School of Engineering & Technology
Asian Institute of Technology
AT82.01 - Computer Programming for Data Science

Cirrhosis Disease Prediction

Report Submission

To

Dr.Chantri Polprasert

By
Group: 11

Munthitra Thadthapong st124022
Sai Haneesha Bestha st124089

Introduction

About Cirrhosis:

Liver cirrhosis is a widespread problem due to high intake of alcohol. Cirrhosis is a condition in which your liver is scarred and permanently damaged. Scar tissue replaces healthy liver tissue and prevents your liver from working normally. As cirrhosis gets worse, your liver begins to fail.
Cirrhosis is classified into four main stages that include:

→**Stage 1: Steatosis**

→ Inflammation of the liver or bile duct is a hallmark of the initial stage of liver disease.

→ Inflammation and symptoms may usually be treated in stage I to stop liver disease from progressing to stage II.

→ **Stage 2: Scarring**

→Stage II is characterized by inflammation or scarring that starts to impede the liver's normal blood flow.

→The liver is unable to function normally as a result, but it may still be able to recover, preventing further damage, and delaying the advancement of the liver disease with treatment.

→ **Stage 3: Fibrosis Liver**

→When liver disease progresses and scar tissue replaces healthy liver tissue, it leads to cirrhosis, which is primarily caused by a lack of treatment.

→This process happens when an infection or sickness that worsens over time (typically years) destroys healthy liver cells.

→ **Stage 4: Cirrhosis**

→The liver's ability to function will cease if it fails at the final stage of the disease. To prevent deaths, this will require immediate medical attention.

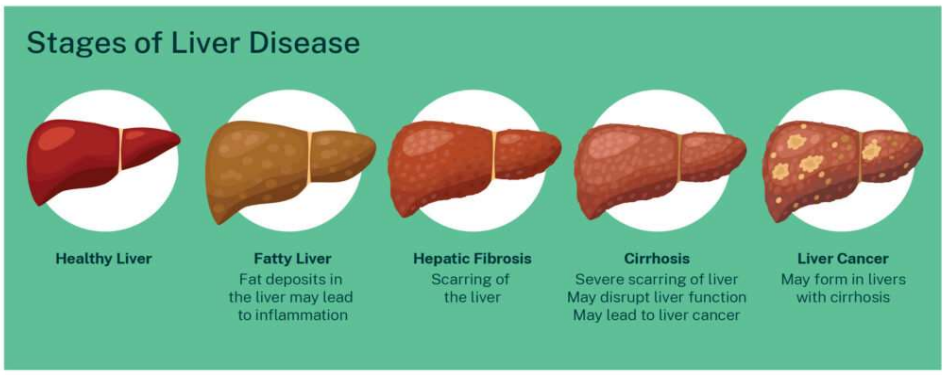


Fig 01: Stages of Cirrhosis of the liver

Reasons for using the Cirrhosis Dataset for this project:

1. The cirrhosis dataset enables comprehensive research on disease progression and treatment, facilitating the development of improved treatment approaches.

2. Data scientists can create predictive models using the dataset to forecast cirrhosis progression, supporting personalized patient care.

3. Analysis of the cirrhosis dataset helps identify crucial risk factors, enabling the implementation of preventive measures to reduce disease incidence.

4. By studying the dataset, patterns in treatment effectiveness can be revealed, leading to the development of targeted treatment strategies for cirrhosis patients.

5. Insights from the cirrhosis dataset aid in understanding the broader public health impact of the disease, informing the development of effective public health policies for prevention and management.

Problem statement

Possible Problem:

Develop a predictive model using the cirrhosis dataset to forecast the most important feature which affects Liver Cirrhosis in Patients with Chronic Liver Disease.

Objectives:

1. Analyze a dataset containing patient attributes, clinical parameters, and disease-related information.

2. Identify the key predictive factors that are significantly associated with liver cirrhosis development.

3. Develop a predictive model to assess the risk of cirrhosis based on data factors.

Significance of the Problem:

Identifying the most important predictive factors for liver cirrhosis is of paramount importance in clinical practice. It can enable healthcare providers to stratify patients based on their risk and make informed decisions about medical interventions and monitoring. Early identification of key predictive factors can also aid in the development of targeted prevention strategies.

Scope of the Project:

This project will focus on analyzing the provided dataset to determine the relative importance of the different attributes in predicting the onset of liver cirrhosis. The scope includes data preprocessing, feature selection, statistical analysis, and the development of a predictive model.

Target Audience:

The primary beneficiaries of this research include healthcare practitioners, clinicians, and researchers working with patients at risk of liver cirrhosis. The findings will inform clinical decision-making and potentially lead to more effective patient care strategies.

Anticipated Outcomes:

The project is expected to result in a better understanding of the most critical predictive factors for liver cirrhosis, which can aid in early diagnosis and improved patient management. The outcomes will facilitate evidence-based medical interventions and contribute to the advancement of liver disease research.

Related Works

1. "Analysis of Risk Factors and Disease Progression in Cirrhosis Patients" - A study investigating the key risk factors associated with the development and progression of cirrhosis, aiming to improve preventive measures and treatment interventions.

2. "Predictive Modeling for Cirrhosis Progression and Treatment Outcomes" - A research paper focusing on the development of predictive models to forecast the progression of cirrhosis and evaluate the effectiveness of different treatment strategies, with implications for personalized patient care.

3. "Long-Term Analysis of Public Health Impact of Cirrhosis: Policy Implications" - A comprehensive analysis exploring the broader public health impact of cirrhosis, providing insights for the development of effective public health policies and initiatives aimed at prevention and management at a larger scale.

4. "Data-Driven Approaches for Personalized Treatment in Cirrhosis Patients" - A study highlighting data-driven approaches for tailoring treatment strategies for patients with cirrhosis, with a focus on optimizing patient outcomes and improving the efficacy of treatment regimens.

Similar Datasets:
1.https://www.kaggle.com/code/arjunbhaybhang/liver-cirrhosis-prediction-with-xgboost-eda
2.https://www.kaggle.com/code/tenebris97/cirrhosis-prediction-cnn-mlp
3.https://www.kaggle.com/code/samshook/liver-cirrhosis-eda-with-python-plotly-express

Datasets

About the Cirrhosis Dataset:

Cirrhosis represents an advanced stage of liver scarring resulting from various liver diseases, including hepatitis and chronic alcoholism. The data gathered from the Mayo Clinic trial on primary biliary cirrhosis (PBC) of the liver, conducted during the period from 1974 to 1984, serves as a vital resource for understanding the clinical background of the trial and the recorded covariates. Additional comprehensive insights can be found in Fleming and Harrington's book "Counting Processes and Survival Analysis" (1991) and in research papers by Dickson et al. (Hepatology, 1989) and Markus et al. (New England Journal of Medicine, 1989).

A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval, met eligibility criteria for the randomized placebo-controlled trial of the drug D-penicillamine. The first 312 cases in the dataset participated in the randomized trial and contain largely complete data. The additional 112 cases did not participate in the clinical trial but consented to have basic measurements recorded and to be followed for survival. Six of those cases were lost to follow-up shortly after diagnosis, so the data here are on an additional 106 cases as well as the 312 randomized participants.

Attribute Information:

This data set has about 19 features. These features are related to the patient’s details like age, sex, etc. and patient’s blood tests like prothrombin, triglycerides, platelets levels, etc. All these factors help in understanding a patient’s chances of liver cirrhosis.

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: triglycerides 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)

Methodology (EDA, Data Preprocessing)

1. Data Types

Numerical data: Includes columns with integer (int) and floating-point (float) values like Age, Cholesterol, Platelets, etc.

Categorical data: Includes columns with object data types like Drug, Sex, Spiders, etc.

2. Handling Missing Values

For numerical data: Our group plans to impute missing values using the median to avoid skewing in the presence of outliers. This is a common approach for numerical data.

For categorical data: Our group plans to impute missing values using the mode, which is the most frequent value in each respective column. This is a common approach for categorical data.

3. Converting Age from Days to Years

The "Age" column contains data in days, which can be difficult to analyze. Your group has decided to convert this column into years for easier interpretation and analysis.

4. Setting up Target and Features

According to the dataset, it has class imbalances. This could make it difficult for our model to train and achieve the desired score. For this demonstration, our group will keep things simple by predicting Cirrhosis and No Cirrhosis. We will define the 'Stage' column as follows: Stage 1 and 2 will be considered No Cirrhosis, labeled as 0, and Stage 3 and 4 will be considered Cirrhosis, labeled as 1." (Label encoding)

Target Variable:

- Create a new binary target variable, often referred to as a binary classification task.
- Use the "Stage" column to determine the values of the new target variable:

  Stage 1 and 2 should be assigned a value of 0, representing "No Cirrhosis."

  Stage 3 and 4 should be assigned a value of 1, representing "Cirrhosis."

Features:

- Features include Drug, Age, Sex, Ascites, Hepatomegaly, Spiders, Edema, Bilirubin, Cholesterol, Albumin, Copper, Alk_Phos, SGOT, Triglycerides, Platelets and Prothrombin.
- Categorical features need to be encoding them into numerical format by using label encoding for Sex, Ascites, Drug, Hepatomegaly and Spiders column. Using one hot encoding for the Edema and Status column.

Detailed Explanation Each Feature :

5. Data Splitting

In order to train and evaluate a machine learning model, it is essential to divide your dataset into two separate sets: a training set and a testing set. In this case, we will use an 80-20 split, which means that 80% of the data will be allocated to the training set, and the remaining 20% will be used for testing.

Preliminary results

Import of Libraries :

Step-1 : Import the necessary libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler, KBinsDiscretizer, OrdinalEncoder
from xgboost import XGBClassifier
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import KFold
```

Step-2 : Check the Versions of libraries that are being used(optional)

```
[ ] import matplotlib
    np.__version__, pd.__version__, sns.__version__, matplotlib.__version__

    ('1.26.0', '2.1.3', '0.13.0', '3.8.1')
```

Loading of Data :

Step-1 : Loading the data set and reading the first 10 entries of the Cirrhosis dataset along with shape of the data set

1. Load Data

```
[ ] df = pd.read_csv('cirrhosis.csv')
    df.head(10)
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	14.5	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	4.0
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	1.1	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	3.0
2	3	1012	D	D-penicillamine	25594	M	N	N	N	S	1.4	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	4.0
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	1.8	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	4.0
4	5	1504	CL	Placebo	13918	F	N	Y	Y	N	3.4	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	3.0
5	6	2503	D	Placebo	24201	F	N	Y	N	N	0.8	248.0	3.98	50.0	944.0	93.00	63.0	NaN	11.0	3.0
6	7	1832	C	Placebo	20284	F	N	Y	N	N	1.0	322.0	4.09	52.0	824.0	60.45	213.0	204.0	9.7	3.0
7	8	2466	D	Placebo	19379	F	N	N	N	N	0.3	280.0	4.00	52.0	4651.2	28.38	189.0	373.0	11.0	3.0
8	9	2400	D	D-penicillamine	15526	F	N	N	Y	N	3.2	562.0	3.08	79.0	2276.0	144.15	88.0	251.0	11.0	2.0
9	10	51	D	Placebo	25772	F	Y	N	Y	Y	12.6	200.0	2.74	140.0	918.0	147.25	143.0	302.0	11.5	4.0

```
[ ] df.shape

    (418, 20)
```

Step-2 : Getting the information about the dataset and type of the columns in that Dataset

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              418 non-null    int64
 1   N_Days          418 non-null    int64
 2   Status          418 non-null    object
 3   Drug            312 non-null    object
 4   Age             418 non-null    int64
 5   Sex             418 non-null    object
 6   Ascites         312 non-null    object
 7   Hepatomegaly    312 non-null    object
 8   Spiders         312 non-null    object
 9   Edema           418 non-null    object
 10  Bilirubin       418 non-null    float64
 11  Cholesterol     284 non-null    float64
 12  Albumin         418 non-null    float64
 13  Copper          310 non-null    float64
 14  Alk_Phos        312 non-null    float64
 15  SGOT            312 non-null    float64
 16  Tryglicerides   282 non-null    float64
 17  Platelets       407 non-null    float64
 18  Prothrombin     416 non-null    float64
 19  Stage           412 non-null    float64
dtypes: float64(10), int64(3), object(7)
memory usage: 65.4+ KB
```

Step-3 : Describing the details of columns of dataset

```
df.describe()
```

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
count	418.000000	418.000000	418.000000	418.000000	284.000000	418.000000	310.000000	312.000000	312.000000	282.000000	407.000000	416.000000	412.000000
mean	209.500000	1917.782297	18533.351675	3.220813	369.510563	3.497440	97.648387	1982.655769	122.556346	124.702128	257.024570	10.731731	3.024272
std	120.810458	1104.672992	3815.845055	4.407506	231.944545	0.424972	85.613920	2140.388824	56.699525	65.148639	98.325585	1.022000	0.882042
min	1.000000	41.000000	9598.000000	0.300000	120.000000	1.960000	4.000000	289.000000	26.350000	33.000000	62.000000	9.000000	1.000000
25%	105.250000	1092.750000	15644.500000	0.800000	249.500000	3.242500	41.250000	871.500000	80.600000	84.250000	188.500000	10.000000	2.000000
50%	209.500000	1730.000000	18628.000000	1.400000	309.500000	3.530000	73.000000	1259.000000	114.700000	108.000000	251.000000	10.600000	3.000000
75%	313.750000	2613.500000	21272.500000	3.400000	400.000000	3.770000	123.000000	1980.000000	151.900000	151.000000	318.000000	11.100000	4.000000
max	418.000000	4795.000000	28650.000000	28.000000	1775.000000	4.640000	588.000000	13862.400000	457.250000	598.000000	721.000000	18.000000	4.000000

EDA , Pre-processing & Feature Engineering:

Step-1 : Learning if the dataset has any “Missing values” using “isna().sum()”

2. EDA & Pre Processing & Feature Engineering

```
[ ] df.isna().sum()

ID                0
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
dtype: int64
```

Step-2 : Selecting the data types that are int and float to find out the missing values

```
[ ] df.select_dtypes(include=(['int64', 'float64'])).isna().sum()

    ID              0
    N_Days          0
    Age             0
    Bilirubin       0
    Cholesterol     134
    Albumin         0
    Copper          108
    Alk_Phos        106
    SGOT            106
    Tryglicerides   136
    Platelets       11
    Prothrombin     2
    Stage           6
    dtype: int64
```

Step-3 : Handling the missing values that are found in step-2.

```
df.select_dtypes(include=(['int64', 'float64'])).isna().sum()
df_num_col = df.select_dtypes(include=(['int64', 'float64'])).columns
for c in df_num_col:
    df[c].fillna(df[c].median(), inplace=True)

df.select_dtypes(include=(['int64', 'float64'])).isna().sum()

ID              0
N_Days          0
Age             0
Bilirubin       0
Cholesterol     0
Albumin         0
Copper          0
Alk_Phos        0
SGOT            0
Tryglicerides   0
Platelets       0
Prothrombin     0
Stage           0
dtype: int64
```

Step-4 : Finding the missing values for categorical values.

```
[ ] # For Categorical type
    df.select_dtypes(include=('object')).isna().sum()

    Status          0
    Drug          106
    Sex             0
    Ascites       106
    Hepatomegaly  106
    Spiders       106
    Edema           0
    dtype: int64
```

Step-5 : Handling the missing values for categorical features**.**

```
df_cat_col = df.select_dtypes(include=('object')).columns
for c in df_cat_col:
    df[c].fillna(df[c].mode().values[0], inplace=True)

df.select_dtypes(include=('object')).isna().sum()
```

```
Status         0
Drug           0
Sex            0
Ascites        0
Hepatomegaly   0
Spiders        0
Edema          0
dtype: int64
```

Step-6 : Checking if the missing values have been handled after the changes made.

```
df.isna().sum()
```

```
ID             0
N_Days         0
Status         0
Drug           0
Age            0
Sex            0
Ascites        0
Hepatomegaly   0
Spiders        0
Edema          0
Bilirubin      0
Cholesterol    0
Albumin        0
Copper         0
Alk_Phos       0
SGOT           0
Tryglicerides  0
Platelets      0
Prothrombin    0
Stage          0
dtype: int64
```

As we can see the missing values have been handled which is why all the features have “0” .As the missing values have been handled well ,the data will not have data misleadings or biased results, which helps in maintaining the data integrity , along with improving the performance of model which we will do in further steps.

Step-7 : Checking if there are any Duplicate Rows.

```
[ ] # Check for duplicated rows
    duplicated_rows = df[df.duplicated()]
    sum_of_duplicated_rows = duplicated_rows.groupby(duplicated_rows.columns.tolist()).size().reset_index(name='count')
    print(sum_of_duplicated_rows)
```

```
Empty DataFrame
Columns: [ID, N_Days, Status, Drug, Age, Sex, Ascites, Hepatomegaly, Spiders, Edema, Bilirubin, Cholesterol, Albumin, Copper, Alk_Phos, SGOT, Tryglicerides, Platelets, Prothrombin, Sta
Index: []

[0 rows x 21 columns]
```

Step-8 : Creating a new column in a DataFrame (df) called 'AgeInYears' by converting the 'Age' column values from days to years and rounding the result.

```
df['AgeInYears'] = (df['Age'] / 365.25).round()
```

```
[ ] df.head()
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	...	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage	A
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	...	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	4.0	
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	...	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	3.0	
2	3	1012	D	D-penicillamine	25594	M	N	N	N	S	...	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	4.0	
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	...	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	4.0	
4	5	1504	CL	Placebo	13918	F	N	Y	Y	N	...	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	3.0	

5 rows × 21 columns

Step-9 : Calculating the stages of Cirrhosis dataset.

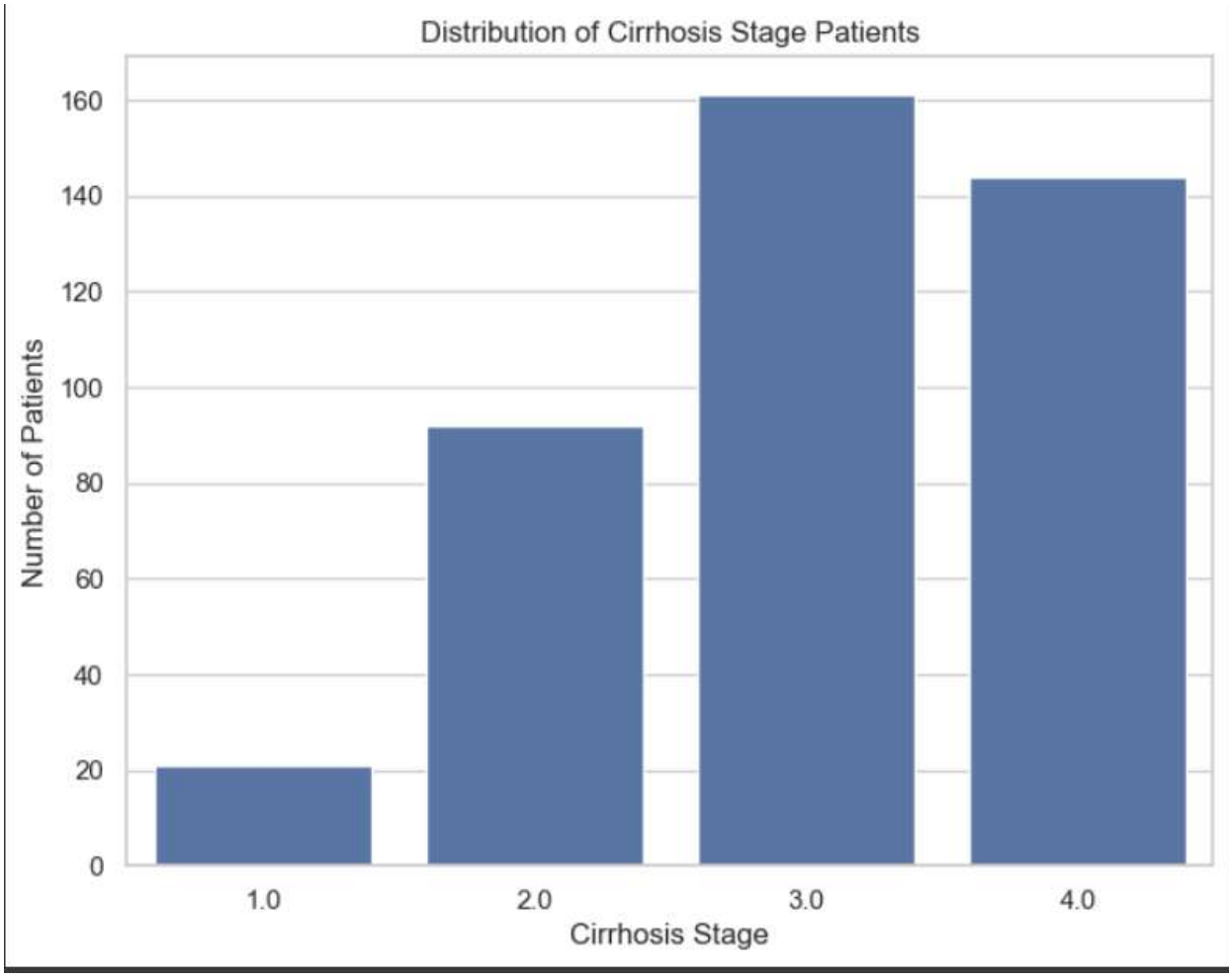
```
df['Stage'].value_counts()
```

```
Stage
3.0    161
4.0    144
2.0     92
1.0     21
Name: count, dtype: int64
```

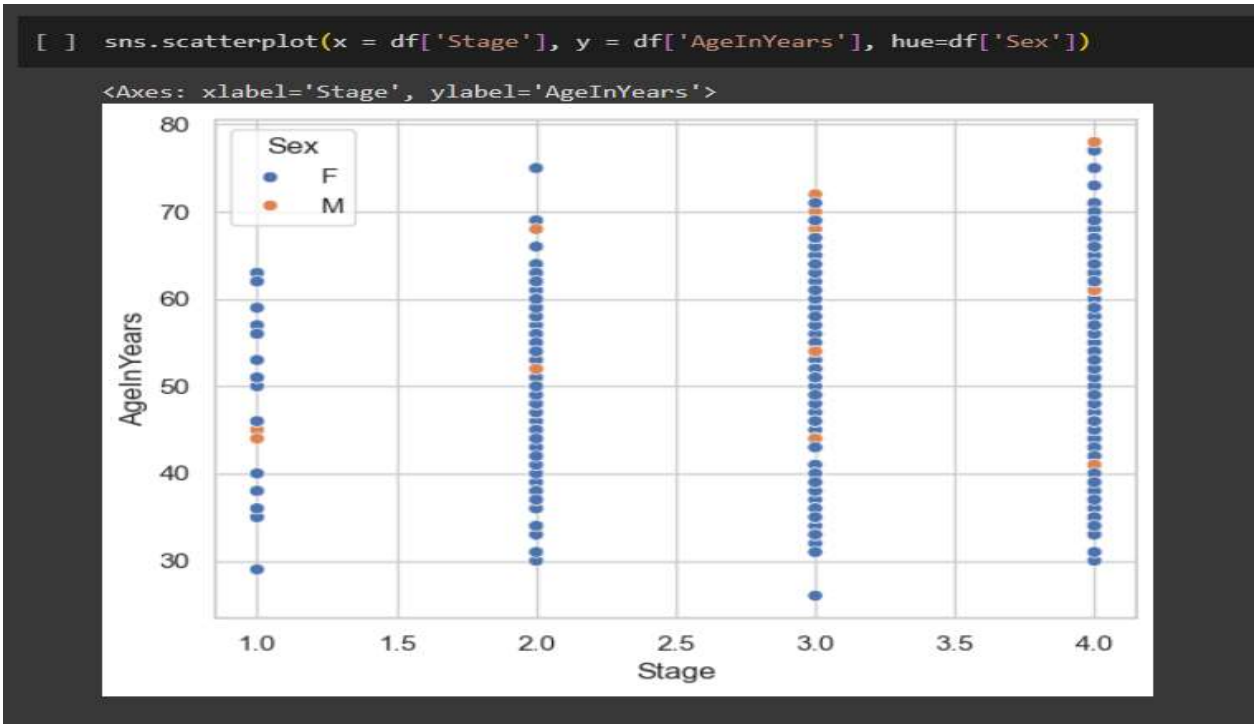
Step-10 : Counting the no:of patients in each stage and plotting the graph of results obtained.

```
[ ] # Count the number of patients in each stage
    stage_counts = df['Stage'].value_counts()

[ ] sns.set(style="whitegrid")
    plt.figure(figsize=(8, 6))
    sns.barplot(x=stage_counts.index, y=stage_counts.values)
    plt.xlabel('Cirrhosis Stage')
    plt.ylabel('Number of Patients')
    plt.title('Distribution of Cirrhosis Stage Patients')
    plt.show()
```

Graph of the results obtained.

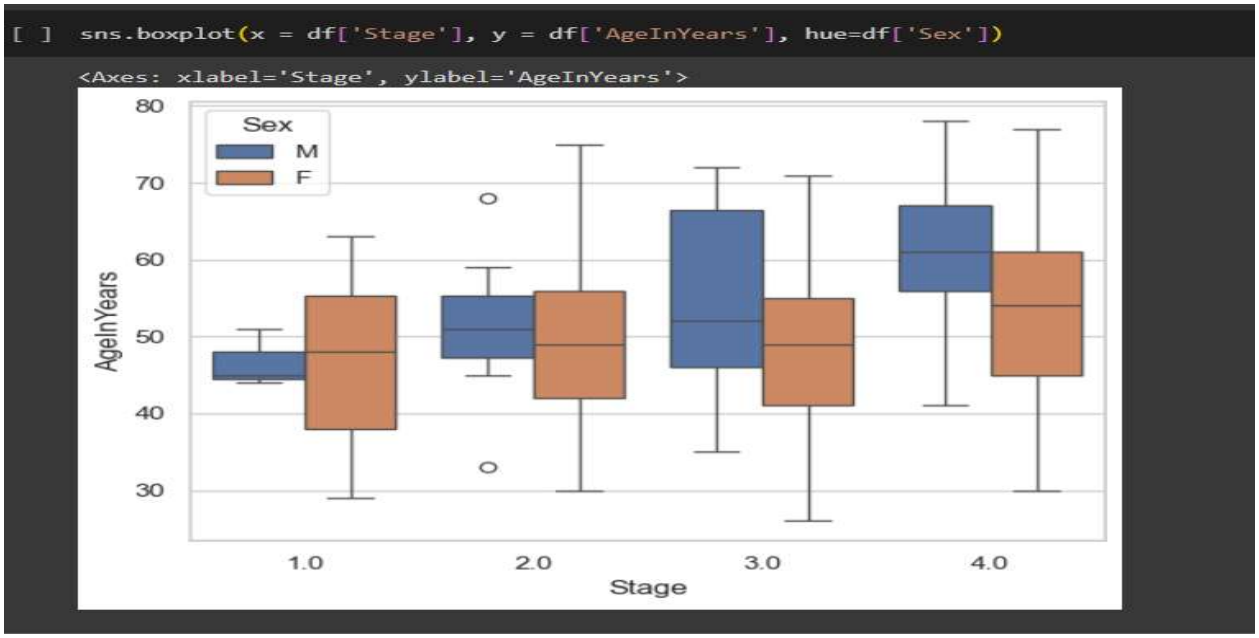


Step-11: Plotting a graph to know which age of patients are likely in which stage of Cirrhosis disease. The plotting is not only based on Age but also based on Male and Female ratio.



As we can see according to plottings the no:of patients who are in different stages of Cirrhosis are more female than the male.

Step-12 : Boxplot of the same information obtained in Step-11



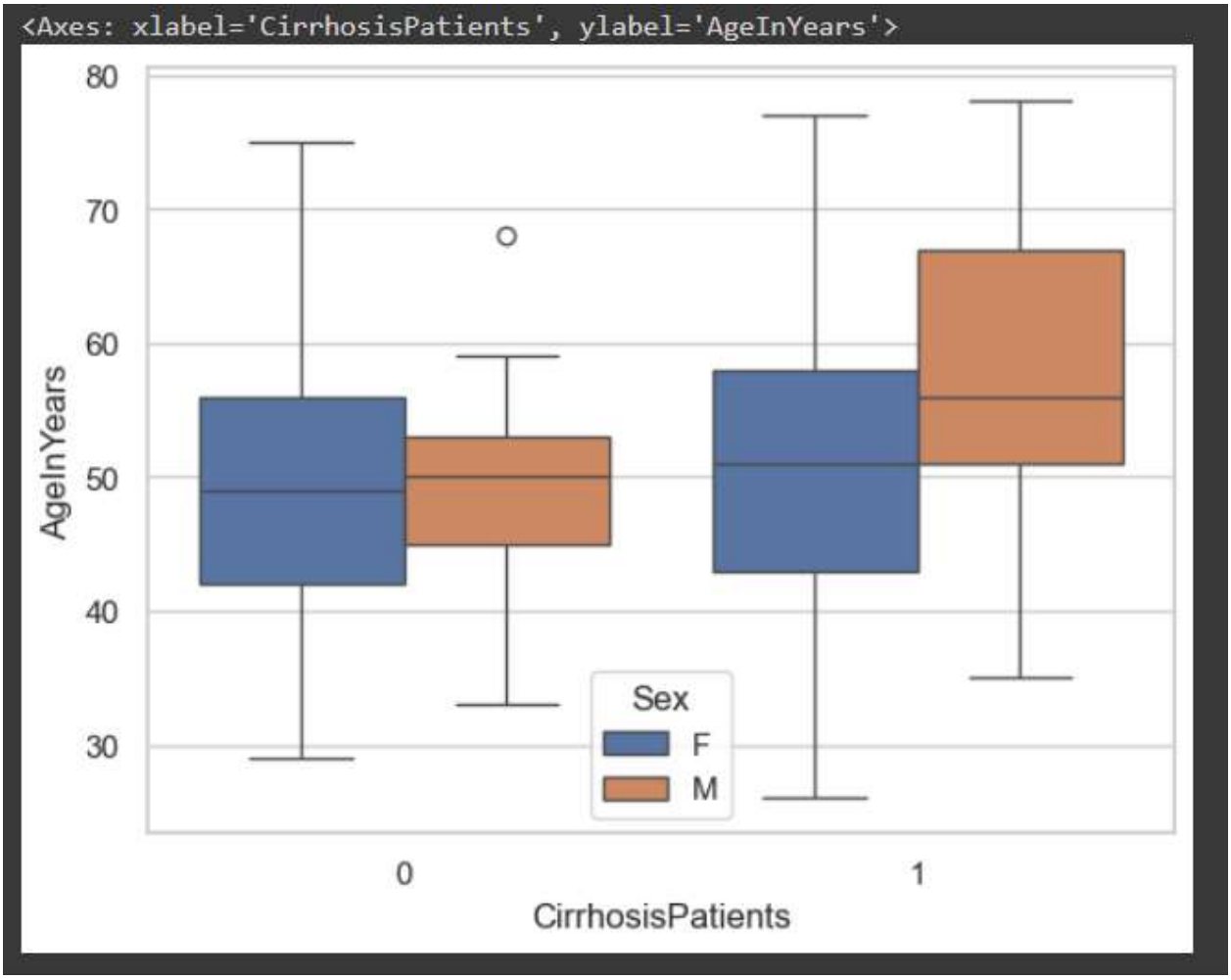
Step-13 : Creating a binary indicator variable based on certain conditions in a DataFrame.

```
# Map stage 3 and 4 to 1, and stage 1 and 2 to 0
df['CirrhosisPatients'] = np.where((df['Stage'] == 3) | (df['Stage'] == 4), 1, 0)
```

```
cirrhosis_df = df[df['CirrhosisPatients'] == 1]
no_cirrhosis_df = df[df['CirrhosisPatients'] == 0]
```

```
sns.boxplot(x = df['CirrhosisPatients'], y = df['AgeInYears'], hue=df['Sex'])
```

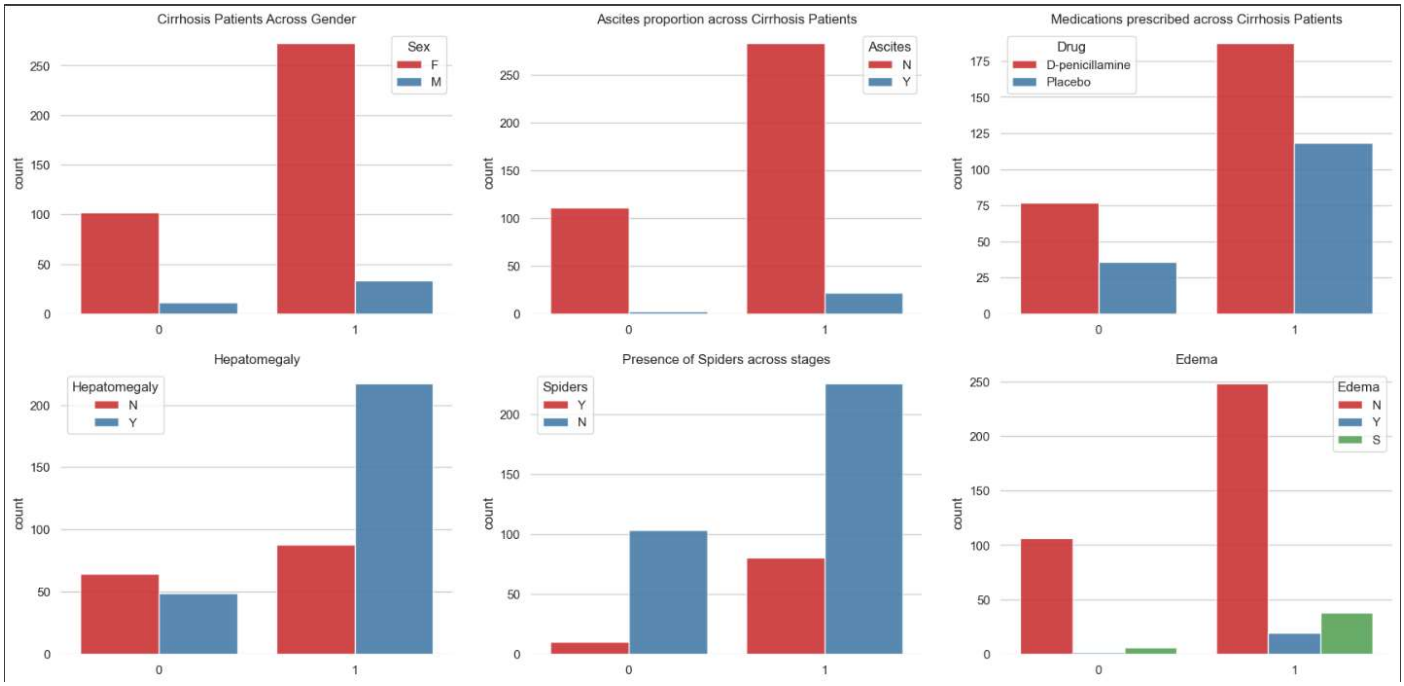
Plotting of the above conditions defined.



Step-14 : The provided code is used to create a 2x3 grid of subplots using Matplotlib and Seaborn to visualize various count distributions across cirrhosis patients in different categories.

```
plt.figure(figsize=(21.2,10))

plt.subplot(2,3,1)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Sex'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
plt.xlabel('')
plt.title('Cirrhosis Patients Across Gender')

plt.subplot(2,3,2)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Ascites'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
plt.xlabel('')
plt.title('Ascites proportion across Cirrhosis Patients')

plt.subplot(2,3,3)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Drug'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
plt.xlabel('')
plt.title('Medications prescribed across Cirrhosis Patients');

plt.subplot(2,3,4)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Hepatomegaly'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
plt.xlabel('')
plt.title('Hepatomegaly');

plt.subplot(2,3,5)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Spiders'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
plt.xlabel('')
plt.title('Presence of Spiders across stages');

plt.subplot(2,3,6)
sns.countplot(x=df['CirrhosisPatients'], hue=df['Edema'], palette='Set1', alpha=0.9)
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False, left=False)
```

Plottings of the above code.



Step-15 : Printing all the Columns in the data set except for AGE,ID and STAGE

```
[ ] df.loc[:, (df.columns != "Age")&(df.columns != "ID")&(df.columns != "Stage")]
```

	N_Days	Status	Drug	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	AgeInYears	CirrhosisPatients
0	400	D	D-penicillamine	F	Y	Y	Y	Y	14.5	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	59.0	1
1	4500	C	D-penicillamine	F	N	Y	Y	N	1.1	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	56.0	1
2	1012	D	D-penicillamine	M	N	N	N	S	1.4	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	70.0	1
3	1925	D	D-penicillamine	F	N	Y	Y	S	1.8	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	55.0	1
4	1504	CL	Placebo	F	N	Y	Y	N	3.4	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	38.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	681	D	D-penicillamine	F	N	Y	N	N	1.2	309.5	2.96	73.0	1259.0	114.70	108.0	174.0	10.9	67.0	1
414	1103	C	D-penicillamine	F	N	Y	N	N	0.9	309.5	3.83	73.0	1259.0	114.70	108.0	180.0	11.2	39.0	1
415	1055	C	D-penicillamine	F	N	Y	N	N	1.6	309.5	3.42	73.0	1259.0	114.70	108.0	143.0	9.9	57.0	1
416	691	C	D-penicillamine	F	N	Y	N	N	0.8	309.5	3.75	73.0	1259.0	114.70	108.0	269.0	10.4	58.0	1
417	976	C	D-penicillamine	F	N	Y	N	N	0.7	309.5	3.29	73.0	1259.0	114.70	108.0	350.0	10.6	53.0	1

418 rows × 19 columns

Step-16 : Printing all the columns in the dataset excluding the features whose data type is object.

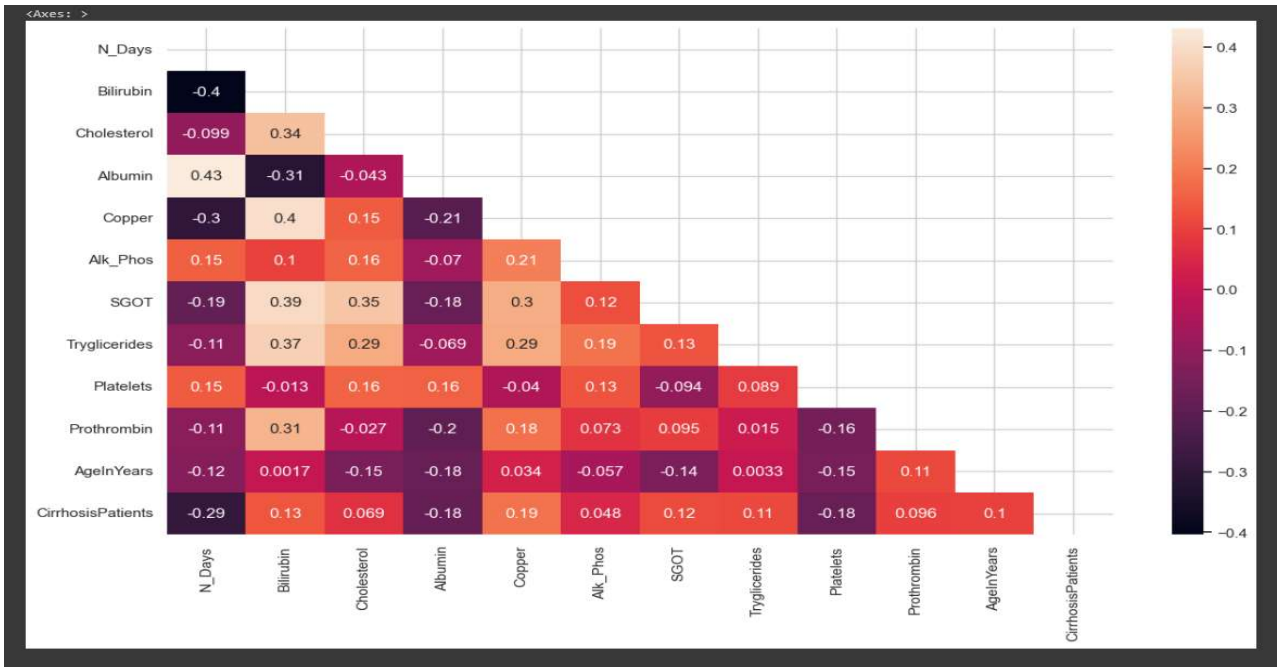
```
df.select_dtypes(exclude = ['object'])
```

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage	AgeInYears	CirrhosisPatients
0	1	400	21464	14.5	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	4.0	59.0	1
1	2	4500	20617	1.1	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	3.0	56.0	1
2	3	1012	25594	1.4	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	4.0	70.0	1
3	4	1925	19994	1.8	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	4.0	55.0	1
4	5	1504	13918	3.4	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	3.0	38.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	414	681	24472	1.2	309.5	2.96	73.0	1259.0	114.70	108.0	174.0	10.9	3.0	67.0	1
414	415	1103	14245	0.9	309.5	3.83	73.0	1259.0	114.70	108.0	180.0	11.2	4.0	39.0	1
415	416	1055	20819	1.6	309.5	3.42	73.0	1259.0	114.70	108.0	143.0	9.9	3.0	57.0	1
416	417	691	21185	0.8	309.5	3.75	73.0	1259.0	114.70	108.0	269.0	10.4	3.0	58.0	1
417	418	976	19358	0.7	309.5	3.29	73.0	1259.0	114.70	108.0	350.0	10.6	4.0	53.0	1

418 rows × 15 columns

Step-17 : Plotting the Upper triangular correlation Matrix

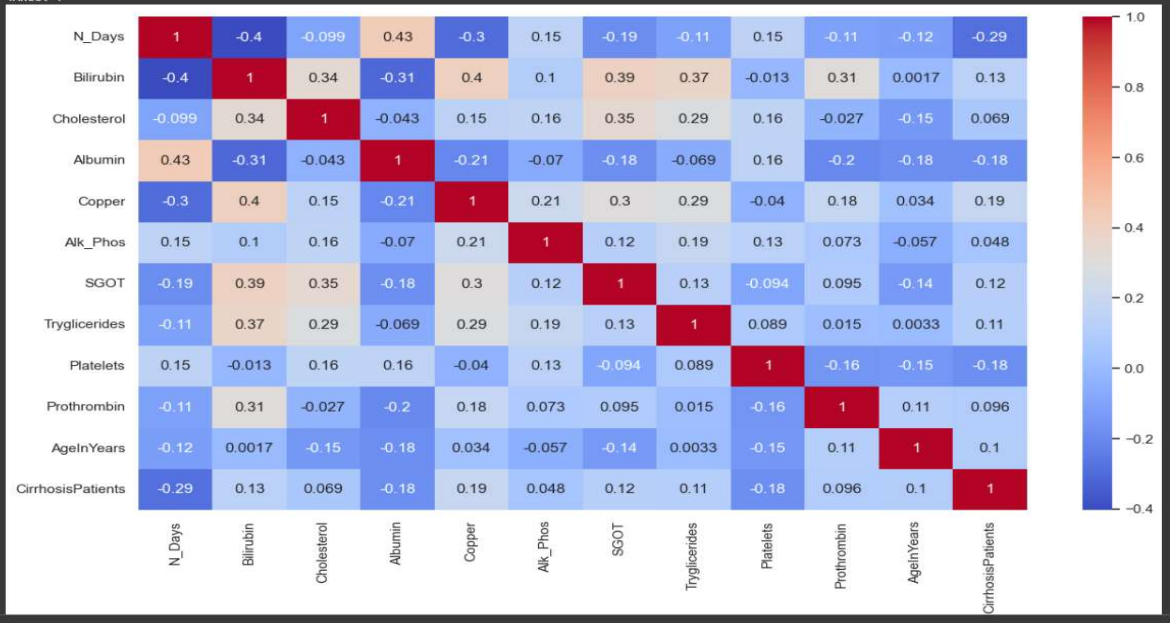
```
[ ] plt.figure(figsize = (15,8))

    df_numerical = df.select_dtypes(exclude = ['object'])

    corr = df_numerical.loc[:, (df_numerical.columns != "Age")&(df_numerical.columns != "ID")&(df_numerical.columns != "Stage")].corr()

    # Getting the Upper Triangle of the co-relation matrix
    matrix = np.triu(corr)

    # using the upper triangle matrix as mask
    sns.heatmap(corr, annot=True, mask=matrix)
```



Step-18 : Heatmap of the Cirrhosis Data

```
plt.figure(figsize = (15,8))
sns.heatmap(corr, annot=True, cmap="coolwarm")
```

```
<Axes: >
```



Step-19 : Checking if there is any imbalance and correcting the imbalance

```
[ ] df['CirrhosisPatients'].value_counts()
```

```
CirrhosisPatients
1    305
0    113
Name: count, dtype: int64
```

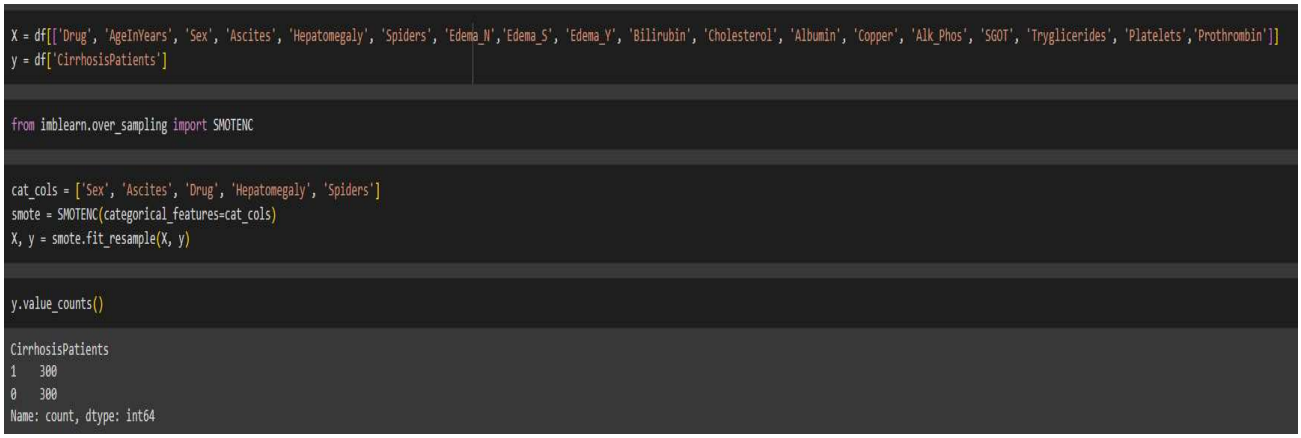
```
#Balance the y

condY = df.CirrhosisPatients == 1  #to 0 or 1
condN = df.CirrhosisPatients == 0

df_Y = df[condY].sample(n=300, random_state=999)
df_N = df[condN] #also 113

df = pd.concat([df_Y, df_N])

df.CirrhosisPatients.value_counts()
```

```
CirrhosisPatients
1    300
0    113
Name: count, dtype: int64
```

Feature Selection:

Feature selection is a crucial step in the process of building machine learning models, and it involves choosing a subset of relevant features from the original set of features. Selecting the right features helps in improving the performance of the model ,it helps in faster training time and also enhances the model interpretability very well .Let's see how it work with our dataset.

Step-1 :Selecting the features and by using class balance we are balancing the classes in the dataset.



```
X = df[['Drug', 'AgeInYears', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema_N','Edema_S', 'Edema_Y', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets','Prothrombin']]
y = df['CirrhosisPatients']

from imblearn.over_sampling import SMOTENC

cat_cols = ['Sex', 'Ascites', 'Drug', 'Hepatomegaly', 'Spiders']
smote = SMOTENC(categorical_features=cat_cols)
X, y = smote.fit_resample(X, y)

y.value_counts()

CirrhosisPatients
1    300
0    300
Name: count, dtype: int64
```

Step-2 : Splitting the dataset into a training and testing set and checking if there are any missing values in the X_train .

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
[ ]  X_train.isna().sum()
```

```
Drug            0
AgeInYears      0
Sex             0
Ascites         0
Hepatomegaly    0
Spiders         0
Edema_N         0
Edema_S         0
Edema_Y         0
Bilirubin       0
Cholesterol     0
Albumin         0
Copper          0
Alk_Phos        0
SGOT            0
Tryglicerides   0
Platelets       0
Prothrombin     0
dtype: int64
```

Step-3 : Finding the unique values for the X_train set

```
# Function for finding each column unique values
def cols_unique(columns):
  for col in columns:
    print(f"{col}: {X_train[col].unique()}\n")
```

```
[ ] cols_unique(X_train)
```

```
Drug: ['D-penicillamine' 'Placebo']

AgeInYears: [35.         64.         61.46273023 55.56849067 54.40168279 43.
 62.         53.71955324 48.         39.         46.21604974 45.41118594
 57.         51.         69.         53.67179456 55.         33.
 75.         43.09518247 38.28450412 49.         50.         39.09652207
 43.29225582 46.35309323 51.81865424 59.         67.         47.
 71.         48.37245791 53.         50.02056628 45.         52.84917914
 41.         50.02468374 53.91262348 35.18577278 60.         31.703699
 38.         39.33387056 42.         56.         52.         49.04030695
 50.60198663 46.         63.         43.27495979 29.         55.51101895
 35.27987914 44.         34.         54.         51.03571192 45.86497291
 57.40048852 44.63001214 48.32113754 56.25370515 55.58607028 70.
 48.65860122 61.         40.90262845 62.91605154 39.99161847 54.7823022
 45.37174825 37.         47.26151349 60.19286831 41.31682419 45.57768121
 36.         45.9590851  31.         65.         61.90751215 51.91134526
 38.1441738  50.36967135 50.56034291 44.19852036 52.17462974 60.54106316
 55.27151504 50.28349013 43.54309321 40.65446979 39.62410371 55.92169136
 41.14683118 59.79958022 49.47789142 48.31065451 50.48235004 46.44141593
 59.24362998 52.93809952 55.75538131 42.95341698 50.33391673 66.
 52.95831967 43.68979078 59.18209929 40.         44.76225005 55.59774391
 68.         54.94334196 56.58400797 58.14349944 61.28110374 47.26910037
 52.69085112 55.08204167 33.62917747 51.52360152 72.         49.0416313
 48.01606858 56.91597554 49.89679437 45.31857085 58.64252044 64.73916536
 50.49513997 58.30659292 78.         51.50170739 55.28417288 53.59241208
 32.         53.83705607 48.09269062 57.42914861 73.         55.52888769
 58.         33.14282862 41.25180805 46.47761332 52.82515386 47.53334137
 30.         44.26258956 46.02309097 59.04310936 46.52180947 42.78800285
 50.11929369 68.44400012 44.82631754 53.97472387 45.87199689 41.38696605
 50.26495924 49.19240224 60.31815386 55.2838181  55.53924087 49.36585802
 56.81020807 35.81163912 42.35841868 56.82613269 45.43675598 40.25249187
```

Model Selection :

It is essential for achieving good generalization performance, avoiding overfitting or underfitting, and ensuring that the chosen model meets the specific requirements of the application.

Step-1 :Standardization and Scaling

```
[ ] # Standardization and Scaling
    num_cols = ['AgeInYears', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']
    enc_cols = ['Sex', 'Ascites', 'Drug', 'Hepatomegaly', 'Spiders']

    remainder_transformer = 'passthrough'

    scaler_encoder = ColumnTransformer(
        transformers=[
            ('standardize', StandardScaler(), num_cols),
            ('ordinal_encoder', OrdinalEncoder(handle_unknown="error"), enc_cols)
        ],
        remainder=remainder_transformer,
        verbose_feature_names_out=False
    ).set_output(transform='pandas')
```

Step-2 :Discretization

```
[ ] # Discretization
    column_to_discretize = 'AgeInYears'

    remainder_transformer = 'passthrough'
    binner = ColumnTransformer(
        transformers=[
            ('binning', KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='uniform'), [column_to_discretize])
        ],
        remainder=remainder_transformer,
        verbose_feature_names_out=False
    ).set_output(transform='pandas')
```

Step-3 :Prediction and Pipeline Creation

```
[ ] #Prediction
    model = XGBClassifier(learning_rate=0.8, max_depth=3, random_state=1, gamma=0, eval_metric='error')
```

```
[ ] pipeline = Pipeline([
        ('scaling', scaler_encoder),
        ('discretization', binner),
        ('prediction', model)])
```

Step-4 : Cross Validation Loop

```
[ ] # Cross-validation loop
    skf = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    acc = []

    for train_index, test_index in skf.split(X_train, y_train):
        X_train_cv, X_test_cv = X_train.iloc[train_index, :], X_train.iloc[test_index, :]
        y_train_cv, y_test_cv = y_train.iloc[train_index], y_train.iloc[test_index]

        pipeline.fit(X_train_cv, y_train_cv)
        score = pipeline.score(X_test_cv, np.ravel(y_test_cv))
        acc.append(score)
        print(f'For Fold {len(acc)} the accuracy is {score}')

    print()
    print('XGboost model Mean Accuracy = ', np.mean(acc))
```

```
For Fold 1 the accuracy is 0.7142857142857143
For Fold 2 the accuracy is 0.7619047619047619
For Fold 3 the accuracy is 0.7380952380952381
For Fold 4 the accuracy is 0.7142857142857143
For Fold 5 the accuracy is 0.7142857142857143
For Fold 6 the accuracy is 0.6666666666666666
For Fold 7 the accuracy is 0.6904761904761905
For Fold 8 the accuracy is 0.8333333333333334
For Fold 9 the accuracy is 0.7619047619047619
For Fold 10 the accuracy is 0.8095238095238095

XGboost model Mean Accuracy =  0.7404761904761905
```

```python
# Tried another model
models = [
    ('Logistic', LogisticRegression()),
    ('Decision Tree', DecisionTreeClassifier()),
    ('Naive Bayes', GaussianNB()),
    ('Random Forest', RandomForestClassifier()),
    ('Support Vector Machine', SVC())
]

# Number of splits for cross-validation
n_splits = 5

# Initialize KFold
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Define parameter grids for each model
param_grid = {
    'Random Forest': {'n_estimators': [50, 100, 200], 'max_depth': [3, 5, 10, 20]},
    'Support Vector Machine': {'C': [0.001, 0.01, 0.1, 1], 'kernel': ['linear', 'rbf']},
    'Logistic': {'C': [0.1, 1, 10], 'penalty': ['l1', 'l2']},
    'Decision Tree': {'max_depth': [3, 5, 7, None], 'min_samples_split': [2, 5, 10]},
    'Naive Bayes': {}  # No hyperparameters to tune for Gaussian Naive Bayes
}
```

```python
# Results dictionary to store the best parameters and scores
results = {}

# Best models and their parameters
best_models = {}

# Perform grid search for each model
for model_name, model in models:
    parameters = param_grid[model_name]
    parameters = {"prediction__" + str(key): val for key, val in parameters.items()}

    pipeline = Pipeline([
    ('scaling', scaler_encoder),
    ('discretization', binner),
    ('prediction', model)])

    # Grid search for mean squared error
    grid_search_acc = GridSearchCV(pipeline, parameters, cv=kf, scoring='accuracy')
    grid_search_acc.fit(X_train, y_train)

    # Grid search for R²
    grid_search_f1 = GridSearchCV(pipeline, parameters, cv=kf, scoring='f1')
    grid_search_f1.fit(X_train, y_train)

    # Store the best parameters and scores
    results[model_name] = {
        'Best Parameters (Accuracy)': grid_search_acc.best_params_,
        'Best Mean Accuracy': (grid_search_acc.best_score_),
        'Best Parameters (F1)': grid_search_f1.best_params_,
        'Best F1': grid_search_f1.best_score_
    }

    # Store the best model
    best_models[model_name] = grid_search_acc.best_estimator_

# Print the information of all models
for model_name, result in results.items():
    print(f"{model_name} - Best Parameters (Accuracy): {result['Best Parameters (Accuracy)']}; Best Mean Accuracy: {result['Best Mean Accuracy']}; Best Parameters (F1): {result['Best Parameters (F1)']}; Best F1: {result['Best F1']}")

# Identify the best model based on R² or another metric
best_model_name = max(results, key=lambda key: results[key]['Best F1'])
best_model = best_models[best_model_name]

# Train the best model on the entire training dataset
best_model.fit(X_train, y_train)
```

Creation of Pipeline:

Step-5 : Importing the model into pickle and predicting the best model with testing set

```
[ ]  import pickle
     import joblib

[ ]  pickle.dump(best_model, open('best_model.pkl', 'wb+'))

[ ]  best_model2 = pickle.load(open('best_model.pkl', 'rb+'))

[ ]  best_model2.predict(X_test)

     array([1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
            0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
            1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
            0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
            0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
            1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
            0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
            1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
            0, 1, 1, 0])
```

Testing: Testing the ypred value with the best model created earlier

```
y_pred = best_model2.predict(X_test)
from sklearn.metrics import classification_report

yhat = best_model2.predict(X_test)
print(classification_report(y_test, yhat))
```

```
              precision    recall  f1-score   support

           0       0.68      0.79      0.73        82
           1       0.80      0.68      0.74        98

    accuracy                           0.73       180
   macro avg       0.74      0.74      0.73       180
weighted avg       0.74      0.73      0.73       180
```

Feature Importance Analysis :

Step-1 :

```
[ ] column = X_train.columns
```

```
[ ] forest_model = best_model['prediction']  # No need for named_steps
    display(forest_model.feature_importances_)
```

```
array([0.06482454, 0.10797968, 0.06177384, 0.11477852, 0.06423155,
       0.06442459, 0.07704057, 0.07566153, 0.1198367 , 0.0830192 ,
       0.00793239, 0.00183802, 0.0237601 , 0.04960886, 0.04336769,
       0.0302372 , 0.00820784, 0.00147718])
```

Step-2 : Checking the feature importance of all the features

```
import pandas as pd

feature_importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': forest_model.feature_importances_
})

# Display the DataFrame
display(feature_importance_df)
```

	Feature	Importance
0	Drug	0.064825
1	AgeInYears	0.107980
2	Sex	0.061774
3	Ascites	0.114779
4	Hepatomegaly	0.064232
5	Spiders	0.064425
6	Edema_N	0.077041
7	Edema_S	0.075662
8	Edema_Y	0.119837
9	Bilirubin	0.083019
10	Cholesterol	0.007932
11	Albumin	0.001838
12	Copper	0.023760
13	Alk_Phos	0.049609
14	SGOT	0.043368
15	Tryglicerides	0.030237
16	Platelets	0.008208
17	Prothrombin	0.001477

Step-3 : Sorting the feature importance from highest to lowest.

```
sorted_feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

display(sorted_feature_importance_df)
```

	Feature	Importance
8	Edema_Y	0.119837
3	Ascites	0.114779
1	AgeInYears	0.107980
9	Bilirubin	0.083019
6	Edema_N	0.077041
7	Edema_S	0.075662
0	Drug	0.064825
5	Spiders	0.064425
4	Hepatomegaly	0.064232
2	Sex	0.061774
13	Alk_Phos	0.049609
14	SGOT	0.043368
15	Tryglicerides	0.030237
12	Copper	0.023760
16	Platelets	0.008208
10	Cholesterol	0.007932
11	Albumin	0.001838
17	Prothrombin	0.001477

Inference:

```
[ ] best_model3 = pickle.load(open('best_model.pkl', 'rb+')) # Load model

print(X_test.iloc[1:3,]) # Get some sample

                Drug  AgeInYears Sex Ascites Hepatomegaly Spiders  Edema_N  \
419  D-penicillamine   41.786871   F       N            Y       N     True
565  D-penicillamine   56.676901   F       N            N       N     True

     Edema_S  Edema_Y  Bilirubin  Cholesterol   Albumin      Copper  \
419    False    False   0.589344   233.312591  3.479838   52.636545
565    False    False   0.893198   247.625885  3.952315  169.449217

        Alk_Phos       SGOT  Tryglicerides   Platelets  Prothrombin
419  1319.388176  72.252730     133.980960  437.256891    10.248906
565   646.476146  61.931979      84.017005  128.425131    10.094898

[ ] y_test.iloc[1:3,]

419    0
565    0
Name: CirrhosisPatients, dtype: int32

[ ] best_model3.predict(X_test.iloc[1:3,])

array([0, 0])
```

Final Result (deployment) :

We have deployed our model in AWS

About AWS for deployment:

AWS deployment strategy for a machine learning model designed to generate reports, outlining **The key steps and services involved in the process.**

1.**Model Training and Containerization**

Train the machine learning model on historical data using your preferred framework.
Containerize the model using Docker for consistency and portability.

2. **Amazon ECR (Elastic Container Registry)**

Store and manage Docker images in Amazon ECR for secure and scalable access.

3. **Amazon ECS (Elastic Container Service)**

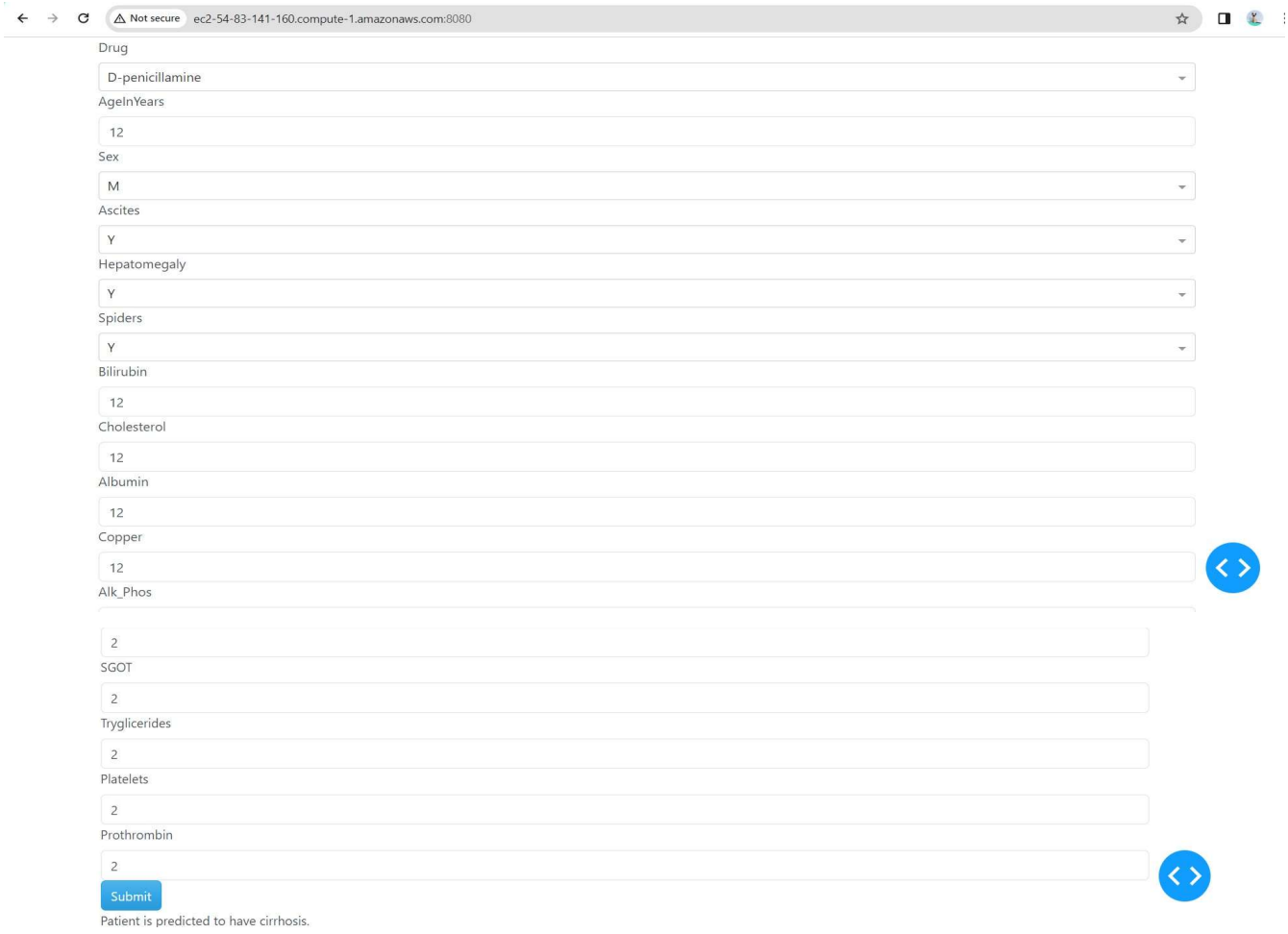
Deploy Docker containers using Amazon ECS for efficient orchestration and scaling.

4. **API Gateway**

Implement an API using Amazon API Gateway to handle requests for model inference.

5. **AWS Lambda (Optional for Serverless Deployment)**
Explore AWS Lambda for serverless deployment, reducing operational overhead.
6. **Amazon S3 (Simple Storage Service)**
Store input data and generated reports in Amazon S3 for durability and accessibility.
7. **Amazon Athena (Optional for Querying Data)**
Leverage Amazon Athena for querying data stored in S3 and generating dynamic reports.
8. **Amazon CloudWatch**
Implement monitoring and logging using Amazon CloudWatch for real-time insights.
9. **IAM Roles**
Ensure appropriate IAM roles are set to manage secure access to AWS resources.

When deployed the model predicts the disease in patients as shown in the figure below



Conclusion:

To sum up, our examination of the cirrhosis dataset has unveiled a myriad of attributes and features pivotal in determining the feasibility and effectiveness of our model. A thorough exploration of these elements has furnished valuable insights into the intricate landscape of cirrhosis prediction and diagnosis.

Our study has pinpointed key predictors and potential markers that significantly enhance our model's accuracy. These findings underscore the importance of employing advanced machine learning techniques to improve the precision of cirrhosis prediction and diagnosis.

Nevertheless, it is imperative to recognize the challenges inherent in the dataset, including issues such as missing values, imbalances, and potential confounding variables. Effectively addressing these challenges is crucial for refining the model and ensuring its robust performance in real-world scenarios.

Future Directions:

Looking ahead, our research should center on several key areas to further enhance the predictive capabilities of our model. Firstly, there should be a concerted effort to collect additional data to fill gaps and augment the dataset's representativeness. Collaboration with medical institutions and the inclusion of diverse patient populations could facilitate this data enrichment.

Furthermore, optimizing the model's performance through the exploration of advanced feature engineering techniques and experimentation with various machine learning algorithms is paramount. This involves delving into the potential of deep learning approaches, ensemble methods, and fine-tuning hyperparameters to achieve optimal predictive accuracy.

Moreover, the development of interpretability tools is crucial for gaining a deeper understanding of the model's decision-making process. In the medical domain, interpretability is essential to instill trust and encourage the adoption of machine learning models by healthcare professionals.

In conclusion, the cirrhosis dataset yields valuable insights into the intricacies of liver disease prediction. By addressing the limitations of the dataset and exploring advanced modeling techniques, our research lays the groundwork for the development of more precise and dependable cirrhosis prediction models in the future.