Final Project Report - An exploration of Biliary Cirrhosis and Treatments

Background

What is Biliary Cholangitis?

Primary Biliary Cholangitis (previously called Primary Biliary Cirrhosis) is an autoimmune disease where bile ducts become swollen and inflamed and block the flow of bile. Bile is a substance that aids with digestion. The bile ducts carry bile from the liver to the small intestine. The swelling and inflammation can lead to scarring of the liver which is cirrhosis. Advanced cirrhosis can lead to liver failure or liver cancer. Medication can slow progression. There is no definite cure at this time.

Why is the data important to us?

Both of us have been very interetsted in the intersection of medicine and machine learning. The idea that the concepts we are learning in our DS majors could have such an impact on the health industry was really special. Another important aspect was that there is so much potential for advancement of medicine, we both agreed that this area is MLs most promising aspect. The cirrhosis data set perticularly stood out to us because the data set included so many interesting attributes and we anted to see how they are connected with the outcomes of the patient.

The data set also includes some important limitations. Most importantly, there are only 418 cases, greatly affecting the types of analysis we could do. This shows realm world problems data scientists face. Thus, our goal is to use our Statistical DS skillset to approach this problem, despite the limitations.

Objectives

Exploring the study, we concluded on two core questions we hope to address in this report:

CRQ1: What predictors, if any affected the status of the patient at the end of the study?

CRQ2: Are there factors not currently included in medical definition that can help distinguish different stages of Biliary Cholangitis?

Clinical Study + Dataset Background :

The dataset is produced from a clinical study by Mayo Clinic run from 1974-1984. The final 'Status' of each patient was observed in 1986. 'Status' was either Dead, Censored, or Censored due to liver transplant. A link to the original study is here: https://faculty.washington.edu/abansal/ShortCourse_DynamicDecisionMaking/Dickson1989_MayoPBCOriginalArticle.pdf

The data contains the following attributes (Data Source):

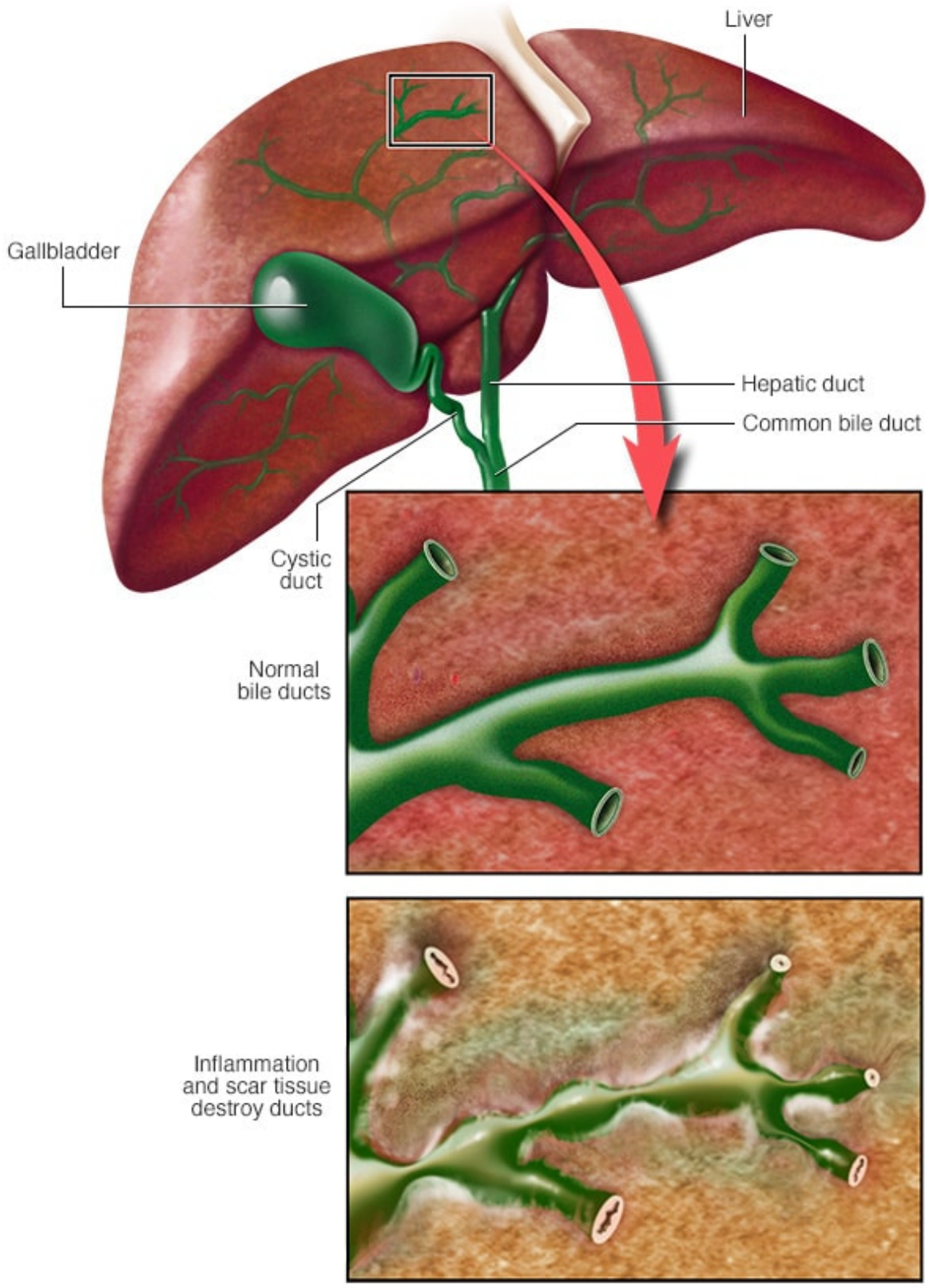


Figure 1: Primary biliary cholangitis (https://www.mayoclinic.org/diseases-conditions/primary-biliary-cholangitis/symptoms-causes/syc-20376874)

1) **ID**: unique identifier

2) **N_Days**: number of days between registration and the earlier of death, transplantation, or study analysis time in July 1986

3) **Status**: status of the patient C (censored), CL (censored due to liver tx), or D (death)

4) **Drug**: type of drug D-penicillamine or placebo

5) **Age**: age in [days]

6) **Sex**: M (male) or F (female)

7) **Ascites**: presence of ascites N (No) or Y (Yes)

8) **Hepatomegaly**: presence of hepatomegaly N (No) or Y (Yes)

9) **Spiders**: presence of spiders N (No) or Y (Yes)

10) **Edema**: presence of edema N (no edema and no diuretic therapy for edema), S (edema present without diuretics, or edema resolved by diuretics), or Y (edema despite diuretic therapy)

11) **Bilirubin**: serum bilirubin in [mg/dl]

12) **Cholesterol**: serum cholesterol in [mg/dl]

13) **Albumin**: albumin in [gm/dl]

14) **Copper**: urine copper in [ug/day]

15) **Alk_Phos**: alkaline phosphatase in [U/liter]

16) **SGOT**: SGOT in [U/ml]

17) **Triglycerides**: triglicerides in [mg/dl]

18) **Platelets**: platelets per cubic [ml/1000]

19) **Prothrombin**: prothrombin time in seconds [s]

20) **Stage**: histologic stage of disease (1, 2, 3, or 4)

For the basic pre-processing, we binary-encoded the categorical variables in the data set and set them as factors.

Goals

Our goal is to explore the data through multiple approaches including regression, classification and clustering. Let us frist clear some definitions and differences overall.

Unsupervised vs Supervised learning

- **Supervised Learning:** Involves training a model on a labeled dataset, where the correct answers are provided, allowing the model to learn the relationships between the input data and output labels.

- **Unsupervised Learning:** Involves training a model on data without labeled responses, enabling the model to identify patterns and structures in the data on its own.

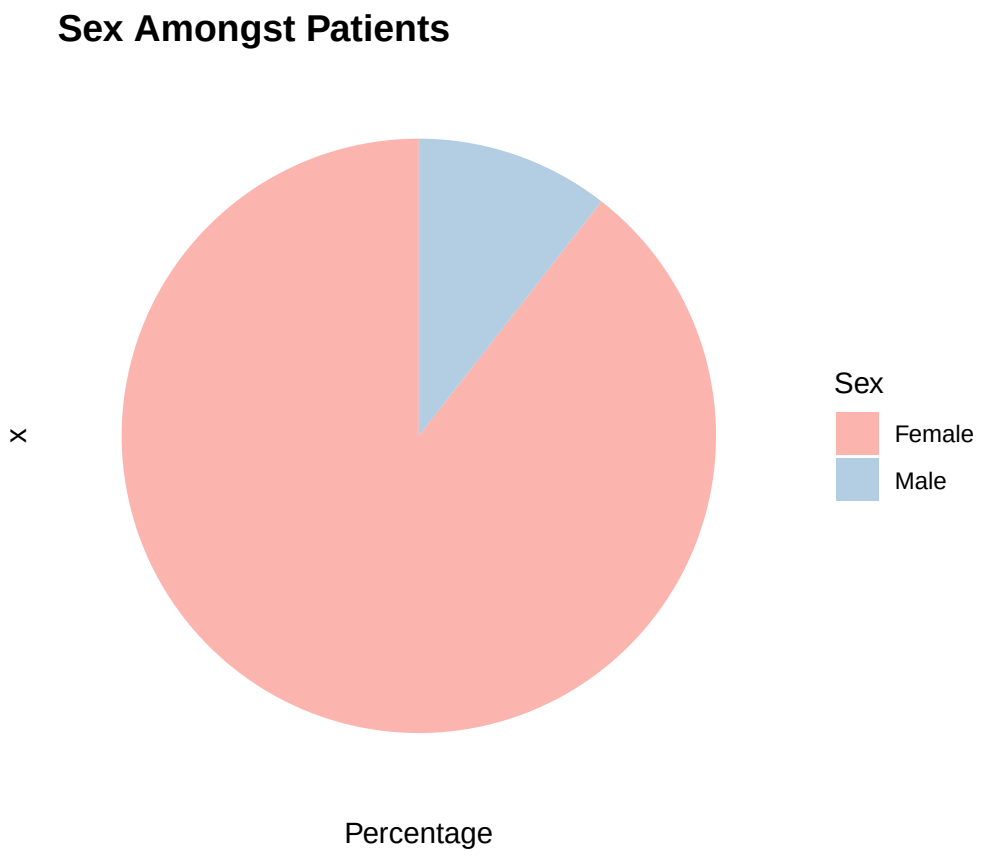
Regression vs Clustering

- **Regression Learning**: A type of machine learning where the model predicts a continuous output. For example, predicting house prices based on features like size and location.
- **Classification Learning**: A type of machine learning where the model categorizes data into discrete classes. For example, determining if an email is spam or not based on its content.

Exploratory Data Analysis

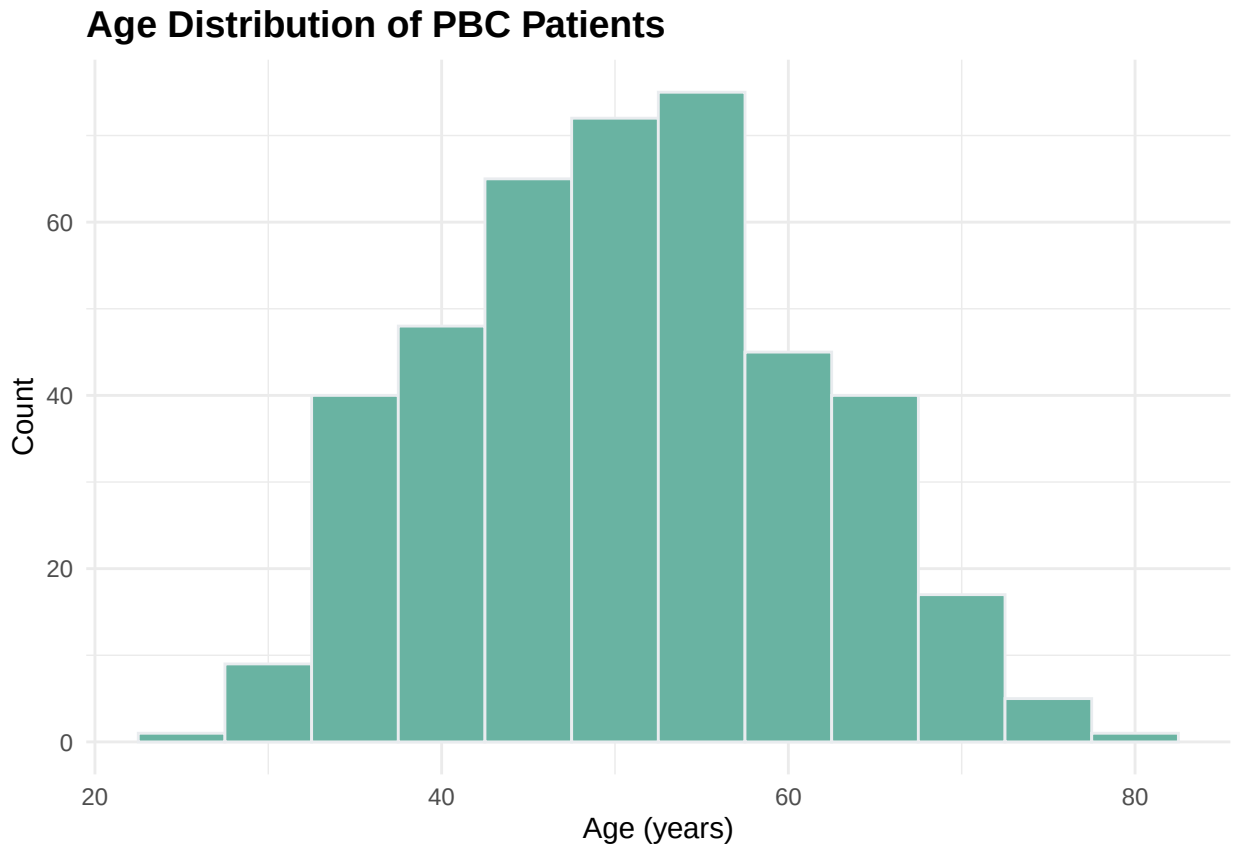
Part 1: Exploring the demographics of the patients

Sex Ratio of Patients



Primary Biliary Cholangitis affects women much more than men at a 10-1 ratio. The data of this clinical study mimics the greater population. Investigating gender-based differences in disease progression can uncover any gender-specific patterns in PBC. This could lead to gender-tailored treatment approaches and a better understanding of the disease's biology, which might differ between males and females.

Age Distribution of Patients



The distribution of ages is quite normal.

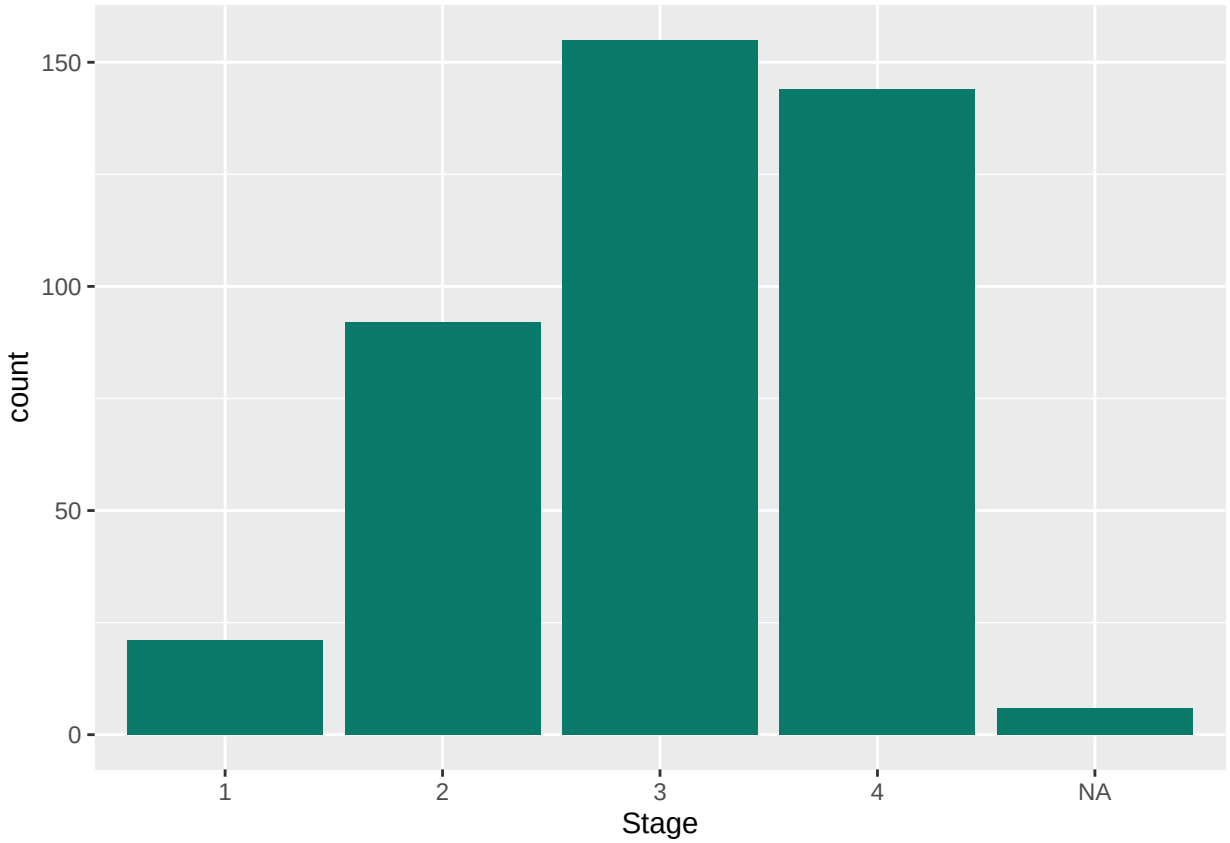
Part 2: Looking at the data itself

NA counts amongst our variables

	NA_CountsPerVar
N_Days	0
Status	0
Drug	106
Age	0
Sex	0
Ascites	106
Hepatomegaly	106
Spiders	106
Edema	0
Bilirubin	0
Cholesterol	134
Albumin	0
Copper	108
Alk_Phos	106
SGOT	106
Tryglicerides	136
Platelets	11
Prothrombin	2
Stage	6
Age_Years	0

An observation is that many of the variables have 106 NAs. This indicates that a good fraction [106 patients] could have been equally tracked and measured in a less extensive way.

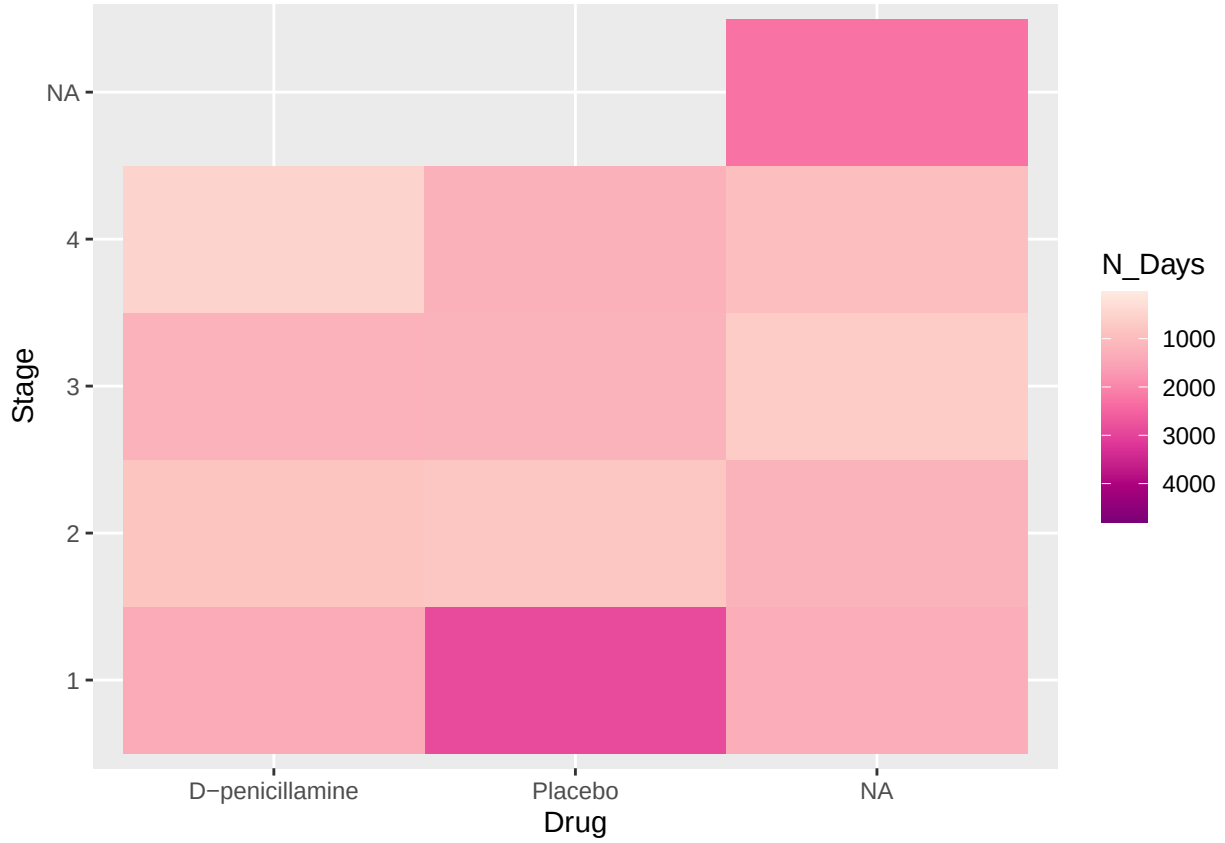
Distribution of Stages of Biliary Cholangitis



This distribution is left skewed and not symmetric. Most patients have stage 3 and then 4 of Biliary Cholangitis.

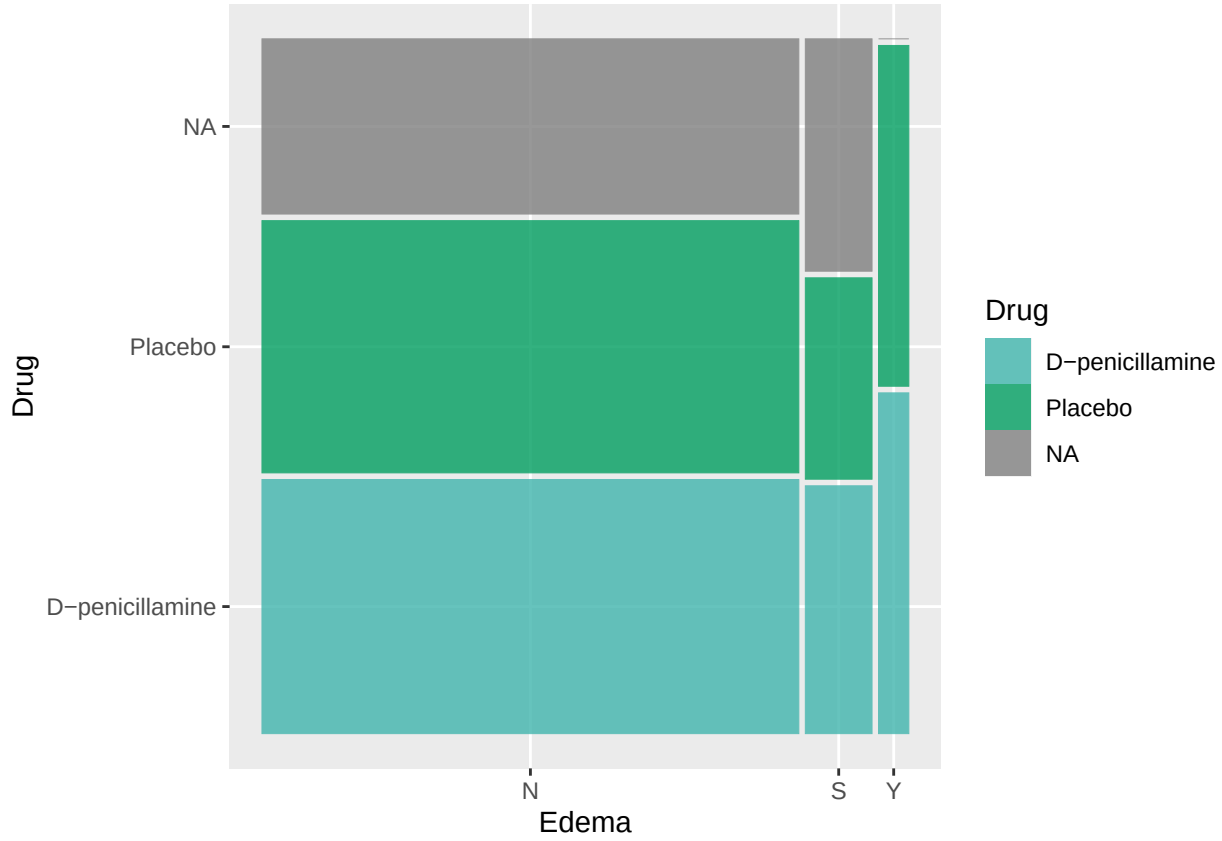
Part 3: Exploring some interactions of variables

How does prescense of a D-penicillamine with stage of Biliary Cholangitis impact number of days till death?



This heat map shows that being a placebo in stage 1 gives you a greater amount of days till death in this sample of patients.

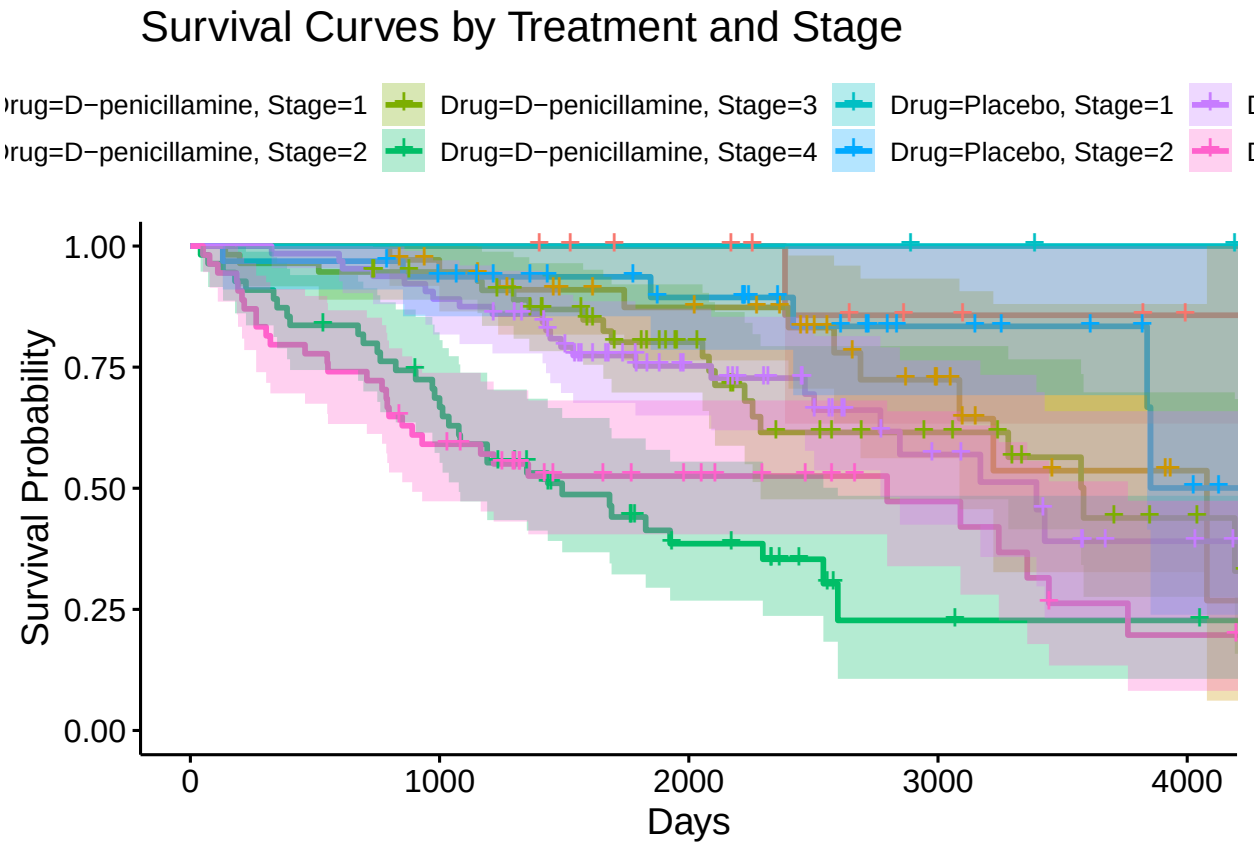
Individuals with Edema, and if they were on the drug



Edema is swelling due to too much liquid trapped in the body's tissues. It's common complication in liver diseases and can significantly impact patient quality of life and survival. Understanding how different levels of edema (none, controlled by diuretics, or persistent despite treatment) affect survival can inform patient management strategies and highlight the need for aggressive interventions in certain cases. From the graph we can see a even split between the placebo and drug for Edema status. There is no NA values for Edema persistent despite diuretics. We cannot directly compare it the splits for other statuses given their NA values if known could change the look of the graph.

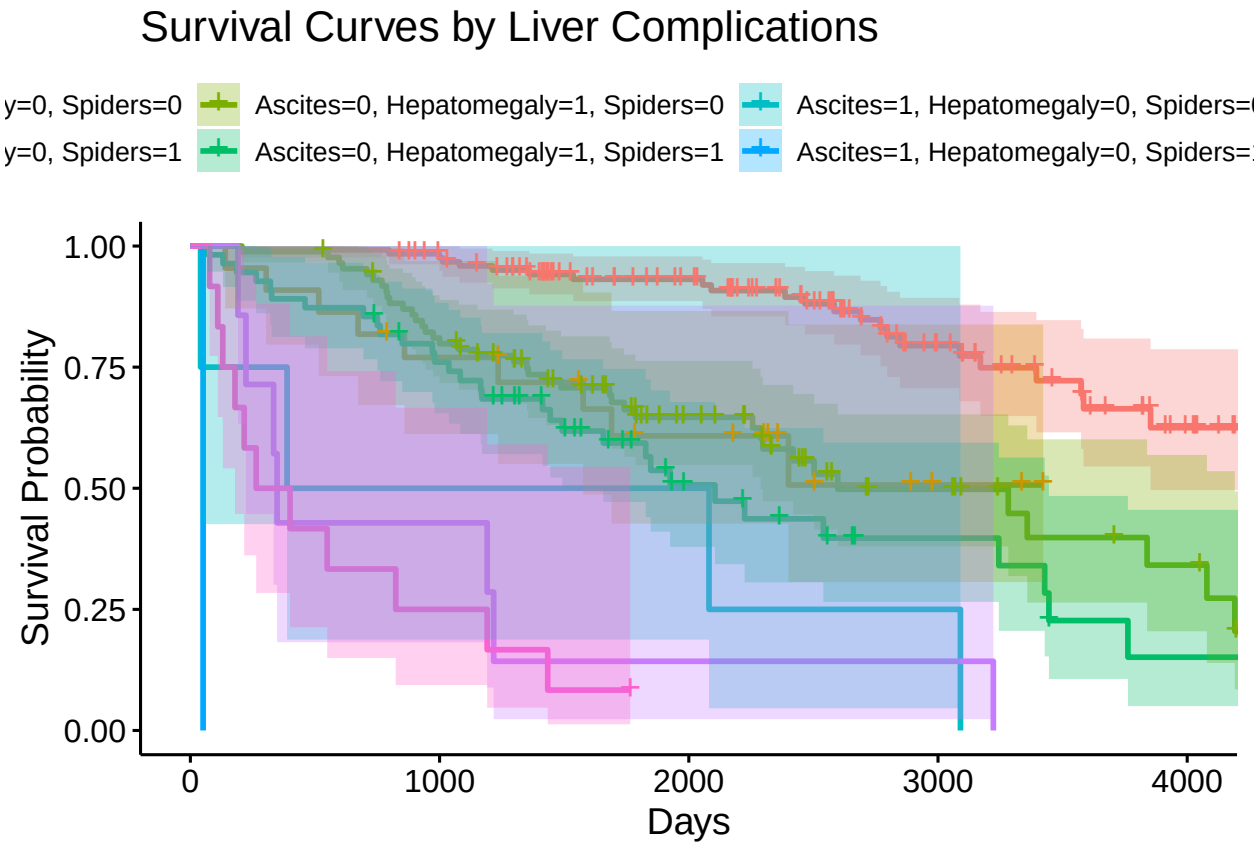
Survival Analysis Based on Treatment and Stage: How does the survival rate differ among patients at different stages of PBC who received D-penicillamine versus placebo?

This question investigates the effectiveness of the drug D-penicillamine compared to a placebo, considering the stage of PBC. By analyzing survival rates across different disease stages and treatment types, we can assess the drug's effectiveness at various disease stages. This information is vital for clinicians to make informed decisions about treatment plans and for researchers to understand the drug's impact.



Impact of Liver Complications on Survival: How do the presence of ascites, hepatomegaly, and spiders relate to survival time?

Exploring how liver-related symptoms (ascites, hepatomegaly, and spiders) affect patient survival provides insights into the severity of these complications in PBC progression. This can help in identifying high-risk patients and understanding the disease's impact on liver function.



Correlation between Biochemical Markers and Disease Stage: What is the correlation between biochemical markers (Bilirubin, Cholesterol, Albumin, Copper, Alk_Phos, SGOT, Triglycerides) and the histologic stage of the disease?

This analysis is crucial to understand how different biochemical markers correlate with the disease's progression. Such correlations can aid in the early detection of disease severity, help in monitoring the disease progression, and potentially guide treatment adjustments.

Part I: Classification Algorithm for the Status of the Patient using CART

To address the first sub-goal of this project, we will be exploring the prediction of the status of a patient at the end of the study. Our main objective through this is to gain a stronger understanding of what factors played in the role of the death of the patient. To do this, we will be using Decision Trees. Let us start by exploring what these are and why we chose to use them.

Introduction

Decision trees are a type of model used in statistics for making predictions based on data. They work by breaking down a dataset into smaller subsets through "splits," resembling a tree with branches. Each branch represents a possible decision or outcome, leading to a final prediction or classification.

The main advantages of decision trees include their simplicity and interpretability, as they are easy to visualize and understand. Another important advantage is that they are able to learn with data with N/A values, something alternatives such as Logistic Regression. Considering the limitation we have with our data size and number of N/A values, this an important advantage. However, there are some downsides such as a tendency to overfit the data, which is something we need to be careful about.

Figure 2: Decision Tree Example (Source)

Pre-processing the data

For this data, we will be dealing with the Status variable. The Status variable was divided into 3 classes:

- Class 0 (D): The patient didn't survive by the end of the observation
- Class 1 (C): The patient is censored, meaning that the observation period ended without the death being recorded
- Class 2 (CL): Similar to Class 1, the patient is censored due to liver transplantation

Thus, we can group Class 1 and 2.

Methodology

Our objective is to create a classifier capable of predicting a patient's outcome. To achieve this, we will be testing our data with the cart algorithm. To ensure model validation, we'll be using a 80% training and 20% testing data division. I will also be stratifying the data based on the status.

To guarantee consistency and reproducibility in our results, we have fixed the seed for our 80/20 data split at 380. With these steps, we are now well-positioned to finalize our training and testing data sets.

The predictors in these models will be guided by the results from the EDA. We will also use a variety of tools to understand the model's performance.

Step 1: Growing the tree

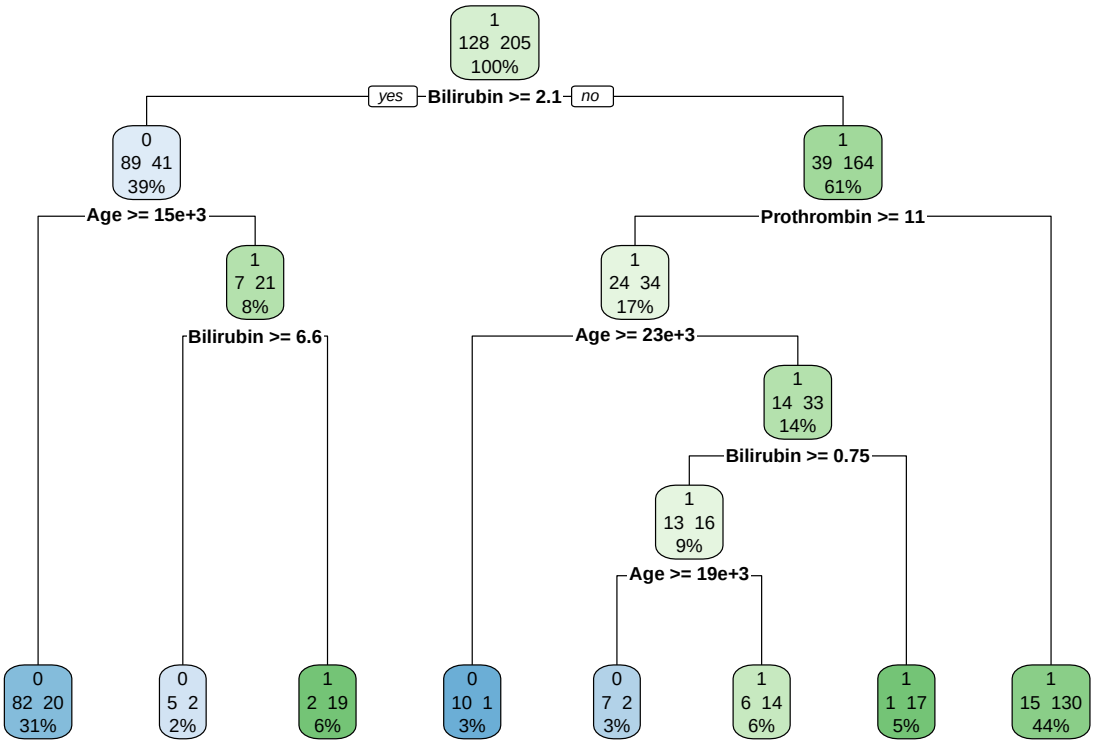
There are useful packages to build decision trees in R: the {tree} and the {rpart} (recursive partitioning) packages.

In this report, we have decided to use {rpart} because it provides more flexibility for surrogate splits and the trees are a bit easier to make attractive looking.

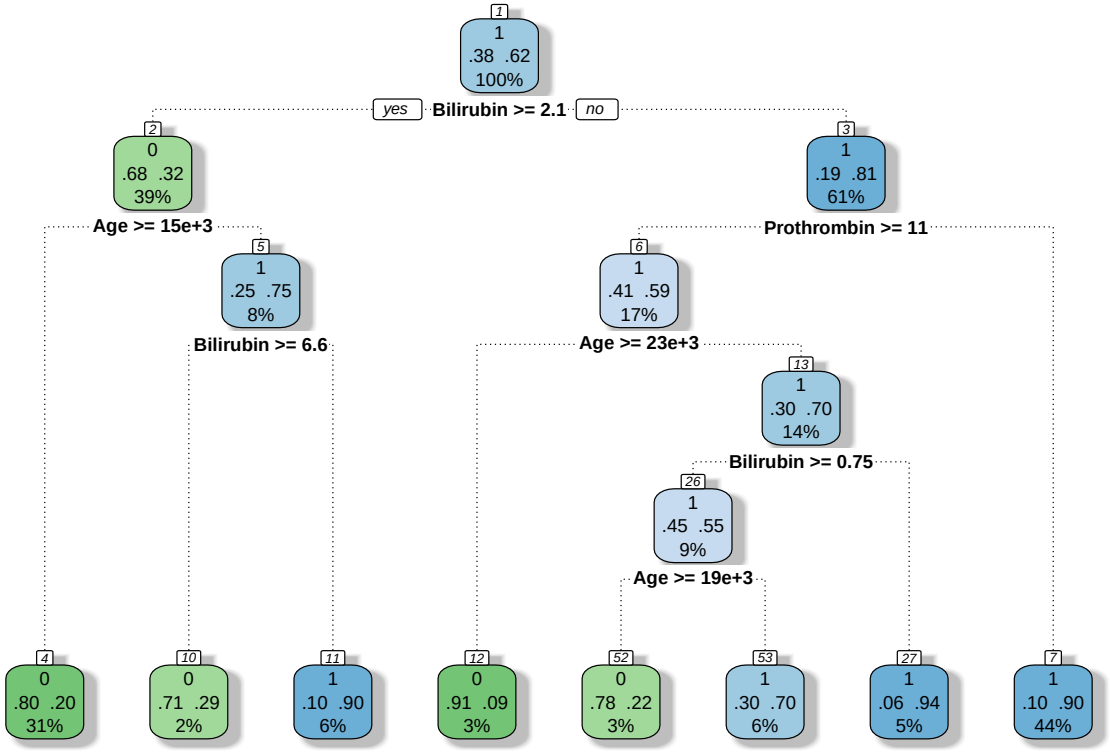
Part 2: Visualizing the tree

With the tree grown, we can now visualize it for an easy understanding of its functioning. This is an important advantage for CART over logistic regression.

The following is a basic diagram for the tree that was just grown.



To gain a further understanding of the data, we can plot a tree yielding Collection Node style trees. This can help us understand how the data is split.



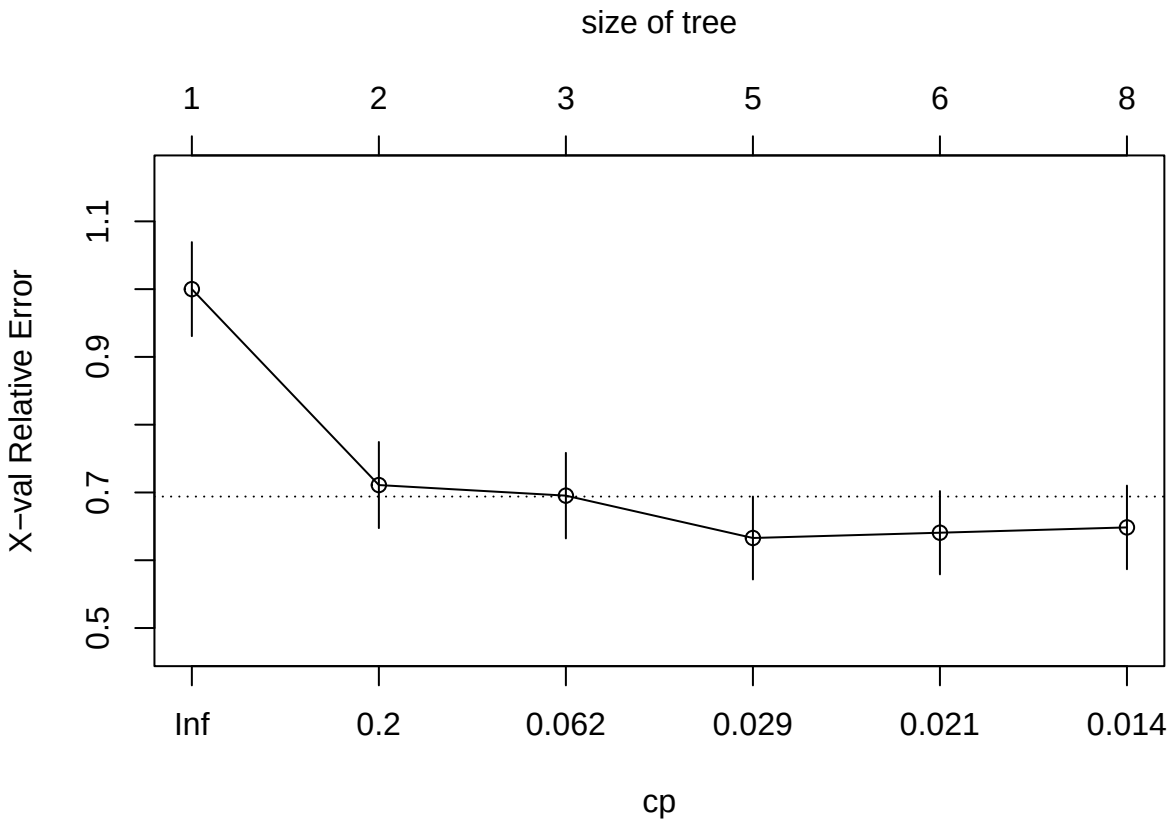
The tree shows us the splits that were done on Age, Bilirubin and Prothrombin. Interestingly, Stage did not contribute in the tree.

Part 3: Pruning the tree

Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances. The first step of pruning a tree is understanding the complexity parameter used. The complexity parameter (cp) in rpart is the minimum improvement in the model needed at each node. This is used when building the tree. We can see the results based on the cross validation from the table below.

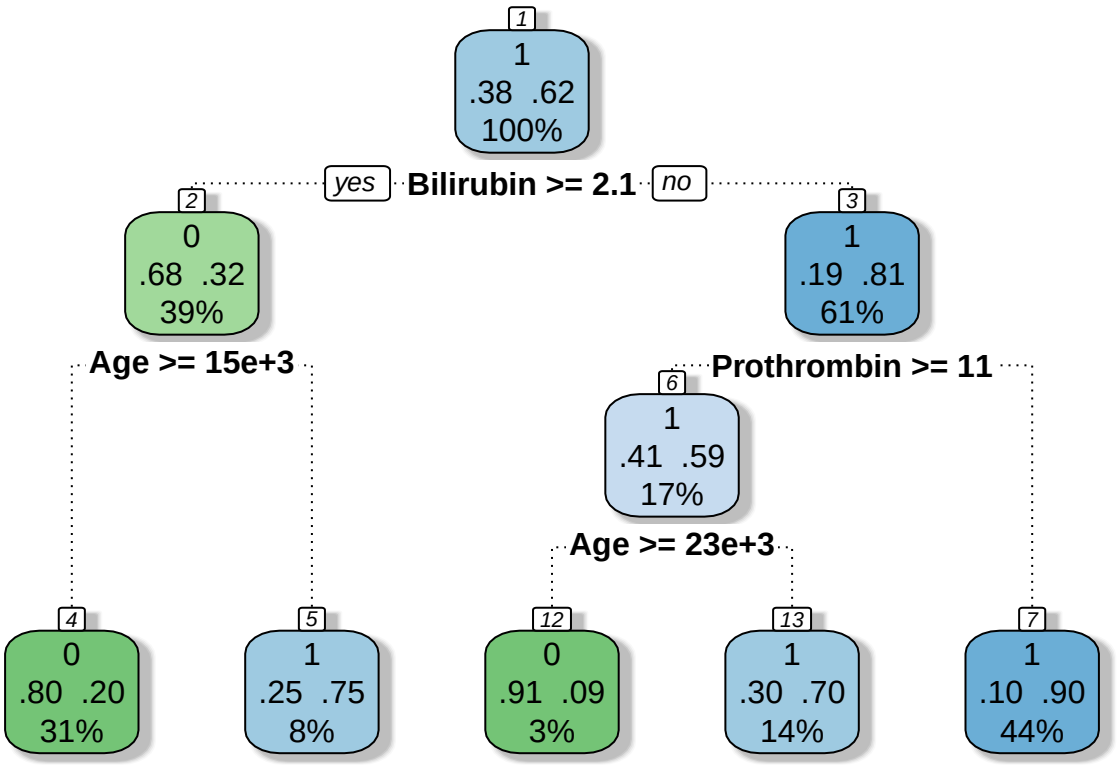
CP	Num. of splits	Rel. Error	Mean Error	Std. Deviation of Error
0.375	0	1.000	1.000	0.069
0.109	1	0.625	0.711	0.064
0.035	2	0.516	0.695	0.063
0.023	4	0.445	0.633	0.061
0.020	5	0.422	0.641	0.061
0.010	7	0.383	0.648	0.062

This can also be visualized in a graph to gain a better understanding of the data. The graph below shows the connection between the cp, size of tree and the x-val relative error.



From the graph we can see that a cp of 0.039 is ideal as it is under the horizontal (dotted) reference line. We can prune the tree with this CP value.

We can plot the pruned tree



We ca see the pruned tree has cut out some leaf nodes. This would help in avoiding overfitting the model. As a reminder, Overfitting is when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

Part 4: Results

Now, we can evaluate the results of the tree on the testing data from the initial 80-20 split. As is true whenever we use validation approaches, we need to test out our model on the testing data set. This will give us a more accurate understanding of how well the model fits the context we're seeking to build our understanding of.

An important part of our results is understanding the role of prediction and inference. In a broad sense, prediction refers to the process of making forecasts about future events or unknown values based on a model while inference generally refers to the process of drawing conclusions from data. For the basic tree, I will be mainly focusing on prediction aspects of the results. However, later in the report, we will also be exploring inference findings.

We can evaluate the results of this through a confusion matrix.

Table 1: Model 1: Confusion Matrix (0=Deceased, 1=Censored)

Prediction/Supervision	0	1
0	19	9
1	14	43

With this, we can also calculate the model's metrics on the test data.

model	accuracy	sensitivity	specificity
rpart2_pred	0.729	0.827	0.576

As you can see the model shows good accuracy with 72.9%. However, the specificity is an issue with 57.6%. Now let us compare this with a different type of model: logistic regression.

Part II: Exploring the predictors for the patient's status using logistic regression

Next, let us explore our question with a different type of model and compare the results with our tree. To do this, we will be using (binary) logistic regression model.

Introduction

Logistic regression is a statistical method used for binary classification, which predicts the probability of an outcome that can be either true or false. This is done by understanding the relationship between a dependent binary variable and one or more independent variables. Logistic regression is easy to implement and interpret. However there are also some drawbacks, the model assumes a linear relationship between the independent variables and the log odds of the dependent variable, which may not always hold true in complex real-world scenarios. Furthermore, unlike the decision tree, linear regression models cannot ignore N/A values.

Methodology

Similar to the tree, we will start by splitting the data set into training and testing sets. Note that the data now only contains instances where the patient was deceased. The training set will be used to train our model, while the testing set will help evaluate its performance. We'll use 80% of the data for training and the remaining 20% for testing. To allow reproducible code, we have fixed the seed at 380.

With this, we will build two candidate models:

- The first model will test the classification based on just the SGOT
- The second model will use a step wise function using various predictors to see the best performance.

Another important consideration is the application of prediction (estimating an outcome based on input variables) and inference (understanding the relationships between variables). We will evaluate the inference through the coefficient analysis and prediction through roc curves and confusion matrix. It is important to note that these metrics will complement each other in our understanding of the data. However, the main focus of this analysis will be prediction and we will work with several metrics to evaluate it.

We will use a variety of tools to understand the model's performance.

Results

Initially, we'll delve into the two preliminary models independently to understand where they stand. Following that, we'll be deploying the best candidate model on our test data. Regarding confusion matrices, we'll employ a basic rule: if the predicted probability of a the patient's status being "deceased" exceeds 0.5, we'll categorize them as deceased (naïve rule).

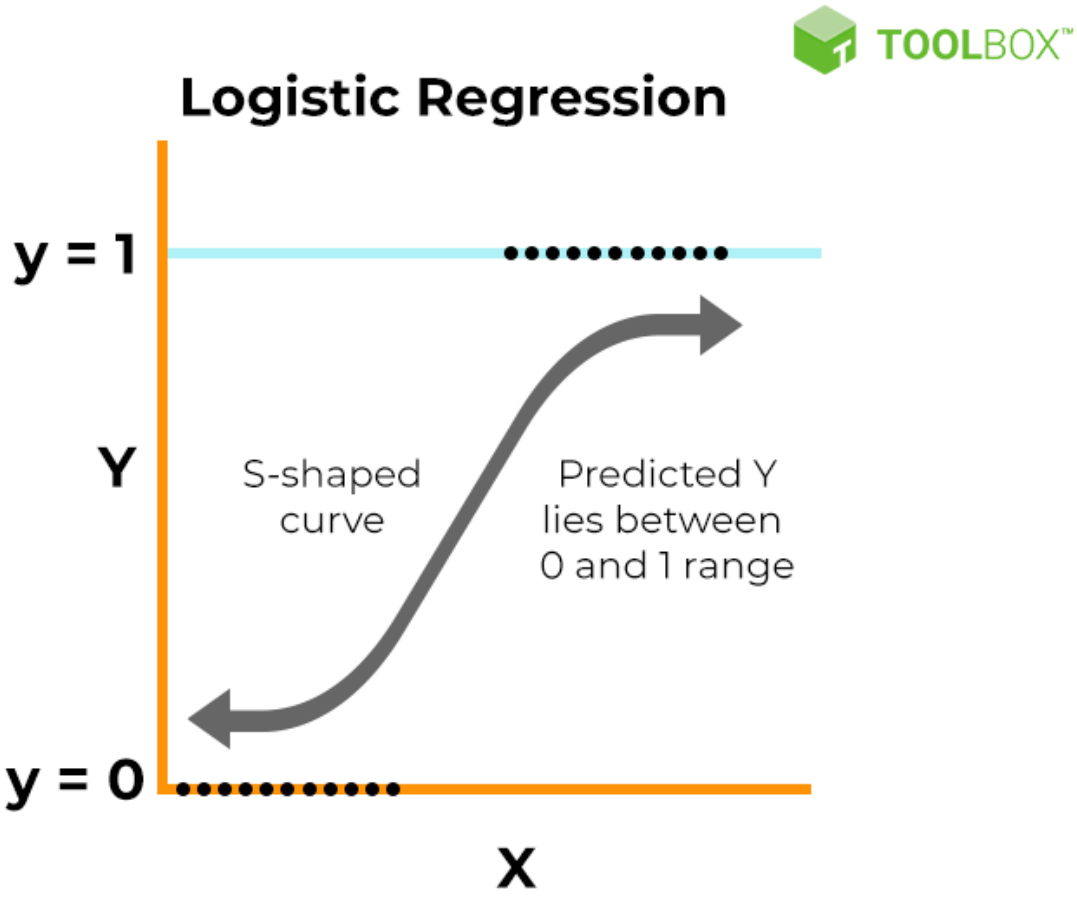


Figure 3: Logistic Regression Model (Source: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/)

Model 1

X	coefficient	prob_odds	Std. Error	z value	Pr(>|z|)
Intercept	1.3728804	3.9467022	0.3566274	3.849622	< 0.001
SGOT	-0.0077204	0.9923094	0.0026073	-2.961056	0.00306586722502071

This table shows us the results of our first model. We can see that, holding other variables constant, a one-unit increase in SGOT is associated with a decrease in the log-odds of the response variable by 0.0076446. Furthermore, the odds-ratio indicates that for each one-unit increase in SGOT, the odds of the event occurring decrease by about 0.76%.

We can also plot the confusion matrix for this model:

Table 2: Model 1 Confusion Matrix

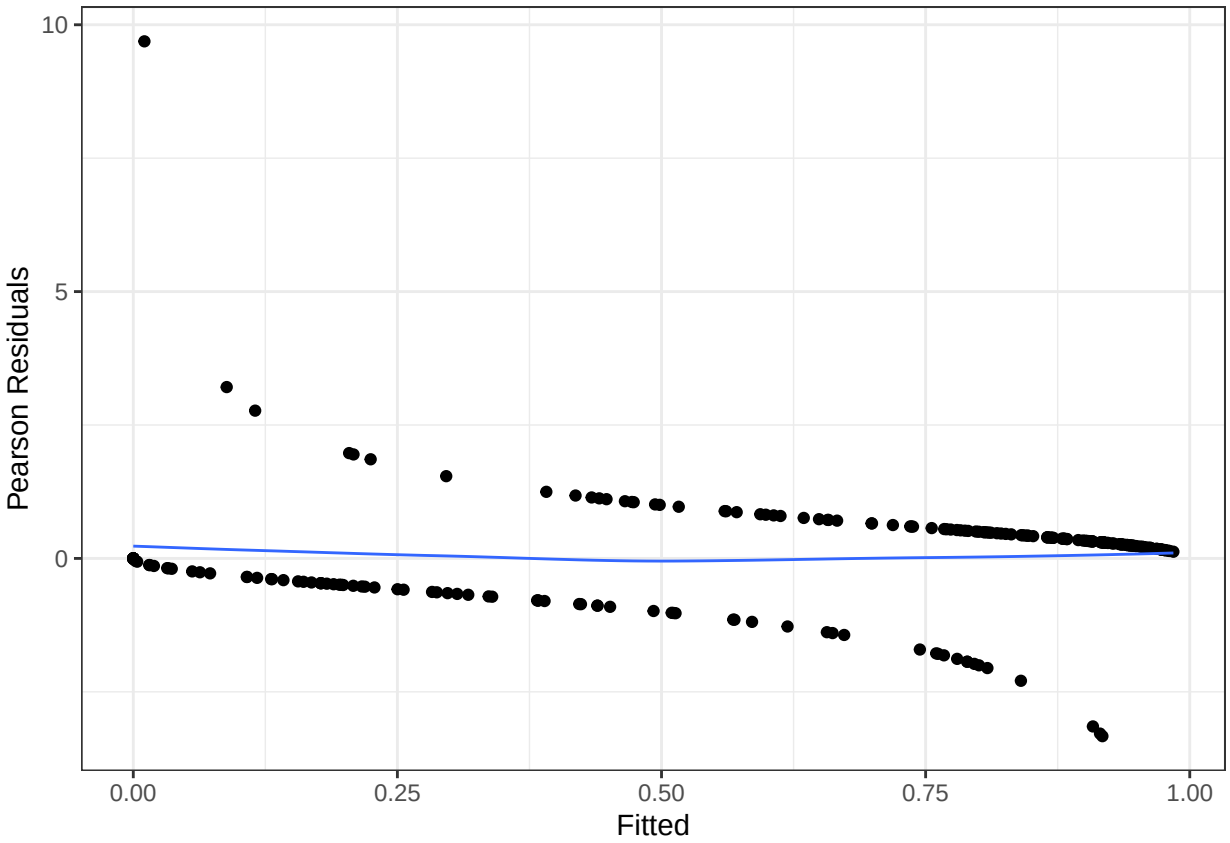
Predicted/Actual	Censored	Deceased
Censored	117	68
Deceased	15	20

We can see that this model tends to over predict censored values. This shows the need of bringing in more factors. Next let us look at our Model 2, which has multiple factors as discussed earlier.

Model 2

	coefficient	prob_odds	Std. Error	z value	Pr(>|z|)
(Intercept)	13.1738171	5.264002e+05	2.5758130	5.1144306	0.0000003
Bilirubin	-0.2146715	8.068064e-01	0.0865790	-2.4794883	0.0131571
Age	-0.0002039	9.997962e-01	0.0000571	-3.5727423	0.0003533
Alk_Phos	-0.0002988	9.997013e-01	0.0000968	-3.0870015	0.0020219
Prothrombin	-0.6086022	5.441109e-01	0.2125486	-2.8633552	0.0041918
Spiders	-0.7375877	4.782662e-01	0.4110588	-1.7943606	0.0727556
EdemaS	-0.0940912	9.101997e-01	0.6115311	-0.1538617	0.8777188
EdemaY	-16.3353205	1.000000e-07	857.1377366	-0.0190580	0.9847948
SGOT	-0.0064630	9.935579e-01	0.0033599	-1.9235635	0.0544093
DrugPlacebo	0.6736154	1.961316e+00	0.3939801	1.7097702	0.0873084
Copper	-0.0038998	9.961078e-01	0.0027552	-1.4154168	0.1569463

This is the role of inference in evaluating our model. The most notable predictors are Bilirubin, Age, Alk_Phos, and Prothrombin, each showing a statistically significant relationship ($p < 0.05$) with the dependent variable. The Intercept and EdemaY have extremely significant p-values, but the practical significance of EdemaY is questionable due to its large standard error. Other variables like Spiders, SGOT, DrugPlacebo, and Copper, while contributing to the model, do not reach conventional levels of statistical significance ($p < 0.05$).



This figure shows us the Tukey-Anscombe plot using Pearson residuals for Model 2. In an ideal fit, the residuals should be evenly distributed about zero with constant mean and variance. The shape of the line suggests that the model is not capturing someunder lying structure in the datain extreme cases.

	GVIF	Df	GVIF^(1/(2*Df))
Bilirubin	1.419	1	1.191
Age	1.232	1	1.110
Alk_Phos	1.033	1	1.017
Prothrombin	1.117	1	1.057
Spiders	1.038	1	1.019
Edema	1.119	2	1.029
SGOT	1.252	1	1.119
Drug	1.091	1	1.044
Copper	1.301	1	1.140

The Variance Inflation Factor (VIF) values for the variables in the model (Bilirubin, Age, Alk_Phos, Prothrombin, Spiders) are all close to 1, indicating minimal multicollinearity. This means that these predictors are relatively independent of each other, enhancing the reliability of the model.

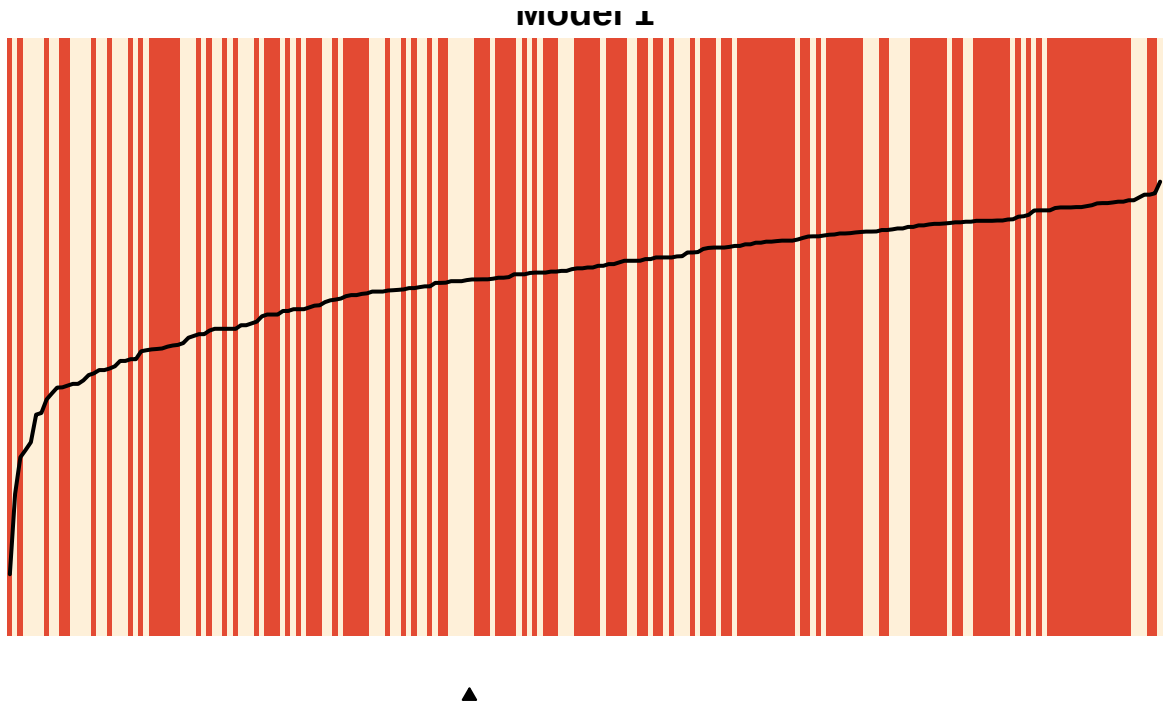
Table 3: Confusion matrix for Model 2

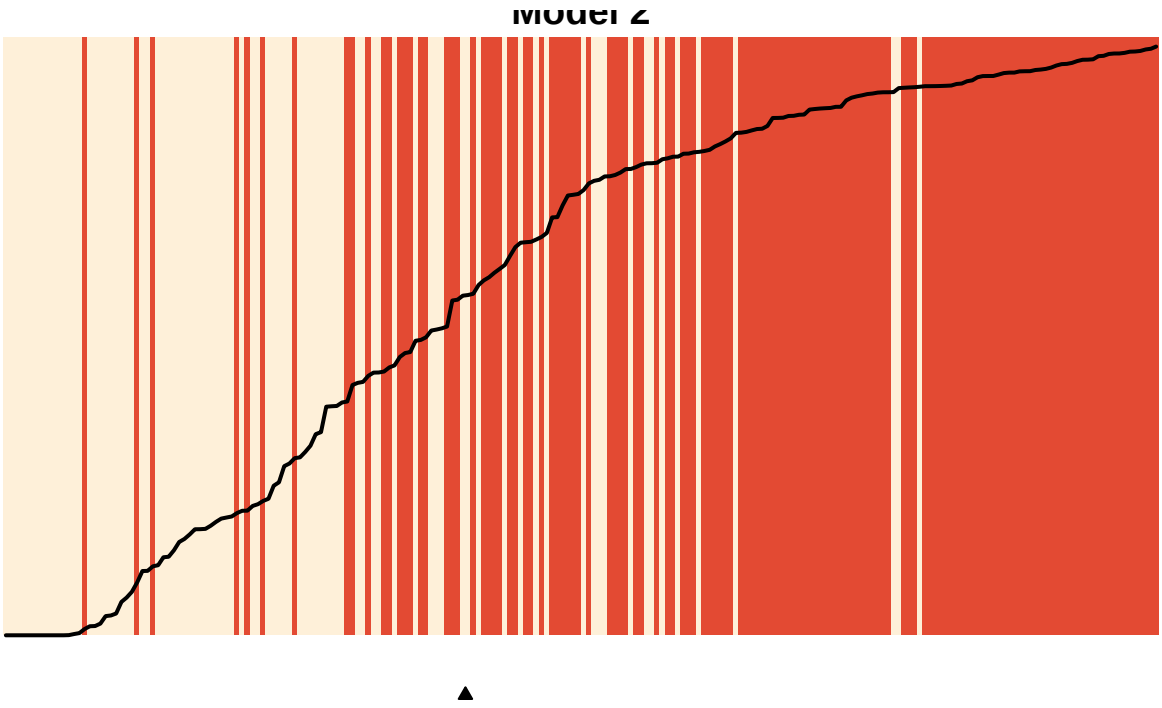
	Censored	Deceased
Censored	115	24
Deceased	17	64

From this confusion matrix we can see the relationships between the True Positive, True Negative, False Positive and False Negative values. From this we can calculate:

- **Accuracy**: Approximately 81.36%
- **Recall**: Approximately 79.01%
- **Precision**: Approximately 72.73%
- **F1 Score**: Approximately 75.74%

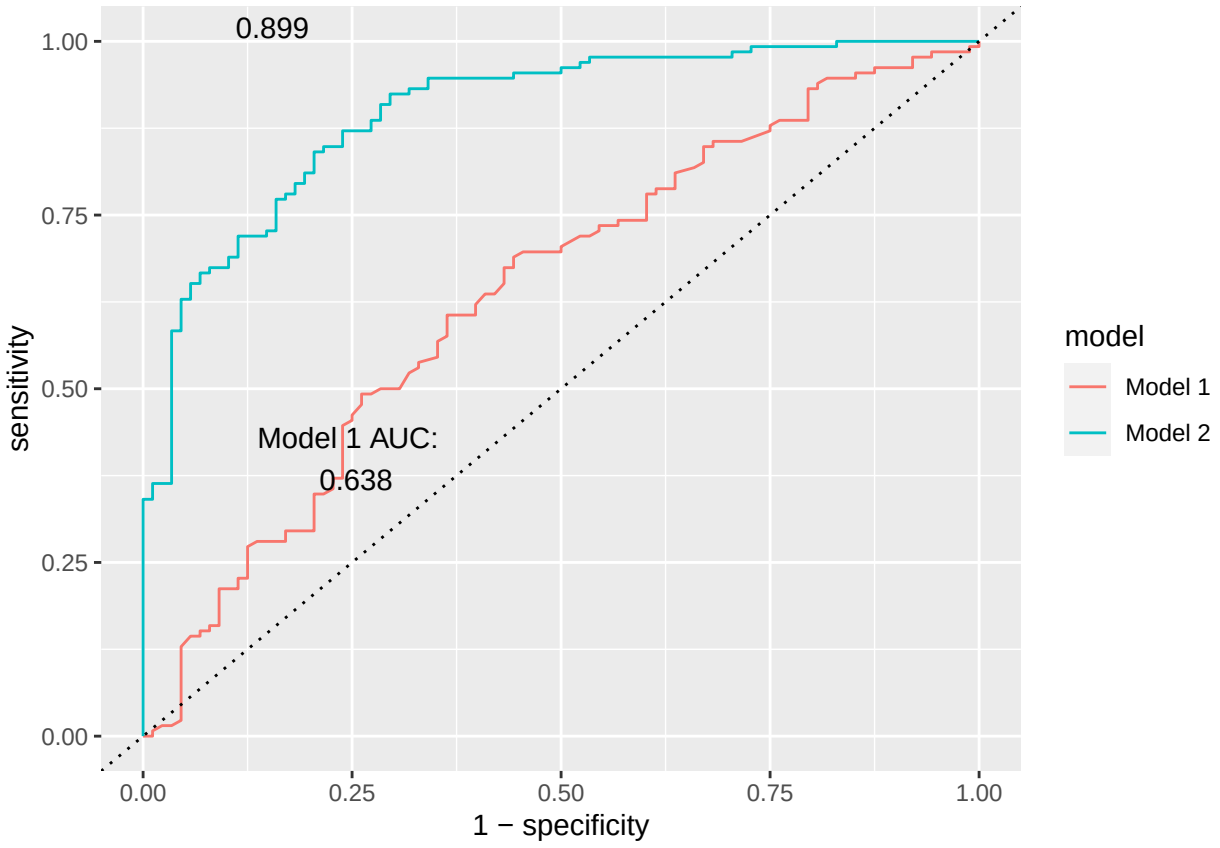
Lastly, let us look at the separation plots for each of the models.





The separation plot assesses the the fit of the model by providing the model's ability to predict occurrences with a high probability and non-occurrences with low probability. The separation plot above separation plot suggests that Model 2 has a reasonably good performance in predicting the patient's status, especially for the observations on the left-most side of the plot compared to Model 1 with the training data. We will later compare this graph with the testing data.

Lastly, let us look at the ROC curves for both the models.



From the graphs we can interpret that:

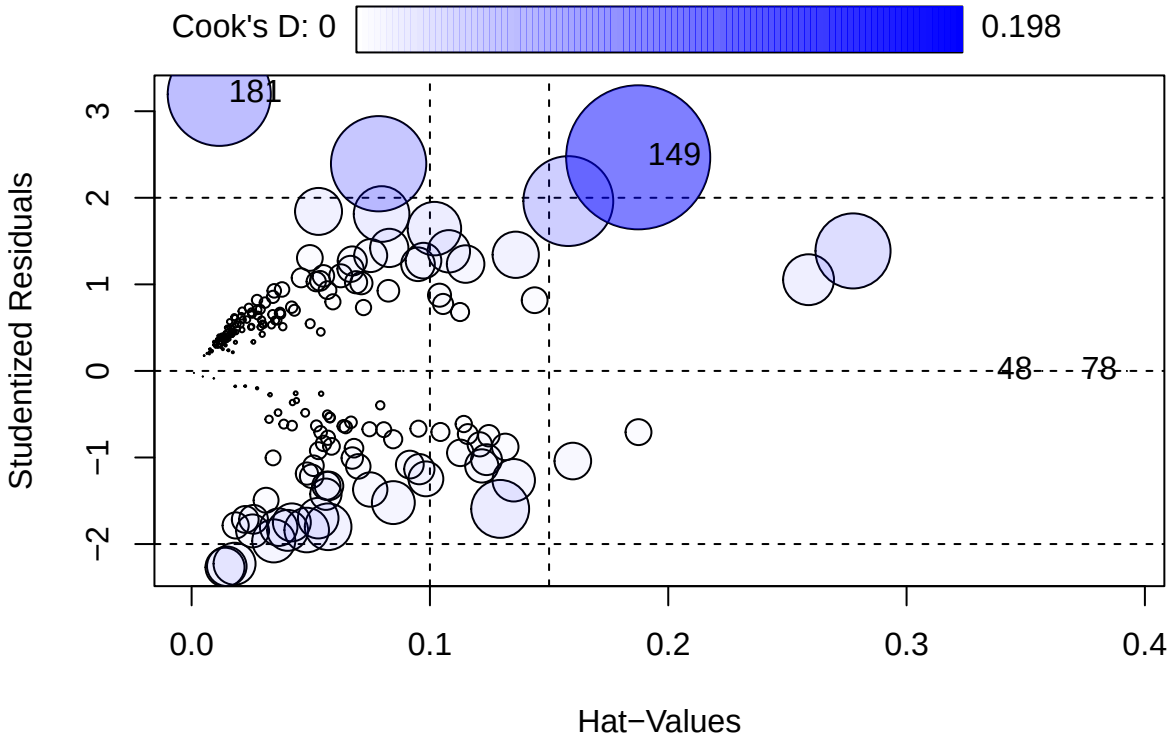
Model 1:

- Its ROC curve is above the line of no discrimination, indicating that the model has some predictive capabilities
- The AUC is 0.638, which is better than random guessing but suggests there's room for improvement.

Model 2:

- The ROC curve for Model 2 is significantly above that of Model 1, and much closer to the top-left corner, indicating better predictive performance.
- The AUC is 0.899, which suggests a good classification performance, and it's notably better than Model 1.
- Its ability to discriminate between positive and negative classes is superior to that of Model 1.

Lastly, let us plot our influence plot.



The influence plot shows several data points with high leverage and large residuals, indicating potential outliers. Some observations, notably those labeled like "149," have significant influence on the regression model due to their Cook's D values.

Testing our model

Using our final model, we now turn to see how well this classifier does on our testing data. Recall that we initially set the test data during the train test split.

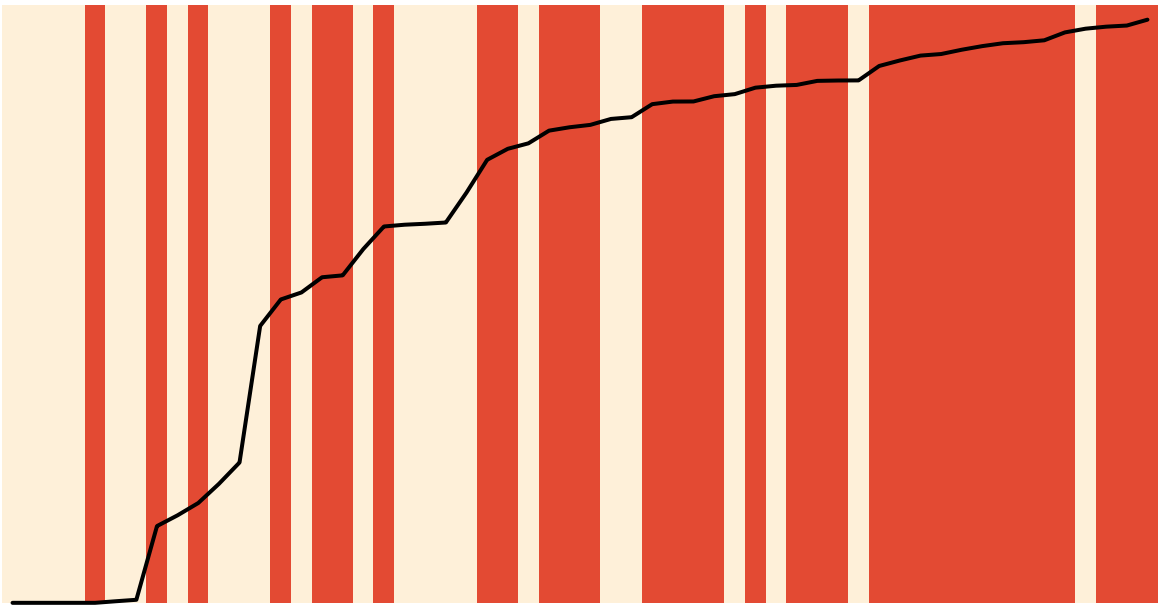
The confusion matrix below shows the performance of our model using the naïve decision rule.

Table 4: Confusion Matrix for Test data

Predicted/Actual	Censored	Deceased
Censored	30	13
Deceased	3	10

- **Accuracy**: Approximately 71.43%
- **Recall**: Approximately 76.92%
- **Precision**: Approximately 43.48%
- **F1 Score**: Approximately 55.56%

This shows that our model is has struggled with overfitting, with an especially low precision score. We will discuss this in the comparison with the tree model, but this is an importnat limitation of our data size as we discussed in the introduction. Lastly, we can plot the separation plot. We can see the separation has increased due to the over fitting we discussed.



▲

Discussion: Comparison between models

Let us recall the differences between the two models:

Logistic Regression:

- Pros: Simple, efficient, good for linear relationships, provides outcome probabilities
- Cons: Assumes linear decision boundary, struggles with complex relationships, sensitive to imbalanced data

Decision Trees:

- Pros: Handles non-linear data, easy to interpret, works with mixed data types, non-parametric, N/A values
- Cons: Prone to overfitting, sensitive to data changes, poor at extrapolation, biased with imbalanced datasets

Now let us compare the performance of our logistic regression model with a decision tree model.

Testing Data Score	CART-based approach	Regression based Approach
Accuracy	72.9%	71.43%
Precision	57.6%	43.48%
Recall	82.70%	76.92%

We can see that for this data, surprisingly, the tree based approach had better results. This is an uncommon observation. We believe this was because of the sparcity of the data and how the rows N/A values prevented over fitting with the small data size.

Part III: Exploring the sub-clusters of the cirrhosis data using unsupervised learning

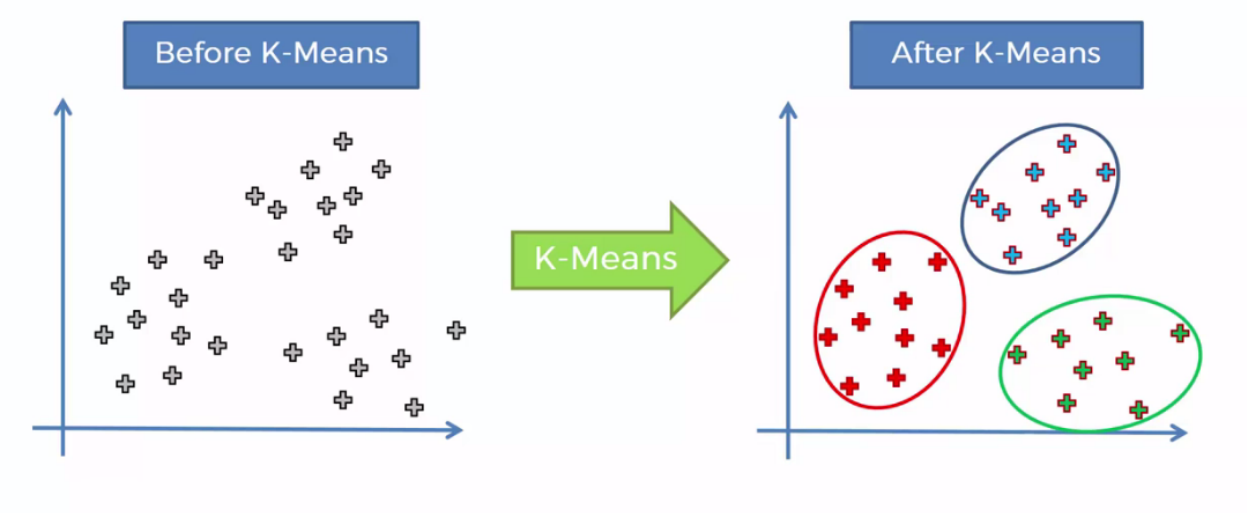
In this section, we sought to if there are sub-collections of patients. We expect sub-collections to mimic medical definitions of the stages of Biliary Cirrhosis but are curious to see if clustering can reveal some newer observations.

Introduction

Biliary Cirrhosis patients are typically clustered in four main clusters (Source):

- Stage 1: There's inflammation and damage to the walls of medium-sized bile ducts.
- Stage 2: There's blockage of the small bile ducts.
- Stage 3: This stage marks the beginning of scarring.
- Stage 4: Cirrhosis has developed. This permanent, severe, scarring and damage to the liver.

In this section we will be using clustering on our data. K-means Clustering is an unsupervised learning technique where data points are grouped based on their similarities. It's commonly used to identify patterns and structures within datasets without prior knowledge of the groups. The main advantage of clustering is its ability to discover hidden patterns in data. However, a significant drawback is the subjectivity in defining the 'similarity' criteria, which can lead to varying results and interpretations.



An important motivation for this part of the study is explore how does “Stage” which is typically defined based on medical professionals’ observations compares to the data that we have in this data set.

Pre-processing the data

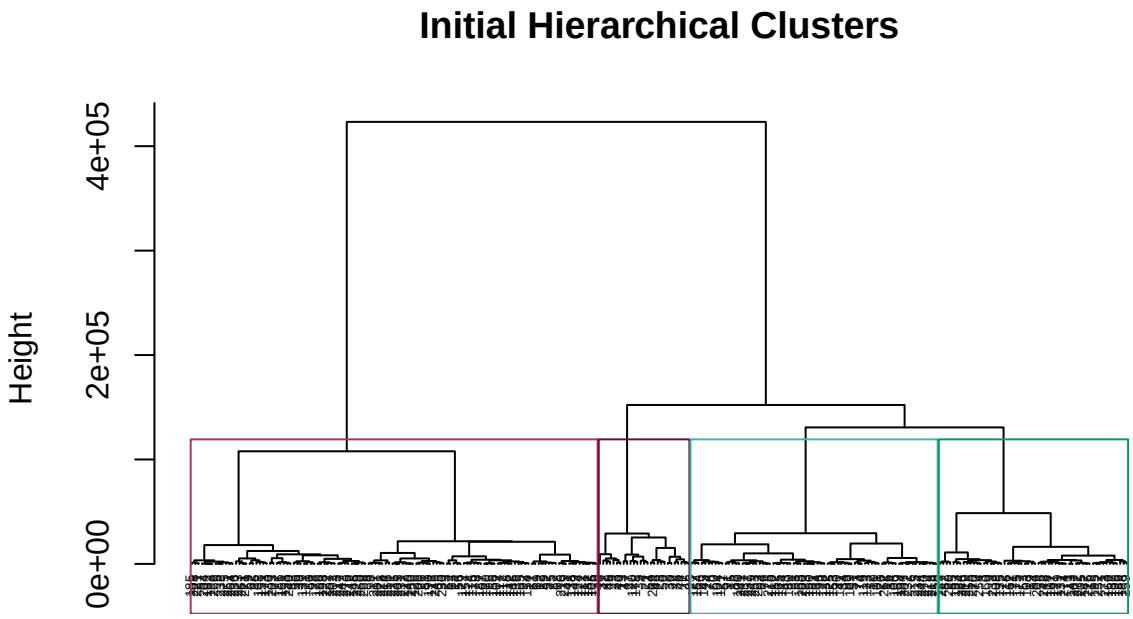
We had to pre-process the data for the best performance. First, we removed the ‘Stage’ column to avoid using outcome-related features in the unsupervised learning. The rows with N/A values were also removed. Lastly, we transformed various categorical variables into numeric formats, which is necessary for clustering algorithms.

Methodology

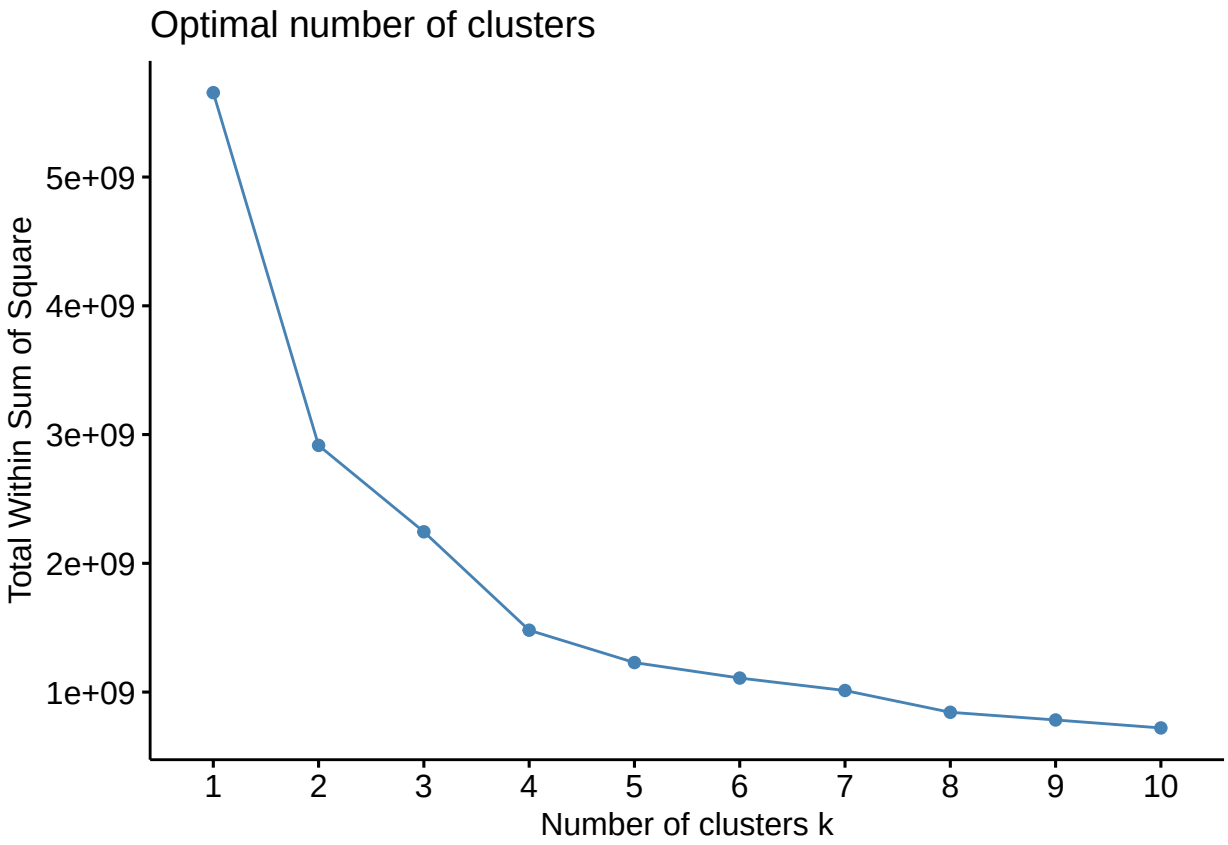
To apply the k-means algorithm. A mixed hierarchical and non-hierarchical was applied. This was done using the hkmeansm module, with k = 4 as our initial value. Based on the data properties, the euclidean (L2) distance works best.

Results

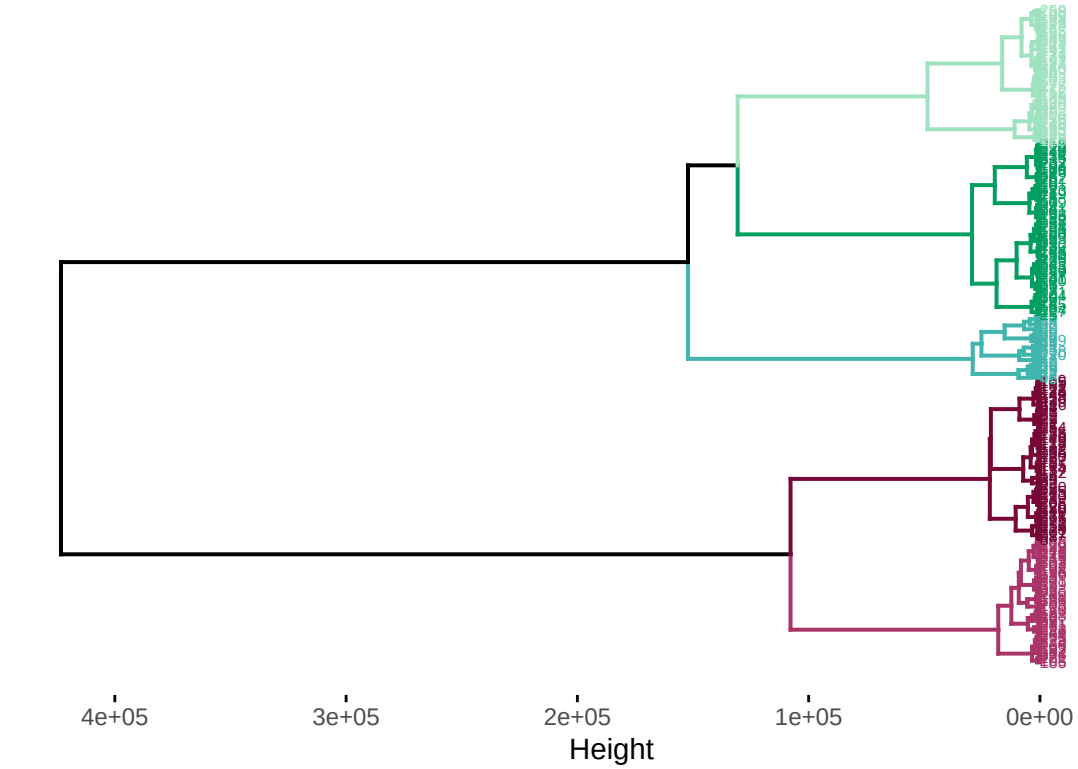
We were able to visualize the results of the k-means algorithm. The first step was making a color palette and plotting the dendrogram tree.



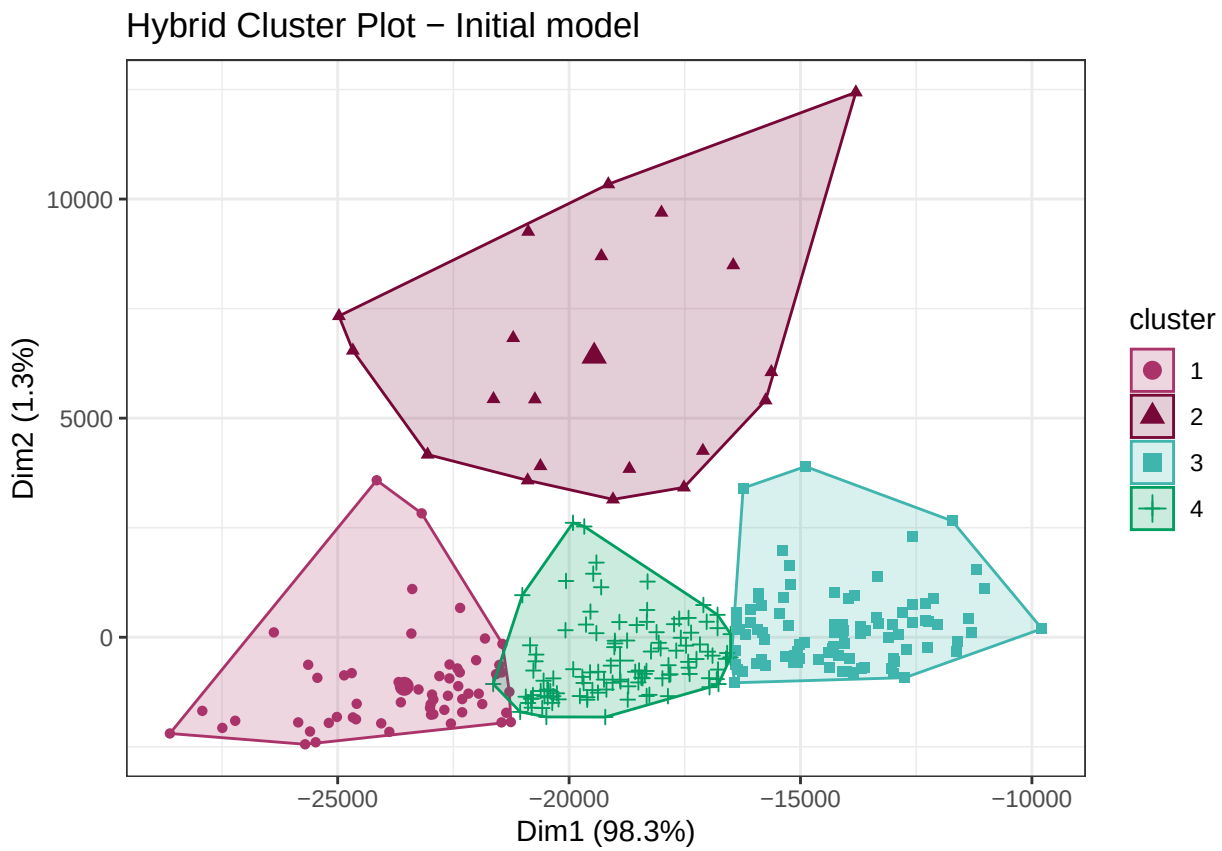
As, you can see, the model was able to create clear clusters for the data. However, it is important to find the best value of k to get the optimal clusters. To do this we can use a scree plot. Here, we’re looking for the number of clusters that corresponds to the “elbow”.



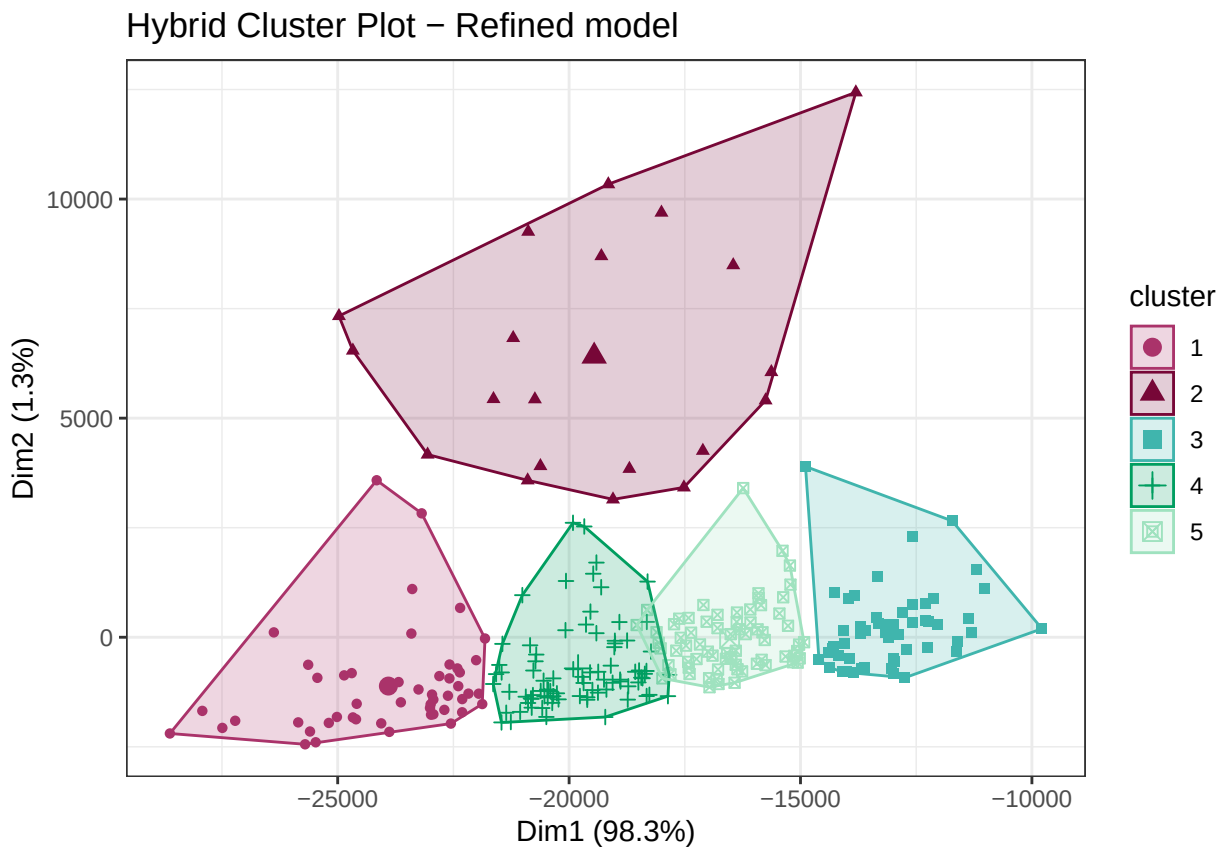
From this, we identified that 5 was the ideal value of k, where the Total Within [Cluster] Sums of Squares begins leveling off. Let us create a new k-means model with k = 5. We can plot this refined model, with a format more easy to visualize.



As you can see, the 5 clusters were identified, which are color coded in our updated diagram. Lastly, let us plot these clusters. Here is a plot of the initial model where k = 4

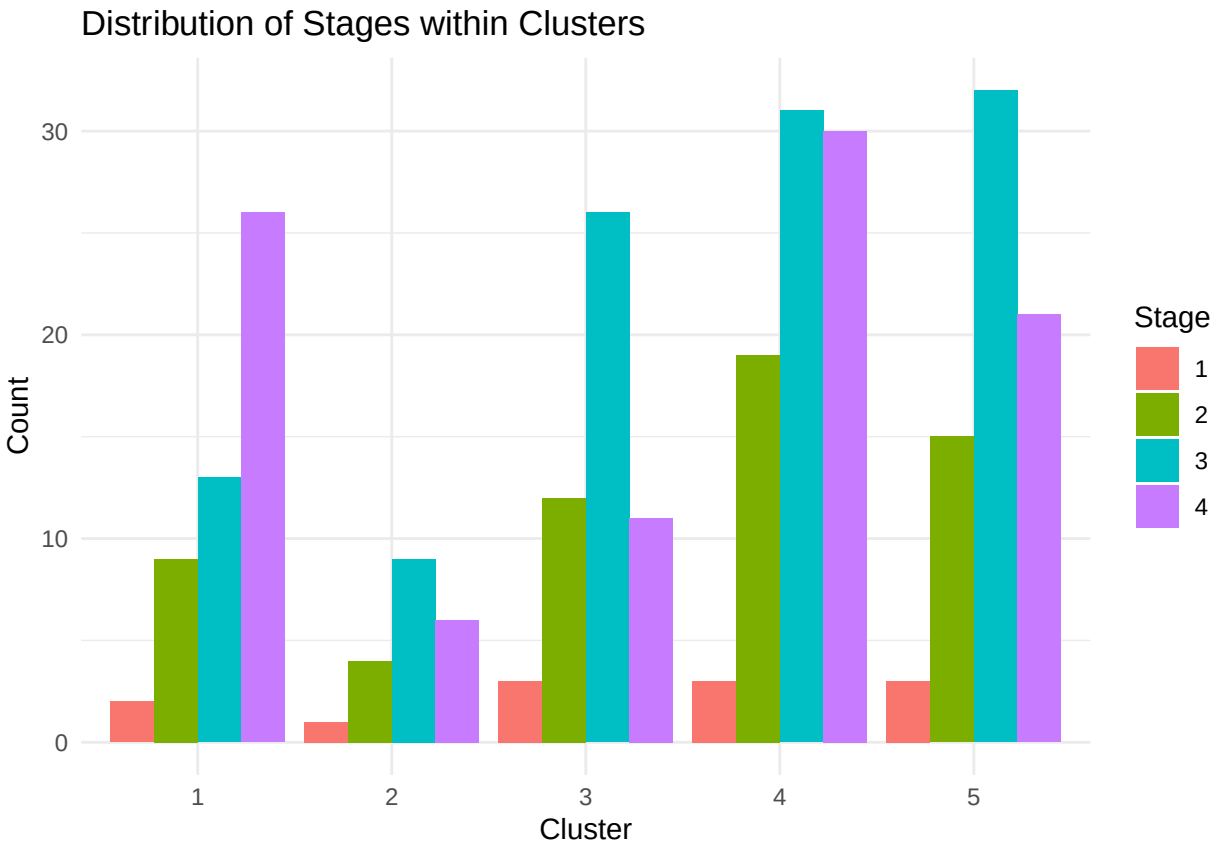


We can also plot our refined model, as you can see the plot below identifies 5 clear clusters.



Now we can go back to our initial goal: comparing how the clusters compare to the Stage variable.

cluster	N_Days	Age	Ascites	Hepatomegaly	Spiders	Bilirubin	Cholesterol	Albumin	Copper
1	1574.080	23836.94	0.1800000	0.5800000	0.2600000	3.994000	329.3400	3.369800	107.82000
2	2579.950	18505.65	0.1000000	0.6500000	0.3500000	4.275000	376.3500	3.372000	149.15000
3	1897.788	12848.35	0.0192308	0.4423077	0.3076923	2.776923	390.1346	3.651923	99.05769
4	1864.627	19675.76	0.0602410	0.5662651	0.3373494	3.667470	356.9880	3.520964	98.03614
5	2288.704	16297.07	0.0281690	0.4225352	0.2253521	2.621127	402.2113	3.557324	86.61972



We can see that the clusters have a somewhat connection to the Stage, but is not as direct as we hypothesized. We were also able to find the cluster summary compared to the rest of the variables and were able to plot that into a table.

Discussion

To conclude, in this report, we were able to explore Biliary Cirrhosis and Treatments through various angles. Though our clustering we were able to find new patterns among the data. Though the logistic and CART algorithms we were able to build models that were able to predict the chance of a patient's death using various predictors.

Along with the analysis in the report project, we attempted to explore various models such as predicting the likelihood of a liver transplant and the number of days until an event. However, due to our data's and report's limitations, such as the limited data quantities, we were unable to get concrete solutions. We hope that our results can be used to build even better models, making healthcare more inclusive, efficient and affordable.

Author Contributions

We collaborated throughout the document using GitHub, but here are the main responsibilities:

Introduction: Nithika

EDA: Nithika

Tree: Mihir

Logistic Regression: Mihir

Clustering: Nithika

Discussion: Mihir

References

Dataset: https://www.kaggle.com/datasets/fedesoriano/cirrhosis-prediction-dataset/data

Study: https://faculty.washington.edu/abansal/ShortCourse_DynamicDecisionMaking/Dickson1989_MayoPBCOriginalArticle.pdf

Code Appendix

```
knitr::opts_chunk$set(
  echo = FALSE,
  warning = FALSE,
  message = FALSE,
  fig.align = "center"
)

library(survival)
library(ggplot2)
library(survival)
library(survminer)
library(kableExtra)
library(factoextra)

#load in the data
library(tidyverse)
#load data
cirrhosis <- read_csv("cirrhosis.csv")
#encode the sex, ascites, hepatomegaly, spiders, edema into binary variables
cirrhosis$Sex <- ifelse(cirrhosis$Sex == "F", 0, 1)
cirrhosis$Ascites <- ifelse(cirrhosis$Ascites == "Y", 1, 0)
cirrhosis$Hepatomegaly <- ifelse(cirrhosis$Hepatomegaly == "Y", 1, 0)
cirrhosis$Spiders <- ifelse(cirrhosis$Spiders == "Y", 1, 0)
#factorize the stage
cirrhosis$Stage <- factor(cirrhosis$Stage)
cirrhosis$Status <- factor(cirrhosis$Status)
cirrhosis$Drug <- factor(cirrhosis$Drug)
cirrhosis$Sex <- factor(cirrhosis$Sex)
#cirrhosis$Ascites <- factor(cirrhosis$Ascites)
#cirrhosis$Hepatomegaly <- factor(cirrhosis$Hepatomegaly)
#cirrhosis$Spiders <- factor(cirrhosis$Spiders)
cirrhosis$Edema <- factor(cirrhosis$Edema)
cirrhosis$Sex <- factor(ifelse(cirrhosis$Sex == 0, "Female", "Male"))

#Drop the first column
cirrhosis <- cirrhosis[,-1]
#Required libraries
library(survival)
library(ggplot2)
library(survival)
library(survminer)


#cirrhosis$Sex <- factor(ifelse(cirrhosis$Sex == 0, "Female", "Male"))

# Preparing the data for pie chart
sex_data <- cirrhosis %>%
            count(Sex) %>%
            mutate(Percentage = n / sum(n) * 100)

# Pie Chart
```

```r
ggplot(sex_data, aes(x="", y=Percentage, fill=Sex)) +
    geom_bar(stat="identity", width=1) +
    coord_polar("y", start=0) +
    scale_fill_brewer(palette="Pastel1") +
    theme_minimal() +
    theme(axis.line = element_blank(),
          axis.text = element_blank(),
          axis.ticks = element_blank(),
          panel.grid = element_blank(),
          plot.title = element_text(size=14, face="bold")) +
    labs(fill="Sex", title=" Sex Amongst Patients")



# Converting age from days to years for better readability
cirrhosis$Age_Years <- cirrhosis$Age / 365.25

# Histogram
ggplot(cirrhosis, aes(x=Age_Years)) +
    geom_histogram(binwidth=5, fill="#69b3a2", color="#e9ecef") +
    theme_minimal() +
    labs(title="Age Distribution of PBC Patients", x="Age (years)", y="Count") +
    theme(plot.title = element_text(size=14, face="bold"))

count <-  data.frame(NA_CountsPerVar = colSums(is.na(cirrhosis)))
kable(count)

cirrhosis %>%
      ggplot(aes(x = Stage)) + geom_bar(fill = "#097969") # need color


ggplot(data = cirrhosis, aes (Drug, Stage, fill = N_Days))+ geom_tile() + scale_fill_distiller(palette =


hi = c("#40B5AD", "#009E60", "#9FE2BF")
library(ggmosaic)
cirrhosis %>%
  ggplot() +
  geom_mosaic(aes( x = product(Edema), fill = Drug)) + scale_fill_manual(values = hi)
surv_object <- Surv(cirrhosis$N_Days, cirrhosis$Status == 'D')
surv_fit <- survfit(surv_object ~ cirrhosis$Drug + cirrhosis$Stage, data = cirrhosis)
ggsurvplot(surv_fit,
           data = cirrhosis,
           conf.int = TRUE,
           #palette = c("#00AFBB", "#E7B800", "#FC4E07"),
           xlab = "Days",
           ylab = "Survival Probability",
           title = "Survival Curves by Treatment and Stage")
surv_fit_complications <- survfit(surv_object ~ cirrhosis$Ascites + cirrhosis$Hepatomegaly + cirrhosis$
ggsurvplot(surv_fit_complications,
           data = cirrhosis,
           conf.int = TRUE,
```

```
            #palette = c("#2E9FDF", "#FC4E07", "#6ACC65"),
            xlab = "Days",
            ylab = "Survival Probability",
            title = "Survival Curves by Liver Complications")
#Combine C and CL status into one variable and binarize
cirrhosisTreeData <- cirrhosis
cirrhosisTreeData$Status <- ifelse(cirrhosisTreeData$Status == "C" | cirrhosisTreeData$Status == "CL",
# Wrangle the Graduate data to set up training and testing datasets
modelCirrhosis <- cirrhosisTreeData %>%
  #drop_na() %>%
  mutate(
    tempID = row_number(),
    .before = Status
  )

## Set seed for reproducibility and slice ----
set.seed(380)
trainingData <- modelCirrhosis %>%
  group_by(Status) %>%
  slice_sample(prop = 0.8)

testingData <- modelCirrhosis %>%
  filter(!(tempID %in% trainingData$tempID))
# Grow Graduate tree via rpart package
library(rpart)
rPartStatus <- rpart(
  formula = Status ~ Drug + Age + Sex + Ascites + Hepatomegaly + Spiders + Edema + Bilirubin + Cholester
  data = trainingData,
  method = "class",
  parms = list(split = "information")
  # We did not need to use the control parameters
)

# Display rpart.plot ----
 library(rpart.plot)
rpart.plot(
  x = rPartStatus,
  type = 2,
  extra = 101
)
# Using the rattle package to visualize the tree ----
library(rattle)

fancyRpartPlot(
  model = rPartStatus,
  main = NULL,
  sub = NULL
)
invisible(capture.output({cpTable <- printcp(rPartStatus)}))

library(kableExtra)

kable(
```

```
  x = cpTable,
  col.names = c("CP", "Num. of splits", "Rel. Error",
                "Mean Error", "Std. Deviation of Error"),
  digits = 3,
  booktabs = TRUE,
  align = "c",
  table.attr = 'data-quarto-disable-processing="true"'
)
plotcp(
  x = rPartStatus,
  minline = TRUE,
  upper = "size"
)
# Prune the rpart Graduate Tree ----
rPartStatus2 <- prune(
  tree = rPartStatus,
  cp = 0.029
)
fancyRpartPlot(
  model = rPartStatus2,
  main = NULL,
  sub = NULL
)
pred_StatusRpart2 <- predict(
  object = rPartStatus2,
  newdata = testingData,
  type = "prob"
)

# Data Wrangling the predictions ----
StatusPrediction <- data.frame(
  rpart2_non_death = pred_StatusRpart2[, 1],
  rpart2_death = pred_StatusRpart2[, 2]
) %>%
  mutate(
    rpart2_pred = ifelse(
      test = rpart2_death > rpart2_non_death,
      yes = 1,
      no = 0
    )
  )

## Set predictions as factors
StatusPrediction$rpart2_pred <- as.factor(StatusPrediction$rpart2_pred)

# Merge supervision column into predictions data frame ----
StatusPrediction <- cbind(
  tempID = testingData$tempID,
  Status = testingData$Status,
  StatusPrediction
)
StatusPrediction$Status <- factor(StatusPrediction$Status)
```

```r
library(yardstick)

# Build confusion matrix for second tree model
conf_matrix <- conf_mat(
  data = StatusPrediction,
  truth = Status,
  estimate = rpart2_pred
)$table

kable(
  conf_matrix,
  col.names = c("Prediction/Supervision", "0", "1"),
  digits = 3,
  booktabs = TRUE,
  caption = "Model 1: Confusion Matrix (0=Deceased, 1=Censored)",
  align = "c"
) %>%
kable_styling(latex_options = "HOLD_position")


accuracy <- accuracy(StatusPrediction, Status, rpart2_pred)
specificity <- specificity(StatusPrediction, Status, rpart2_pred)
sensitivity <- sensitivity(StatusPrediction, Status, rpart2_pred)
# Build a data frame with model metrics ----
StatusPreds <- StatusPrediction %>%
  dplyr::select(tempID, Status, contains("_pred")) %>%
  pivot_longer(
    cols = !c(tempID, Status),
    names_to = "model",
    values_to = "prediction"
  )

accuracy <- StatusPreds %>%
  group_by(model) %>%
  accuracy(
    truth = Status,
    estimate = prediction
  )

sensitivity <- StatusPreds %>%
  group_by(model) %>%
  sensitivity(
    truth = Status,
    estimate = prediction,
    event_level = "second"
  )

specificity <- StatusPreds %>%
  group_by(model) %>%
  specificity(
    truth = Status,
    estimate = prediction,
    event_level = "second"
```

```
  )

modelMetrics <- bind_rows(
  accuracy,
  sensitivity,
  specificity
)
# Make a nice looking table of model metrics ----
modelMetrics %>%
  dplyr::select(model, .metric, .estimate) %>%
  pivot_wider(
    id_cols = model,
    names_from = .metric,
    values_from = .estimate
  ) %>%
  kable(
    digits = 3,
    booktabs = TRUE,
    align = "c",
    table.attr = 'data-quarto-disable-processing="true"'
  )
cirrhosisRegression <- cirrhosis
cirrhosisRegression$Status <- ifelse(cirrhosisRegression$Status == "C" | cirrhosisRegression$Status == 
#model data
LRmodelData <- cirrhosisRegression %>%
  drop_na() %>%
  mutate(
    tempID = row_number(),
    .before = Status
  )

# Set seed for reproducibility and slice
set.seed(380)
trainingData <- LRmodelData %>%
  group_by(Status) %>%  # group_by() function ensures that the data
  slice_sample(prop = 0.80)

testingData <- LRmodelData %>%
  filter(!(tempID %in% trainingData$tempID))

trainingResults <- trainingData
# Form Candidate Model 1
model1 <- glm(
  formula = Status ~ SGOT,
  data = trainingData,
  family = binomial
)
# Lower bound (Intercept only)
lower <- glm(
  formula = Status ~ 1,
  data = trainingData,
  family = binomial
)
```

```r
# Upper bound
upper <- glm(
  formula = Status ~ Drug + Age + Sex + Ascites + Hepatomegaly + Spiders + Edema + Bilirubin + Cholester
  data = trainingData,
  family = binomial
)


# Stepwise search
model2 <- step(
  object = lower,
  scope = list(
    lower = lower,
    upper = upper
  ),
  data = trainingData,
  direction = "both",
  k = 2
)
# Model 1 Coefficient Table
as.data.frame(summary(model1)$coefficients) %>%
  rownames_to_column(var = "X") %>%
  rename(coefficient = Estimate) %>%
  mutate(
    prob_odds = case_when(
      coefficient == "(Intercept)" ~ exp(coefficient)/(1 + exp(coefficient)),
      .default = exp(coefficient)
    ),
    .after = coefficient
  ) %>%
  mutate(
    `Pr(>|z|)` = ifelse(
      test = `Pr(>|z|)` < 0.001,
      yes = paste("< 0.001"),
      no = `Pr(>|z|)`
    ),
    X = case_when(
      X == "(Intercept)" ~ "Intercept",
      grepl(x = X, pattern = "SGOT") ~ "SGOT"
    )
  ) %>%
  kable()

library(janitor)
# Building confidence intervals for Model 1 coefficients
model1CI <- confint(
  object = model1,
  parm = "SGOT",
  level = 0.9
)

trainingResults <- trainingData %>%
  ungroup() %>%
```

```r
  mutate(model1Pred = predict(object = model1, newdata = ., type = "response"))

# Apply naïve rule ----
trainingResults <- trainingResults %>%
  mutate(
    model1Class = case_when(
      model1Pred > 0.5 ~ "Censored",
      .default = "Deceased"
    )
  )

#Confusion Matrix for Model 1
trainingResults %>%
  mutate(Patient_status = ifelse(Status == 1, "Censored", "Deceased")) %>%
  tabyl(var1 = model1Class, var2 = Patient_status) %>%
  adorn_title(
    placement = "combined",
    row_name = "Predicted",
    col_name = "Actual"
  ) %>%
  kable(
    booktabs = TRUE,
    align = "c",
    caption = "Model 1 Confusion Matrix"
  )%>%kable_styling(latex_options = "HOLD_position")

#Coeff for model 2
as.data.frame(summary(model2)$coefficients) %>%
  rename(coefficient = Estimate) %>%
  mutate(
    prob_odds = case_when(
      coefficient == "(Intercept)" ~ exp(coefficient)/(1 + exp(coefficient)),
      TRUE ~ exp(coefficient)
    ),
    .after = coefficient
  ) %>%
  kable()

#do the Tukey-Anscombe plot
ggplot(
  data = data.frame(
    residuals = residuals(model2, type = "pearson"),
    fitted = fitted(model2)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point() +
  geom_smooth(
    formula = y ~ x,
    method = stats::loess,
    method.args = list(degree = 1),
    se = FALSE,
    linewidth = 0.5
```

```
  ) +
  theme_bw() +
  labs(
    x = "Fitted",
    y = "Pearson Residuals"
  )
#plot the gvif
as.data.frame(car::vif(model2)) %>%
  kable(
    digits = 3,
    align = "lcccc",
    booktab = TRUE,
    format.args = list(big.mark = ","),
    table.attr = 'data-quarto-disable-processing="true"',
    label = "GVIF analsyis"
  )

#Store the predicted and actual values for Model 2
trainingResults$model2Pred <- predict(model2, type = "response")
trainingResults$model2Class <- ifelse(trainingResults$model2Pred > 0.5, "Censored", "Deceased")
trainingResults$Actual <- ifelse(trainingData$Status == 1, "Censored", "Deceased")

# Create confusion matrix using table
confusionMatrixRegression <- table(Predicted = trainingResults$model2Class, Actual = trainingResults$Ac

kable(confusionMatrixRegression, caption = "Confusion matrix for Model 2") %>%
  kable_classic(latex_options = "HOLD_position")

library(pROC)
library(separationplot)
# Fit ROC Curves for later
## Model 1
model1ROC <- roc(
  formula = Status ~ model1Pred,
  data = trainingResults
)
model1ROC_df <- data.frame(
  threshold = model1ROC$thresholds,
  sensitivity = model1ROC$sensitivities,
  specificity = model1ROC$specificities,
  model = "Model 1"
)
## Model 2
model2ROC <- roc(
  formula = Status ~ model2Pred,
  data = trainingResults
)
model2ROC_df <- data.frame(
  threshold = model2ROC$thresholds,
  sensitivity = model2ROC$sensitivities,
  specificity = model2ROC$specificities,
  model = "Model 2"
)
```

```r
# Convert 'Actual' column to numeric 0/1
trainingResults <- trainingResults %>%
  mutate(
    actualNum = if_else(Actual == "Deceased", 0, 1)
  )


#Sepeation Plot
par(mar = c(4,0,0,0))
separationplot(
  pred = trainingResults$model1Pred,
  actual = trainingResults$actualNum,
  type = "rect",
  line = TRUE,
  lwd2 = 2,
  show.expected = TRUE,
  newplot = FALSE,
  heading = "Model 1"
)

#Sepeation Plot
par(mar = c(4,0,0,0))
separationplot(
  pred = trainingResults$model2Pred,
  actual = trainingResults$actualNum,
  type = "rect",
  line = TRUE,
  lwd2 = 2,
  show.expected = TRUE,
  newplot = FALSE,
  heading = "Model 2"
)
## Merge into existing data frame
rocData <- rbind(model1ROC_df, model2ROC_df)

## AUC Data
aucData <- data.frame(
  model = c("Model 1", "Model 2"),
  auc = c(model1ROC$auc, model2ROC$auc)
)
#ROC plot
ggplot(
  data = rocData,
  mapping = aes(x = 1 - specificity, y = sensitivity, color = model)
) +
  geom_path() +
  geom_abline(
    slope = 1,
    intercept = 0,
    linetype = "dotted"
  ) +
  geom_text(
  inherit.aes = FALSE,
```

```
  data = aucData,
  mapping = aes(label = paste(model, "AUC: \n", round(auc, 3))),
  x = c(0.25, 0.15),
  y = c(0.4, 1.05)
)
# Influence Plot for Model 2
idCases <- car::influencePlot(model2)
# Set up testing data results
testingData <- testingData %>%
  mutate(
    gradNum = case_when(
      Status == 0 ~ 0,
      Status == 1 ~ 1
    ),
    .after = Status
  )
testingData$predict <- predict(
  object = model2,
  newdata = testingData,
  type = "response"
)
testingData <- testingData %>%
  mutate(
    model2Class = case_when(
      predict > 0.5 ~ "Censored",
      .default = "Deceased"
    )
  )
testingData$Status <- ifelse(testingData$Status == 1, "Censored", "Deceased")

# Build Confusion Matrix for Testing Data
testingData %>%
  tabyl(var1 = model2Class, var2 = Status) %>%
  adorn_title(
    placement = "combined",
    row_name = "Predicted",
    col_name = "Actual"
  ) %>%
  kable(
    caption = "Confusion Matrix for Test data"
  )%>%
kable_styling(latex_options = "HOLD_position")
#Sepeation Plot
par(mar = c(4,0,0,0))
separationplot(
  pred = testingData$predict,
  actual = testingData$gradNum,
  type = "rect",
  line = TRUE,
  lwd2 = 2,
  show.expected = TRUE,
  newplot = FALSE
)
```

```r
#do the ml, take about observational stages vs now quantify, look into the different variables and what
cirrhosisCluster <- cirrhosis
cirrhosisCluster2 <- na.omit(cirrhosis)

cirrhosisCluster <- cirrhosisCluster %>%
                                dplyr::select(-Stage)

cirrhosisCluster <- na.omit(cirrhosisCluster) #cannot have NA values in clustering

cirrhosisCluster$Age_Years <- round(cirrhosisCluster$Age_Years) #round age to whole numbers


#reode sex... recode others later if need be #female is 0

#cirrhosisCluster  <- cirrhosisCluster %>%
#  select( -c(Status, Drug, Edema)) %>%

# mutate(Sex = ifelse(Sex == 'Female', 0, 1))


  cirrhosisCluster  <- cirrhosisCluster %>%
    mutate(Sex = ifelse(Sex == 'Female', 0, 1)) %>%
          mutate(Transplant = ifelse(Status == "CL", 1, 0)) %>%
              mutate(Status = ifelse(Status %in% c('C', 'CL'), 0 , 1)) %>%
                    mutate(Drug = ifelse(Drug == "D-penicillamine", 0, 1)) %>%
                        mutate(EdemaDiurectics = ifelse(Edema %in% c('S', 'Y'), 1, 0)) %>%
                              mutate(NoEdemaORD = ifelse(Edema == 'N' , 1, 0)) %>%
                                        mutate(EdemaANDD = ifelse(Edema == "Y", 1, 0)) %>%
                                                mutate(EdemaORD = ifelse(Edema == "S", 1, 0)) %>%
                                                                      dplyr::select(-Edema)




#Data is already factored

#mutate(Sex = ifelse(Sex == 'Female', 0, 1)) %>%

   #mutate(Status = ifelse(Status %in% c('C', 'CL'), 0 , 1)) %>%
          #    mutate(Drug = ifelse(Drug == "D-penicillamine", 0, 1))


#???:

#make column yes no edema #then yes no under treatment

#same for transplant stuff
distCirrhosis <- dist(
  x = cirrhosisCluster,
  method = "euclidean"
)
```

```r
hybridCirrhosis <- hkmeans(
  x = cirrhosisCluster,
  k = 4,
  hc.metric = "euclidean",
  hc.method = "ward.D",
  iter.max = 10
)


StagesPalette <- c("#AA336A", "#770737", "#40B5AD", "#009E60", "#9FE2BF")

## MAKE A NEW PALLETE TO VISUALIZE
# Plot the initial dendrogram for hybrid approach ----
set.seed(380)
hkmeans_tree(
  hkmeans = hybridCirrhosis,
  rect.col = StagesPalette,
  cex = 0.4,
  main = "Initial Hierarchical Clusters"
)

# Create scree plot for choosing k ----
library(factoextra)
set.seed(380)
fviz_nbclust(
  x = cirrhosisCluster,
  diss = NULL,
  FUNcluster = kmeans,
  method = "wss",
  k.max = 10
)



hybridCirrhosis2 <- hkmeans(
  x = cirrhosisCluster,
  k = 5,
  hc.metric = "euclidean",
  hc.method = "ward.D",
  iter.max = 10
)

# Plot the final dendrogram for hybrid approach ----
# library(factoextra)
fviz_dend(
  x = hybridCirrhosis2,
  cex = 0.4,
  palette = StagesPalette,
  rect = FALSE,
  horiz = TRUE,
  repel = TRUE,
  main = "Final Dendrogram"
)
```

```r
# Plot the final clustering for hybrid approach ----
# library(factoextra)
fviz_cluster(
  object = hybridCirrhosis,
  stand = FALSE,
  geom = "point",
  main = "Hybrid Cluster Plot - Initial model"
) +
  scale_color_manual(values = StagesPalette) +
  scale_fill_manual(values = StagesPalette) +
  theme_bw()


# Plot the final clustering for hybrid approach ----
# library(factoextra)
fviz_cluster(
  object = hybridCirrhosis2,
  stand = FALSE,
  geom = "point",
  main = "Hybrid Cluster Plot - Refined model"
) +
  scale_color_manual(values = StagesPalette) +
  scale_fill_manual(values = StagesPalette) +
  theme_bw()

cirrhosisCluster2$cluster <- hybridCirrhosis2$cluster
# Calculate mean (or median, etc.) for each variable in each cluster
library(dplyr)
cluster_summary <- cirrhosisCluster2 %>%
  group_by(cluster) %>%
  summarise_all(funs(mean(., na.rm = TRUE))) # Replace mean with median or any other function as necess
kable(cluster_summary %>% dplyr::select(-Status, -Drug, - Sex, -Edema, -Stage), label = "Cluster Summary
ggplot(cirrhosisCluster2, aes(x = factor(cluster), fill = factor(Stage))) +
  geom_bar(position = "dodge") +
  labs(title = "Distribution of Stages within Clusters",
       x = "Cluster",
       y = "Count",
       fill = "Stage") +
  theme_minimal()
```