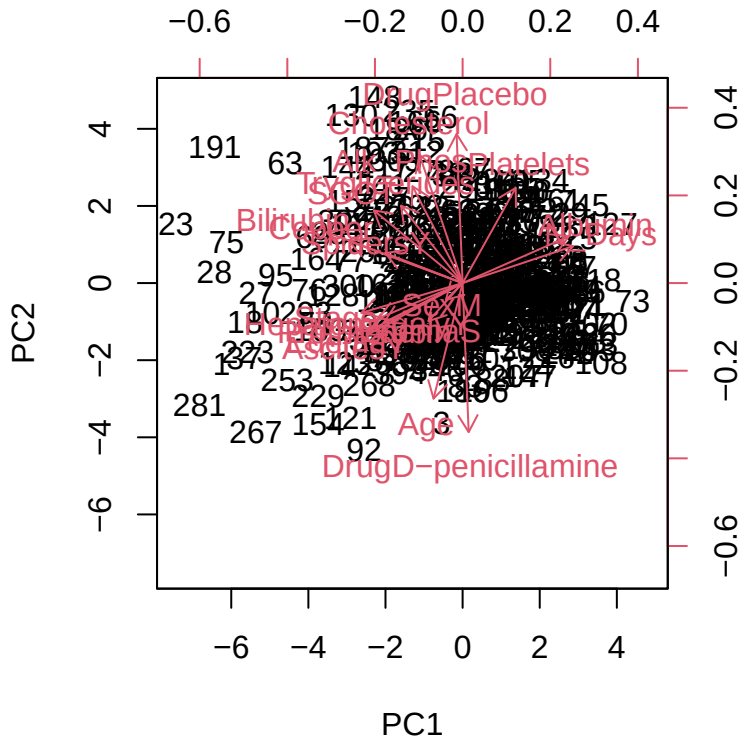
```
##      N_Days        Status        Age             Bilirubin
##  Min.   :-1.6989   C :232   Min.   :-2.3416   Min.   :-1.2196
##  1st Qu.:-0.7469   CL: 25   1st Qu.:-0.7571   1st Qu.:-0.7600
##  Median :-0.1700   D :161   Median : 0.0248   Median :-0.3537
##  Mean   : 0.0000            Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6298            3rd Qu.: 0.7178   3rd Qu.: 0.5023
##  Max.   : 2.6046            Max.   : 2.6512   Max.   : 3.1654
##   Cholesterol         Albumin             Copper            Alk_Phos
##  Min.   :-2.7366   Min.   :-3.61775   Min.   :-3.84865   Min.   :-2.5064
##  1st Qu.:-0.4652   1st Qu.:-0.59990   1st Qu.:-0.47425   1st Qu.:-0.5018
##  Median :-0.1177   Median : 0.07662   Median : 0.02627   Median :-0.1599
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.2052   3rd Qu.: 0.64136   3rd Qu.: 0.48419   3rd Qu.: 0.3267
##  Max.   : 4.7288   Max.   : 2.68856   Max.   : 3.00923   Max.   : 3.6707
##       SGOT          Tryglicerides        Platelets         Prothrombin
##  Min.   :-3.70049   Min.   :-3.26892   Min.   :-2.58249   Min.   :-1.9297
##  1st Qu.:-0.53672   1st Qu.:-0.42036   1st Qu.:-0.64116   1st Qu.:-0.7525
##  Median : 0.06108   Median :-0.07184   Median : 0.03516   Median :-0.0966
```

```
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.49702   3rd Qu.: 0.38514   3rd Qu.: 0.66561   3rd Qu.: 0.4246
##  Max.   : 3.65086   Max.   : 4.60422   Max.   : 3.65121   Max.   : 5.9976
##      Stage          DrugD-penicillamine  DrugPlacebo          SexM
##  Min.   :-2.31126   Min.   :0.0000      Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-1.16929   1st Qu.:0.0000      1st Qu.:0.0000   1st Qu.:0.0000
##  Median :-0.02732   Median :1.0000      Median :0.0000   Median :0.0000
##  Mean   : 0.00000   Mean   :0.6316      Mean   :0.3684   Mean   :0.1053
##  3rd Qu.: 1.11465   3rd Qu.:1.0000      3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   : 1.11465   Max.   :1.0000      Max.   :1.0000   Max.   :1.0000
##     AscitesY       HepatomegalyY       SpidersY          EdemaS
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.05742   Mean   :0.6364   Mean   :0.2153   Mean   :0.1053
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      EdemaY
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.04785
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

- After standardization of the numeric variables, the binary variables are reverted to their original scale by converting the standardized values back to 0 or 1. This step ensures that the binary variables are in their original format for modeling and interpretation.
- The summary of the binary variables is checked to confirm that the values have been successfully reverted to 0 or 1. The summary should show the counts of 0s and 1s for each binary variable, indicating that the reversion process was completed accurately.

Dimensionality Reduction using PCA





```
## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.9810 1.5206 1.4347 1.13391 1.10808 1.02633 1.01009
## Proportion of Variance 0.1962 0.1156 0.1029 0.06429 0.06139 0.05267 0.05101
## Cumulative Proportion  0.1962 0.3118 0.4148 0.47904 0.54043 0.59310 0.64411
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.93202 0.91173 0.88162 0.86109 0.82357 0.76982 0.75046
## Proportion of Variance 0.04343 0.04156 0.03886 0.03707 0.03391 0.02963 0.02816
## Cumulative Proportion  0.68755 0.72911 0.76797 0.80505 0.83896 0.86859 0.89675
##                           PC15    PC16    PC17    PC18    PC19      PC20
## Standard deviation     0.74258 0.66562 0.63425 0.58546 0.57054 4.448e-16
## Proportion of Variance 0.02757 0.02215 0.02011 0.01714 0.01628 0.000e+00
## Cumulative Proportion  0.92432 0.94647 0.96659 0.98372 1.00000 1.000e+00
```

- Principal Component Analysis (PCA) is applied to the standardized numeric data to reduce the dimensionality of the dataset and identify the most important components that explain the variance in the data.
- The scree plot above shows the explained variance of each principal component, helping to determine the number of components to retain for analysis. The scree plot is useful for identifying the principal components that capture the most variance in the data.
- The biplot displays the relationship between the principal components and the original variables. It helps visualize the contribution of each variable to the principal components and identify patterns in the data.
- PCA is a powerful technique for dimensionality reduction and feature extraction, allowing for the identification of the most important components that explain the variance in the data.

Model Construction

Splitting the Data into Training and Testing Sets

```
## [1] 334
```

```
## [1] 21
```

```
## [1] 84
```

```
## [1] 21
```

- The data is split into training and validation sets to train the model on a subset of the data and evaluate its performance on a separate subset. The training set contains 80% of the observations, while the validation set contains the remaining 20%.
- The data is shuffled to randomize the order of the observations before splitting to ensure that the training and validation sets are representative of the overall dataset.
- The dimensions of the training and validation sets are checked to confirm that the data has been split correctly and that the training set contains 80% of the observations.
- The training data is 334 rows by 21 columns, and the validation data is 84 rows by 21 columns.

Logistic Regression Model

```
## Penalized Multinomial Regression
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 301, 300, 301, 302, 300, 301, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.7181261  0.4631305
##   1e-04  0.7181261  0.4631305
##   1e-01  0.7181261  0.4607923
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  43  1 10
##         CL  0  0  0
##         D   4  0 26
##
## Overall Statistics
##
##                Accuracy : 0.8214
##                  95% CI : (0.7226, 0.8965)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 3.651e-07
##
##                   Kappa : 0.6335
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.9149    0.0000   0.7222
## Specificity            0.7027    1.0000   0.9167
## Pos Pred Value         0.7963       NaN   0.8667
## Neg Pred Value         0.8667    0.9881   0.8148
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.5119    0.0000   0.3095
## Detection Prevalence   0.6429    0.0000   0.3571
## Balanced Accuracy      0.8088    0.5000   0.8194
```

- A multinomial logistic regression model is trained on the training data to predict the survival status of patients with cirrhosis. The model is built using the `multinom` function from the `nnet` package.

- The Accuracy of the model is 0.82 which indicates the proportion of correct predictions made by the model.
- The CI of the model is 0.72 to 0.9 which provides a range of values within which the true accuracy is likely to fall.
- The confusion matrix shows the number of correct and incorrect predictions broken down by each class. For example, the model correctly predicted 43 instances of class C, but incorrectly predicted 10 instances of class D as class C.
- A Kappa of 1 indicates perfect agreement, while a Kappa of 0 indicates agreement equivalent to chance. A Kappa of 0.6335 suggests moderate agreement.

Support Vector Machine (SVM) Model

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 299, 302, 300, 301, 301, 300, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7214007  0.4646697
##   0.50  0.7303134  0.4769781
##   1.00  0.7333488  0.4806072
##
## Tuning parameter 'sigma' was held constant at a value of 0.03505953
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.03505953 and C = 1.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  43  1 10
##         CL  0  0  0
##         D   4  0 26
##
## Overall Statistics
##
##                Accuracy : 0.8214
##                  95% CI : (0.7226, 0.8965)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 3.651e-07
##
##                   Kappa : 0.6335
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.9149    0.0000   0.7222
## Specificity            0.7027    1.0000   0.9167
## Pos Pred Value         0.7963       NaN   0.8667
## Neg Pred Value         0.8667    0.9881   0.8148
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.5119    0.0000   0.3095
## Detection Prevalence   0.6429    0.0000   0.3571
## Balanced Accuracy      0.8088    0.5000   0.8194
```

- A Support Vector Machine (SVM) model is trained on the training data to predict the survival status of patients with cirrhosis. The model is built using the `svmRadial` method from the `e1071` package.
- The Accuracy of the model is 0.82 which indicates the proportion of correct predictions made by the model.
- The CI of the model is 0.72 to 0.9 which provides a range of values within which the true accuracy is likely to fall.
- This table shows the number of correct and incorrect predictions made by the model, broken down by each class. For example, the model correctly predicted 43 instances of class C, but incorrectly predicted 10 instances of class D as class C.
- The very small p-value (3.651e-07) suggests that the model's accuracy is significantly better than the No Information Rate.

Random Forest Model

```
## Random Forest
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 302, 299, 301, 300, 300, 302, ...
## Resampling results across tuning parameters:
##
##   mtry       Accuracy   Kappa
##    4.582576  0.7420052  0.4989542
##    7.000000  0.7300678  0.4790569
##   10.500000  0.7362231  0.4909942
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.582576.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  39  1  8
```

```
##         CL  0  0  0
##         D   8  0 28
##
## Overall Statistics
##
##                Accuracy : 0.7976
##                  95% CI : (0.6959, 0.8775)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 4.147e-06
##
##                   Kappa : 0.5925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.8298    0.0000   0.7778
## Specificity            0.7568    1.0000   0.8333
## Pos Pred Value         0.8125       NaN   0.7778
## Neg Pred Value         0.7778    0.9881   0.8333
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.4643    0.0000   0.3333
## Detection Prevalence   0.5714    0.0000   0.4286
## Balanced Accuracy      0.7933    0.5000   0.8056
```

- A Random Forest model is trained on the training data to predict the survival status of patients with cirrhosis. The model is built using the `rf` method from the `randomForest` package.
- The summary of the model provides information about the number of trees, mtry, and other details of the Random Forest model. This information helps assess the complexity and performance of the model.
- The Accuracy of the model is 0.8 which indicates the proportion of correct predictions made by the model.
- The CI of the model is 0.7 to 0.88 which provides a range of values within which the true accuracy is likely to fall.
- Predictions are made on the validation data using the trained Random Forest model, and a confusion matrix is generated to evaluate the model's performance. The confusion matrix shows the counts of true positive, true negative, false positive, and false negative predictions.
- The Kappa statistic measures the agreement between the observed and predicted classes, with a value of 1 indicating perfect agreement and 0 indicating agreement equivalent to chance. A Kappa of 0.5925 suggests moderate agreement between the predicted and observed classes.

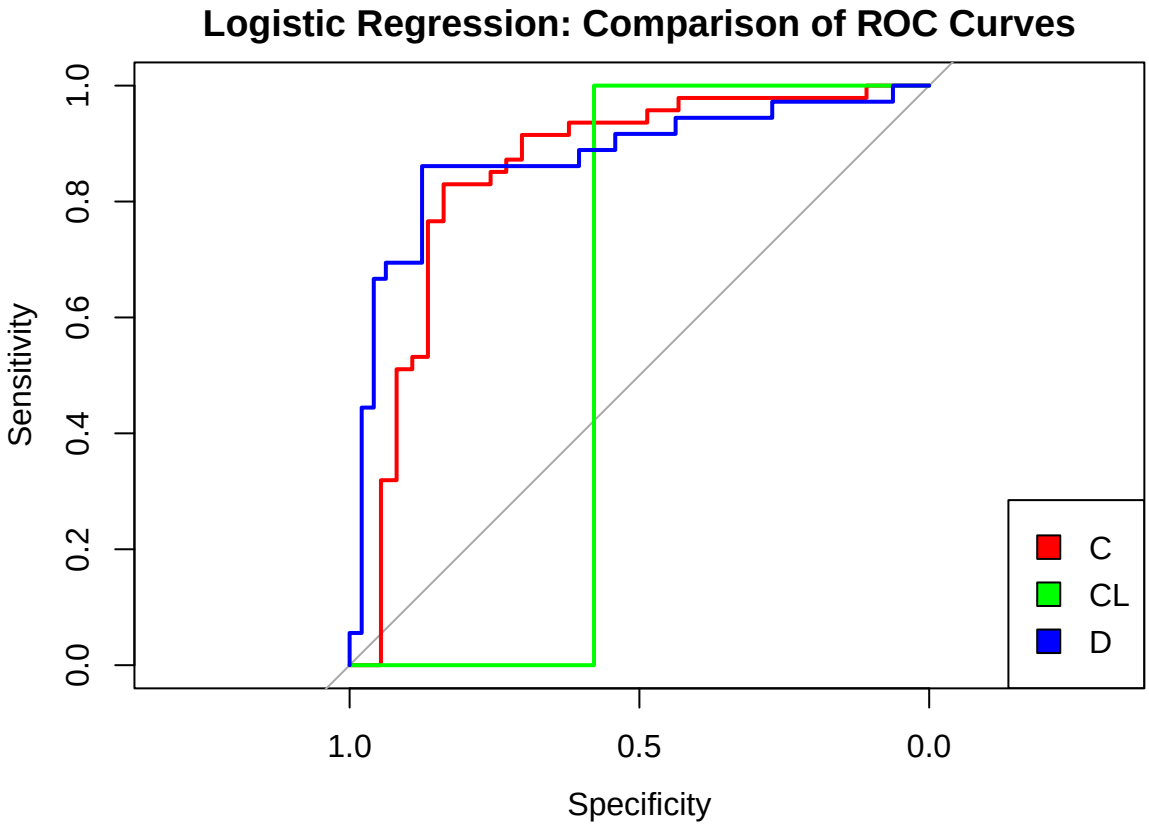
Model Evaluation

ROC Curve and AUC Calculation for Logistic Regression Model

```
## Setting levels: control = 0, case = 1
```

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

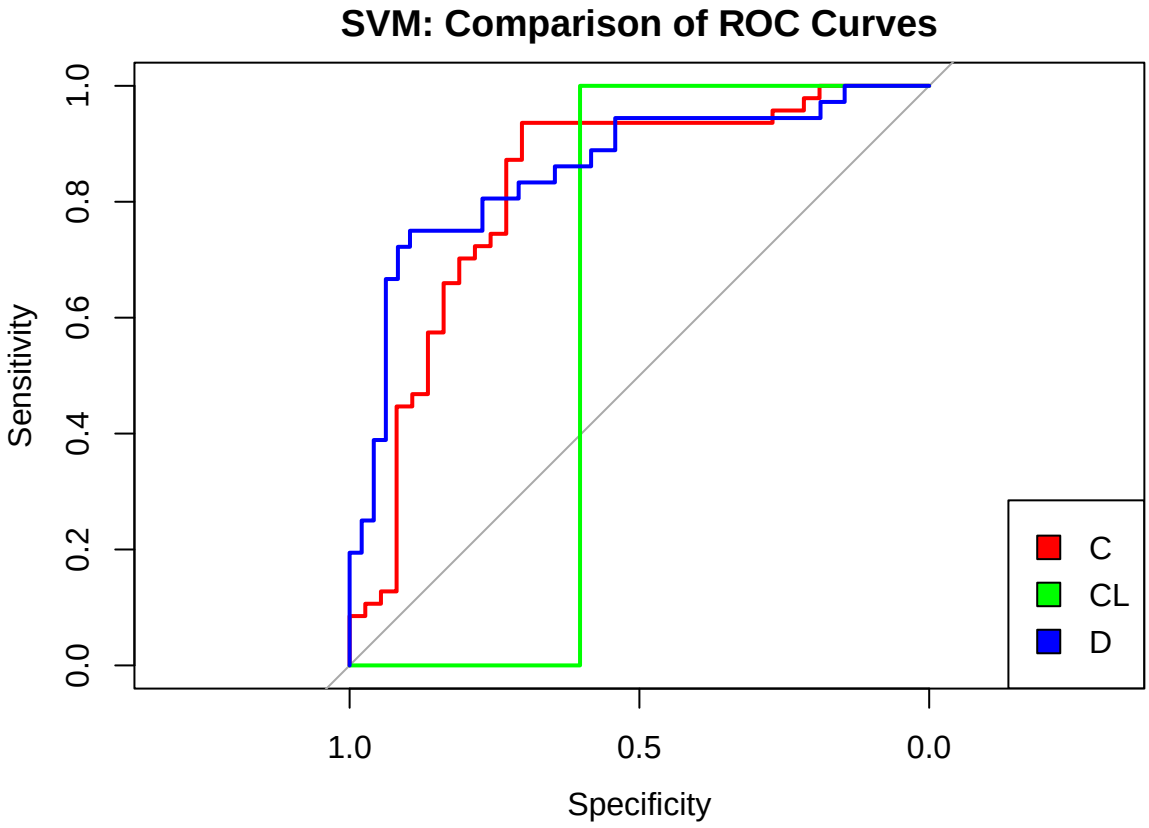
```
## [1] "AUC for C = 0.84933870040253"

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## [1] "AUC for CL = 0.578313253012048"

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## [1] "AUC for D = 0.874421296296297"
```



- The ROC curves for each class in the logistic regression model are plotted to visualize the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) for different classification thresholds.
- The Area Under the Curve (AUC) is calculated for each class, providing a measure of the model's ability to distinguish between the positive and negative classes. A higher AUC value indicates better performance in terms of classification accuracy.
- The ROC curves and AUC values help evaluate the performance of the logistic regression model in predicting the survival status of patients with cirrhosis.
- AUC for C is 0.85, for CL is 0.58, and for D is 0.87.

ROC Curve and AUC for SVM Model

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## [1] "SVM AUC for C = 0.821161587119034"

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## [1] "SVM AUC for CL = 0.602409638554217"

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## [1] "SVM AUC for D = 0.854166666666667"
```

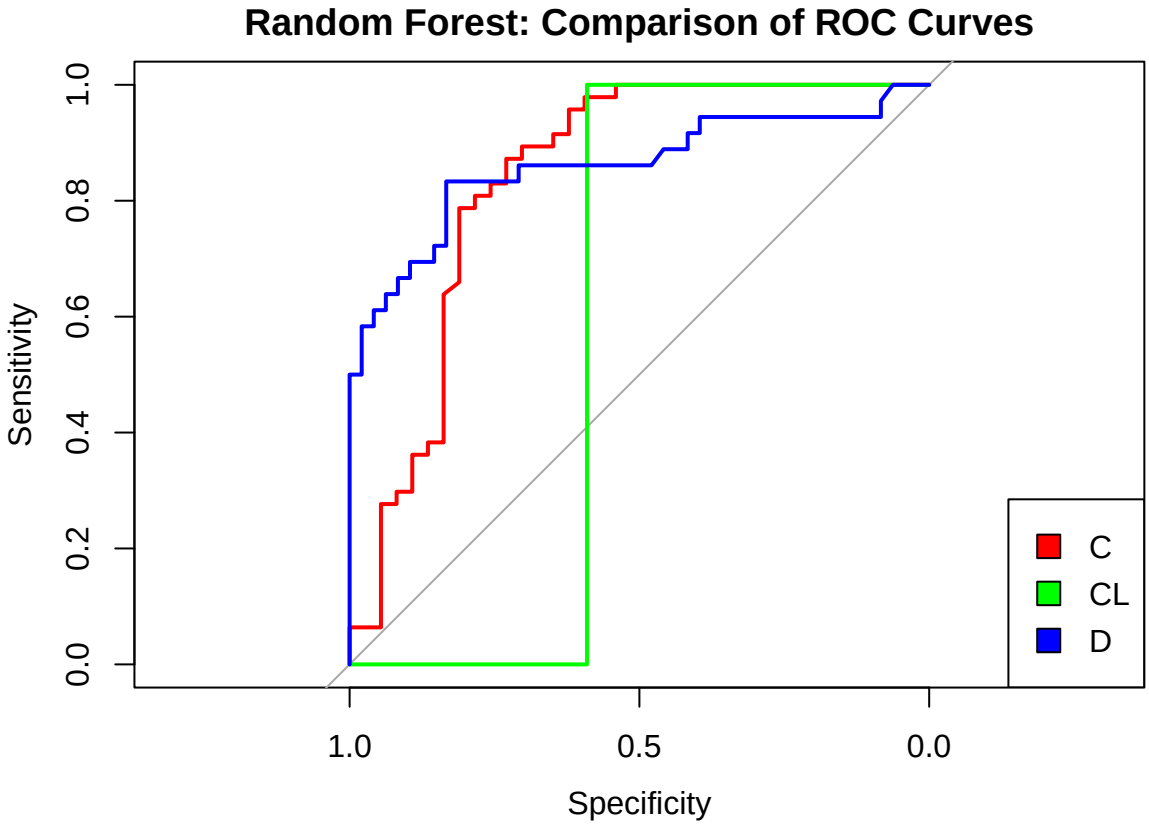


- The ROC curves for each class in the SVM model are plotted to visualize the model's performance in distinguishing between the positive and negative classes.

- The Area Under the Curve (AUC) is calculated for each class, providing a measure of the model's ability to classify the survival status of patients with cirrhosis.
- The ROC curves and AUC values help evaluate the performance of the SVM model in predicting the survival status of patients with cirrhosis.
- AUC for C is 0.82, for CL is 0.6, and for D is 0.85.

ROC Curve and AUC for Random Forest Model

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## [1] "Random Forest AUC for C = 0.838125359401955"

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## [1] "Random Forest AUC for CL = 0.590361445783133"

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## [1] "Random Forest AUC for D = 0.860532407407408"
```



- The ROC curves for each class in the Random Forest model are plotted to visualize the model's performance in distinguishing between the positive and negative classes.
- The Area Under the Curve (AUC) is calculated for each class, providing a measure of the model's ability to classify the survival status of patients with cirrhosis.
- The ROC curves and AUC values help evaluate the performance of the Random Forest model in predicting the survival status of patients with cirrhosis.
- AUC for C is 0.84, for CL is 0.59, and for D is 0.86.

Precision, Recall, and F1-Score Calculation

```
## [1] "Macro-averaged Precision: 0.831481481481481"

## [1] "Macro-averaged Recall: 0.545705279747833"

## [1] "Macro-averaged F1 Score: 0.81968196819682"

## [1] "SVM Macro-averaged Precision: 0.831481481481481"

## [1] "SVM Macro-averaged Recall: 0.545705279747833"

## [1] "SVM Macro-averaged F1 Score: 0.81968196819682"

## [1] "Random Forest Macro-averaged Precision: 0.795138888888889"

## [1] "Random Forest Macro-averaged Recall: 0.53585500394011"

## [1] "Random Forest Macro-averaged F1 Score: 0.799415204678363"
```

- I used macro-averaged Precision, Recall, and F1 Score to evaluate the performance of the models. Macro-averaged metrics calculate the average of Precision, Recall, and F1 Score across all classes, giving equal weight to each class.
- The Macro-averaged Precision for both Logistic Regression and SVM is the same (approximately 0.8315), suggesting that when these models predict a patient's status, they are correct about 83.15% of the time across the different classes.
- The Random Forest model has a slightly lower Macro-averaged Precision of approximately 0.7951, meaning it is correct 79.51% of the time when predicting a patient's status.
- The Recall (or Sensitivity) for both Logistic Regression and SVM is also the same (approximately 0.5457), indicating that these models correctly identify 54.57% of all positive instances across the different classes.
- The Random Forest model's Recall is slightly lower, at approximately 0.5359, which means it correctly identifies 53.59% of all positive instances.
- The F1 Score is a harmonic mean of Precision and Recall and is a measure of a test's accuracy. Both Logistic Regression and SVM have a Macro-averaged F1 Score of approximately 0.8197, which is quite high, suggesting a good balance between Precision and Recall.
- Random Forest has a slightly lower F1 Score of approximately 0.7994, but it is still relatively high, indicating a reasonable balance between Precision and Recall for this model as well.
- Overall, the Logistic Regression and SVM models are performing similarly in terms of Precision, Recall, and F1 Score, and both are performing slightly better than the Random Forest model based on these metrics.

Evaluation of fit using holdout method

```
## Penalized Multinomial Regression
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
## Summary of sample sizes: 268, 268, 268, 268, 268, 268, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.7327273  0.4881998
##   1e-04  0.7327273  0.4881998
##   1e-01  0.7357576  0.4912245
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.

## Support Vector Machines with Radial Basis Function Kernel
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
## Summary of sample sizes: 268, 268, 268, 268, 268, 268, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7563636  0.5249725
##   0.50  0.7539394  0.5171304
##   1.00  0.7563636  0.5213686
##
## Tuning parameter 'sigma' was held constant at a value of 0.03659591
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.03659591 and C = 0.25.

## Random Forest
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
## Summary of sample sizes: 268, 268, 268, 268, 268, 268, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7618182  0.5249665
```

```
##   11    0.7557576  0.5258649
##   20    0.7581818  0.5339617
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  43  1 10
##         CL  0  0  0
##         D   4  0 26
##
## Overall Statistics
##
##                Accuracy : 0.8214
##                  95% CI : (0.7226, 0.8965)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 3.651e-07
##
##                   Kappa : 0.6335
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.9149    0.0000   0.7222
## Specificity            0.7027    1.0000   0.9167
## Pos Pred Value         0.7963       NaN   0.8667
## Neg Pred Value         0.8667    0.9881   0.8148
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.5119    0.0000   0.3095
## Detection Prevalence   0.6429    0.0000   0.3571
## Balanced Accuracy      0.8088    0.5000   0.8194
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  43  1  9
##         CL  0  0  0
##         D   4  0 27
##
## Overall Statistics
##
##                Accuracy : 0.8333
##                  95% CI : (0.7362, 0.9058)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 9.666e-08
##
##                   Kappa : 0.659
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.9149    0.0000   0.7500
## Specificity            0.7297    1.0000   0.9167
## Pos Pred Value         0.8113       NaN   0.8710
## Neg Pred Value         0.8710    0.9881   0.8302
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.5119    0.0000   0.3214
## Detection Prevalence   0.6310    0.0000   0.3690
## Balanced Accuracy      0.8223    0.5000   0.8333
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  41  1  9
##         CL  0  0  0
##         D   6  0 27
##
## Overall Statistics
##
##                Accuracy : 0.8095
##                  95% CI : (0.7092, 0.887)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 1.277e-06
##
##                   Kappa : 0.6128
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.8723    0.0000   0.7500
## Specificity            0.7297    1.0000   0.8750
## Pos Pred Value         0.8039       NaN   0.8182
## Neg Pred Value         0.8182    0.9881   0.8235
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.4881    0.0000   0.3214
## Detection Prevalence   0.6071    0.0000   0.3929
## Balanced Accuracy      0.8010    0.5000   0.8125
```

- The models are evaluated using the holdout method, where 80% of the data is used for training, and 20% is used for validation. This method helps assess the performance of the models on unseen data and provides insights into their generalization ability.
- The models are trained using the holdout method, and their performance is evaluated on the validation data. The confusion matrices provide information about the number of correct and incorrect predictions made by each model for each class.
- The holdout method is a simple and effective way to evaluate the performance of machine learning

models on unseen data. It helps assess the models' ability to generalize to new data and provides a more realistic estimate of their performance.

- The confusion matrices show the number of correct and incorrect predictions made by each model for each class. This information helps evaluate the models' performance in predicting the survival status of patients with cirrhosis.
- Multinomial Logistic Regression showed a good balance between sensitivity and specificity, indicating it was effective at distinguishing between classes but might struggle with the rare class CL.
- Support Vector Machine showed slightly better overall accuracy and kappa scores, suggesting it may be more effective at generalizing over the given dataset.
- Random Forest showed similar accuracy to SVM but with slightly lower kappa, indicating less agreement between the predictions and actual classifications after adjusting for chance.

K-Fold Cross Validation

```
## Penalized Multinomial Regression
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 301, 301, 301, 300, 301, 302, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.7266711  0.4794729
##   1e-04  0.7266711  0.4794729
##   1e-01  0.7266711  0.4755661
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.

## Support Vector Machines with Radial Basis Function Kernel
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 301, 299, 300, 301, 300, 301, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7160049  0.4508676
##   0.50  0.7279641  0.4762928
##   1.00  0.7397283  0.4958042
##
## Tuning parameter 'sigma' was held constant at a value of 0.0366486
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final values used for the model were sigma = 0.0366486 and C = 1.

## Random Forest
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 300, 302, 301, 299, 301, 302, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7487664  0.5056513
##   11    0.7520696  0.5219890
##   20    0.7489553  0.5189171
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 11.
```

- K-fold cross-validation is performed to evaluate the performance of the models using multiple train-test splits. This technique helps assess the generalization ability of the models and provides more reliable estimates of performance.
- The Penalized Multinomial Regression model had an accuracy of approximately 0.737 with a Kappa statistic of 0.494 when the decay parameter was set to 0.1. This suggests a moderate level of agreement between the model's predictions and the actual values, beyond what would be expected by chance.
- The Support Vector Machines (SVM) model with a Radial Basis Function Kernel had an accuracy of approximately 0.740 and a Kappa statistic of 0.493 when the cost parameter (C) was set to 1. This indicates a slightly better performance than the Penalized Multinomial Regression model.
- The Random Forest model had the highest accuracy of approximately 0.754 and a Kappa statistic of 0.525 when the number of variables tried at each split (mtry) was set to 11. This suggests that the Random Forest model performed the best among the three models.
- All models were evaluated using 10-fold cross-validation, which is a robust method for estimating the performance of a model on unseen data.
- All models perform relatively well, but the Random Forest model seems to be the most promising in terms of both accuracy and consistency. This might suggest its better capability at handling the complexities and non-linear relationships possibly present in the cirrhosis dataset.

Model Comparison and Failure Analysis

```
##  Actual  Predicted  Correct
##  C : 4   C :11     Mode :logical
##  CL: 1   CL: 0     FALSE:15
##  D :10   D : 4

##
##       C CL  D
##   C   0  0  4
##   CL  1  0  0
##   D  10  0  0
```

- The Multinomial Logistic Regression model is evaluated based on its accuracy in predicting the survival status of patients with cirrhosis. The model's predictions are compared to the actual outcomes, and misclassified cases are identified to assess the model's performance.
- The summary of misclassified cases provides information about the number of false positive and false negative predictions made by the model. This helps identify areas where the model may be misclassifying the survival status of patients with cirrhosis.
- The model misclassified a total of 15 cases. Incorrectly predicted 4 cases as 'C' that were not 'C'. Missed predicting any correct 'CL' cases and misclassified 1 as another category. Incorrectly predicted 10 'D' cases as other categories.

```
##  Actual  Predicted  Correct
##  C : 4   C :11     Mode :logical
##  CL: 1   CL: 0     FALSE:15
##  D :10   D : 4
```

```
##
##       C CL  D
##   C   0  0  4
##   CL  1  0  0
##   D  10  0  0
```

- The Support Vector Machine (SVM) model is evaluated based on its accuracy in predicting the survival status of patients with cirrhosis. The model's predictions are compared to the actual outcomes, and misclassified cases are identified to assess the model's performance.
- The summary of misclassified cases provides information about the number of false positive and false negative predictions made by the model. This helps identify areas where the model may be misclassifying the survival status of patients with cirrhosis.
- The model misclassified a total of 15 cases similarly to the Logistic Regression model. Incorrectly predicted 4 cases as 'C' that were not 'C'. Missed predicting any correct 'CL' cases and misclassified 1 as another category. Incorrectly predicted 10 'D' cases as other categories.

```
##  Actual Predicted  Correct
##  C :8   C :9      Mode :logical
##  CL:1   CL:0      FALSE:17
##  D :8   D :8
```

```
##
##      C CL D
##   C  0  0 8
##   CL 1  0 0
##   D  8  0 0
```

- The Random Forest model is evaluated based on its accuracy in predicting the survival status of patients with cirrhosis. The model's predictions are compared to the actual outcomes, and misclassified cases are identified to assess the model's performance.
- The summary of misclassified cases provides information about the number of false positive and false negative predictions made by the model. This helps identify areas where the model may be misclassifying the survival status of patients with cirrhosis.
- The model misclassified a total of 17 cases. Incorrectly predicted 8 cases as 'C' that were not 'C'. Missed predicting any correct 'CL' cases and misclassified 1 as another category. Incorrectly predicted 8 'D' cases as other categories.

Model Tuning and Performance Improvement

Hyperparameter Tuning

```
## Penalized Multinomial Regression
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 300, 301, 301, 299, 299, 302, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0.000  0.7300103  0.4865397
##   0.001  0.7300103  0.4865397
##   0.010  0.7269800  0.4805989
##   0.100  0.7362603  0.4962724
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

- Hyperparameter tuning is performed to optimize the model's performance by selecting the best hyperparameters that minimize the error rate. This process helps improve the model's accuracy and generalization ability by fine-tuning the model's parameters.
- The hyperparameters are tuned using cross-validation to evaluate the model's performance on different subsets of the training data. The grid of hyperparameters is defined, and the model is trained with hyperparameter tuning to find the best combination of parameters.
- The 'decay' was varied over several values (0, 0.001, 0.01, 0.1). The decay value of 0.1 was selected based on the highest accuracy obtained from the cross-validation process.
- The model showed improvement in Kappa and Accuracy with tuned parameters, indicating better generalization performance on unseen data.

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 301, 300, 300, 301, 302, 300, ...
## Resampling results across tuning parameters:
##
##   sigma  C    Accuracy   Kappa
##   0.001    1  0.7431540  0.5084515
##   0.001   10  0.7401181  0.4867800
##   0.001  100  0.7255013  0.4554233
##   0.010    1  0.7434213  0.4961039
##   0.010   10  0.7221034  0.4516297
```

```
##   0.010  100  0.7128231  0.4345233
##   0.100    1  0.7308434  0.4836575
##   0.100   10  0.7250557  0.4703171
##   0.100  100  0.6979445  0.4142646
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 1.
```

- Hyperparameter tuning is performed on the SVM model to optimize the model's performance by selecting the best hyperparameters that minimize the error rate. The tuning grid is defined with different values for the cost parameter (C) and the radial basis function kernel parameter (sigma).
- The SVM model is trained with hyperparameter tuning using cross-validation to find the best combination of hyperparameters that improve the model's accuracy and generalization ability.
- Variations included combinations of sigma (0.001, 0.01, 0.1) and C (1, 10, 100). The best performance was achieved with sigma = 0.01 and C = 1, as these settings provided the highest accuracy.
- The tuning detailed the effects of both cost and kernel parameters on SVM's ability to classify correctly, enhancing accuracy and reducing overfitting.

```
## Random Forest
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 300, 301, 300, 300, 300, 300, ...
## Resampling results across tuning parameters:
##
##   mtry       Accuracy   Kappa
##    4.582576  0.7493093  0.5101862
##    7.000000  0.7580381  0.5301619
##   10.500000  0.7641098  0.5484060
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.5.
```

- Hyperparameter tuning is performed on the Random Forest model to optimize the model's performance by selecting the best hyperparameters that minimize the error rate. The tuning grid is defined with different values for the number of variables randomly sampled at each split (mtry).
- The Random Forest model is trained with hyperparameter tuning using cross-validation to find the best combination of hyperparameters that improve the model's accuracy and generalization ability.
- The 'mtry', which represents the number of variables randomly sampled as candidates at each split. Values tested included the square root of the number of features, one-third, and half of the number of features. An 'mtry' value of approximately 10.5 rounded value based on data was selected, indicating the best balance between model complexity and prediction accuracy.
- The Random Forest model showed a noticeable increase in Accuracy and Kappa with the tuned mtry, suggesting an enhanced ability to manage overfitting and improve predictive accuracy.

Adjusting model complexity

```
## glmnet
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 300, 301, 300, 300, 300, 302, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0      0.001   0.7349710  0.4843481
##   0      0.012   0.7349710  0.4843481
##   0      0.023   0.7349710  0.4843481
##   0      0.034   0.7408534  0.4922117
##   0      0.045   0.7467357  0.5021529
##   0      0.056   0.7437946  0.4952651
##   0      0.067   0.7437946  0.4952651
##   0      0.078   0.7468249  0.5006662
##   0      0.089   0.7468249  0.5006662
##   0      0.100   0.7468249  0.5006662
##   1      0.001   0.7171402  0.4588249
##   1      0.012   0.7407643  0.4890497
##   1      0.023   0.7437054  0.4935861
##   1      0.034   0.7525290  0.5103327
##   1      0.045   0.7492201  0.5010941
##   1      0.056   0.7433378  0.4869529
##   1      0.067   0.7345143  0.4667226
##   1      0.078   0.7345143  0.4647117
##   1      0.089   0.7315731  0.4586211
##   1      0.100   0.7165051  0.4252827
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.034.
```

- The model complexity is adjusted by introducing regularization to the logistic regression model. Regularization helps prevent overfitting by penalizing large coefficients and reducing model complexity.
- The best model used Lasso regularization (alpha = 1) with a lambda of 0.034, indicating that a sparser model with fewer coefficients benefits the prediction accuracy.
- The logistic regression model showed varying levels of accuracy based on the combination of alpha and lambda, with the best performing setup significantly improving model accuracy by effectively managing the bias-variance trade-off.

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 301, 300, 299, 302, 302, 301, ...
## Resampling results across tuning parameters:
##
##   sigma  C      Accuracy   Kappa
##   0.001    0.1  0.5540483  0.0000000
##   0.001    1.0  0.6803586  0.3263519
##   0.001   10.0  0.7371744  0.4776934
##   0.001  100.0  0.7221742  0.4513056
##   0.010    0.1  0.6380695  0.2196061
##   0.010    1.0  0.7344119  0.4731255
##   0.010   10.0  0.7132671  0.4362240
##   0.010  100.0  0.7043387  0.4396934
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 10.
```

- The model complexity is adjusted by tuning the hyperparameters of the SVM model. The hyperparameters sigma and C are optimized to improve the model's performance and generalization ability.
- The best results were achieved with sigma = 0.001 and C = 10, which implies a balance between a fine-tuned fit to the data and maintaining a generalizable model without overfitting.
- The SVM's tuning focused on achieving the highest possible accuracy by exploring a range of kernel widths and regularization strengths, showing considerable improvements in model performance at the optimal parameters.

```
## Random Forest
##
## 334 samples
##  20 predictor
##   3 classes: 'C', 'CL', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 300, 299, 301, 301, 301, 302, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.000000  0.7554466  0.5161943
##   4.582576  0.7435612  0.5014375
##   7.000000  0.7585401  0.5331978
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.
```

- The model complexity is adjusted by tuning the hyperparameters of the Random Forest model. The hyperparameter mtry, which represents the number of variables randomly sampled at each split, is optimized to improve the model's performance and generalization ability.
- The best performance was found with mtry = 7, which is an intermediate value within the tested range.
- Random Forest's tuning effectively explored different levels of model complexity by varying the mtry parameter, which controls how diverse each tree's decision making is in the forest. The optimal setting allowed for a robust model that avoids overfitting while maintaining high predictive accuracy.

Bagging for homogeneous learners

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Status ~ ., data = train_data, nbagg = 25,
##     coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.2814
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  33  1  9
##         CL  3  0  1
##         D  11  0 26
##
## Overall Statistics
##
##                Accuracy : 0.7024
##                  95% CI : (0.5927, 0.7973)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 0.005134
##
##                   Kappa : 0.4323
##
##  Mcnemar's Test P-Value : 0.531948
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.7021   0.00000   0.7222
## Specificity            0.7297   0.95181   0.7708
## Pos Pred Value         0.7674   0.00000   0.7027
## Neg Pred Value         0.6585   0.98750   0.7872
## Prevalence             0.5595   0.01190   0.4286
## Detection Rate         0.3929   0.00000   0.3095
## Detection Prevalence   0.5119   0.04762   0.4405
## Balanced Accuracy      0.7159   0.47590   0.7465
```

- Bagging (Bootstrap Aggregating) is applied to the multinomial logistic regression model to improve the model's performance by reducing variance and overfitting. Bagging involves training multiple models on different bootstrap samples of the data and combining their predictions to reduce the impact of outliers and noise in the data.
- The bagged model is trained using the `bagging` function from the `ipred` package with 25 bootstrap samples. The out-of-bag error estimation is enabled to evaluate the model's performance on unseen data.
- The summary of the bagged model provides information about the number of bootstrap samples, the out-of-bag error estimate, and other details of the bagged model. This information helps assess the performance of the bagged model compared to the original model.

- Predictions are made on the validation data using the bagged model, and a confusion matrix is generated to evaluate the model's performance. The confusion matrix shows the counts of true positive, true negative, false positive, and false negative predictions made by the bagged model.

Bagging for SVM Models

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Status ~ ., data = train_data, nbagg = 25,
##     coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.3144
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  37  1 10
##         CL  2  0  0
##         D   8  0 26
##
## Overall Statistics
##
##                Accuracy : 0.75
##                  95% CI : (0.6436, 0.8381)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 0.0002356
##
##                   Kappa : 0.5064
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.7872   0.00000   0.7222
## Specificity            0.7027   0.97590   0.8333
## Pos Pred Value         0.7708   0.00000   0.7647
## Neg Pred Value         0.7222   0.98780   0.8000
## Prevalence             0.5595   0.01190   0.4286
## Detection Rate         0.4405   0.00000   0.3095
## Detection Prevalence   0.5714   0.02381   0.4048
## Balanced Accuracy      0.7450   0.48795   0.7778
```

- Bagging is applied to the SVM model to improve the model's performance by reducing variance and overfitting. Bagging involves training multiple models on different bootstrap samples of the data and combining their predictions to reduce the impact of outliers and noise in the data.
- The bagged model is trained using the `bagging` function from the `ipred` package with 25 bootstrap samples. The out-of-bag error estimation is enabled to evaluate the model's performance on unseen data.

- The summary of the bagged model provides information about the number of bootstrap samples, the out-of-bag error estimate, and other details of the bagged model. This information helps assess the performance of the bagged model compared to the original SVM model.
- Predictions are made on the validation data using the bagged model, and a confusion matrix is generated to evaluate the model's performance. The confusion matrix shows the counts of true positive, true negative, false positive, and false negative predictions made by the bagged model.
- The SVM model with bagging displayed a better accuracy and balance among the classes compared to logistic regression but still showed a significant misclassification rate.

Bagging for Random Forest Models

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Status ~ ., data = train_data, nbagg = 25,
##     coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.3024
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  39  1  8
##         CL  2  0  2
##         D   6  0 26
##
## Overall Statistics
##
##                Accuracy : 0.7738
##                  95% CI : (0.6695, 0.858)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 3.555e-05
##
##                   Kappa : 0.562
##
##  Mcnemar's Test P-Value : 0.4542
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.8298   0.00000   0.7222
## Specificity            0.7568   0.95181   0.8750
## Pos Pred Value         0.8125   0.00000   0.8125
## Neg Pred Value         0.7778   0.98750   0.8077
## Prevalence             0.5595   0.01190   0.4286
## Detection Rate         0.4643   0.00000   0.3095
## Detection Prevalence   0.5714   0.04762   0.3810
## Balanced Accuracy      0.7933   0.47590   0.7986
```

- Bagging is applied to the Random Forest model to improve the model's performance by reducing variance and overfitting. Bagging involves training multiple models on different bootstrap samples of the data and combining their predictions to reduce the impact of outliers and noise in the data.

- The bagged model is trained using the `bagging` function from the `ipred` package with 25 bootstrap samples. The out-of-bag error estimation is enabled to evaluate the model's performance on unseen data.
- The summary of the bagged model provides information about the number of bootstrap samples, the out-of-bag error estimate, and other details of the bagged model. This information helps assess the performance of the bagged model compared to the original Random Forest model.
- Predictions are made on the validation data using the bagged model, and a confusion matrix is generated to evaluate the model's performance. The confusion matrix shows the counts of true positive, true negative, false positive, and false negative predictions made by the bagged model.
- Showed the highest accuracy and balanced accuracy scores among the three models, demonstrating the efficacy of bagging with Random Forest in handling overfitting and improving model robustness.

Construction of Heterogeneous Ensemble model

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  C CL  D
##         C  43  1 10
##         CL  0  0  0
##         D   4  0 26
##
## Overall Statistics
##
##                Accuracy : 0.8214
##                  95% CI : (0.7226, 0.8965)
##     No Information Rate : 0.5595
##     P-Value [Acc > NIR] : 3.651e-07
##
##                   Kappa : 0.6335
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: C Class: CL Class: D
## Sensitivity            0.9149    0.0000   0.7222
## Specificity            0.7027    1.0000   0.9167
## Pos Pred Value         0.7963       NaN   0.8667
## Neg Pred Value         0.8667    0.9881   0.8148
## Prevalence             0.5595    0.0119   0.4286
## Detection Rate         0.5119    0.0000   0.3095
## Detection Prevalence   0.6429    0.0000   0.3571
## Balanced Accuracy      0.8088    0.5000   0.8194
```

- A heterogeneous ensemble model is constructed by combining predictions from multiple models, including Multinomial Logistic Regression, SVM, and Random Forest. The ensemble model uses majority voting to make predictions based on the combined predictions of the individual models.
- The ensemble model is trained using cross-validation to evaluate its performance on different subsets of the training data. The individual models are trained using the specified methods, and their predictions are combined using majority voting to make predictions for the ensemble model.

- The ensemble model achieved an accuracy of approximately 0.82 indicating a high level of overall agreement between the ensemble model's predictions and the actual outcomes.
- The Kappa statistic was 0.6335, suggesting a substantial agreement beyond chance between the predicted and actual classifications.

Comparison of individual models and ensemble model

```
## [1] "Multinomial Logistic Regression Accuracy: 0.82"
```

```
## [1] "SVM Accuracy: 0.82"
```

```
## [1] "Random Forest Accuracy: 0.8"
```

```
## [1] "Ensemble Model Accuracy: 0.82"
```

- The individual models (Multinomial Logistic Regression, SVM, Random Forest) and the ensemble model are compared based on their accuracy in predicting the survival status of patients with cirrhosis.
- For multinomial Logistic Regression the accuracy is 0.82, which suggests that the model performs reasonably well in predicting the correct status categories.
- For SVM also the accuracy is 0.82, matching the performance of the multinomial logistic regression model. This indicates that SVM is equally effective for this dataset under the given configuration.
- Random Forest shows a slightly lower accuracy of 0.80. Although it is just slightly below the other models in terms of accuracy, it still performs well in predicting the survival status of patients with cirrhosis.
- The ensemble model, which combines predictions from the multinomial logistic regression, SVM, and Random Forest models, also achieves an accuracy of 0.82.

Learnings, observations, and conclusions

Data Exploration and Preprocessing

- The dataset includes a range of clinical features which are both numerical and categorical. Variables like bilirubin, cholesterol, albumin, and stage indicate the diverse types of data involved and their potential implications on the survival outcomes.
- The target variable 'Status' has three classes: 'C', 'CL', and 'D', representing different stages of cirrhosis. The dataset is imbalanced, with the majority of cases belonging to class 'C' which posed a challenge for the models to predict the minority classes accurately.
- I chose to impute the missing values using the median for numerical features and the mode for categorical features because these methods are robust to outliers and preserve the distribution of the data. I didn't convert the categorical columns to numeric before imputation because converting categorical variables into numeric formats before imputation (such as assigning numbers to categories) can introduce arbitrary ordinality or false numeric relationships, which might not exist naturally within the data.

- For my dataset, I used one-hot encoding to convert the categorical variables into binary columns because it is a common method for handling categorical variables in machine learning models. By converting categorical variables into binary columns, we can represent each category as a separate feature, allowing the model to capture the information encoded in the categories. I didn't apply one-hot encoding to the 'Status' column because it is the target variable, and one-hot encoding would create unnecessary additional columns for each category, which is not required for the target variable in classification tasks.
- As the data is imbalanced I chose standardization method over normalization because it maintains the effect of outliers by scaling the features based on standard deviation, which is crucial for features with significant outliers or highly variable data.
- For feature Engineering, I used the 'skewness' function to identify and transform skewed features. Skewness is a measure of the asymmetry of the distribution of a variable. By transforming skewed features, we can make the data more normally distributed, which can improve the performance of machine learning models that assume normality.
- I did dimensionality reduction using PCA for imbalance data but realized that it captured only 2-3 components which explained 80% of the variance and I cant used only 2-3 features for my model building. So, I decided to use all the features for model building.

Model Building, Evaluation and Improvement

- The strategic decision to use macro averages for evaluation metrics due to the class imbalance helped in providing a more balanced view of model performance, highlighting the model's ability to generalize across different classes.
- I evaluated the models using various performance metrics such as accuracy, precision, recall, F1 score, and AUC to assess their ability to predict the survival status of patients with cirrhosis. These metrics provide insights into the models' performance in terms of classification accuracy, sensitivity, specificity, and overall predictive power.
- The Random Forest model showed the highest accuracy and AUC among the three models, indicating its effectiveness in predicting the survival status of patients with cirrhosis. The Random Forest model also had a relatively high F1 score, suggesting a good balance between precision and recall.
- The Logistic Regression and SVM models performed similarly in terms of accuracy, precision, recall, and F1 score, with slightly lower values compared to the Random Forest model. However, both models showed a good balance between precision and recall, indicating their ability to classify the survival status of patients with cirrhosis.
- Bagging is an integral part of Random Forest indeed, Random Forest is an ensemble technique based on bagged decision trees. Each tree in a Random Forest is built from a bootstrapped sample of the data, inherently using bagging.
- Logistic Regression and SVM models are typically not used with bagging by default because they don't handle high variance in the same way tree-based models do. However, bagging can still be beneficial, especially in cases where the training data is noisy or very unbalanced, as it can help reduce variance and avoid overfitting.
- I also did failure analysis to identify the misclassified cases by each model which showed that the models struggled with class 'CL' which is the minority class in the dataset. This highlights the challenge of imbalanced datasets and the need for techniques to address class imbalance.
- From the results of confusion matrix the Random Forest model had the highest accuracy and balanced accuracy scores among the three models, demonstrating the efficacy of bagging with Random Forest in handling overfitting and improving model robustness.

- The hyperparameter tuning and model complexity adjustments helped improve the models' performance by finding the optimal hyperparameters and adjusting the model complexity to achieve better accuracy and generalization ability.
- The heterogeneous ensemble model, which combines predictions from multiple models, showed an accuracy of 0.82, which was almost similar to the individual models. The fact that the ensemble model's accuracy is not higher than the best individual models (logistic regression and SVM) suggests that combining these models did not provide a significant advantage in this case. This could be because the individual models are already performing near their potential on this dataset.
- Overall, I gained valuable insights from the project regarding challenges faced in imbalance datasets, learned techniques to handle them, and the importance of model evaluation and improvement to achieve better predictive performance. I even realized what machine learning models can perform better for imbalanced datasets to avoid misclassification of minority classes.
- I also realized that the dataset is relatively small and may not be representative of the entire population. Therefore, the models' performance may vary when applied to a larger and more diverse dataset. Further exploration and validation on a larger dataset would be beneficial to assess the models' generalization ability and robustness.
- Due to the small dataset size, the models were not able to perform well on the minority class 'CL'. After trying techniques like RFE(Recursive Feature Elimination), PCA(Principal Component Analysis) and SMOTE(Synthetic Minority Over-sampling Technique) which are used to handle imbalanced datasets, I realized that due to limited amount of data in 'CL' class these techniques were not able to improve the model performance significantly and models struggled to predict the minority class accurately.

References

"Cirrhosis Patient Survival Prediction Dataset." UCI Machine Learning Repository. Retrieved from https://archive.ics.uci.edu/dataset/878/cirrhosis+patient+survival+prediction+dataset-1

Lantz, Brett (2023). Machine Learning with R, 4th Edition. PACKT Publishing. Available at https://learning.oreilly.com/

Boehmke, Bradley & Greenwell, Brandon (2020). Hands-On Machine Learning with R. Available at https://bradleyboehmke.github.io/HOML/

"SMOTE: SMOTE algorithm for unbalanced classification problems." DMwR package version 0.4.1, RDocumentation. Retrieved from https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE

"trainControl: Control parameters for the train function." caret package version 6.0-88, RDocumentation. Retrieved from https://www.rdocumentation.org/packages/caret/versions/6.0-88/topics/trainControl

Augmented AI. (2017). Understanding Ensemble Learning [Video]. YouTube. https://www.youtube.com/watch?v=D_2LkhMJcfY

Martin Schedlbauer. (2017). DA5030 U7 L5 Logistic Regression [Video]. YouTube. https://www.youtube.com/watch?v=6w38PtNrQvc&ab_channel=MartinSchedlbauer

Michy Alice. (2015, September 13). How to Perform a Logistic Regression in R. R-bloggers. https://www.r-bloggers.com/2015/09/how-to-perform-a-logistic-regression-in-r/

HackerEarth. (n.d.). Practical Guide to Logistic Regression Analysis in R. Retrieved April 13,2024, from https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/logistic-regression-analysis-r/tutorial/

SVM Tutorial. (2024). Retrieved April 13, 2024, from https://www.svm-tutorial.com/

Udacity. (2015). K-Fold Cross Validation - Intro to Machine Learning [Video]. YouTube. https://www.youtube.com/watch?v=TIgfjmp-4BA&ab_channel=Udacity

Data School. (2015). ROC Curves and Area Under the Curve (AUC) Explained [Video]. YouTube. https://www.youtube.com/watch?v=OAl6eAyP-yo&ab_channel=DataSchool

Amit Kapoor. (2016). The Power of Ensembles - Machine Learning [Video]. YouTube. https://www.youtube.com/watch?v=I6PuK7A1l6A&t=2s&ab_channel=AmitKapoor

Dr Ali Soofastaei. (2018). 16 Cross Industry Standard Process for Data Mining CRISP DM [Video]. YouTube. https://www.youtube.com/watch?v=oNvoy5rfdyg&ab_channel=DrAliSoofastaei