Project on liver cirrhosis prediction using Machine Learning

- **Aim:-**To create a Data science Project, where we will be predicting the the stage of liver Cirrhosis using 18 clinical features. Cirrhosis damages the liver from a variety of causes leading to scarring and liver failure

  Prediction on liver cirrhosis with help of :-

  Train Dataset - It consists of a total of 6801 data points. • Test Dataset - You must predict the stage of cirrhosis of 3201 data points.

- Steps to be taken in the project is sub-divided into the following sections. These are:
  - Importing the libraries such as 'numpy', 'pandas','sklearn. model' etc.
  - Loading Dataset as a CSV file for training & testing the models.
  - Splitting the data set into independent & dependent sets.
  - Checking if still any null values or any other data types other than float and integers are present into the dataset or not.
  - Importing the train_test_split model from sklearn.model for splitting data into train & test sets.
  - Applying the different kinds of ML Algorithms .which gives Best accuracy of model.
  - Also checking with new data set for predicting the values.
- Steps of creating ML model:-
- Importing numpy as np & pandas as pd for loading and reading the data-set & using matplotlib.pyplot and Seaborn for visualization of data.

```
[1]
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as snr
```

- Loading the csv-dataset in the variable name 'data_train' Then viewing the data with data_train.head()

```
[2] data_train=pd.read_csv('/content/train_dataset.csv')
    data_train.head()
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	7135	1654	CL	D-penicillamine	19581	F	N	N	Y	N	0.3	279.0	2.96	84.0	1500.8	99.43	109.0	293.0	10.2	4.0
1	7326	41	C	D-penicillamine	22880	F	NaN	N	NaN	N	0.3	NaN	2.96	NaN	1835.4	26.35	131.0	308.0	10.8	1.0
2	7254	297	D	NaN	27957	F	N	N	NaN	N	0.3	328.0	2.64	4.0	NaN	NaN	116.0	194.0	10.3	3.0
3	3135	1872	C	D-penicillamine	21111	F	NaN	Y	Y	N	0.3	302.0	2.02	49.0	NaN	26.35	NaN	NaN	10.5	4.0
4	2483	939	CL	D-penicillamine	18061	F	NaN	NaN	NaN	N	0.5	344.0	3.11	91.0	NaN	104.56	NaN	306.0	11.4	2.0

- Checking the data such as number of columns, rows and type of data(float,integer) with help of data_train.info()

```
[4] data_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6800 entries, 0 to 6799
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             6800 non-null   int64
 1   N_Days         6800 non-null   int64
 2   Status         6800 non-null   object
 3   Drug           4775 non-null   object
 4   Age            6800 non-null   int64
 5   Sex            6800 non-null   object
 6   Ascites        4554 non-null   object
 7   Hepatomegaly   4373 non-null   object
 8   Spiders        4210 non-null   object
 9   Edema          6800 non-null   object
 10  Bilirubin      6800 non-null   float64
 11  Cholesterol    3699 non-null   float64
 12  Albumin        6800 non-null   float64
 13  Copper         4644 non-null   float64
 14  Alk_Phos       4302 non-null   float64
 15  SGOT           4698 non-null   float64
 16  Tryglicerides  3988 non-null   float64
 17  Platelets      6462 non-null   float64
 18  Prothrombin    6645 non-null   float64
 19  Stage          6800 non-null   float64
dtypes: float64(10), int64(3), object(7)
memory usage: 1.0+ MB
```

We observe that the above data have integer, object and float.

```
[6] data_train.shape

(6800, 20)
```

Train data have 6800 Rows and 20 columns

- ❖ Now checking data have Nan value or not.

```
[ ] #missing values columns wise
    data_train.isnull().sum(axis=0).sort_values()

ID                0
Albumin           0
Bilirubin         0
Sex               0
Edema             0
Status            0
N_Days            0
Age               0
Stage             0
Prothrombin     155
Platelets       338
Drug           2025
SGOT           2102
Copper         2156
Ascites        2246
Hepatomegaly   2427
Alk_Phos       2498
Spiders        2590
Tryglicerides  2812
Cholesterol    3101
dtype: int64
```

```
[ ] data_train.isnull().mean()*100

ID               0.000000
N_Days           0.000000
Status           0.000000
Drug            29.779412
Age              0.000000
Sex              0.000000
Ascites         33.029412
Hepatomegaly    35.691176
Spiders         38.088235
Edema            0.000000
Bilirubin        0.000000
Cholesterol     45.602941
Albumin          0.000000
Copper          31.705882
Alk_Phos        36.735294
SGOT            30.911765
Tryglicerides   41.352941
Platelets        4.970588
Prothrombin      2.279412
Stage            0.000000
dtype: float64
```

We observe that the above data have Nan value. And we see that maximum 41.5% Nan value in Triglycerides .we cleaning the Nan value before working on it.

- We remove the Nan value with help of fillna(method='ffill)

```
[ ] # we use forward data for removing Nan value data
    new_data_train=data_train.fillna(method='ffill')
    new_data_train
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	7135	1654	CL	D-penicillamine	19581	F	N	N	Y	N	0.3	279.0	2.96	84.0	1500.8	99.43	109.0	293.0	10.2	4.0
1	7326	41	C	D-penicillamine	22880	F	N	N	Y	N	0.3	279.0	2.96	84.0	1835.4	26.35	131.0	308.0	10.8	1.0
2	7254	297	D	D-penicillamine	27957	F	N	N	Y	N	0.3	328.0	2.64	4.0	1835.4	26.35	116.0	194.0	10.3	3.0
3	3135	1872	C	D-penicillamine	21111	F	N	Y	Y	N	0.3	302.0	2.02	49.0	1835.4	26.35	116.0	194.0	10.5	4.0
4	2483	939	CL	D-penicillamine	18061	F	N	Y	Y	N	0.5	344.0	3.11	91.0	1835.4	104.56	116.0	306.0	11.4	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6795	4622	1762	C	Placebo	20909	M	N	N	N	N	0.9	360.0	2.43	274.0	3444.4	131.59	130.0	314.0	12.9	1.0
6796	4446	2955	C	Placebo	28650	F	Y	N	N	N	0.3	360.0	2.99	43.0	3444.4	131.59	238.0	399.0	10.6	4.0
6797	4082	584	C	Placebo	24102	F	N	Y	N	N	0.3	218.0	2.79	43.0	3444.4	26.35	123.0	103.0	9.3	1.0
6798	2248	1426	CL	D-penicillamine	19791	F	N	Y	Y	N	0.3	218.0	1.96	12.0	289.0	72.95	134.0	354.0	10.6	2.0
6799	4826	125	D	D-penicillamine	25245	F	Y	N	Y	N	5.9	218.0	3.65	12.0	289.0	26.35	69.0	288.0	11.1	4.0

8800 rows × 20 columns

```
[ ] new_data_train.isnull().sum(axis=0).sort_values()# we can check the data having Nan data
```

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

We observe that the above data have fully remove Nan value.

- Now ,Main focus convert the categorical data into Numerical data with help of one hot encoding method.

```
[ ] # we convert the categorical data into numerical data
    my_data=pd.get_dummies(new_data_train,columns=['Status','Drug','Sex','Ascites','Hepatomegaly','Spiders','Edema'],drop_first=True)
    my_data.head()
```

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	...	Stage	Status_CL	Status_D	Drug_Placebo	Sex_M	Ascites_Y	Hepatomegaly_Y	Spiders_Y	Edema_S	Edema_Y
0	7135	1654	19581	0.3	279.0	2.96	84.0	1500.8	99.43	109.0	...	4.0	1	0	0	0	0	0	1	0	0
1	7326	41	22880	0.3	279.0	2.96	84.0	1835.4	26.35	131.0	...	1.0	0	0	0	0	0	0	1	0	0
2	7254	297	27957	0.3	328.0	2.64	4.0	1835.4	26.35	116.0	...	3.0	0	1	0	0	0	0	1	0	0
3	3135	1872	21111	0.3	302.0	2.02	49.0	1835.4	26.35	116.0	...	4.0	0	0	0	0	0	1	1	0	0
4	2483	939	18061	0.5	344.0	3.11	91.0	1835.4	104.56	116.0	...	2.0	1	0	0	0	0	1	1	0	0

5 rows × 22 columns

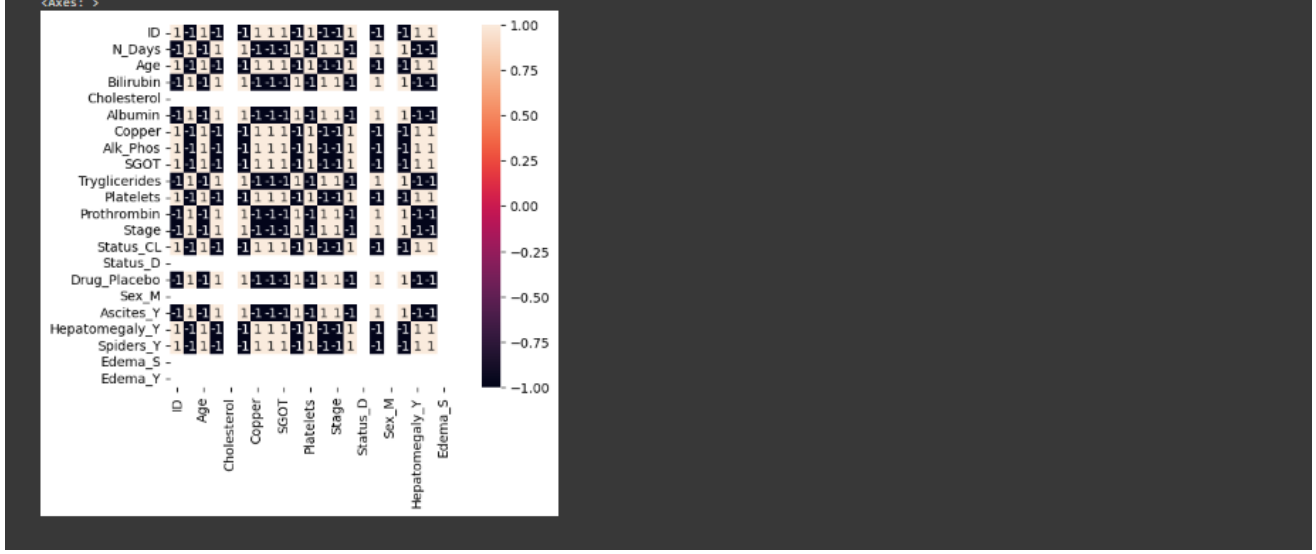
```
my_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3200 entries, 0 to 3199
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              3200 non-null   int64
 1   N_Days          3200 non-null   int64
 2   Age             3200 non-null   int64
 3   Bilirubin       3200 non-null   float64
 4   Cholesterol     3200 non-null   float64
 5   Albumin         3200 non-null   float64
 6   Copper          3200 non-null   float64
 7   Alk_Phos        3200 non-null   float64
 8   SGOT            3200 non-null   float64
 9   Tryglicerides   3200 non-null   float64
 10  Platelets       3200 non-null   float64
 11  Prothrombin     3200 non-null   float64
 12  Status_CL       3200 non-null   uint8
 13  Status_D        3200 non-null   uint8
 14  Drug_Placebo    3200 non-null   uint8
 15  Sex_M           3200 non-null   uint8
 16  Ascites_Y       3200 non-null   uint8
 17  Hepatomegaly_N  3200 non-null   uint8
 18  Hepatomegaly_Y  3200 non-null   uint8
 19  Spiders_Y       3200 non-null   uint8
 20  Edema_S         3200 non-null   uint8
 21  Edema_Y         3200 non-null   uint8
dtypes: float64(9), int64(3), uint8(10)
memory usage: 331.4 KB
```

Finally we observe the data are fully cleaned.

- ❖ Now we check the data dependency.

```
plt.figure(figsize=(5,5))
snr.heatmap(my_data[4:6].corr(),annot=True)
```

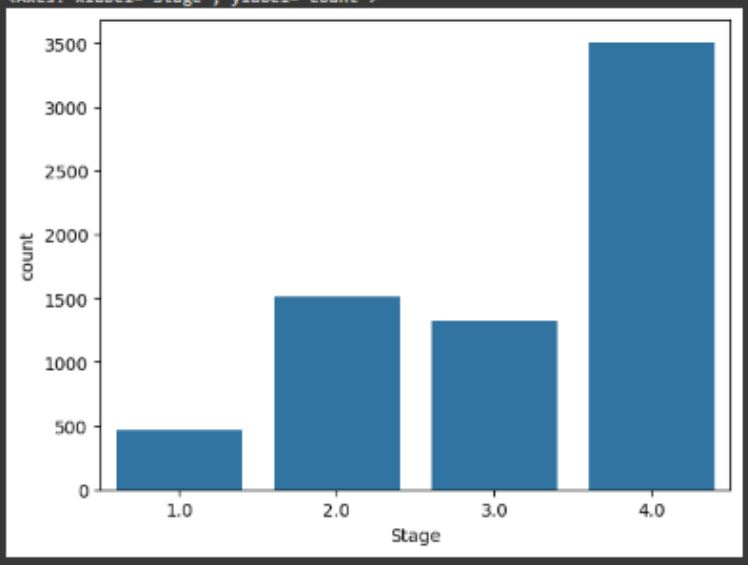


We see that data dependent each other.

- ❖ Visualizing the liver cirrhosis with various Stage like Status, Drug, Sex, Hepatomegaly, Cholesterol Platelets, Stage etc.

```
[ ] snr.countplot(x='Stage',data=data_train)
```

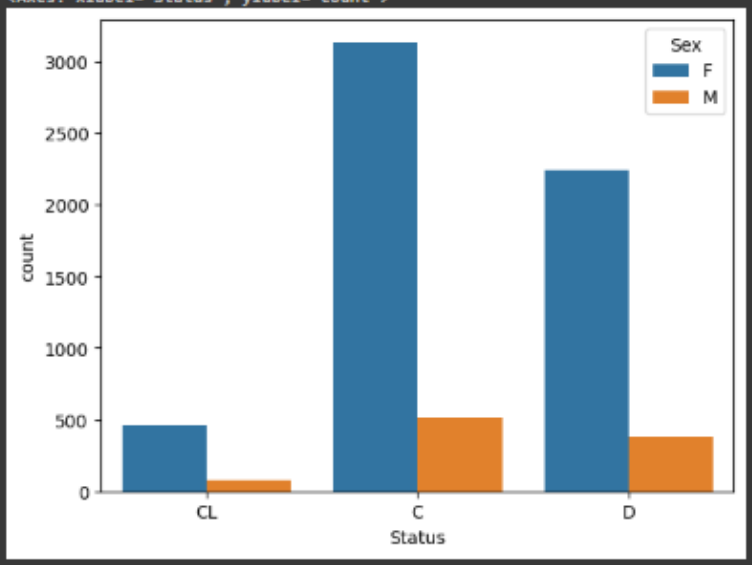
```
<Axes: xlabel='Stage', ylabel='count'>
```



As per Visualizing the above graph, liver cirrhosis in stage 4 is more than the other stage.

```
[ ] snr.countplot(data=data_train, x="Status",hue='Sex')
```

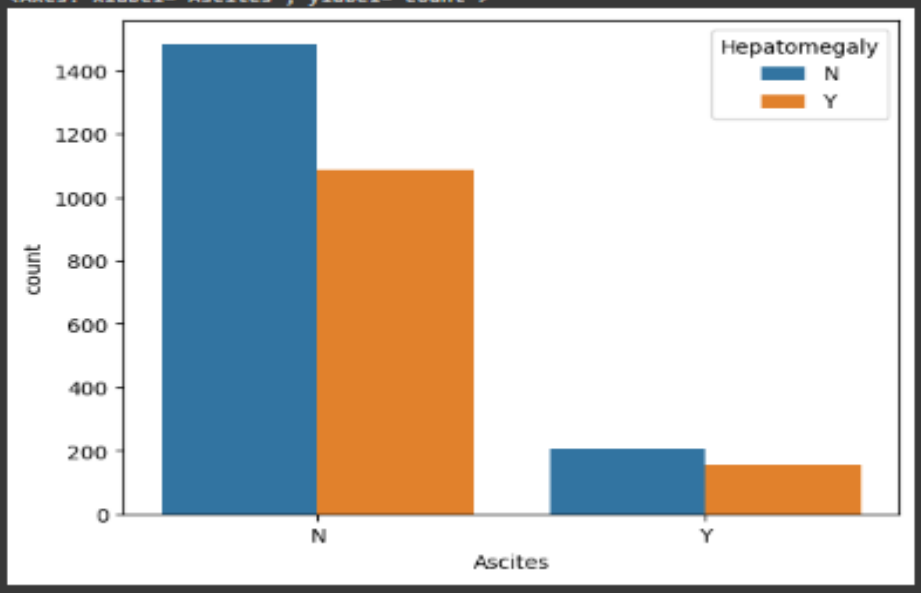
```
<Axes: xlabel='Status', ylabel='count'>
```



As per Visualizing the above graph, censored due to liver transplant in female is more.

```
[ ]  snr.countplot(data=data_train, x="Ascites",hue='Hepatomegaly')
```

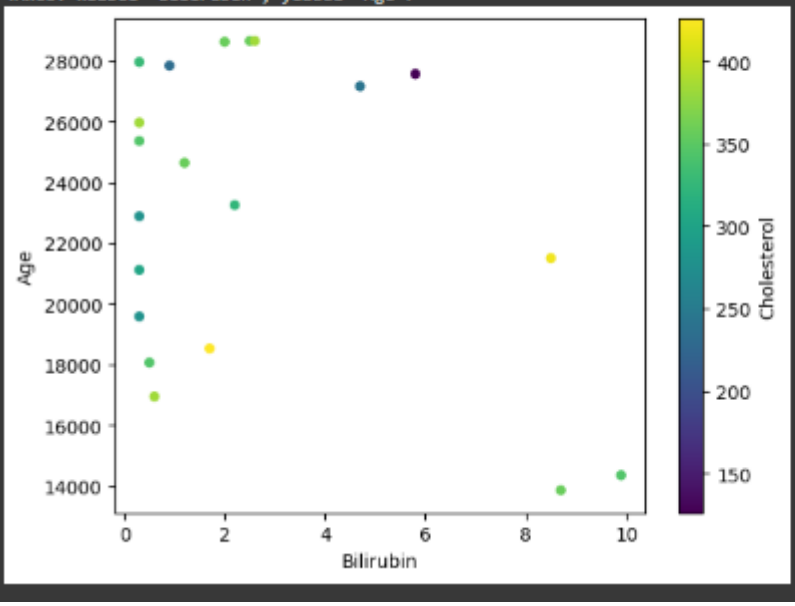
```
<Axes: xlabel='Ascites', ylabel='count'>
```



As per Visualizing the above graph, People have no presence of Hepatomegaly as well as Ascites.

```
[ ]  my_data.iloc[0:20].plot.scatter(x='Bilirubin',y='Age',c='Cholesterol')
```

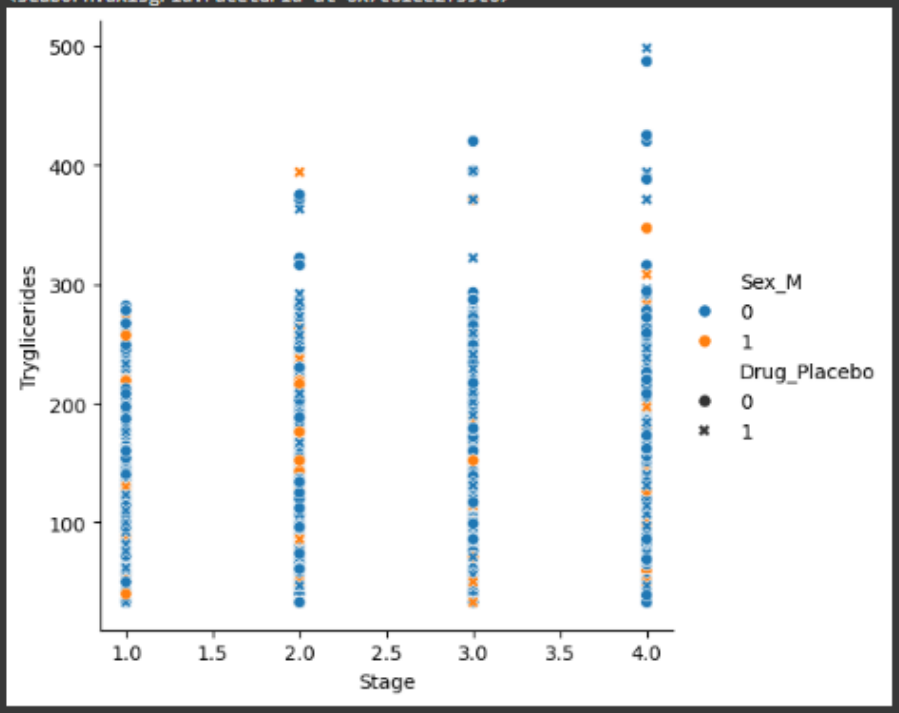
```
<Axes: xlabel='Bilirubin', ylabel='Age'>
```



We observed that cholesterol having more with less of Bilirubin

```
snr.relplot(x='Stage',y='Tryglicerides',hue='Sex_M',style='Drug_Placebo',data=my_data)
```

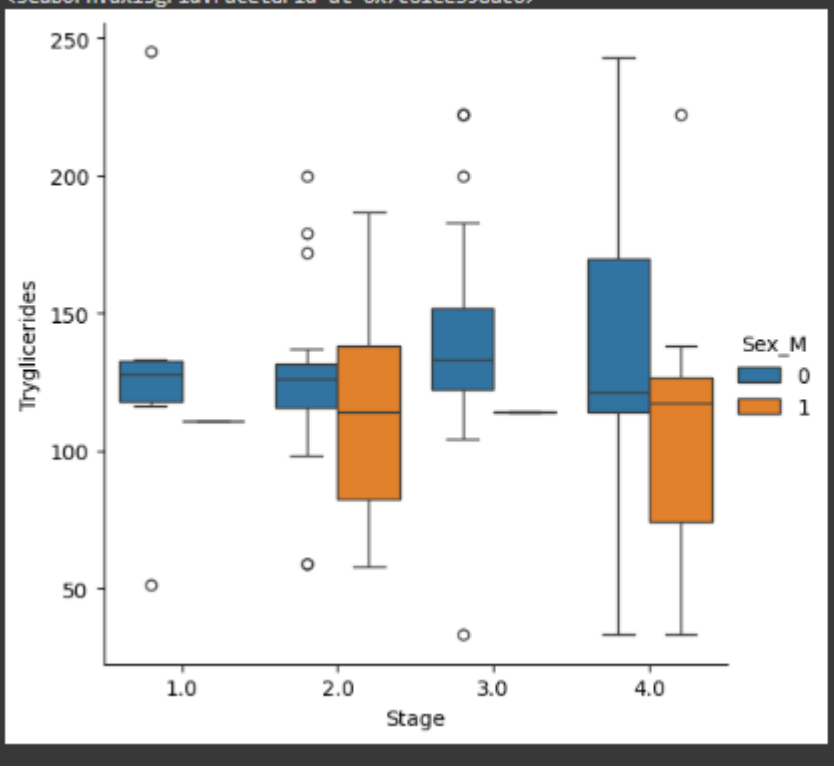
```
<seaborn.axisgrid.FacetGrid at 0x7c61ee2f59c0>
```



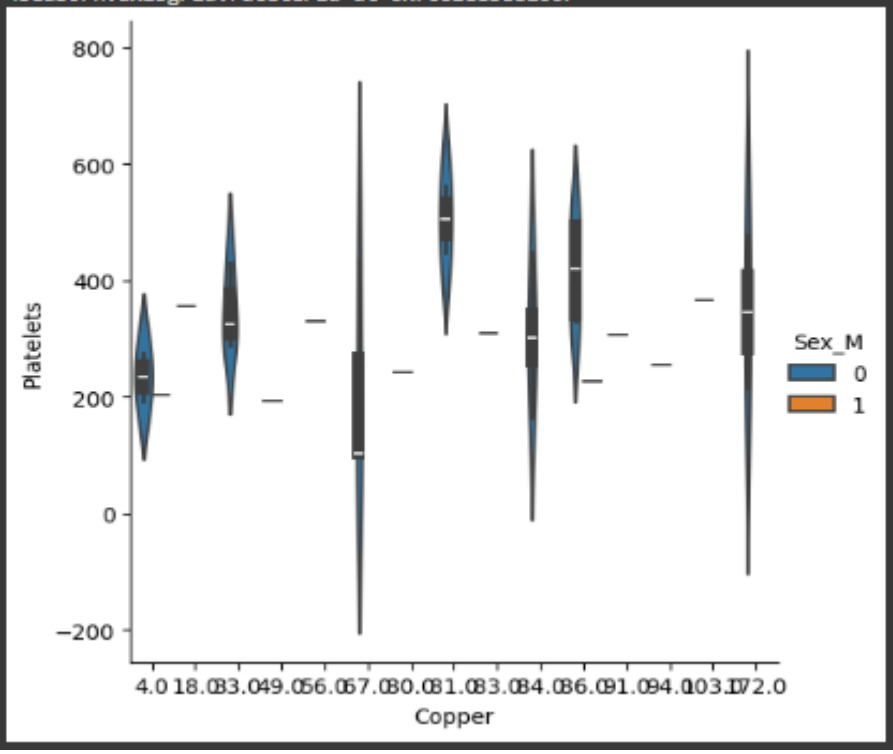
We observe the above graph,stage 4 have more Tryglicerides in female.

```
[ ] snr.catplot(x='Stage',y='Tryglicerides',hue='Sex_M',data=my_data.iloc[0:110],kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7c61ee398ac0>
```



```
[ ] <seaborn.axisgrid.FacetGrid at 0x7c61ee388100>
```



- ❖ Splitting the dataset into dependent & independent sets

```
[ ] #Divide the data into dependent and independent set
    x=my_data.drop(columns=['Stage'])
    y=my_data['Stage']
```

- ➢ Importing train_test_split from sklearn.model library for splitting the data into train and test sets. (we consider train dataset).

```
[ ] #spilting the data into training and testing set.
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=0)
```

- ➢ Importing KNN from sklearn Libaray & then activating the Machine learning Model .Then used knn.fit() to training the model by providing train & test sets as x & y. And then predicted the trained model with help of MLM & the checked score as knn.score(x,y)

```
# by using knn for machine learning model
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5) #where k=5
```

```
[ ] knn.fit(x_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_predict=knn.predict(x_test)
```

- Checking the accuracy with help of confusion Matrix.

```
#checking the accuracy with help of confusion matrix
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_predict)
ac=accuracy_score(y_test,y_predict)
```

```
print(cm)
print(ac)
```

```
[[  5  20  21  41]
 [ 17  66  62 165]
 [ 18  60  94  74]
 [ 28 139 128 422]]
0.43161764705882355
```

-In the above model we can see that the accuracy obtained is 43% which is not good , we can try using different models to see if we can get better accuracy than this or not.

- Now applying new algorithm DecisionTreeclassifier ,then checked score.

```
# by using Decision tree for machine learning model
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier()
```

```
tree.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
tree_predict=tree.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,tree_predict)
ac=accuracy_score(y_test,tree_predict)
```

```
print(cm)
print(ac)
```

```
[[ 13  19  24  31]
 [ 22  87  46 155]
 [ 29  40  83  94]
 [ 44 177 124 372]]
0.40808823529411764
```

In the above model we can see that the accuracy obtained is 40% ,is less than **KNN**.But we can try using **SVM** to see if we can get better accuracy than this or not.

- Applying Support Vector Machines

```
from sklearn import svm
clt=svm.SVC(kernel='linear')
```

```
clt.fit(x_train,y_train)
```

```
SVC(kernel='linear')
```

```
predictions=clt.predict(x_test)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
cm=confusion_matrix(y_test,predictions)
ac=accuracy_score(y_test,predictions)
```

```
print(cm)
print(ac)
```

```
[[  0   0  21  66]
 [  0   0  31 279]
 [  0   0  67 179]
 [  0   0  95 622]]
0.5066176470588235
```

In the above model we can see that the accuracy obtained 50%

Algorithms	accuracy
Support vector machine	50%
KNN	40%
Decision Tree classifier	43%

Conclusion-Support vector machine algorithms is better than KNN and Decision Tree.

- Now recalling the test data set.
- Loading the csv-dataset in the variable name 'data_test' Then viewing the data with data_test.head()

```
data_test=pd.read_csv('/content/test_dataset.csv')
data_test.head()
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
0	3870	41	C	Placebo	22553	F	N	NaN	N	N	1.4	247.0	3.62	NaN	NaN	108.65	NaN	169.0	11.6
1	3462	1811	C	D-penicillamine	16223	F	N	Y	N	N	0.3	311.0	2.80	92.0	1748.1	NaN	129.0	321.0	11.5
2	1632	954	C	D-penicillamine	27100	F	N	N	N	N	0.4	NaN	3.56	NaN	NaN	43.52	NaN	296.0	10.3
3	722	1969	D	Placebo	17039	F	N	Y	N	N	1.2	NaN	3.16	NaN	617.1	113.76	NaN	125.0	10.9
4	1000	2721	D	D-penicillamine	17738	F	NaN	NaN	NaN	N	3.2	NaN	2.36	89.0	1782.4	NaN	129.0	138.0	10.6

- Cleaning the test dataset.

```
# we use forward data for removing Nan value data
new_data_test=data_test.fillna(method='bfill')
new_data_test
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
0	3870	41	C	Placebo	22553	F	N	Y	N	N	1.4	247.0	3.62	92.0	1748.1	108.65	129.0	169.0	11.6
1	3462	1811	C	D-penicillamine	16223	F	N	Y	N	N	0.3	311.0	2.80	92.0	1748.1	43.52	129.0	321.0	11.5
2	1632	954	C	D-penicillamine	27100	F	N	N	N	N	0.4	206.0	3.56	89.0	617.1	43.52	129.0	296.0	10.3
3	722	1969	D	Placebo	17039	F	N	Y	N	N	1.2	206.0	3.16	89.0	617.1	113.76	129.0	125.0	10.9
4	1000	2721	D	D-penicillamine	17738	F	N	N	N	N	3.2	206.0	2.36	89.0	1782.4	37.87	129.0	138.0	10.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3195	2601	4513	D	Placebo	18293	M	N	N	N	S	0.3	422.0	2.75	106.0	2063.9	111.68	166.0	563.0	10.9
3196	53	3281	D	D-penicillamine	12149	F	N	N	Y	N	4.7	422.0	4.30	106.0	1203.2	32.49	166.0	234.0	10.2
3197	386	1933	D	D-penicillamine	17084	F	N	Y	N	N	0.8	408.0	3.56	9.0	2358.1	124.70	131.0	234.0	11.6
3198	920	2350	D	Placebo	28650	M	N	N	N	Y	15.3	258.0	3.21	32.0	1472.7	113.24	130.0	110.0	10.3
3199	2047	2691	D	D-penicillamine	13765	F	N	N	N	N	0.3	NaN	3.42	63.0	1598.8	NaN	126.0	103.0	10.8

3200 rows × 19 columns

- Splitting into test & train sets as x1_test & x1_train. Then we find the liver cirrhosis prediction using Machine Learning(SVM)

```
[ ]  #now cleaning the data then use for prediction
     #now spilting into x1 test and x1_train set

[ ]  #spilting the data into training and testing set.
     from sklearn.model_selection import train_test_split
     x1_train, x1_test = train_test_split(df, test_size=0.1, random_state=0)
```

Applying SVM algorithms for predictions.

```
ty_predict=clt.predict(x1_test)

51] ty_predict

array([4., 2., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 2., 4., 4., 4., 4., 4., 4., 4., 4., 2., 4., 3., 2., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 2., 4., 4., 4., 4., 4., 4., 1., 4., 4., 4., 4.,
       2., 4., 2., 4., 4., 4., 4., 4., 4., 4., 1., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 2., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 2.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 2., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 2., 2., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       2., 4., 4., 4., 4., 4., 4., 4., 2., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 1., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 2., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 2., 4., 4., 4., 2., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 2., 4., 3., 4., 4., 4., 3., 4., 4., 4., 4., 4.,
       4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 4., 2., 4., 4.])
```

Conclusion:- In this test data set we analysed the data we found the max. liver disease in stage 4.

Thank you