Liver fibrosis staging from elastography images

P.YASASWINI

BU22CSEN0300380

CSV FILE :

ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders
1	400	D	D-penicillamine	21464	F	Y	Y	Y
2	4500	C	D-penicillamine	20617	F	N	Y	Y
3	1012	D	D-penicillamine	25594	M	N	N	N
4	1925	D	D-penicillamine	19994	F	N	Y	Y
5	1504	CL	Placebo	13918	F	N	Y	Y
6	2503	D	Placebo	24201	F	N	Y	N
7	1832	C	Placebo	20284	F	N	Y	N
8	2466	D	Placebo	19379	F	N	N	N
9	2400	D	D-penicillamine	15526	F	N	N	Y
10	51	D	Placebo	25772	F	Y	N	Y
11	3762	D	Placebo	19619	F	N	Y	Y
12	304	D	Placebo	21600	F	N	N	Y
13	3577	C	Placebo	16688	F	N	N	N
14	1217	D	Placebo	20535	M	Y	Y	N
15	3584	D	D-penicillamine	23612	F	N	N	N
16	3672	C	Placebo	14772	F	N	N	N
17	769	D	Placebo	19060	F	N	Y	N
18	131	D	D-penicillamine	19698	F	N	Y	Y
19	4232	C	D-penicillamine	18102	F	N	Y	N
20	1356	D	Placebo	21898	F	N	Y	N
21	3445	C	Placebo	23445	M	N	Y	Y
22	673	D	D-penicillamine	20555	F	N	N	Y
23	264	D	Placebo	20442	F	Y	Y	Y
24	4079	D	D-penicillamine	16261	M	N	Y	N
25	4127	C	Placebo	16463	F	N	N	N
26	1444	D	Placebo	19002	F	N	Y	Y

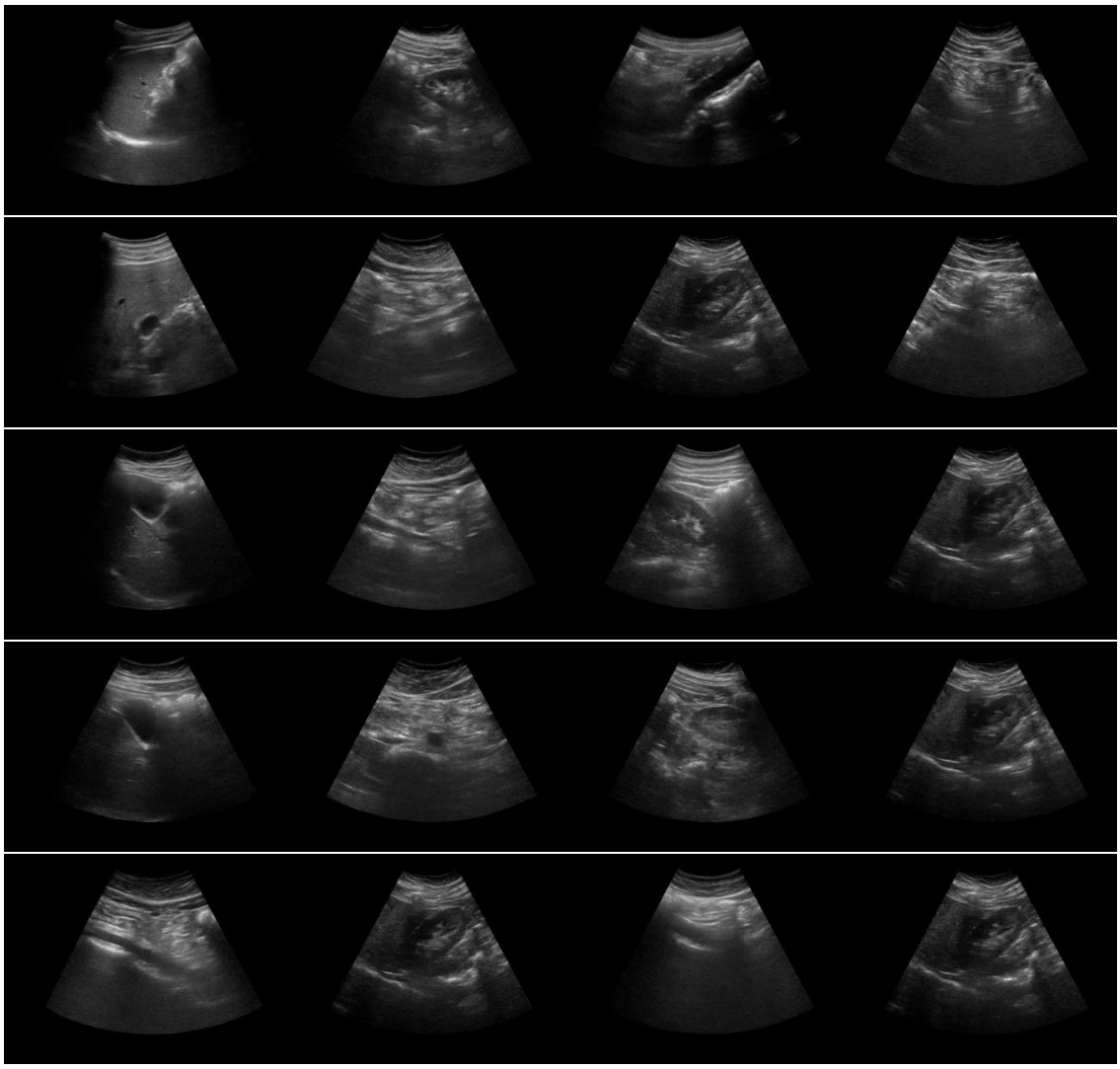
27	77	D	Placebo	19884	F	Y	Y	Y
28	549	D	Placebo	16417	F	Y	Y	Y
29	4509	C	Placebo	23331	F	N	N	N
30	321	D	Placebo	15116	F	N	Y	Y
31	3839	D	Placebo	15177	F	N	Y	N
32	4523	C	Placebo	19722	F	N	Y	N
33	3170	D	Placebo	18731	F	N	N	N
34	3933	C	D-penicillamine	19015	F	N	N	N
35	2847	D	Placebo	17758	F	N	N	N
36	3611	C	Placebo	20604	F	N	N	N
37	223	D	D-penicillamine	22546	F	Y	Y	N
38	3244	D	Placebo	13378	F	N	Y	Y
39	2297	D	D-penicillamine	20232	F	N	Y	N
40	4467	C	D-penicillamine	17046	F	N	N	N
41	1350	D	D-penicillamine	12285	F	N	Y	N
42	4453	C	Placebo	12307	F	N	Y	Y
43	4556	C	D-penicillamine	17850	F	N	N	N
44	3428	D	Placebo	13727	F	N	Y	Y
45	4025	C	Placebo	15265	F	N	N	N
46	2256	D	D-penicillamine	16728	F	N	Y	N
47	2576	C	Placebo	17323	F	N	N	N
48	4427	C	Placebo	17947	M	N	N	N
49	708	D	Placebo	22336	F	N	Y	N
50	2598	D	D-penicillamine	19544	F	N	Y	N
51	3853	D	Placebo	19025	F	N	N	N
52	2386	D	D-penicillamine	18460	M	N	N	N
53	1000	D	D-penicillamine	24621	F	N	Y	N
54	1434	D	D-penicillamine	14317	F	Y	Y	Y
55	1360	D	D-penicillamine	24020	M	N	N	N
56	1847	D	Placebo	12279	F	N	Y	Y
57	3282	D	D-penicillamine	19567	F	N	Y	N
58	4459	C	D-penicillamine	16279	M	N	N	N
59	2224	D	D-penicillamine	14754	F	N	Y	Y
60	4365	C	D-penicillamine	21324	F	N	N	N

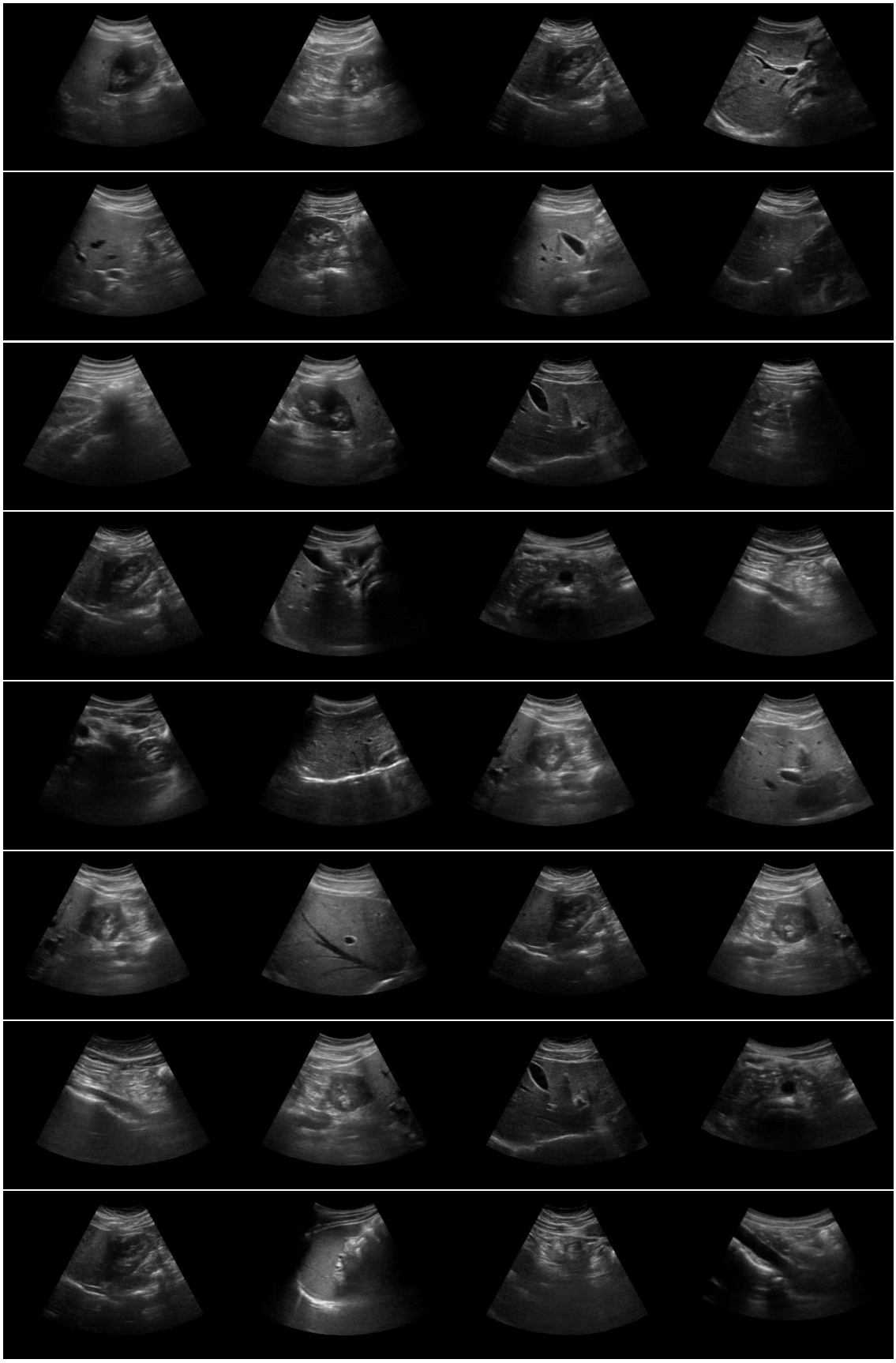
61	4256	C	Placebo	16034	M	N	N	N
62	3090	D	Placebo	22173	F	Y	N	N
63	859	D	Placebo	17031	F	N	N	Y
64	1487	D	Placebo	22977	F	N	Y	N
65	3992	C	D-penicillamine	14684	F	N	N	N
66	4191	D	D-penicillamine	16967	M	N	Y	N
67	2769	D	Placebo	18733	F	N	N	N
68	4039	C	D-penicillamine	11912	F	N	N	N
69	1170	D	D-penicillamine	18021	F	N	Y	Y
70	3458	C	D-penicillamine	20600	F	N	N	N
71	4196	C	Placebo	17841	F	N	Y	N
72	4184	C	Placebo	11868	F	N	N	N
73	4190	C	Placebo	14060	F	N	N	N
74	1827	D	D-penicillamine	18964	F	N	Y	Y
75	1191	D	D-penicillamine	15895	F	Y	Y	Y
76	71	D	D-penicillamine	18972	F	N	Y	Y
77	326	D	Placebo	18199	F	N	Y	Y
78	1690	D	D-penicillamine	17512	F	N	Y	N
79	3707	C	D-penicillamine	16990	F	N	Y	N
80	890	D	Placebo	24622	M	N	Y	N
81	2540	D	D-penicillamine	23107	F	N	Y	Y
82	3574	D	D-penicillamine	24585	F	N	N	N
83	4050	C	D-penicillamine	20459	F	N	Y	N
84	4032	C	Placebo	20392	F	N	N	N
85	3358	D	Placebo	17246	F	N	Y	N
86	1657	D	D-penicillamine	19270	F	N	Y	Y
87	198	D	D-penicillamine	13616	F	N	N	N
88	2452	C	Placebo	15119	F	N	N	N
89	1741	D	D-penicillamine	19155	F	N	Y	N
90	2689	D	D-penicillamine	12227	M	N	N	N
91	460	D	Placebo	16658	F	N	Y	Y
92	388	D	D-penicillamine	28018	F	Y	N	N

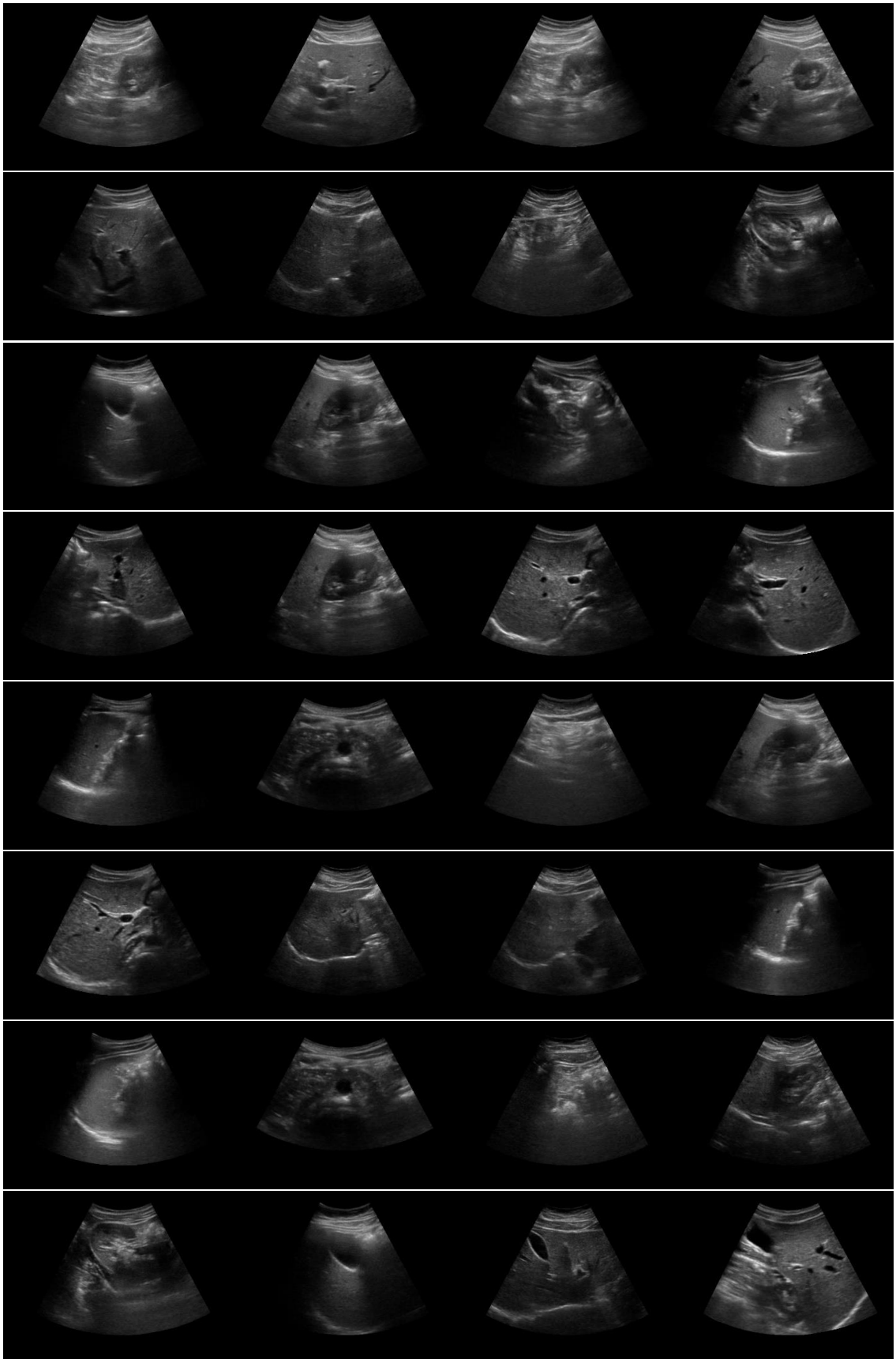
93	3913	C	D-penicillamine	13344	F	N	N	N
94	750	D	D-penicillamine	19693	F	N	Y	Y
95	130	D	Placebo	16944	F	Y	Y	Y
96	3850	C	D-penicillamine	17841	F	N	N	N
97	611	D	Placebo	26259	M	N	Y	N
98	3823	C	D-penicillamine	10550	F	N	N	N
99	3820	C	Placebo	17703	M	N	N	N

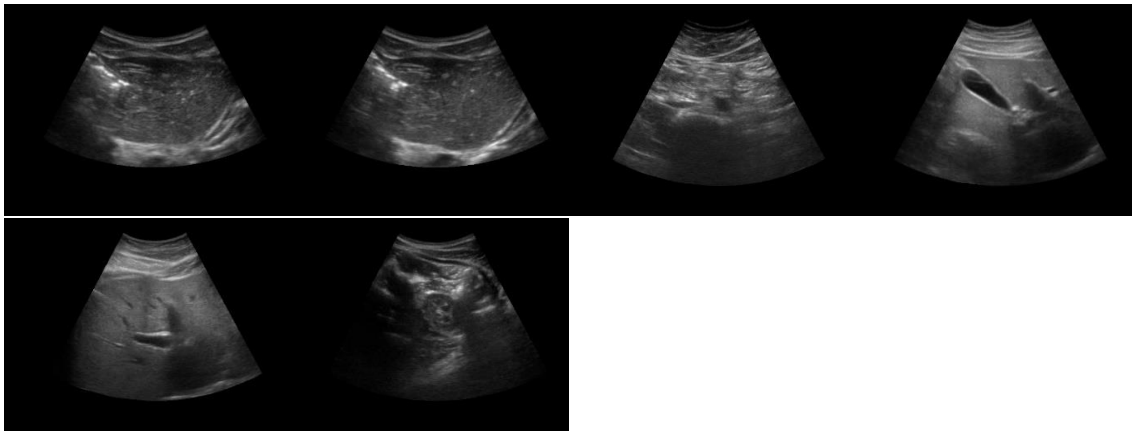
IMAGE DATSETS:

F0:

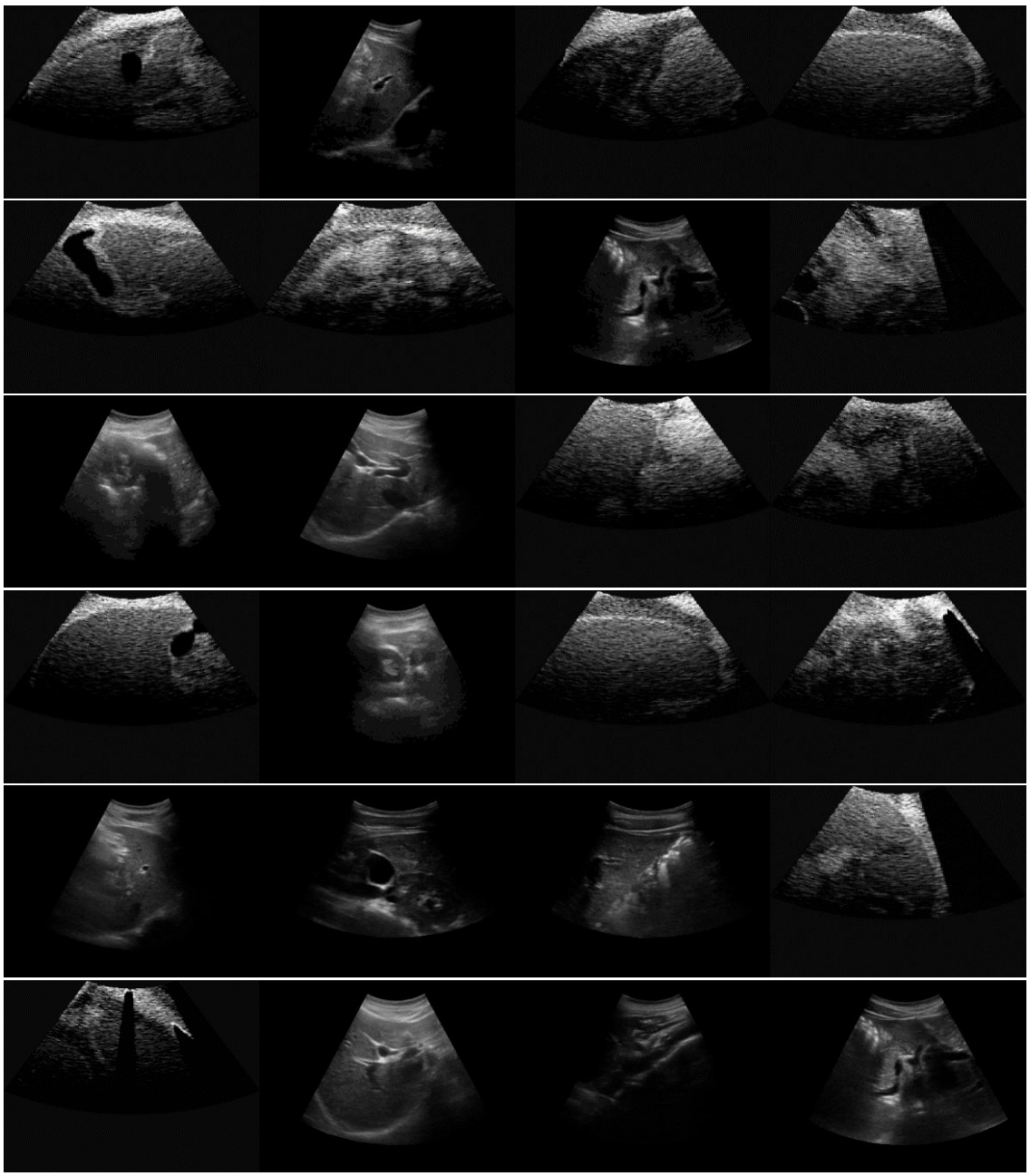


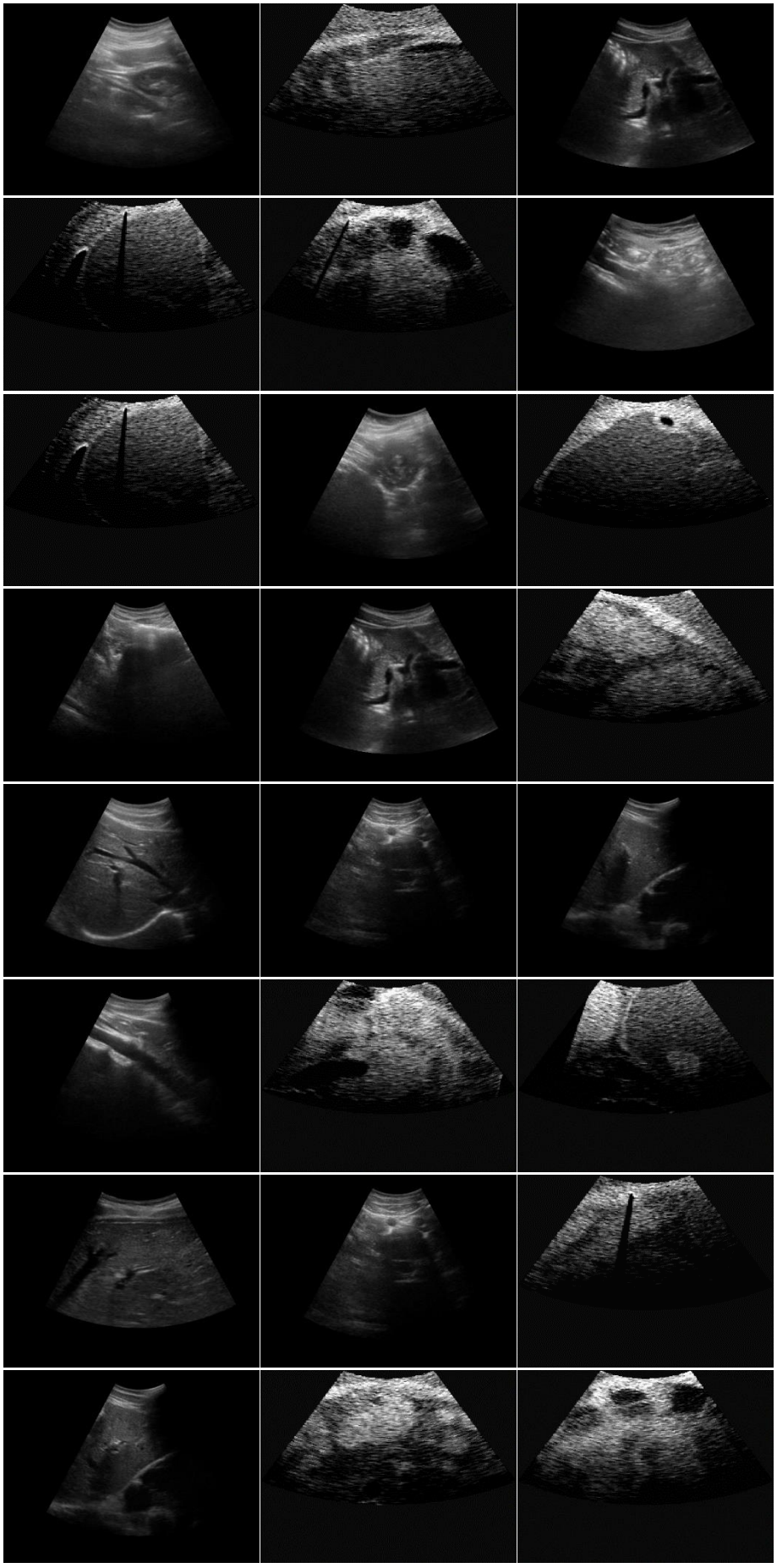


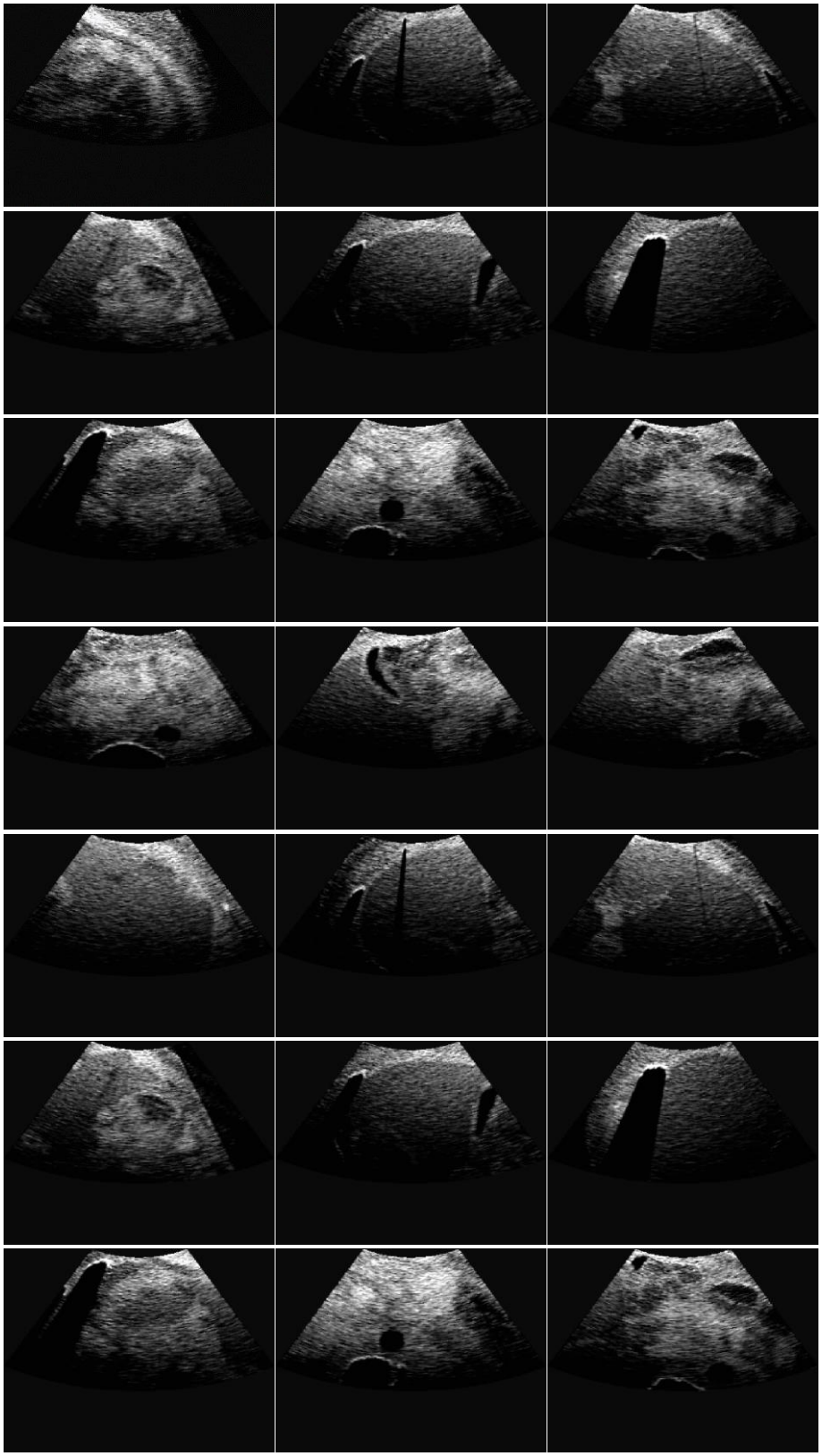


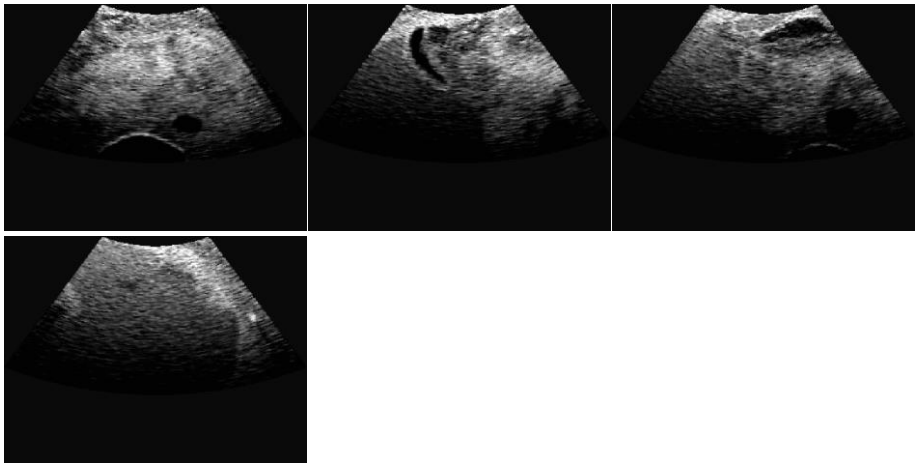


F1:









PYTHON CODE IMPLEMENTATION :

```
import pandas as pd
import matplotlib.pyplot as plot
import numpy as np
data=pd.read_csv('/content/cirrhosis.csv')
data.describe()
```

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Trygliceride	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

```
data.shape
data.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
```

```
data.columns
```

```
Index(['ID', 'N_Days', 'Status', 'Drug', 'Age', 'Sex', 'Ascites',
       'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol',
       'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
       'Prothrombin', 'Stage'],
      dtype='object')
```

```
data.isnull()
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	False	False	False	True	False	False	True	True	True	False	False	True	False	True	True	True	True	False	False	False
414	False	False	False	True	False	False	True	True	True	False	False	True	False	True	True	True	True	False	False	False
415	False	False	False	True	False	False	True	True	True	False	False	True	False	True	True	True	True	False	False	False
416	False	False	False	True	False	False	True	True	True	False	False	True	False	True	True	True	True	False	False	False
417	False	False	False	True	False	False	True	True	True	False	False	True	False	True	True	True	True	False	False	False

18 rows × 20 columns

```
data.isnull().sum()
```

	0
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

```
data.dropna()
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
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
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
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
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	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
307	308	1153	C	D-penicillamine	22347	F	N	Y	N	N	0.4	246.0	3.58	24.0	797.0	91.00	113.0	288.0	10.4
308	309	994	C	Placebo	21294	F	N	N	N	N	0.4	260.0	2.75	41.0	1166.0	70.00	82.0	231.0	10.8
309	310	939	C	D-penicillamine	22767	F	N	N	N	N	1.7	434.0	3.35	39.0	1713.0	171.00	100.0	234.0	10.2
310	311	839	C	D-penicillamine	13879	F	N	N	N	N	2.0	247.0	3.16	69.0	1050.0	117.00	88.0	335.0	10.5
311	312	788	C	Placebo	12109	F	N	N	Y	N	6.4	576.0	3.79	186.0	2115.0	136.00	149.0	200.0	10.8

276 rows × 20 columns

```
data.dropna(how='any')
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
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
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
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
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	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
307	308	1153	C	D-penicillamine	22347	F	N	Y	N	N	0.4	246.0	3.58	24.0	797.0	91.00	113.0	288.0	10.4
308	309	994	C	Placebo	21294	F	N	N	N	N	0.4	260.0	2.75	41.0	1166.0	70.00	82.0	231.0	10.8
309	310	939	C	D-penicillamine	22767	F	N	N	N	N	1.7	434.0	3.35	39.0	1713.0	171.00	100.0	234.0	10.2
310	311	839	C	D-penicillamine	13879	F	N	N	N	N	2.0	247.0	3.16	69.0	1050.0	117.00	88.0	335.0	10.5
311	312	788	C	Placebo	12109	F	N	N	Y	N	6.4	576.0	3.79	186.0	2115.0	136.00	149.0	200.0	10.8

276 rows × 20 columns

```
data.dropna(how='any')
data.dropna(how='all')
data.dropna(axis=1)
```

	ID	N_Days	Status	Age	Sex	Edema	Bilirubin	Albumin
0	1	400	D	21464	F	Y	14.5	2.60
1	2	4500	C	20617	F	N	1.1	4.14
2	3	1012	D	25594	M	S	1.4	3.48
3	4	1925	D	19994	F	S	1.8	2.54
4	5	1504	CL	13918	F	N	3.4	3.53
...	...	...	...	...	...	...	...	...
413	414	681	D	24472	F	N	1.2	2.96
414	415	1103	C	14245	F	N	0.9	3.83
415	416	1055	C	20819	F	N	1.6	3.42
416	417	691	C	21185	F	N	0.8	3.75
417	418	976	C	19358	F	N	0.7	3.29

418 rows × 8 columns

```
df=pd.DataFrame(data)
df
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
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
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
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
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	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	414	681	D	NaN	24472	F	NaN	NaN	NaN	N	1.2	NaN	2.96	NaN	NaN	NaN	NaN	174.0	10.9
414	415	1103	C	NaN	14245	F	NaN	NaN	NaN	N	0.9	NaN	3.83	NaN	NaN	NaN	NaN	180.0	11.2
415	416	1055	C	NaN	20819	F	NaN	NaN	NaN	N	1.6	NaN	3.42	NaN	NaN	NaN	NaN	143.0	9.9
416	417	691	C	NaN	21185	F	NaN	NaN	NaN	N	0.8	NaN	3.75	NaN	NaN	NaN	NaN	269.0	10.4
417	418	976	C	NaN	19358	F	NaN	NaN	NaN	N	0.7	NaN	3.29	NaN	NaN	NaN	NaN	350.0	10.6

418 rows × 20 columns

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data = data.dropna()
x = data.iloc[:, [2, 3]].values
y = data.iloc[:, 4].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25, random_state=0)
print(f"Training Set (x_train):\n{x_train}")
print(f"Test Set (x_test):\n{x_test}")
```

```
Training Set (x_train):
[['CL' 'Placebo']
 ['D' 'Placebo']
 ['CL' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'D-penicillamine']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['CL' 'Placebo']
 ['C' 'D-penicillamine']
 ['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'Placebo']
```

```
['D' 'D-penicillamine']
['CL' 'Placebo']
['D' 'D-penicillamine']
['C' 'Placebo']
['C' 'Placebo']
['D' 'D-penicillamine']
['D' 'Placebo']
['D' 'Placebo']
['C' 'Placebo']
['CL' 'Placebo']
['C' 'D-penicillamine']
['C' 'D-penicillamine']
['C' 'Placebo']
['C' 'Placebo']
['D' 'D-penicillamine']
['C' 'Placebo']
['D' 'Placebo']
['C' 'Placebo']
['C' 'Placebo']
['D' 'D-penicillamine']
['D' 'D-penicillamine']
['D' 'Placebo']
['D' 'Placebo']
['C' 'Placebo']
['D' 'Placebo']
['C' 'Placebo']
['D' 'D-penicillamine']
['D' 'D-penicillamine']
['D' 'Placebo']
['D' 'D-penicillamine']
```

```
Test Set (x_test):
[['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['D' 'Placebo']
 ['CL' 'Placebo']
 ['C' 'D-penicillamine']
 ['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['CL' 'Placebo']
 ['C' 'Placebo']
 ['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['CL' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['D' 'D-penicillamine']
```

```
['D' 'D-penicillamine']
['D' 'D-penicillamine']
['C' 'D-penicillamine']
['C' 'Placebo']
['CL' 'Placebo']
['CL' 'D-penicillamine']
['D' 'Placebo']
['D' 'D-penicillamine']
['C' 'D-penicillamine']
['CL' 'D-penicillamine']
['C' 'D-penicillamine']
['D' 'Placebo']
['C' 'D-penicillamine']
['D' 'D-penicillamine']
['D' 'D-penicillamine']
['C' 'D-penicillamine']
['C' 'D-penicillamine']
['D' 'D-penicillamine']
['C' 'Placebo']
['D' 'Placebo']
['D' 'Placebo']
['C' 'Placebo']
['C' 'D-penicillamine']
['C' 'Placebo']
['C' 'Placebo']
['CL' 'Placebo']
['C' 'Placebo']
['D' 'Placebo']
['C' 'D-penicillamine']
['C' 'D-penicillamine']
['D' 'D-penicillamine']
['D' 'D-penicillamine']]
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data=pd.read_csv('/content/cirrhosis.csv')
print(data.info())
print(data.columns)
data = data.dropna()
numerical_cols = data.select_dtypes(include=np.number).columns
print(f"Numerical Columns: {numerical_cols}")
x = data.loc[:,numerical_cols].values
y = data.iloc[:, 4].values
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25, random_state=0)
print(f"Training Set (x_train):\n{x_train}")
print(f"Test Set (x_test):\n{x_test}")
st_x = StandardScaler()
x_train = st_x.fit_transform(x_train)]
x_test = st_x.transform(x_test)
```

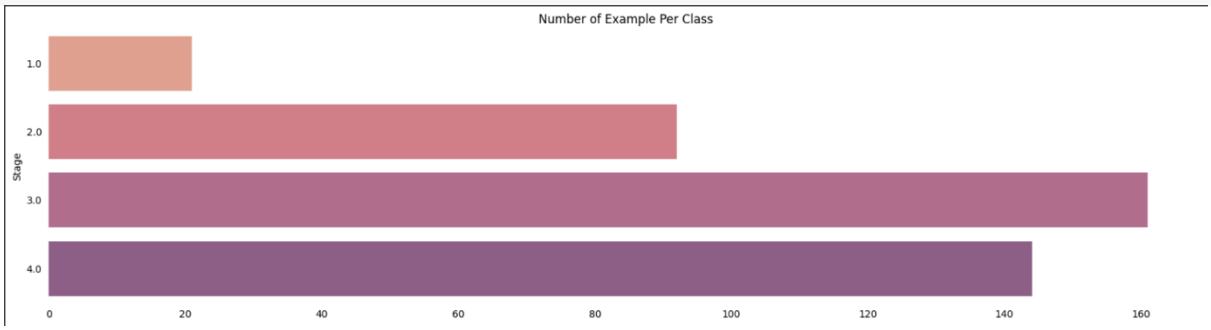
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             418 non-null    int64
 1   N_Days         418 non-null    int64
 2   Status         418 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
None
Index(['ID', 'N_Days', 'Status', 'Drug', 'Age', 'Sex', 'Ascites',
       'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol',
       'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
       'Prothrombin', 'Stage'],
      dtype='object')

Numerical Columns: Index(['ID', 'N_Days', 'Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper',
       'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin',
       'Stage'],
      dtype='object')
Training Set (x_train):
[[2.8800e+02 1.0670e+03 1.7874e+04 ... 2.9800e+02 9.6000e+00 2.0000e+00]
 [1.4800e+02 1.4270e+03 1.1273e+04 ... 3.3000e+02 9.8000e+00 3.0000e+00]
 [1.1100e+02 2.3500e+03 1.5031e+04 ... 4.6700e+02 1.0700e+01 3.0000e+00]
 ...
 [1.3500e+02 3.1500e+03 1.5694e+04 ... 4.4500e+02 1.1000e+01 2.0000e+00]
 [5.6000e+01 1.8470e+03 1.2279e+04 ... 3.6500e+02 1.0600e+01 2.0000e+00]
 [2.0000e+02 2.3180e+03 1.1773e+04 ... 3.3200e+02 9.9000e+00 3.0000e+00]]
Test Set (x_test):
[[2.1500e+02 1.0800e+03 1.5037e+04 5.9000e+00 1.2760e+03 3.8500e+00
  1.4100e+02 1.2040e+03 2.0305e+02 1.5700e+02 2.1600e+02 1.0700e+01
  3.0000e+00]
 [1.8000e+01 1.3100e+02 1.9698e+04 1.1400e+01 1.7800e+02 2.8000e+00
  5.8800e+02 9.6100e+02 2.8055e+02 2.0000e+02 2.8300e+02 1.2400e+01
  4.0000e+00]
 [6.5000e+01 3.9920e+03 1.4684e+04 1.2000e+00 2.5600e+02 3.6000e+00
  7.4000e+01 7.2400e+02 1.4105e+02 1.0800e+02 4.3000e+02 1.0000e+01
  1.0000e+00]
 [8.5000e+01 3.3580e+03 1.7246e+04 2.1000e+00 2.6200e+02 3.4800e+00
  5.8000e+01 2.0450e+03 8.9900e+01 8.4000e+01 2.2500e+02 1.1500e+01
  4.0000e+00]
 [2.2100e+02 2.0500e+03 2.0684e+04 9.0000e-01 3.6000e+02 3.6500e+00
  7.2000e+01 3.1860e+03 9.4550e+01 1.5400e+02 2.6900e+02 9.7000e+00
  4.0000e+00]
 [2.4300e+02 9.3000e+02 2.4650e+04 8.0000e+00 4.6800e+02 2.8100e+00
  1.3900e+02 2.0090e+03 1.9840e+02 1.3900e+02 2.3300e+02 1.0000e+01
  4.0000e+00]
 [1.0500e+02 3.0920e+03 1.2433e+04 1.1000e+00 4.6400e+02 4.2000e+00
  3.8000e+01 1.6440e+03 1.5190e+02 1.0200e+02 3.4800e+02 1.0300e+01
  3.0000e+00]
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
data_set = pd.read_csv('/content/cirrhosis.csv')
data_set = data_set.dropna()
x = data_set.iloc[:, [2, 3]].values
y = data_set.iloc[:, 4].values
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25, random_state=0)
print(f"Training Set (x_train):\n{x_train}")
print(f"Test Set (x_test):\n{x_test}")
```

```
Training Set (x_train):
[['CL' 'Placebo']
 ['D' 'Placebo']
 ['CL' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'D-penicillamine']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
```

```
Test Set (x_test):
[['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['D' 'Placebo']
 ['CL' 'Placebo']
 ['C' 'D-penicillamine']
 ['D' 'Placebo']
 ['D' 'D-penicillamine']
 ['C' 'Placebo']
 ['C' 'D-penicillamine']
 ['C' 'D-penicillamine']
 ['C' 'Placebo']
 ['D' 'Placebo']
 ['C' 'Placebo']
 ['D' 'D-penicillamine']
 ['CL' 'Placebo']
 ['C' 'Placebo']
```

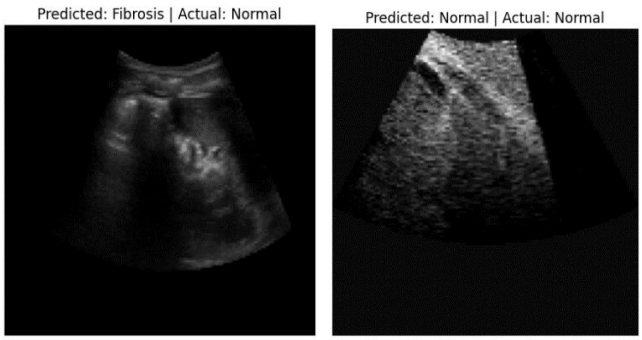
```
plt.figure(figsize=(21,5))
sns.countplot(y=df['Stage'], palette="flare", alpha=0.8, )
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False,
left=False)
plt.xlabel('')
plt.title('Number of Example Per Class')
```



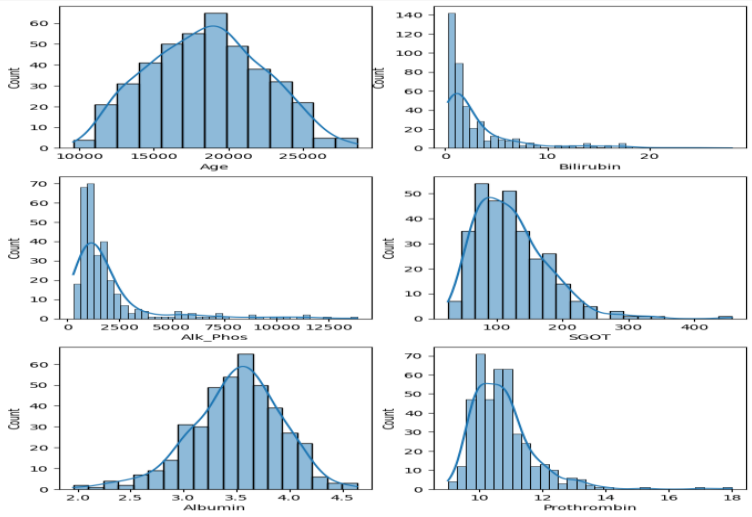

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
image_files = ["/content/e5180.jpg", "/content/e5816.jpg",
"/content/e5916.jpg",
               "/content/e6302.jpg", "/content/e697.jpg",
"/content/e7501.jpg"]

labels = [0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1]
image_size = (128, 128)
X = []
y = []
for i, file_path in enumerate(image_files):
    img = cv2.imread(file_path)
    if img is not None:
        img = cv2.resize(img, image_size)
```

```
        img = img / 255.0
        X.append(img)
        y.append(labels[i])
    else:
        print(f"Error loading image: {file_path}")
X = np.array(X)
y = np.array(y)
y = to_categorical(y, num_classes=2)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=8,
validation_split=0.1)
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
predictions = model.predict(X_test)
predicted_labels = np.argmax(predictions, axis=1)
true_labels = np.argmax(y_test, axis=1)
for i in range(len(X_test)):
    plt.imshow(X_test[i])
    plt.title(f"Predicted: {'Fibrosis' if predicted_labels[i] == 1 else
'Normal'} | Actual: {'Fibrosis' if true_labels[i] == 1 else 'Normal'}")
    plt.axis('off')
    plt.show()
```

```
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 1.0000 - loss: 0.0788 - val_accuracy: 1.0000 - val_loss: 0.6538
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 320ms/step - accuracy: 1.0000 - loss: 0.0238 - val_accuracy: 1.0000 - val_loss: 0.6339
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 240ms/step - accuracy: 1.0000 - loss: 0.0471 - val_accuracy: 1.0000 - val_loss: 0.5513
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.5000 - loss: 1.2312
Test Accuracy: 50.00%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
```




```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
numerical_columns = ['Age', 'Bilirubin', 'Alk_Phos', 'SGOT', 'Albumin',
'Prothrombin']
num_train = df[numerical_columns]
rows = (len(num_train.columns) + 1) // 2
_, axes = plt.subplots(rows, 2, figsize=(8, rows * 4))
for i, col in enumerate(num_train.columns):
    hor = i % 2
    ver = i // 2
    sns.histplot(data=num_train[col], kde=True, ax=axes[ver, hor])
plt.show()
```




```
import pandas as pd
import numpy as np
from sklearn.metrics import multilabel_confusion_matrix
csv_file_path = '/content/liver_cirrhosis.csv'
df = pd.read_csv(csv_file_path)
df["Stage"] = df["Stage"].astype(str).apply(lambda x: eval(x))
y_true = np.array(df["Stage"].tolist())
np.random.seed(42)
y_pred = np.random.randint(0, 2, size=y_true.shape)
conf_matrices = multilabel_confusion_matrix(y_true, y_pred)
label_names = ["Cholesterol", "Albumin", "Stage"]
```

```
conf_matrices_dict = {label: conf_matrices[i] for i, label in
enumerate(label_names)}
for label, matrix in conf_matrices_dict.items():
    print(f"Confusion Matrix for {label}:")
    print(matrix)
    print()
```

```
Confusion Matrix for Cholesterol:
[[12502 12498]
 [    0     0]]

Confusion Matrix for Albumin:
[[8324 8411]
 [4174 4091]]

Confusion Matrix for Stage:
[[16559     0]
 [ 8441     0]]
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
data = pd.read_csv('/content/liver_cirrhosis.csv')
X = data[['Cholesterol', 'Albumin']]
y = data['Stage']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.8421333333333333
```

```
from sklearn.metrics import precision_score
y_pred = model.predict(X_test)
precision = precision_score(y_test, y_pred, average='weighted')
print(f'Precision: {precision}')
```

```
Precision: 0.8422693688657297
```

```
from sklearn.metrics import recall_score
y_pred = model.predict(X_test)
recall = recall_score(y_test, y_pred, average='weighted')
print(f'Recall: {recall}')
```

```
Recall: 0.8421333333333333
```

```
from sklearn.metrics import f1_score
y_pred = model.predict(X_test)
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'F1 Score: {f1}')
```

```
F1 Score: 0.8421274711080007
```