Project on Liver Disease Prediction using Machine Learning Models

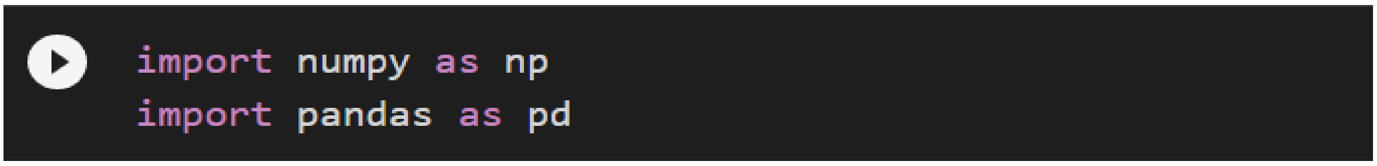
- Introduction: This is a Data Science project, where we'll be predicting the Liver Disease using machine learning models with the help of csv-dataset(s) provided which contains 18 clinical features which causes liver Cirrhosis. Cirrhosis damages the liver from a variety of cause leading to scarring and liver failure.

- The project is sub-divided into the following sections. These are:

  - Loading necessary libraries such as 'numpy', 'pandas', 'sklearn. model' etc.
  - Loading Dataset(s) as a CSV file. Here we are using two different files for training & testing the models.
  - Data cleaning was performed by dropping useless data & changing string values to integer values. I have replaced the null values of data set with '0'.
  - Visualisation of Liver Disease using Tableau.
  - Splitting the data set into independent & dependent sets (only train data set was taken in use).
  - Importing the train_test_split model from sklearn.model for splitting data into train & test sets.
  - Importing the Decision Tree Classifier & then training the model with the help of fit ().
  - Predicting the trained model & then checking accuracy of the model using confusion matrix & accuracy score.
  - Then recalled test_dataset & splitted the data into testing & training using X1_train & X1_test.
  - Trained the test_dataset with tain_dataset using two different classifiers.
  - Finally, predicted the stage of liver disease for test_dataset.

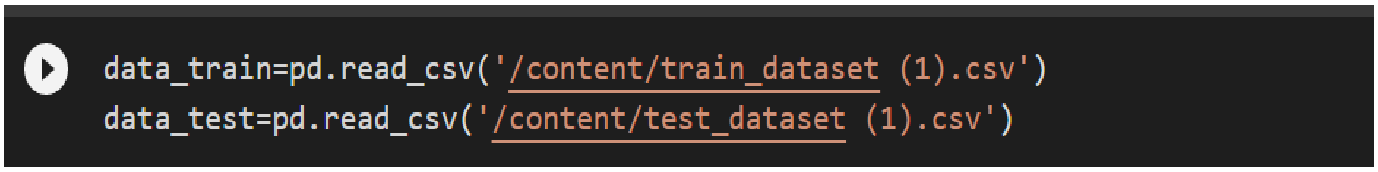
Steps for creating a model:

- Step-1: Importing numpy as np & pandas as pd for loading and reading the data-set.

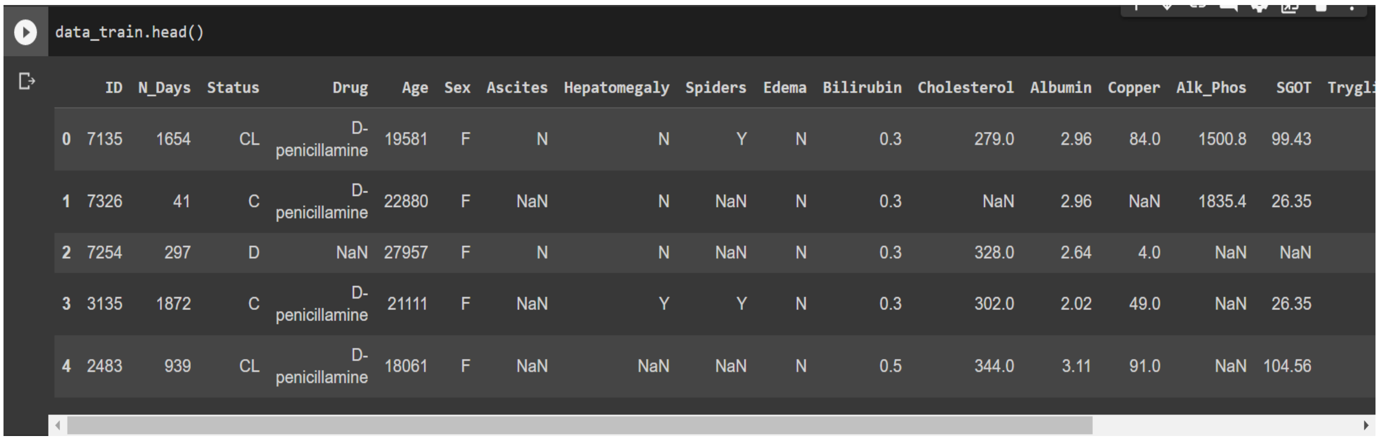


```
import numpy as np
import pandas as pd
```

- Step-2: Loading the csv-dataset(s) in the variable name(s) 'data_train' & 'data_test'. Then viewing the data(s) with data_train.head() & data_test.head().



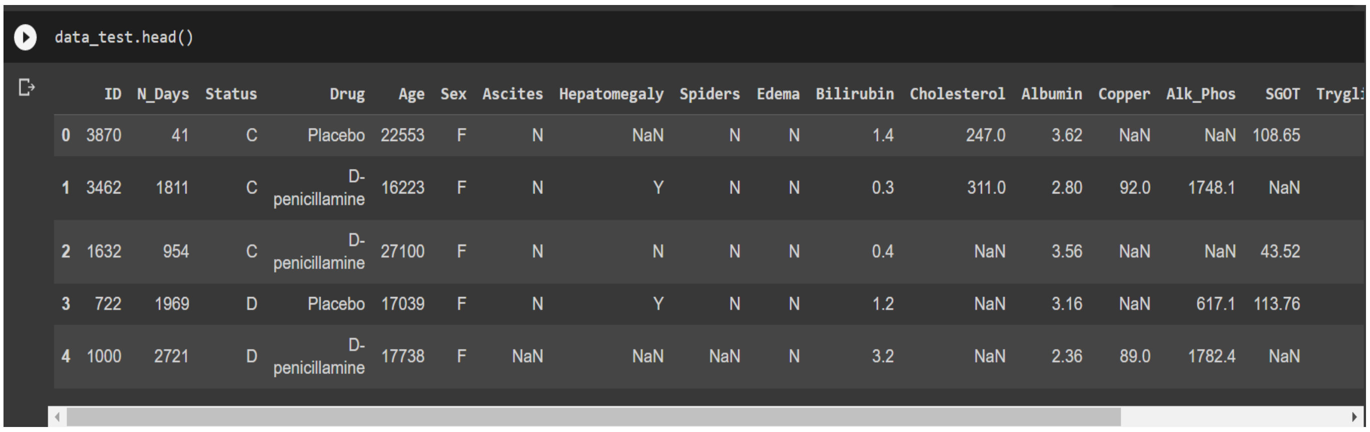
```
data_train=pd.read_csv('/content/train_dataset (1).csv')
data_test=pd.read_csv('/content/test_dataset (1).csv')
```



```
data_train.head()
```

```
   ID    N_Days  Status  Drug                Age    Sex  Ascites  Hepatomegaly  Spiders  Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos  SGOT    Trygl
0  7135  1654    CL      D-penicillamine     19581  F    N        N             Y        N      0.3        279.0        2.96     84.0    1500.8    99.43
1  7326  41      C       D-penicillamine     22880  F    NaN      N             NaN      N      0.3        NaN          2.96     NaN     1835.4    26.35
2  7254  297     D       NaN                 27957  F    N        N             NaN      N      0.3        328.0        2.64     4.0     NaN       NaN
3  3135  1872    C       D-penicillamine     21111  F    NaN      Y             Y        N      0.3        302.0        2.02     49.0    NaN       26.35
4  2483  939     CL      D-penicillamine     18061  F    NaN      NaN           NaN      N      0.5        344.0        3.11     91.0    NaN       104.56
```

-Viewing train dataset

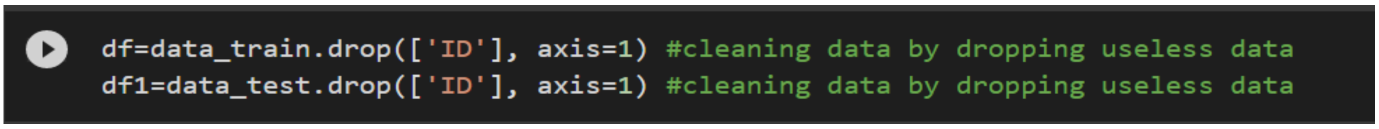


```
data_test.head()
```

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

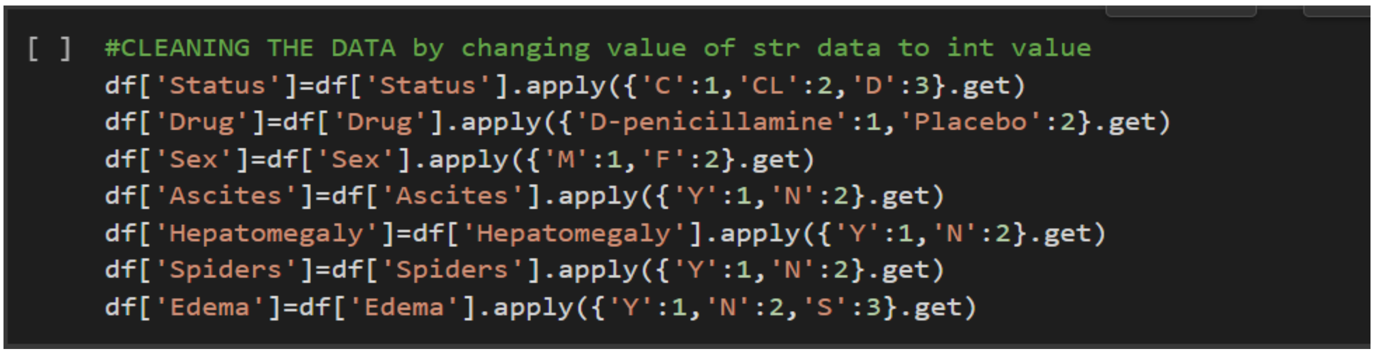
-Viewing test dataset

- Step-3: Cleaning the datasets by dropping useless column & changing any categorical values to numerical value. Also, by replacing null values into zero(0).



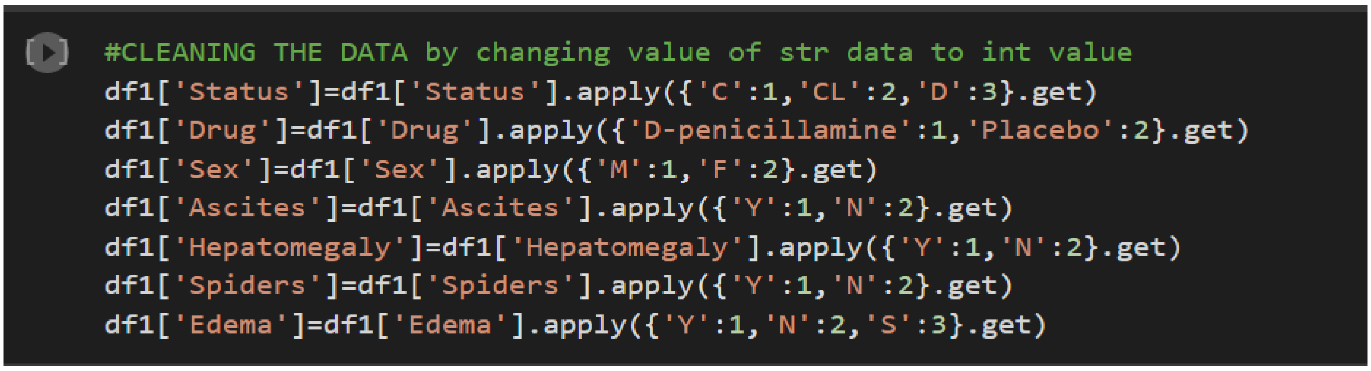
```
df=data_train.drop(['ID'], axis=1) #cleaning data by dropping useless data
df1=data_test.drop(['ID'], axis=1) #cleaning data by dropping useless data
```

-Dropping useless column.



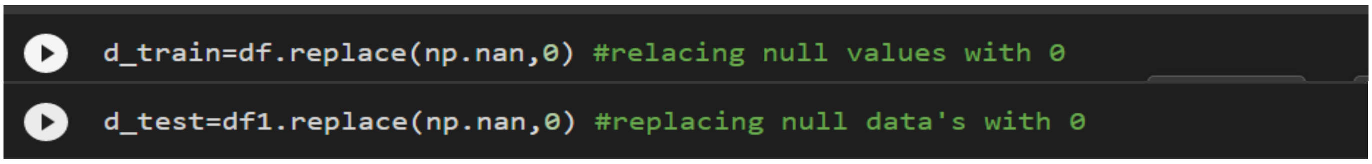
```
[ ]  #CLEANING THE DATA by changing value of str data to int value
     df['Status']=df['Status'].apply({'C':1,'CL':2,'D':3}.get)
     df['Drug']=df['Drug'].apply({'D-penicillamine':1,'Placebo':2}.get)
     df['Sex']=df['Sex'].apply({'M':1,'F':2}.get)
     df['Ascites']=df['Ascites'].apply({'Y':1,'N':2}.get)
     df['Hepatomegaly']=df['Hepatomegaly'].apply({'Y':1,'N':2}.get)
     df['Spiders']=df['Spiders'].apply({'Y':1,'N':2}.get)
     df['Edema']=df['Edema'].apply({'Y':1,'N':2,'S':3}.get)
```

-Changing the string values to integer values (train_dataset).



```
#CLEANING THE DATA by changing value of str data to int value
df1['Status']=df1['Status'].apply({'C':1,'CL':2,'D':3}.get)
df1['Drug']=df1['Drug'].apply({'D-penicillamine':1,'Placebo':2}.get)
df1['Sex']=df1['Sex'].apply({'M':1,'F':2}.get)
df1['Ascites']=df1['Ascites'].apply({'Y':1,'N':2}.get)
df1['Hepatomegaly']=df1['Hepatomegaly'].apply({'Y':1,'N':2}.get)
df1['Spiders']=df1['Spiders'].apply({'Y':1,'N':2}.get)
df1['Edema']=df1['Edema'].apply({'Y':1,'N':2,'S':3}.get)
```

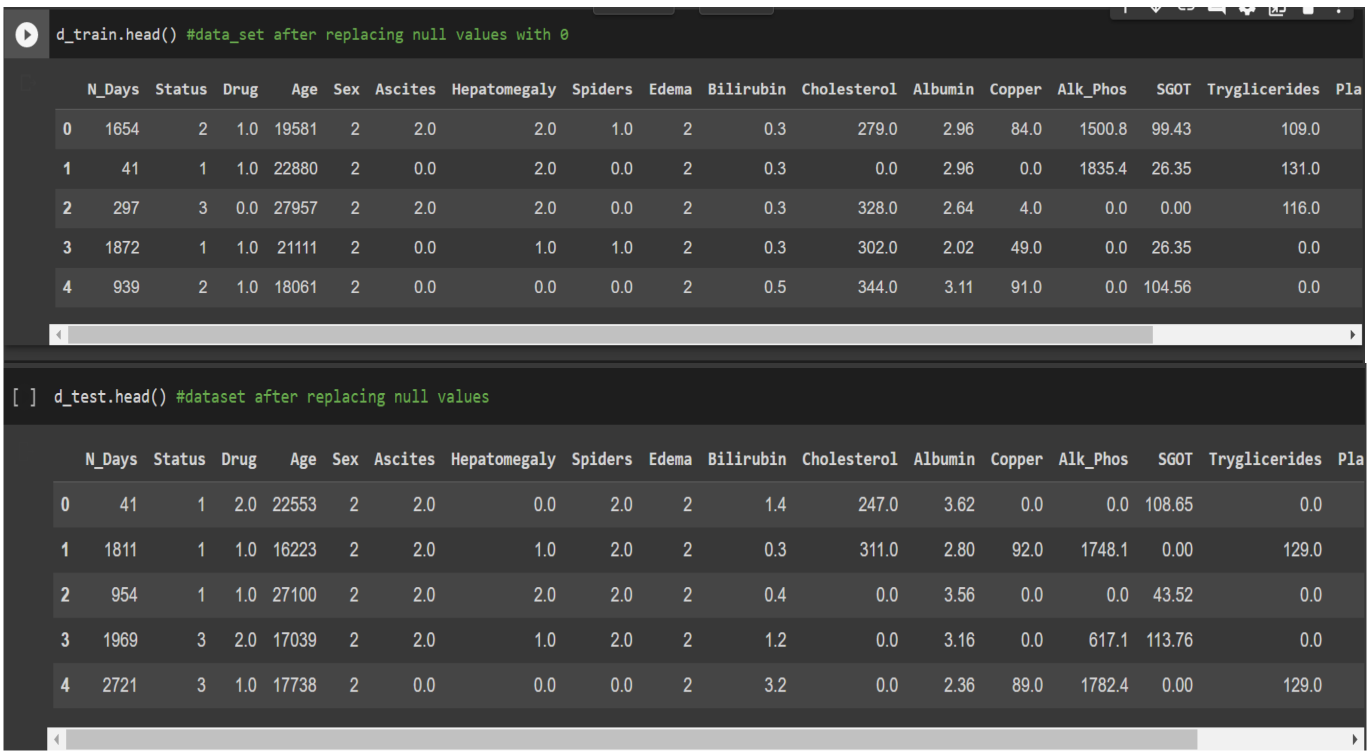
- Changing the string values to integer values (test_dataset).



```
d_train=df.replace(np.nan,0) #relacing null values with 0
```

```
d_test=df1.replace(np.nan,0) #replacing null data's with 0
```

-Replacing the null values with zero(0) in both the datasets.



```
d_train.head() #data_set after replacing null values with 0
```

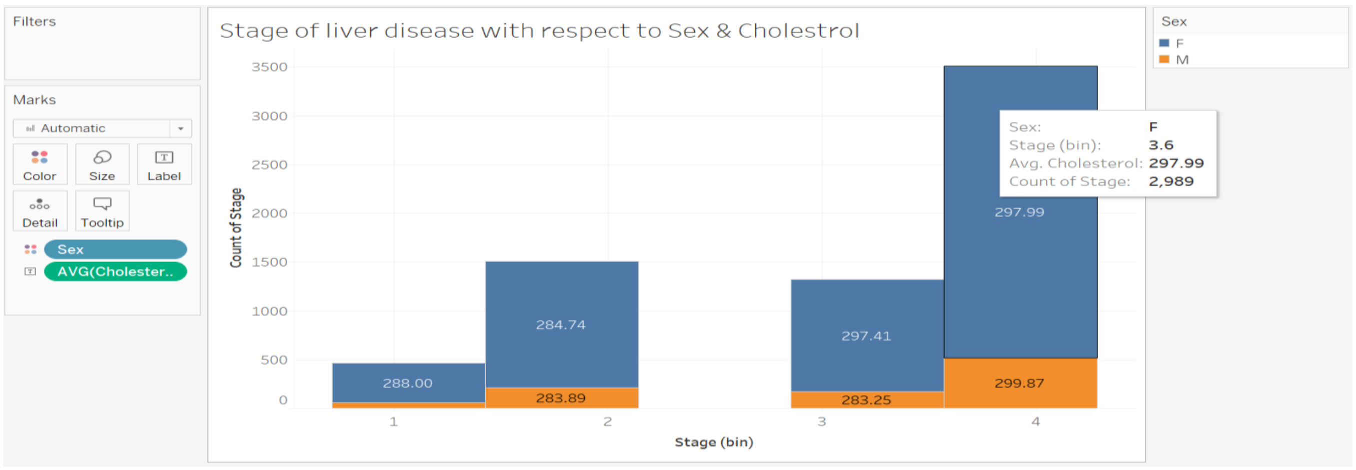
```
   N_Days  Status  Drug    Age  Sex  Ascites  Hepatomegaly  Spiders  Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  Tryglicerides  Pla
0    1654       2   1.0  19581    2      2.0           2.0      1.0      2        0.3        279.0     2.96    84.0    1500.8   99.43          109.0
1      41       1   1.0  22880    2      0.0           2.0      0.0      2        0.3          0.0     2.96     0.0    1835.4   26.35          131.0
2     297       3   0.0  27957    2      2.0           2.0      0.0      2        0.3        328.0     2.64     4.0       0.0    0.00          116.0
3    1872       1   1.0  21111    2      0.0           1.0      1.0      2        0.3        302.0     2.02    49.0       0.0   26.35            0.0
4     939       2   1.0  18061    2      0.0           0.0      0.0      2        0.5        344.0     3.11    91.0       0.0  104.56            0.0
```

```
[ ] d_test.head() #dataset after replacing null values
```

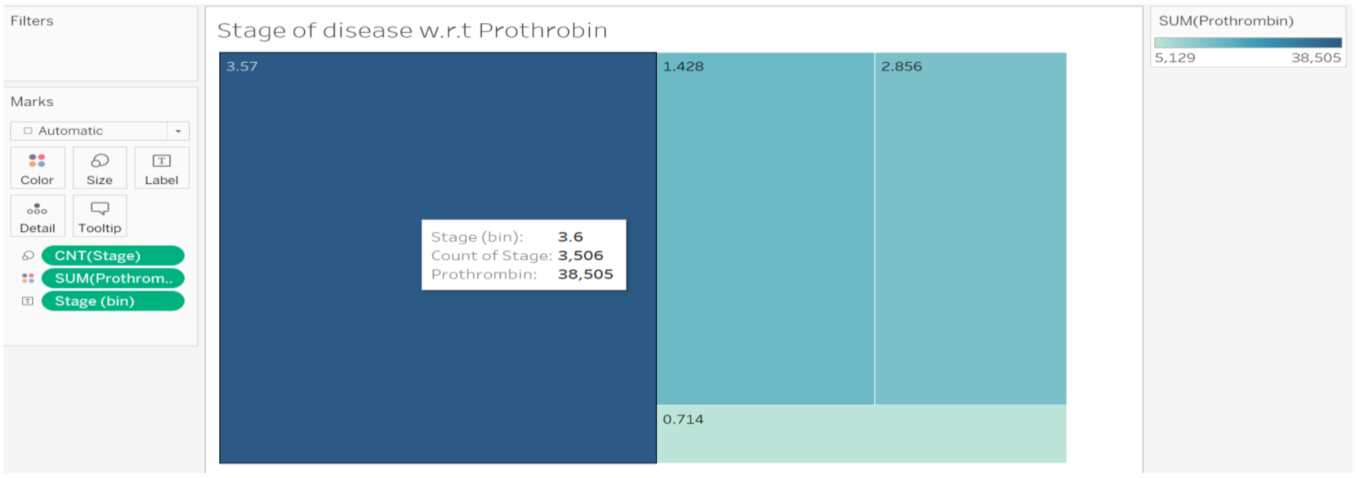
```
   N_Days  Status  Drug    Age  Sex  Ascites  Hepatomegaly  Spiders  Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  Tryglicerides  Pla
0      41       1   2.0  22553    2      2.0           0.0      2.0      2        1.4        247.0     3.62     0.0       0.0  108.65            0.0
1    1811       1   1.0  16223    2      2.0           1.0      2.0      2        0.3        311.0     2.80    92.0    1748.1    0.00          129.0
2     954       1   1.0  27100    2      2.0           2.0      2.0      2        0.4          0.0     3.56     0.0       0.0   43.52            0.0
3    1969       3   2.0  17039    2      2.0           1.0      2.0      2        1.2          0.0     3.16     0.0     617.1  113.76            0.0
4    2721       3   1.0  17738    2      0.0           0.0      0.0      2        3.2          0.0     2.36    89.0    1782.4    0.00          129.0
```

-Viewing the datasets after all necessary cleanings.

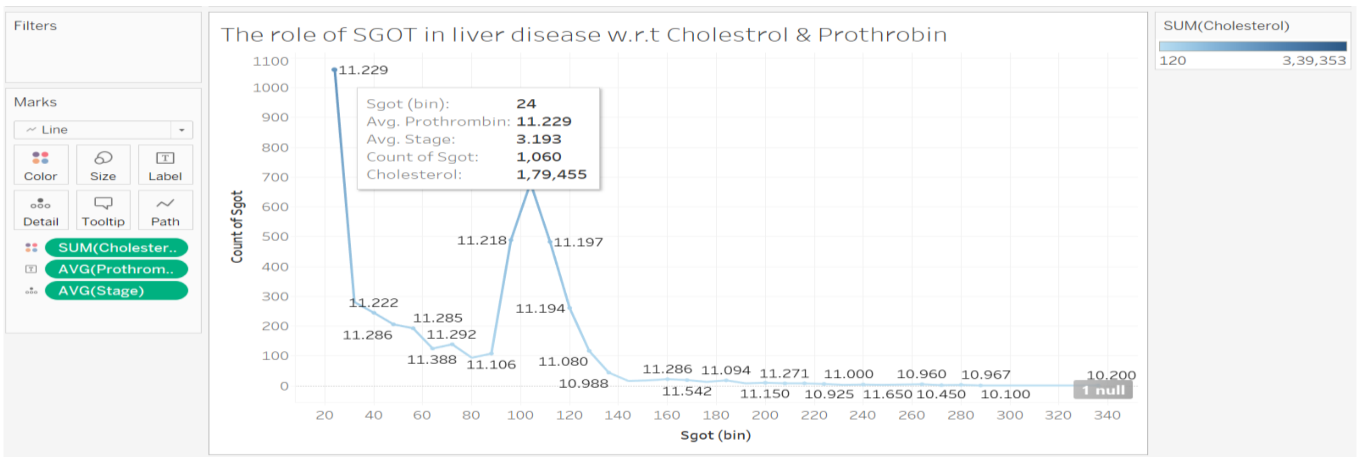
➢ <u>Step-4</u>: Visualising the liver disease data using Tableau to obtain some insights of liver damaging.



-From the above visualisation we can see that rate of liver disease is more in Females as compares to Males. Also, the cholesterol level in stage 4 cirrhosis people is more as compared to stage 1 people.

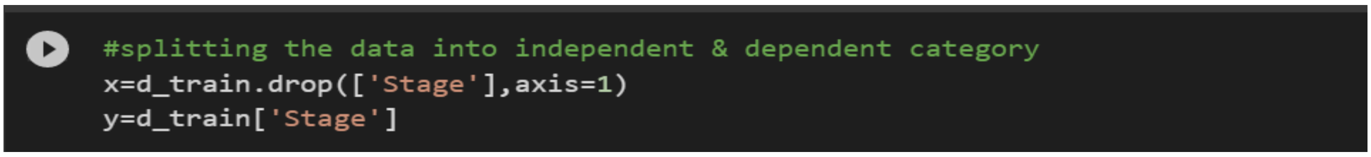


-Visualisation of stage of disease with respect to Prothrombin(coagulation of blood). From the above visualisation we can see that the rate of blood coagulation is more in stage 4 as compared to stage 1.



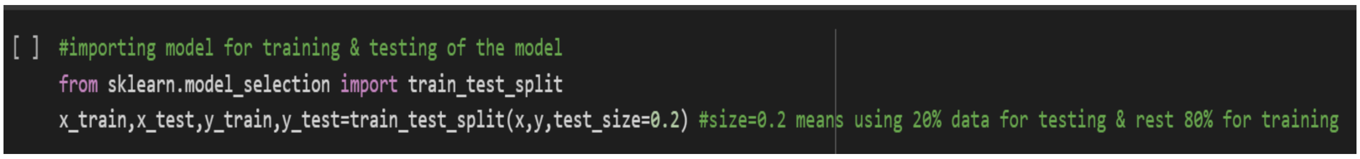
-From the above visualisation we can that the SGOT(an enzyme release by damaged liver) is occurring more in stage 4 patient which can further also results in cancer & other body disease. Due to more number of SGOT the cholesterol level & Prothrombin is also high.

➢ Step-5: Splitting the dataset into dependent & independent sets (taken only train dataset).



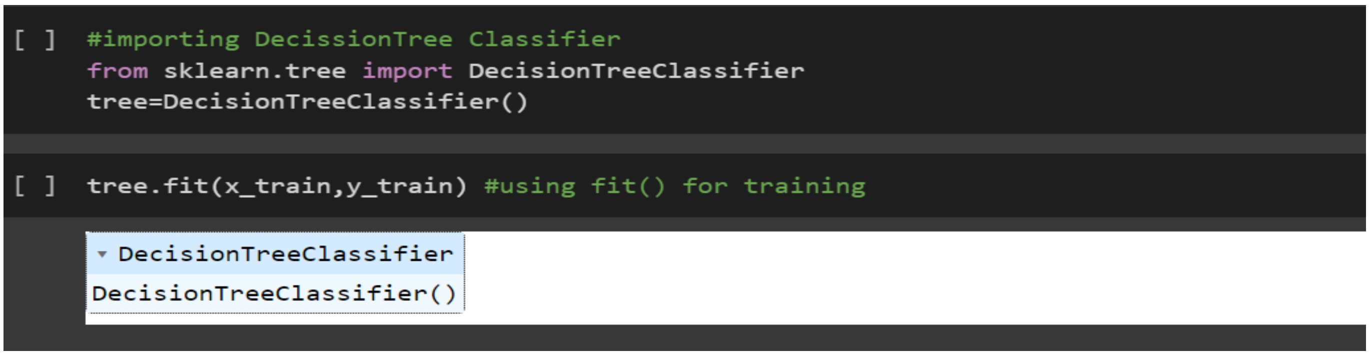
```
#splitting the data into independent & dependent category
x=d_train.drop(['Stage'],axis=1)
y=d_train['Stage']
```

➢ Step-6: Importing train_test_split from sklearn.model library for splitting the data into train and test sets.



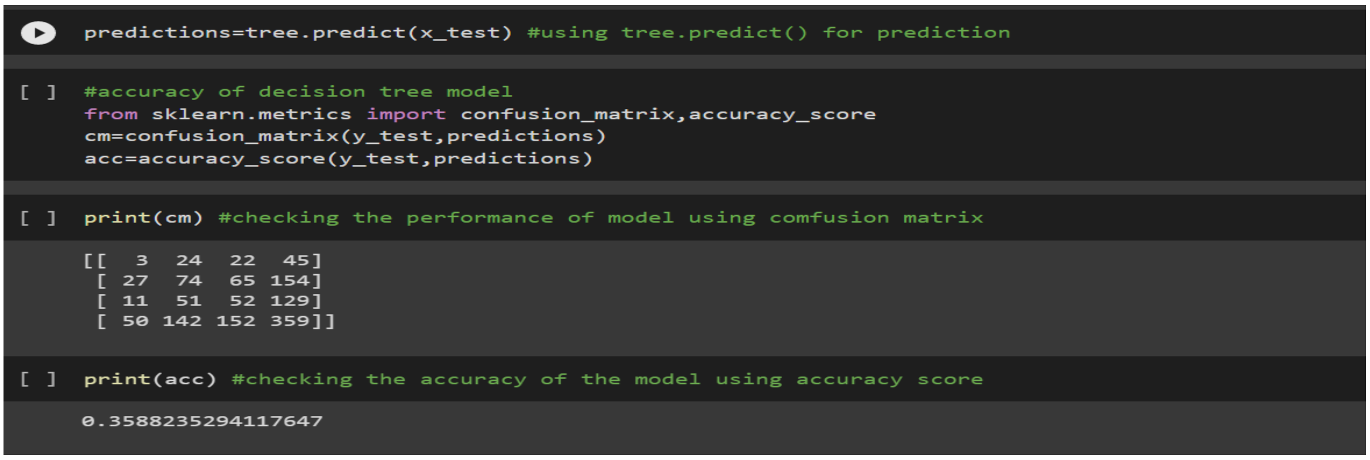
```
[ ] #importing model for training & testing of the model
    from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2) #size=0.2 means using 20% data for testing & rest 80% for training
```

➢ Step-7: Importing DecisionTreeClassifier from sklearn.model & then activating it by storing into the variable name tree. Then used tree.fit() to train the model by providing train & test sets as x & y.



```
[ ] #importing DecissionTree Classifier
    from sklearn.tree import DecisionTreeClassifier
    tree=DecisionTreeClassifier()

[ ] tree.fit(x_train,y_train) #using fit() for training

    ▾ DecisionTreeClassifier
    DecisionTreeClassifier()
```

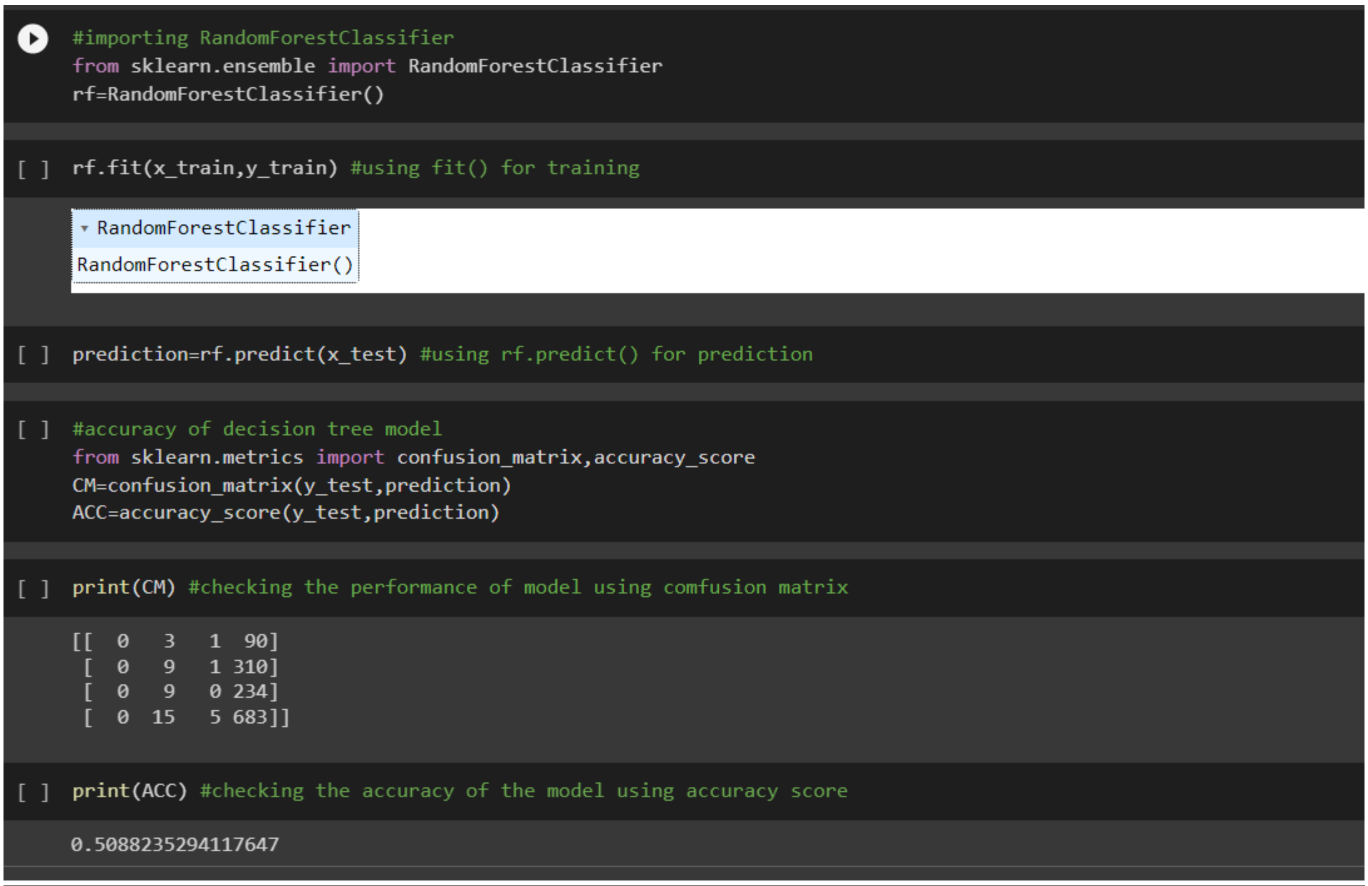
➢ Step-8: Predicting the trained model & the checked accuracy of the model using confusion_matrix & accuracy_score.



```
predictions=tree.predict(x_test) #using tree.predict() for prediction

[ ] #accuracy of decision tree model
    from sklearn.metrics import confusion_matrix,accuracy_score
    cm=confusion_matrix(y_test,predictions)
    acc=accuracy_score(y_test,predictions)

[ ] print(cm) #checking the performance of model using confusion matrix

    [[  3  24  22  45]
     [ 27  74  65 154]
     [ 11  51  52 129]
     [ 50 142 152 359]]

[ ] print(acc) #checking the accuracy of the model using accuracy score

    0.3588235294117647
```

-In the above model we can see that the accuracy is only 35% which not so good.

- <u>So I have also used RandomForestClassifier for obtaining better accuracy of the model.</u>



```
#importing RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
```

```
rf.fit(x_train,y_train) #using fit() for training
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
prediction=rf.predict(x_test) #using rf.predict() for prediction
```

```
#accuracy of decision tree model
from sklearn.metrics import confusion_matrix,accuracy_score
CM=confusion_matrix(y_test,prediction)
ACC=accuracy_score(y_test,prediction)
```

```
print(CM) #checking the performance of model using comfusion matrix
```

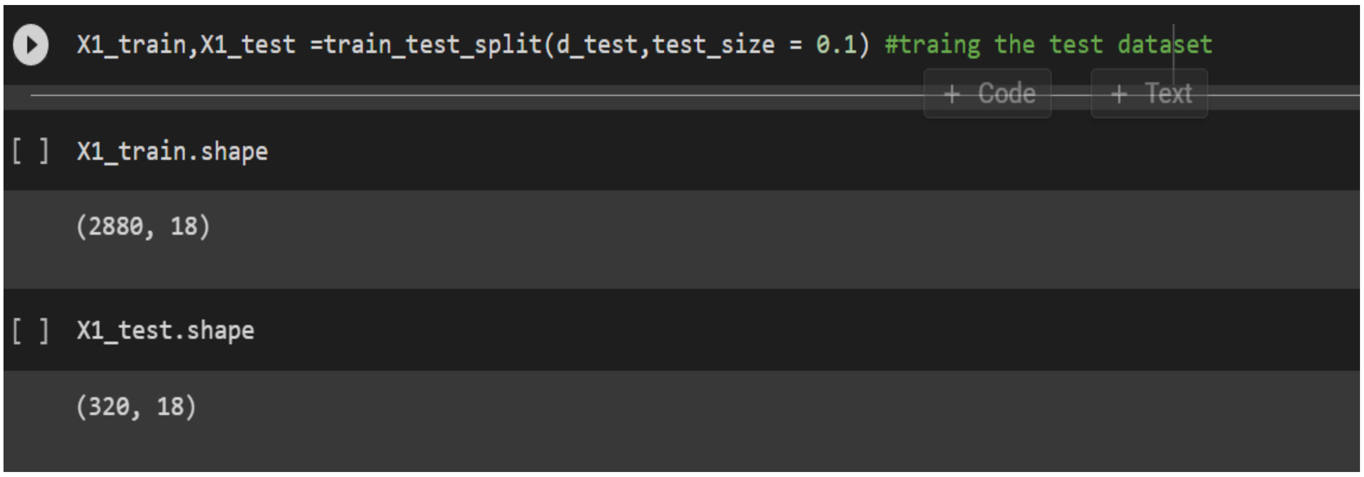
```
[[  0   3   1  90]
 [  0   9   1 310]
 [  0   9   0 234]
 [  0  15   5 683]]
```

```
print(ACC) #checking the accuracy of the model using accuracy score
```

```
0.5088235294117647
```

-The above accuracy of 50% was obtained using RandomForestClassifier which is relatively much better as compared to DecisionTree.

- <u>Step-8</u>: Recalling test_dataset as d_test & then splitting into test & train sets as X1_test & X1_train.



```
X1_train,X1_test =train_test_split(d_test,test_size = 0.1) #traing the test dataset
```

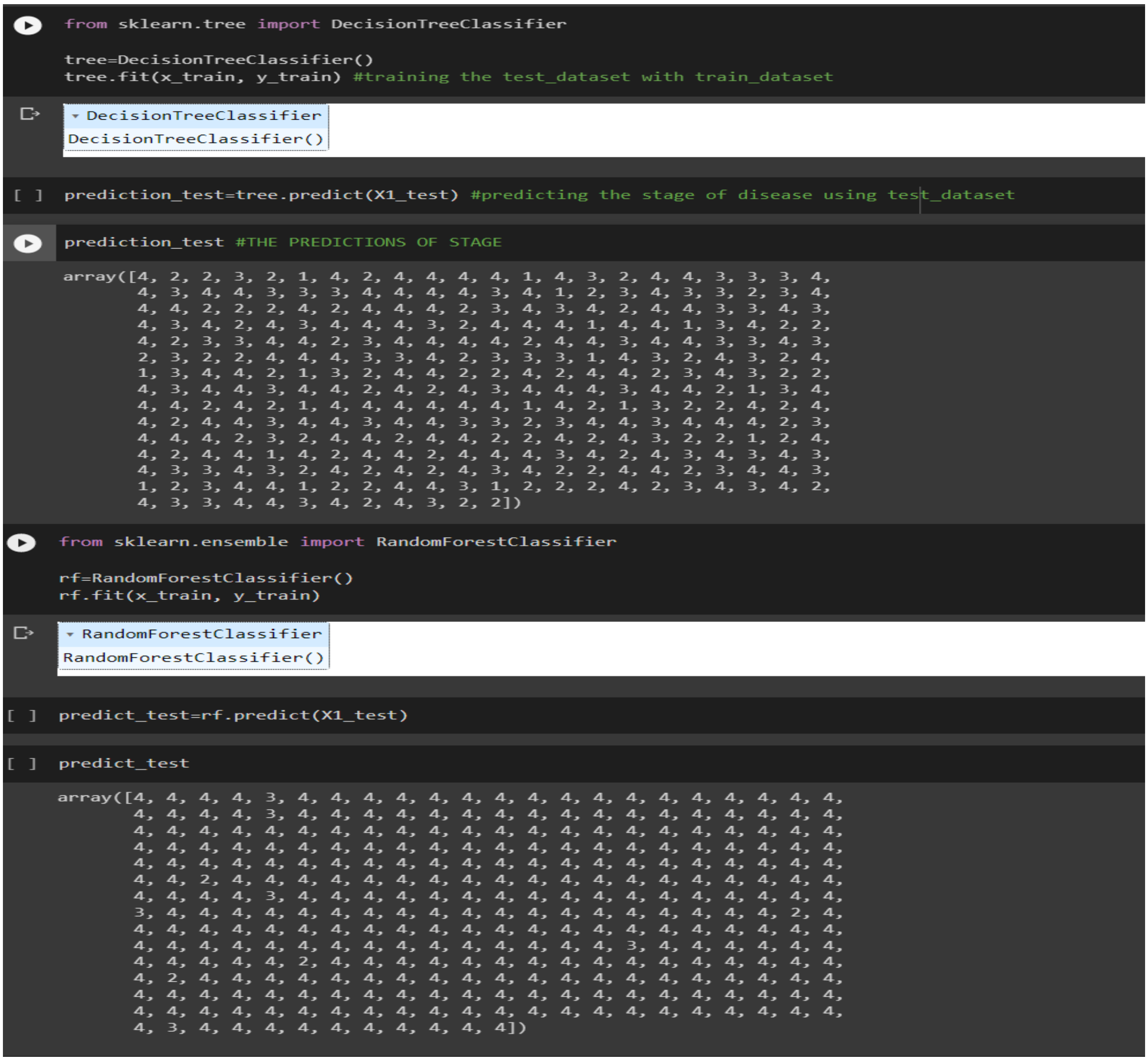
```
X1_train.shape
```

```
(2880, 18)
```

```
X1_test.shape
```

```
(320, 18)
```

➢ Step-9: Predicting the stage of liver disease of test_dataset using DecisionTreeClassifier & RandomForestClassifier.



```
from sklearn.tree import DecisionTreeClassifier

tree=DecisionTreeClassifier()
tree.fit(x_train, y_train) #training the test_dataset with train_dataset
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
prediction_test=tree.predict(X1_test) #predicting the stage of disease using test_dataset
```

```
prediction_test #THE PREDICTIONS OF STAGE
```

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

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()
rf.fit(x_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
predict_test=rf.predict(X1_test)
```

```
predict_test
```

```
array([4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 3, 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, 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, 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, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       3, 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, 4, 4, 3, 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, 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,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4])
```

-From the above two predictions we can consider RandomForestClassifier over DecisionTreeClassifier because the accuracy the was more in RandomForest as compared to DecisionTree.

➢ Conclusion: In the test_dataset where the stage of the liver disease needs to predicted, there we can use the predictions of RandomForestClassifier because it have the better accuracy of 50% whereas the accuracy rate of DecisionTree was in 30's.