PYTHON FOR DATA ANALYSIS PROJECT WORK

AIM : To develop a predictive model to identify the Severity of Cirrhosis Disease in an Individual based on Clinical and Laboratory Factors Using Machine Learning Model

About Columns

1. Id - Id of the patient
2. N_Days - Number of days between registration and the earlier of death, transplantation, or study analysis time in 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. Tryglicerides - 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, or 3)

The Categorical Columns of Data are: ['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Stage']

The Numerical Columns of Data are: ['N_Days', 'Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']

In [1]:
```
#importing necessary libraries :
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
```

In [3]:
```
#Loading the csv :
try:
    df = pd.read_csv("C:\\Users\mtab\Downloads\cirrhosis.csv")
    print("Data set is loaded successfully...")

except FileNotFoundError as e:
    print(f"Error: {e}, Check the file path...")

except pd.errors.ParserError as e:
    print(f"Error with the file: {e}")
```

```
Data set is loaded successfully...
```

Overview of the dataset

In [4]:
```
#analyzing the dataset :
df.info()
#summary of dataframes struc
```

```
<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
```

In [5]:
```
df.shape
```

Out[5]:
```
(418, 20)
```

In [6]:
```
#dataset has 418 records and 20 fields
```

In [7]:
```
#reading the first and last 5 records of the cirrhosis dataset :
df.head(5)
```

Out[7]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	14.5	261.0	2.60
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	1.1	302.0	4.14
2	3	1012	D	D-penicillamine	25594	M	N	N	N	S	1.4	176.0	3.48
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	1.8	244.0	2.54
4	5	1504	CL	Placebo	13918	F	N	Y	Y	N	3.4	279.0	3.53

In [8]:
```
df.tail(5)
```

Out[8]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Co
413	414	681	D	NaN	24472	F	NaN	NaN	NaN	N	1.2	NaN	2.96	
414	415	1103	C	NaN	14245	F	NaN	NaN	NaN	N	0.9	NaN	3.83	
415	416	1055	C	NaN	20819	F	NaN	NaN	NaN	N	1.6	NaN	3.42	
416	417	691	C	NaN	21185	F	NaN	NaN	NaN	N	0.8	NaN	3.75	
417	418	976	C	NaN	19358	F	NaN	NaN	NaN	N	0.7	NaN	3.29	

In [9]:
```
#checking for any null values and handling them :

df.isnull().sum()    #null values observed in 8 columns
```

Out[9]:
```
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
```

In [10]:
```
#filling null values using median and mode
# mode --> categorical values
# median --> numerical values

df['Ascites'] = df['Ascites'].fillna(df['Ascites'].mode()[0])
df['Hepatomegaly'] = df['Hepatomegaly'].fillna(df['Hepatomegaly'].mode()[0])
df['Spiders'] = df['Spiders'].fillna(df['Spiders'].mode()[0])
df['Drug'] = df['Drug'].fillna(df['Drug'].mode()[0])
df['Stage'] = df['Stage'].fillna(df['Stage'].mode()[0])

df['Cholesterol'] = df['Cholesterol'].fillna(df['Cholesterol'].median())
df['Copper'] = df['Copper'].fillna(df['Copper'].median())
df['SGOT'] = df['SGOT'].fillna(df['SGOT'].median())
df['Tryglicerides'] = df['Tryglicerides'].fillna(df['Tryglicerides'].median())
df['Platelets'] = df['Platelets'].fillna(df['Platelets'].median())
df['Prothrombin'] = df['Prothrombin'].fillna(df['Prothrombin'].median())
df['Alk_Phos'] = df['Alk_Phos'].fillna(df['Alk_Phos'].median())
```

In [11]:
```
#rechecking for null values
df.isnull().sum()
```

Out[11]:
```
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
```

In [12]:
```
#looking for any duplicate rows and dropping if any
num_duplicates = df.duplicated().sum()
print(f'Number of duplicate rows: {num_duplicates}')
```

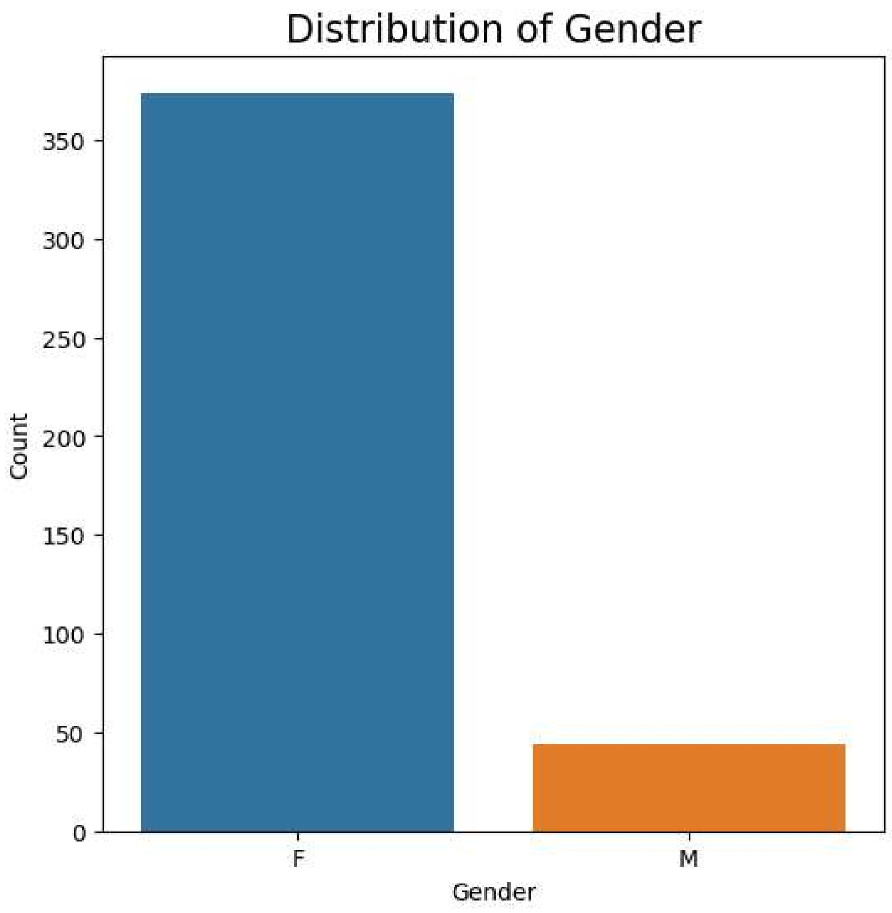
```
Number of duplicate rows: 0
```

In [13]:
```
df['Age']= df['Age']/365  #converting age(given-in days) into years
```

Visual Analysis of the Dataset

Distribution of Gender

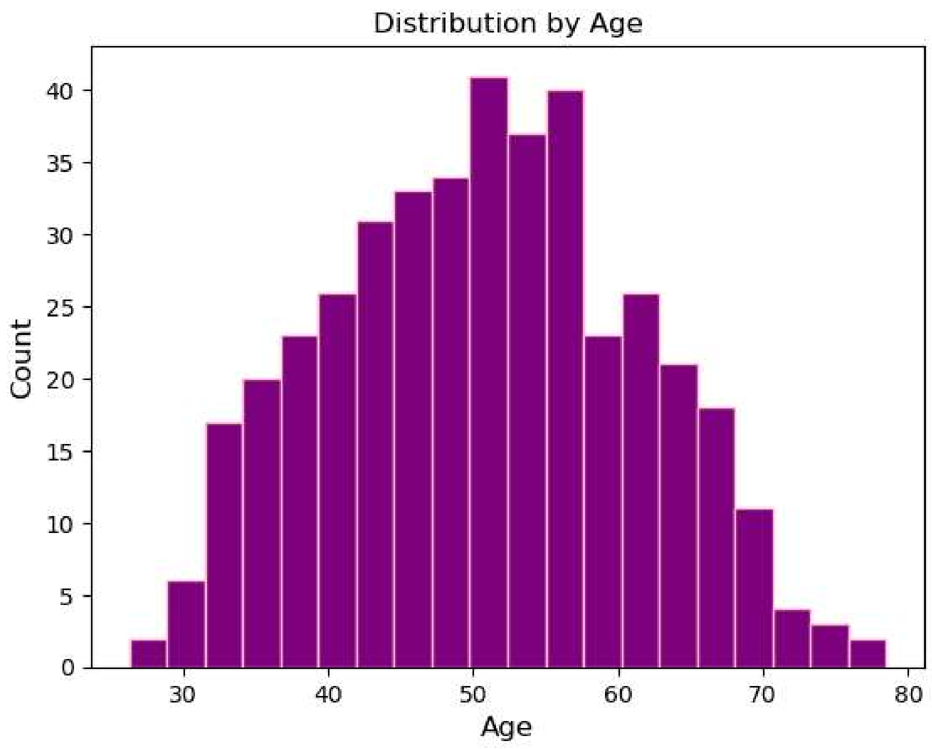
```
In [14]: plt.figure(figsize=(6, 6))
         sns.countplot(x='Sex', data=df)
         plt.title('Distribution of Gender',fontsize=16)
         plt.xlabel('Gender')
         plt.ylabel('Count')
         plt.show()
```



Interpretation : Among the total number of individuals diagnosed with cirrhosis, approximately 375-380 are female patients and 40-43 male patients. This indicates a greater occurrence of cirrhosis in females as compared to males.

Distribution By Age

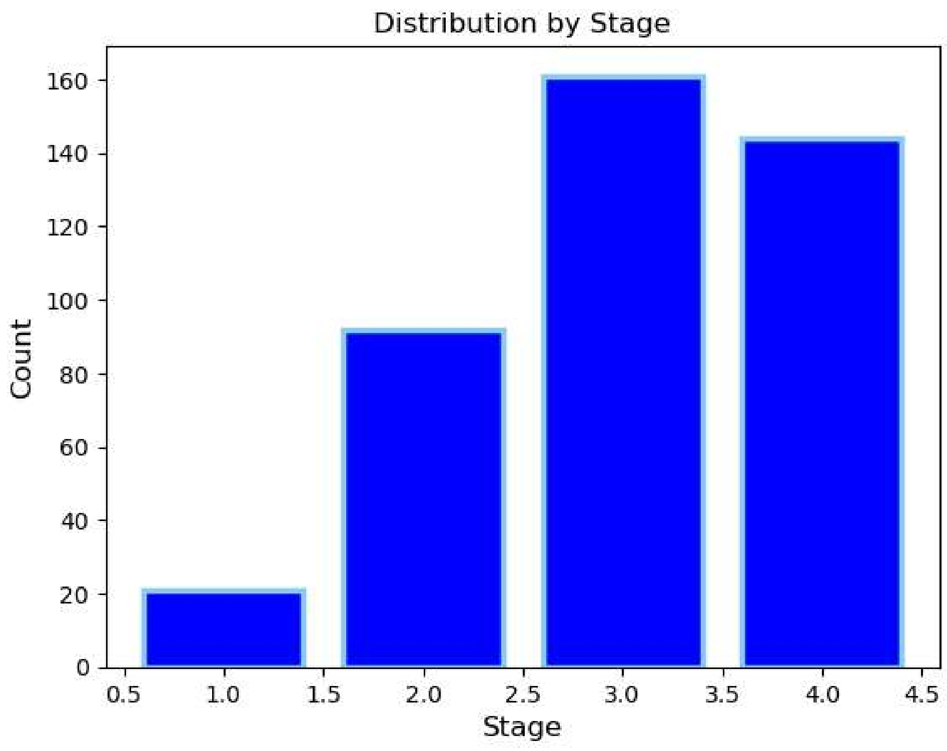
In [15]:
```
plt.hist(df['Age'], bins=20, color='purple', edgecolor='pink')


plt.title('Distribution by Age', fontsize=12)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()
```



Interpretation : From the above graph we can say that there is a gradual increase in the number of people suffering from cirrhosis as the age of an individual increases from 30-50 years of age, following which the number of people with cirrhosis starts to go down.

count of people vs the stage of cirrhosis

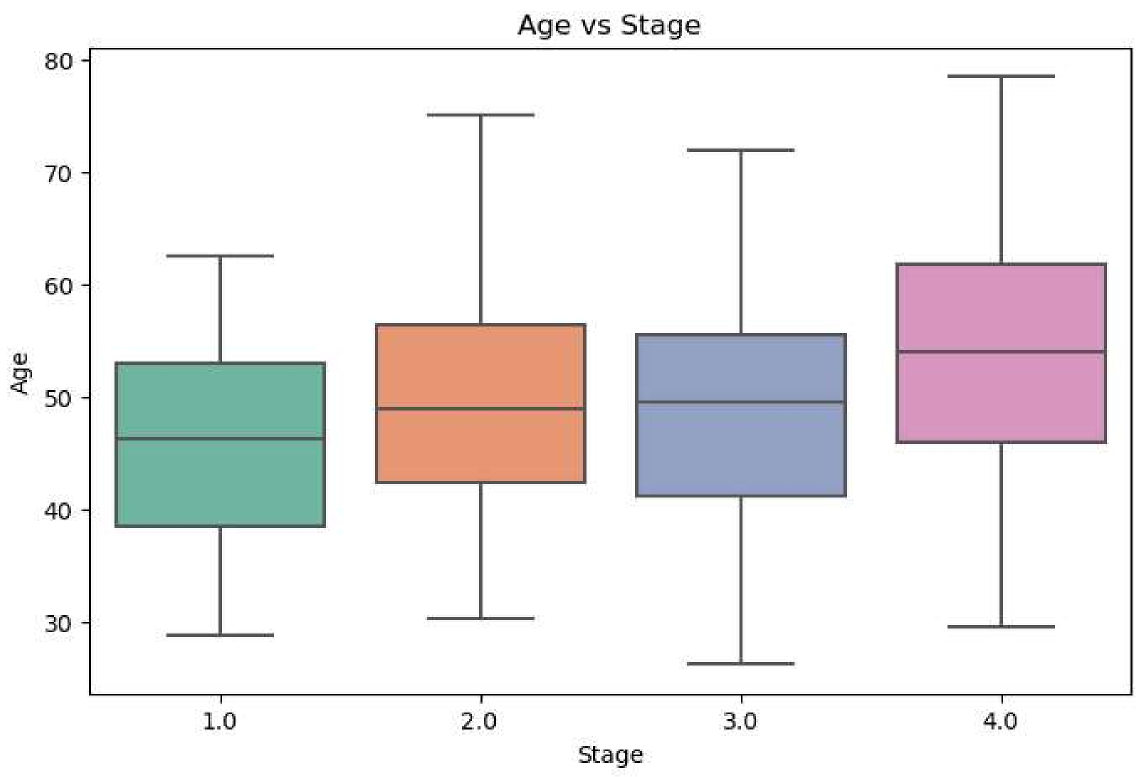
In [16]:
```
stage_counts = df['Stage'].value_counts()

plt.bar(stage_counts.index, stage_counts.values, color='blue', edgecolor='skyblue',linewidth=2.5)
plt.title('Distribution by Stage', fontsize=12)
plt.xlabel('Stage', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()
```



Interpretation : Among the total number of people, the highest number of cases is observed in stage 3 cirrhosis with a count of approximately 160 individuals, followed by around 150 people suffering from stage 4 cirrhosis. The count for stage 2 cirrhosis is relatively low, with a record of around 90 people, while only 20 people are recorded in stage 1 cirrhosis.

Relationship between Age vs Stage

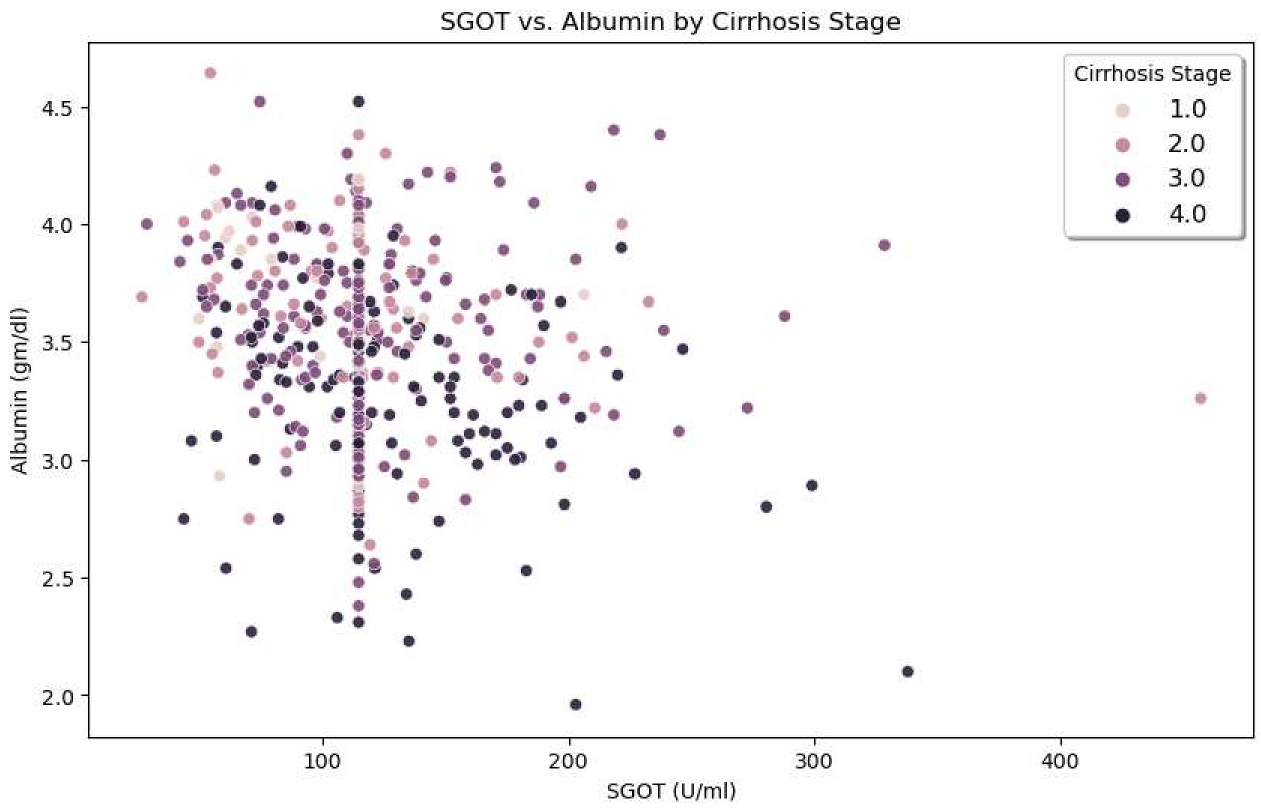
In [17]:
```
#Age vs Stage
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='Stage', y='Age', palette='Set2')
plt.title('Age vs Stage')
plt.xlabel('Stage')
plt.ylabel('Age')
plt.show()
```



```
Stage 1: The ages in Stage 1 are mostly concentrated between 30 and 50, with a few extending to
around 60.

Stage 2: The range of ages in Stage 2 is similar to Stage 1, but the median age seems slightly
higher (around 48).

Stage 3: In Stage 3, the median age is slightly higher than in the previous stages. The range of
ages is also wider, with a few individuals between the age of 60-70 years.

Stage 4: Stage 4 shows the highest median age among all stages (around 53-54 years). The range is
also quite wide, suggesting that individuals at this stage can be quite diverse in terms of age.
```

SGOT VS ALMBUMIN by Cirrhosis Stage

In [18]:
```python
#SGOT vs Albumin
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='SGOT', y='Albumin', hue='Stage', alpha=0.9)
plt.title('SGOT vs. Albumin by Cirrhosis Stage')
plt.xlabel('SGOT (U/ml)')
plt.ylabel('Albumin (gm/dl)')
plt.legend(title='Cirrhosis Stage',loc="best",shadow=True,fontsize="large")
plt.show()
```



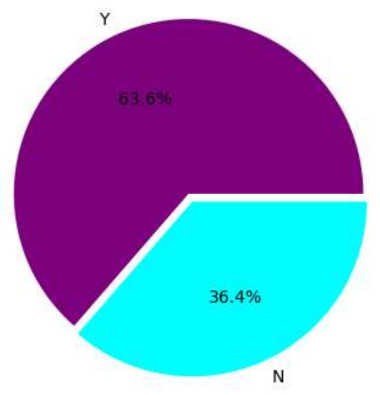
There is a negative correlation between SGOT and Albumin levels, As SGOT levels increase, indicating greater liver damage, albumin levels tend to decrease.

The scatter plot shows that individuals with more advanced cirrhosis stages (3.0 and 4.0) generally have lower albumin levels compared to those in earlier stages (1.0 and 2.0).

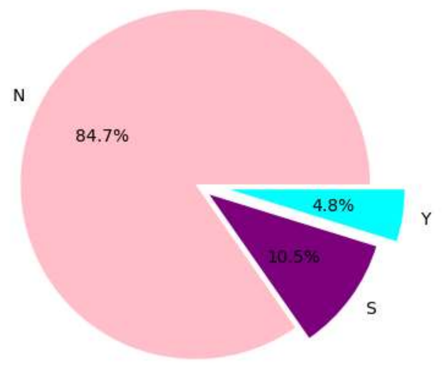
pie chart plot

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt


fig, axs = plt.subplots(2, 2, figsize=(16, 8))

# Pie chart for Hepatomegaly
hepatomegaly_counts = df['Hepatomegaly'].value_counts()
axs[0, 0].pie(hepatomegaly_counts, labels=hepatomegaly_counts.index,
               autopct='%1.1f%%', colors=['purple', 'cyan'], explode=[0, 0.05])
axs[0, 0].set_title('Proportion of Patients with Hepatomegaly')

# Pie chart for Edema
edema_counts = df['Edema'].value_counts()
axs[0, 1].pie(edema_counts, labels=edema_counts.index,
               autopct='%1.1f%%', colors=['pink', 'purple', 'cyan'], explode=[0, 0.1, 0.2])
axs[0, 1].set_title('Proportion of Patients with Edema')

# Pie chart for Drug
drug_counts = df['Drug'].value_counts()
axs[1, 0].pie(drug_counts, labels=drug_counts.index,
               autopct='%1.1f%%', colors=['pink', 'cyan'], explode=[0, 0.05])
axs[1, 0].set_title('Drug Distribution')

# Pie chart for Ascites
ascites_counts = df['Ascites'].value_counts()
axs[1, 1].pie(ascites_counts, labels=ascites_counts.index,
               autopct='%1.1f%%', colors=['purple', 'cyan'], explode=[0, 0.05])
axs[1, 1].set_title('Ascites Distribution')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()


#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
```

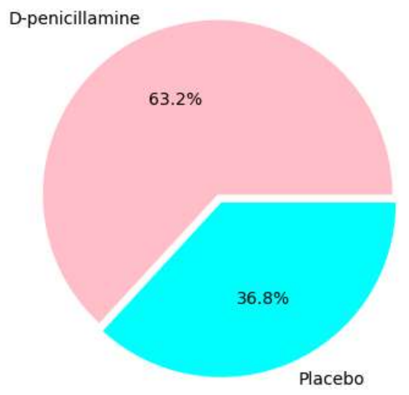
Proportion of Patients with Hepatomegaly



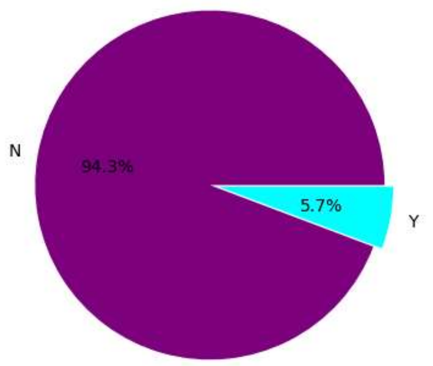
Proportion of Patients with Edema



Drug Distribution



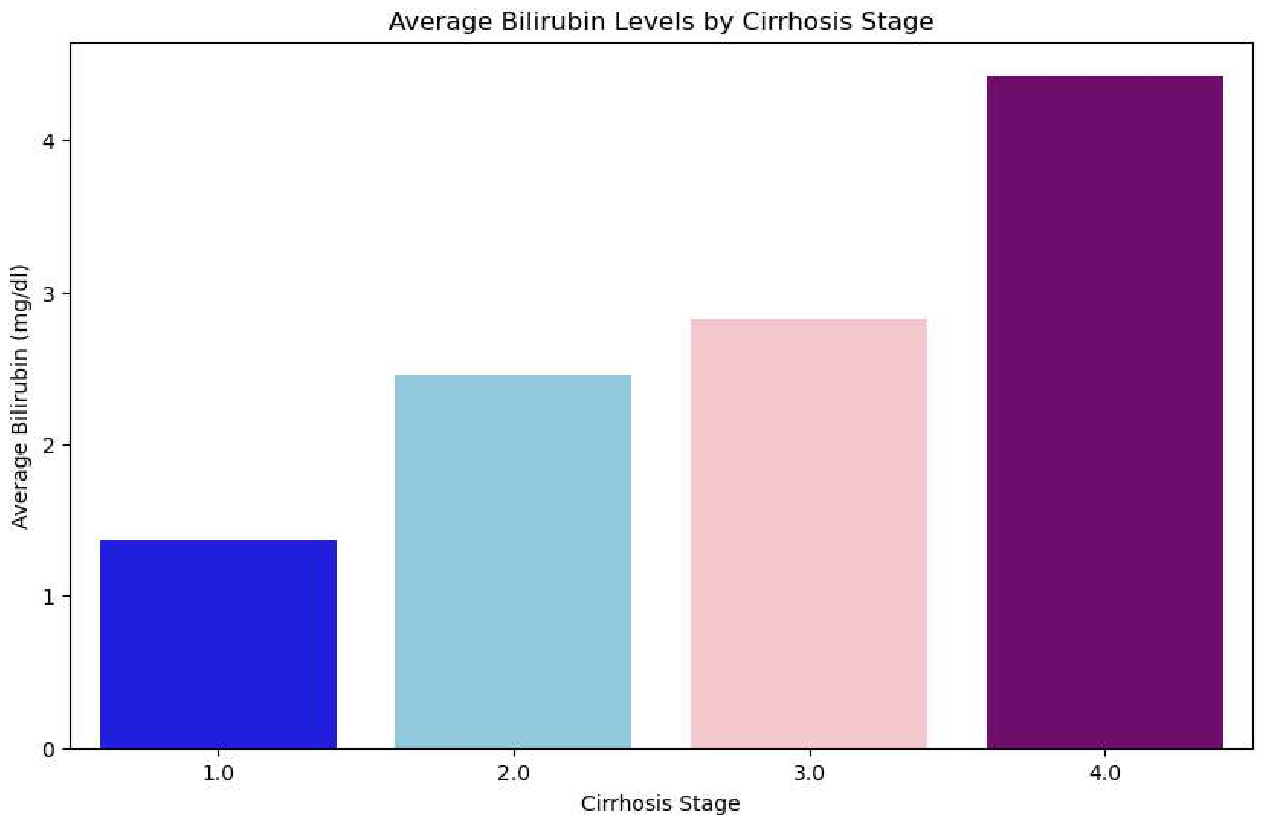
Ascites Distribution



Plot for Average Bilirubin Levels by Cirrhosis Stage

In [20]:
```
clrs = ['blue','skyblue','pink','purple']

plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='Stage', y='Bilirubin', ci=None, palette=clrs)
plt.title('Average Bilirubin Levels by Cirrhosis Stage')
plt.xlabel('Cirrhosis Stage')
plt.ylabel('Average Bilirubin (mg/dl)')
plt.show()
```

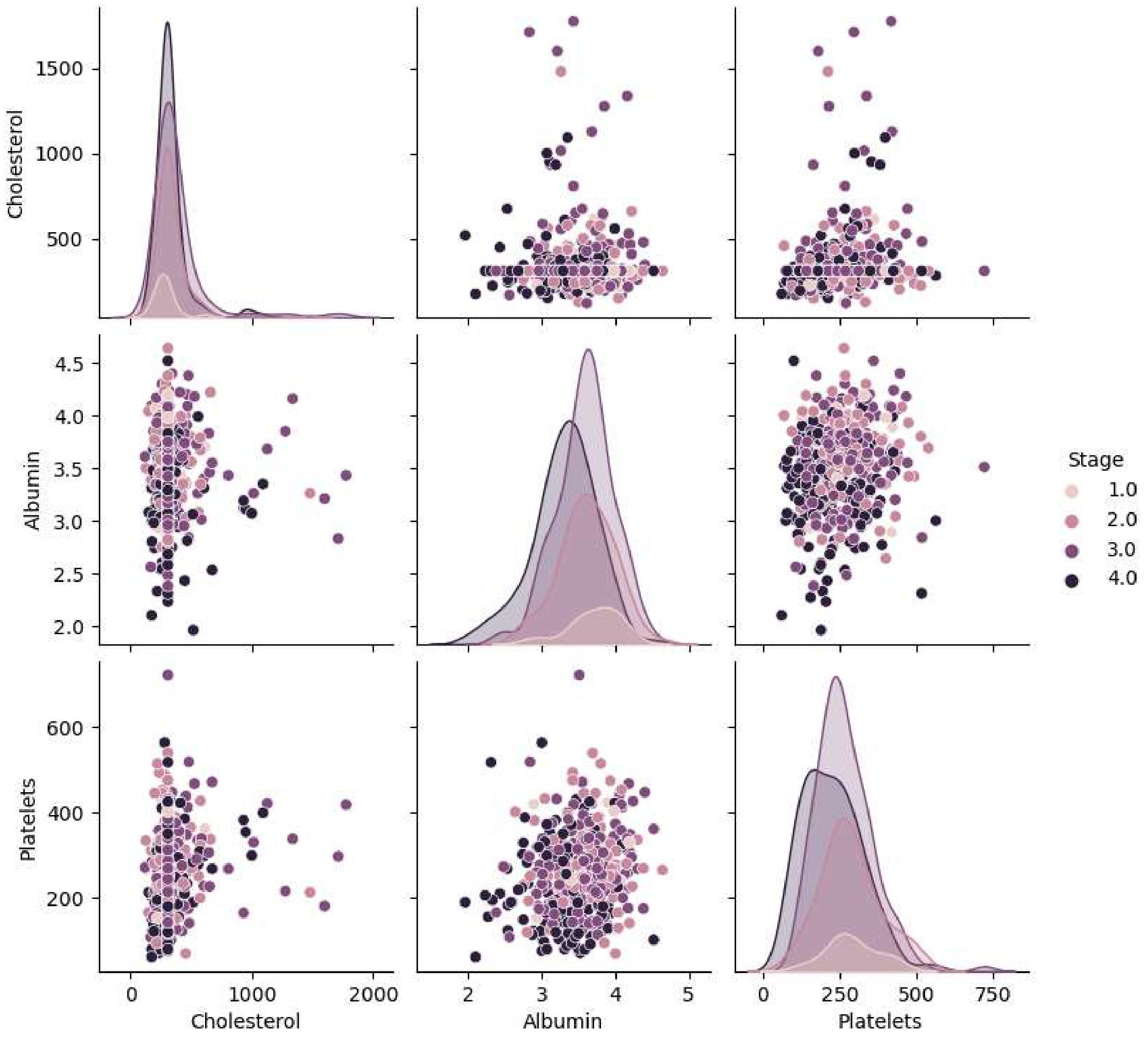
```
C:\Users\mtab\AppData\Local\Temp\ipykernel_11292\3671046248.py:4: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Stage', y='Bilirubin', ci=None, palette=clrs)
```



Bilirubin levels are relatively low in this early stage of cirrhosis. While the levels reach their highest point in Stage 4, indicating significant liver damage.

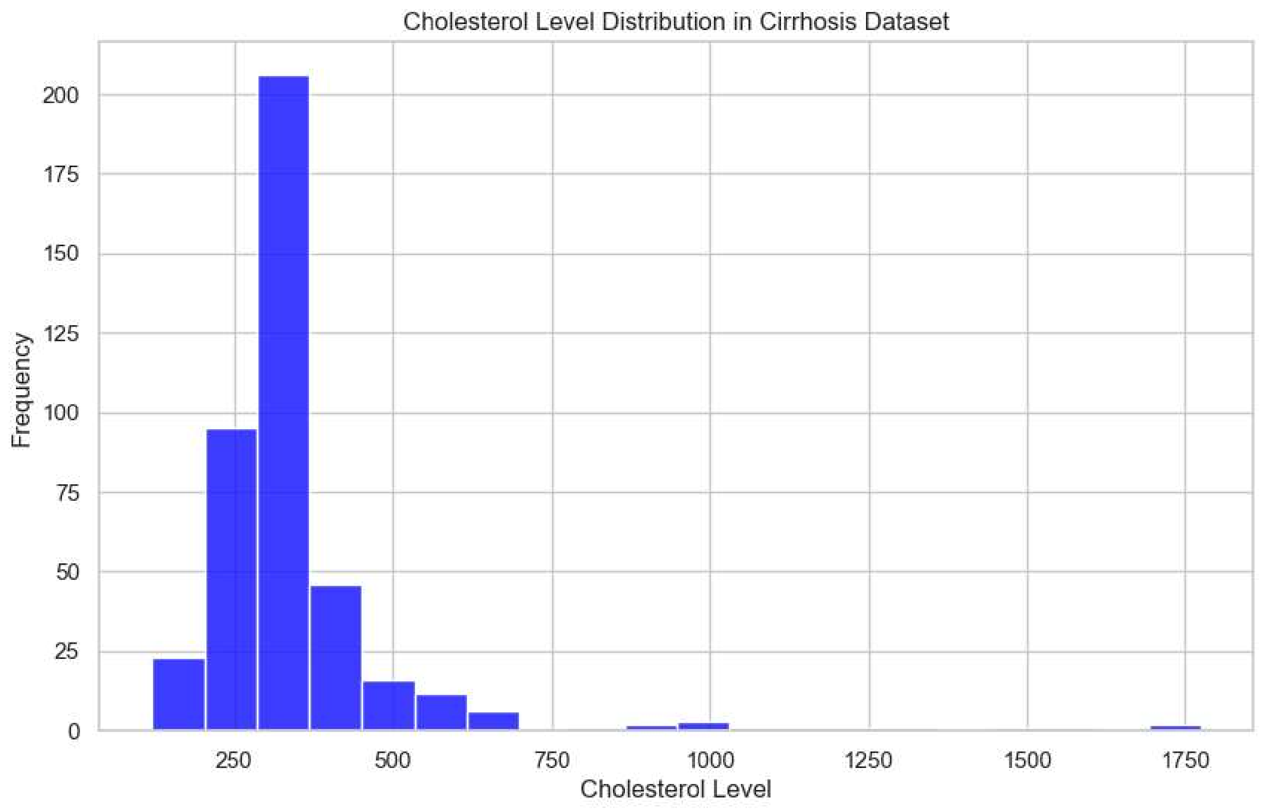
Pair plot

In [21]:
```
# Selecting relevant numerical columns along with the Stage column
s_Cols = ['Cholesterol', 'Albumin','Platelets', 'Stage']
subset_data = df[s_Cols]

sns.pairplot(subset_data, hue='Stage')
plt.show()
```



In [22]:
```
#This graph shows how Cholesterol, Albumin, and Platelet levels change
#as the disease (cirrhosis) progresses. We can see that there are some weak relationships between th
#but they don't tell us much about the severity of the disease.
```

In [23]:
```
sns.set(style="whitegrid")

# Plotting the distribution of cholesterol levels
plt.figure(figsize=(10, 6))
sns.histplot(df['Cholesterol'], bins=20,color='blue')
plt.title('Cholesterol Level Distribution in Cirrhosis Dataset')
plt.xlabel('Cholesterol Level')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [24]:
```
# Defining the cholesterol range # as from the above plot we can say max. people suffering
#from cirrhosis have cholestrol level btw 250-500
lower_th = 250
upper_th = 500

# Creating a boolean mask for cholesterol levels
m= (df['Cholesterol'] >= lower_th) & (df['Cholesterol'] <= upper_th)

# Filter the DataFrame based on the mask
filtered_df = df[m]
num_rows = filtered_df.shape[0]
print(f"\nNumber of patients with Cholesterol between {lower_th} and {upper_th}: {num_rows}")
```

```
Number of patients with Cholesterol between 250 and 500: 311
```

In [25]:
```
#viewing the first 5 row of the above
print("\n\nFiltered Rows (Cholesterol between 250 and 500):")
print(filtered_df.iloc[:5])
```

```


Filtered Rows (Cholesterol between 250 and 500):
   ID  N_Days Status             Drug        Age Sex Ascites Hepatomegaly  \
0   1     400      D  D-penicillamine  58.805479   F       Y            Y
1   2    4500      C  D-penicillamine  56.484932   F       N            Y
4   5    1504     CL          Placebo  38.131507   F       N            Y
6   7    1832      C          Placebo  55.572603   F       N            Y
7   8    2466      D          Placebo  53.093151   F       N            N

  Spiders Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  \
0       Y     Y       14.5        261.0     2.60   156.0    1718.0  137.95
1       Y     N        1.1        302.0     4.14    54.0    7394.8  113.52
4       Y     N        3.4        279.0     3.53   143.0     671.0  113.15
6       N     N        1.0        322.0     4.09    52.0     824.0   60.45
7       N     N        0.3        280.0     4.00    52.0    4651.2   28.38

   Tryglicerides  Platelets  Prothrombin  Stage
0          172.0      190.0         12.2    4.0
1           88.0      221.0         10.6    3.0
4           72.0      136.0         10.9    3.0
6          213.0      204.0          9.7    3.0
7          189.0      373.0         11.0    3.0
```

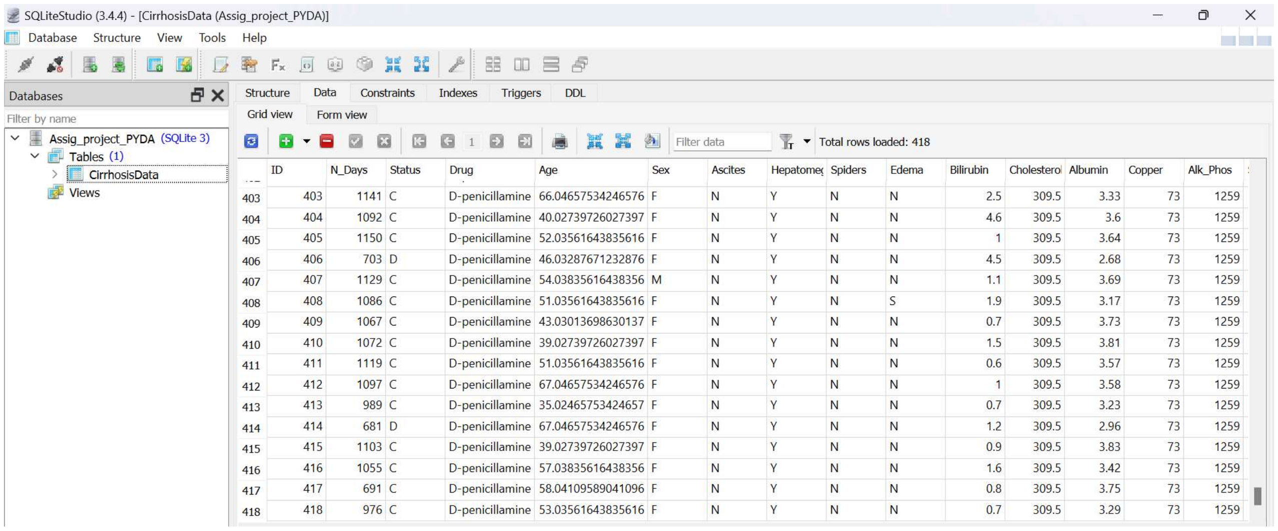
In [26]:
```
# Counting occurrences of each stage with filetred cholestrol levels :
stage_counts = filtered_df['Stage'].value_counts()

print("\nStage Counts:")
print(stage_counts)
```

```

Stage Counts:
3.0    121
4.0    106
2.0     73
1.0     11
Name: Stage, dtype: int64
```

Pushing data into database

In [ ]:
```
import sqlite3

conn = sqlite3.connect('Project_PYDA.db')

df.to_sql('CirrhosisData', conn, if_exists='replace', index=False)


conn.close()

print("DataFrame pushed to SQLite database successfully!")
```

```
DataFrame pushed to SQLite database successfully!
```

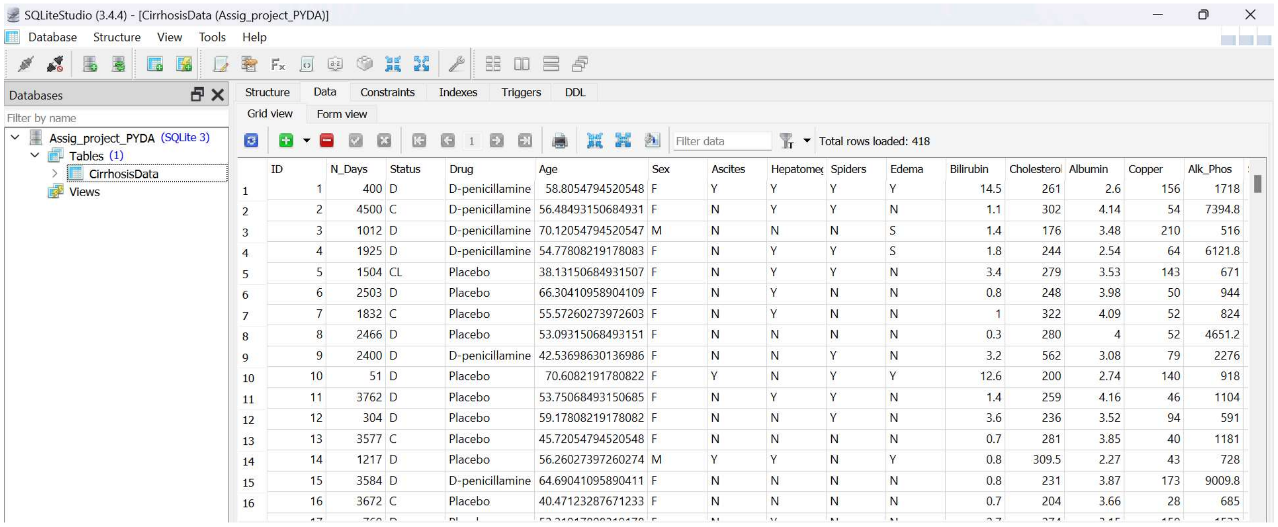


SQLiteStudio (3.4.4) - [CirrhosisData (Assig_project_PYDA)]

Databases: Assig_project_PYDA (SQLite 3) › Tables (1) › CirrhosisData; Views

Total rows loaded: 418

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomeg	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos
403	403	1141	C	D-penicillamine	66.04657534246576	F	N	Y	N	N	2.5	309.5	3.33	73	1259
404	404	1092	C	D-penicillamine	40.02739726027397	F	N	Y	N	N	4.6	309.5	3.6	73	1259
405	405	1150	C	D-penicillamine	52.03561643835616	F	N	Y	N	N	1	309.5	3.64	73	1259
406	406	703	D	D-penicillamine	46.03287671232876	F	N	Y	N	N	4.5	309.5	2.68	73	1259
407	407	1129	C	D-penicillamine	54.03835616438356	M	N	Y	N	N	1.1	309.5	3.69	73	1259
408	408	1086	C	D-penicillamine	51.03561643835616	F	N	Y	N	S	1.9	309.5	3.17	73	1259
409	409	1067	C	D-penicillamine	43.03013698630137	F	N	Y	N	N	0.7	309.5	3.73	73	1259
410	410	1072	C	D-penicillamine	39.02739726027397	F	N	Y	N	N	1.5	309.5	3.81	73	1259
411	411	1119	C	D-penicillamine	51.03561643835616	F	N	Y	N	N	0.6	309.5	3.57	73	1259
412	412	1097	C	D-penicillamine	67.04657534246576	F	N	Y	N	N	1	309.5	3.58	73	1259
413	413	989	C	D-penicillamine	35.02465753424657	F	N	Y	N	N	0.7	309.5	3.23	73	1259
414	414	681	D	D-penicillamine	67.04657534246576	F	N	Y	N	N	1.2	309.5	2.96	73	1259
415	415	1103	C	D-penicillamine	39.02739726027397	F	N	Y	N	N	0.9	309.5	3.83	73	1259
416	416	1055	C	D-penicillamine	57.03835616438356	F	N	Y	N	N	1.6	309.5	3.42	73	1259
417	417	691	C	D-penicillamine	58.04109589041096	F	N	Y	N	N	0.8	309.5	3.75	73	1259
418	418	976	C	D-penicillamine	53.03561643835616	F	N	Y	N	N	0.7	309.5	3.29	73	1259



SQLiteStudio (3.4.4) - [CirrhosisData (Assig_project_PYDA)]

Databases: Assig_project_PYDA (SQLite 3) › Tables (1) › CirrhosisData; Views

Total rows loaded: 418

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomeg	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos
1	1	400	D	D-penicillamine	58.8054794520548	F	Y	Y	Y	Y	14.5	261	2.6	156	1718
2	2	4500	C	D-penicillamine	56.48493150684931	F	N	Y	Y	N	1.1	302	4.14	54	7394.8
3	3	1012	D	D-penicillamine	70.12054794520547	M	N	N	N	S	1.4	176	3.48	210	516
4	4	1925	D	D-penicillamine	54.77808219178083	F	N	Y	Y	S	1.8	244	2.54	64	6121.8
5	5	1504	CL	Placebo	38.13150684931507	F	N	Y	Y	N	3.4	279	3.53	143	671
6	6	2503	D	Placebo	66.30410958904109	F	N	Y	N	N	0.8	248	3.98	50	944
7	7	1832	C	Placebo	55.57260273972603	F	N	Y	N	N	1	322	4.09	52	824
8	8	2466	D	Placebo	53.09315068493151	F	N	N	N	N	0.3	280	4	52	4651.2
9	9	2400	D	D-penicillamine	42.53698630136986	F	N	N	Y	N	3.2	562	3.08	79	2276
10	10	51	D	Placebo	70.6082191780822	F	Y	N	Y	Y	12.6	200	2.74	140	918
11	11	3762	D	Placebo	53.75068493150685	F	N	Y	Y	N	1.4	259	4.16	46	1104
12	12	304	D	Placebo	59.17808219178082	F	N	N	Y	N	3.6	236	3.52	94	591
13	13	3577	C	Placebo	45.72054794520548	F	N	N	N	N	0.7	281	3.85	40	1181
14	14	1217	D	Placebo	56.26027397260274	M	Y	Y	N	Y	0.8	309.5	2.27	43	728
15	15	3584	D	D-penicillamine	64.69041095890411	F	N	N	N	N	0.8	231	3.87	173	9009.8
16	16	3672	C	Placebo	40.47123287671233	F	N	N	N	N	0.7	204	3.66	28	685

Model Building

In [27]:
```python
# Label encoding for categorical features
df['Status'] = df['Status'].map({'C': 0, 'CL': 1, 'D': 2})
df['Drug'] = df['Drug'].map({'D-penicillamine': 0, 'Placebo': 1})
df['Sex'] = df['Sex'].map({'F': 0, 'M': 1})
df['Ascites'] = df['Ascites'].map({'N': 0, 'Y': 1})
df['Hepatomegaly'] = df['Hepatomegaly'].map({'N': 0, 'Y': 1})
df['Spiders'] = df['Spiders'].map({'N': 0, 'Y': 1})
df['Edema'] = df['Edema'].map({'N': 0, 'S': 1, 'Y': 2})
```

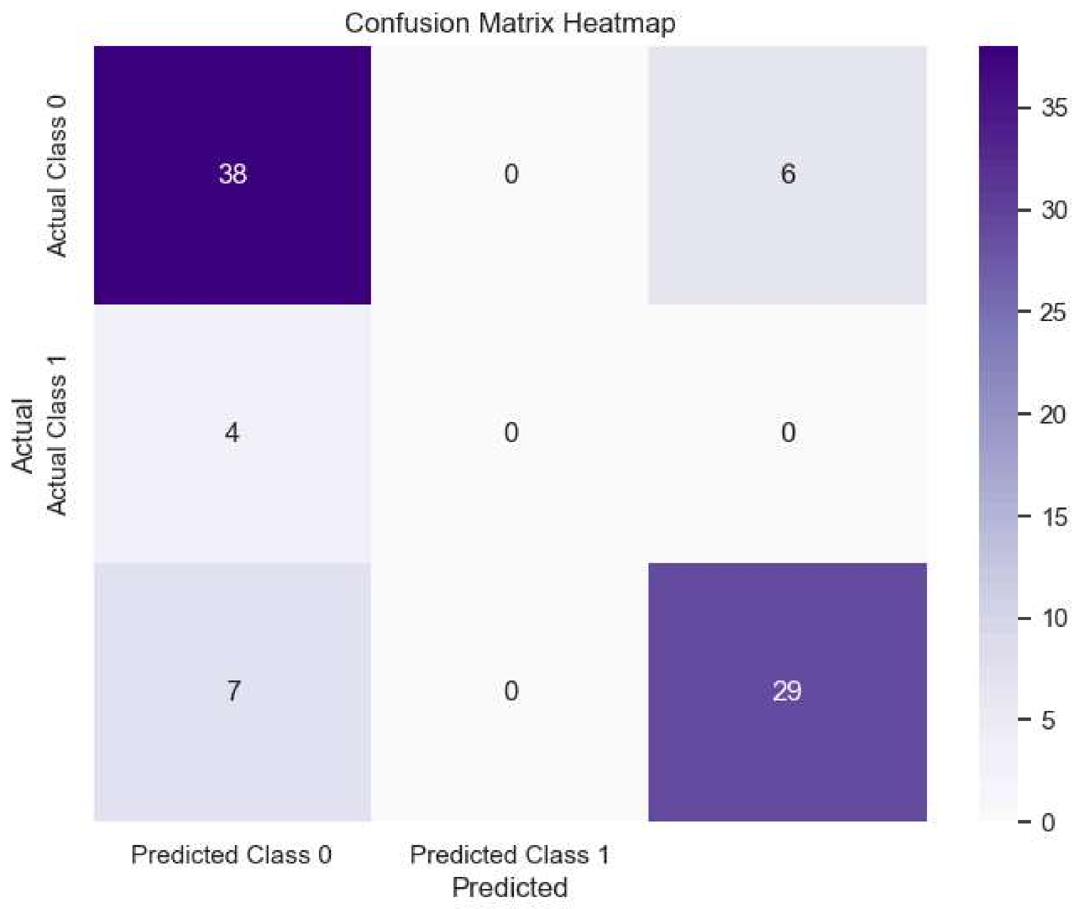
In [32]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix


X = df.drop(['ID', 'Status', 'Drug', 'N_Days'], axis=1)  # Using all features except these
y = df['Status']

# Scaling the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# fits the scaler to X and transforms it, resulting in X_scaled, where each feature will
#have a mean of 0 and a standard deviation of 1.python


# Splitting  the data into training and testing sets - (80:20)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Logistic Regression :

In [33]:
```
model = LogisticRegression(max_iter=5000, random_state=42)  #A logistic regression model is created

# Training  the model
model.fit(X_train, y_train)

# Making  predictions on the test set
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy (train-test split): {accuracy:.2f}')


print("Classification Report (train-test split):")
print(classification_report(y_test, y_pred))

cm1 = confusion_matrix(y_test, y_pred)
print("Confusion_Matrix : \n",cm1)
```

```
Accuracy (train-test split): 0.80
Classification Report (train-test split):
              precision    recall  f1-score   support

           0       0.78      0.86      0.82        44
           1       0.00      0.00      0.00         4
           2       0.83      0.81      0.82        36

    accuracy                           0.80        84
   macro avg       0.53      0.56      0.54        84
weighted avg       0.76      0.80      0.78        84

Confusion_Matrix :
 [[38  0  6]
 [ 4  0  0]
 [ 7  0 29]]

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted s
amples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted s
amples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted s
amples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
In [34]: # Creating  a heatmap for the confusion matrix for models
         plt.figure(figsize=(8, 6))
         sns.heatmap(cm1, annot=True, fmt='d', cmap='Purples',
                     xticklabels=['Predicted Class 0', 'Predicted Class 1'],
                     yticklabels=['Actual Class 0', 'Actual Class 1'])
         plt.title('Confusion Matrix Heatmap')
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.show()
```

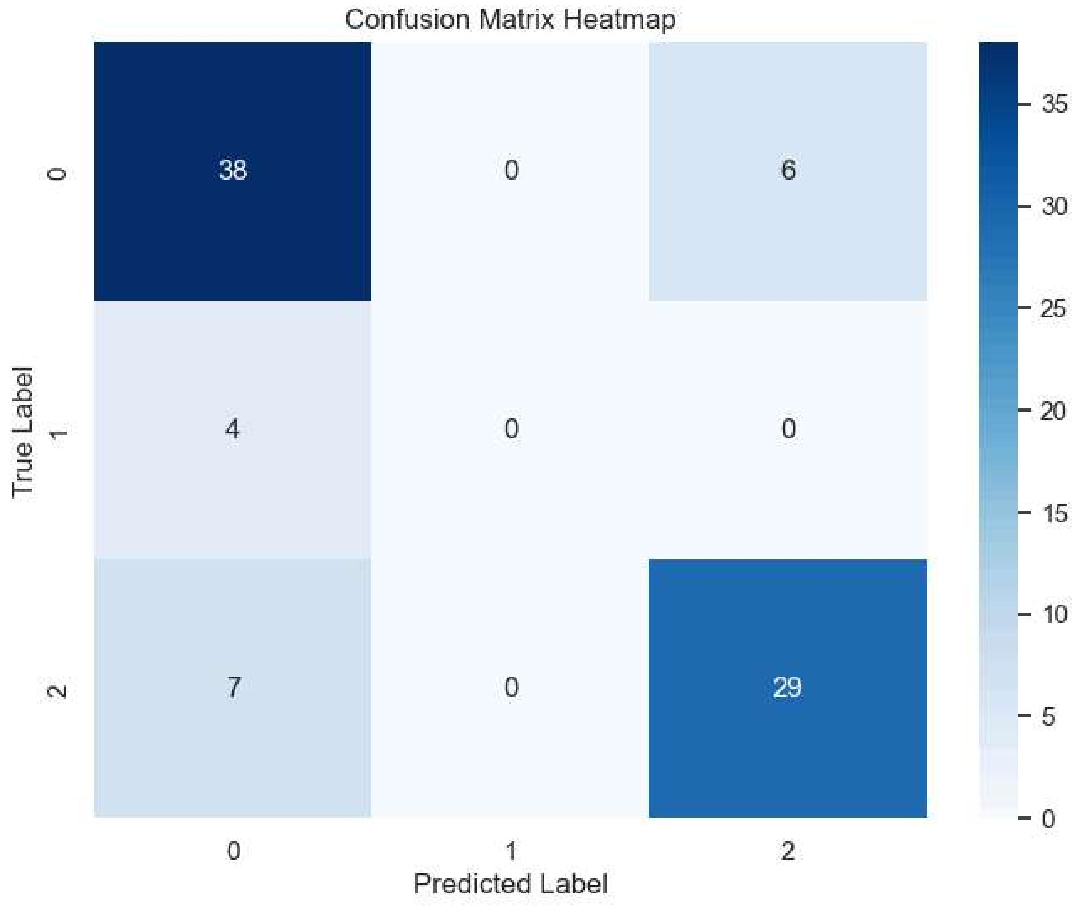


Random Forest :

In [35]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

model2 = RandomForestClassifier()
model2.fit(X_train, y_train)

#accuracy calculation :
y_train_pred = model2.predict(X_train)
training_accuracy = accuracy_score(y_train, y_train_pred)

print("Training Accuracy:", training_accuracy)


#Calculating test accuracy
y_test_pred = model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
print("Test Accuracy:", test_accuracy)


cm_test = confusion_matrix(y_test, y_test_pred)
print("Confusion Matrix (test set):\n", cm_test)
```

```
Training Accuracy: 1.0
Test Accuracy: 0.7976190476190477
Confusion Matrix (test set):
 [[38  0  6]
 [ 4  0  0]
 [ 7  0 29]]
```

In [36]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues', xticklabels=model2.classes_, yticklabels=mo
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



Conclusion :

After evaluating the performance of two machine learning models - Logistic Regression, and Random Forest on the Cirrhosis dataset, Logistic Regression emerged as the most suitable model with an accuracy of aboutr 80%.

In [ ]: