ADVANCE PYTHON PROGRAMMING - CA2 - PROJECT

ATCHAYA.B.V - E8122019 and ROHITH.V - E8122018

DATASET : INDIAN LIVER PATIENT RECORD

PROBLEM STATEMENT : To determine which patients have liver disease and which ones do not by using the patient records.

QUESTION 1

1.Identify a research problem of your own interest. Formulate good research questions for meaningful data analysis and perform the following on the collected dataset.

A. Exploratory data analysis based on the research questions.

In [2]:

```
# Importing Liabraries
import pandas as pd
import numpy as np
```

In [3]:

```
# Reading the Data from the CSV file
liver=pd.read_csv("Z:\Term 2\Advance python programmig\Indian Liver Patient Dataset (ILPD).csv")
liver.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         583 non-null    int64  
 1   Gender                      583 non-null    object 
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64  
 5   Alamine_Aminotransferase    583 non-null    int64  
 6   Aspartate_Aminotransferase  583 non-null    int64  
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Dataset                     583 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

In [4]:

```
# Checking if there is some null values or not
liver.isnull().sum()
```

Out[4]:

```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  4
Dataset                     0
dtype: int64
```

In [5]:

```
# Total number of columns in the dataset
liver.columns
```

Out[5]:

```
Index(['Age', 'Gender', 'Total_Bilirubin', 'Direct_Bilirubin',
       'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
       'Aspartate_Aminotransferase', 'Total_Protiens', 'Albumin',
       'Albumin_and_Globulin_Ratio', 'Dataset'],
      dtype='object')
```

B. Statistical analysis on the significant features.

In [6]:

```
# Viewing the first five entries
liver.head()
```

Out[6]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	A
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	

In [7]:

```
# To know more about the dataset
liver.describe()
```

Out[7]:

irect_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Da
583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.0(
1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	1.2(
2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.4(
0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.0(
0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	1.0(
0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.0(
1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000	2.0(
19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.0(

In [8]:

```
pd.get_dummies(liver['Gender']).head()
```

Out[8]:

	Female	Male
0	1	0
1	0	1
2	0	1
3	0	1
4	0	1

In [9]:

```
# Filling the NA values with mean value
liver["Albumin_and_Globulin_Ratio"]=liver.Albumin_and_Globulin_Ratio.fillna(liver["Albumin_and_Globulin_Ratio"].mean())
```

In [10]:

```
# Viewing the data after filling null values
liver.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         583 non-null    int64  
 1   Gender                      583 non-null    object 
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64  
 5   Alamine_Aminotransferase    583 non-null    int64  
 6   Aspartate_Aminotransferase  583 non-null    int64  
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  583 non-null    float64
 10  Dataset                     583 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

In [11]:

```
liver.isnull().sum()
```

Out[11]:

```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  0
Dataset                     0
dtype: int64
```

In [12]:

```
liver.describe()
```

Out[12]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumi
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.00000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.14185
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.79551
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.90000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.60000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.10000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.80000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.50000

C. Extract the features of interest based on relevant feature engineering techniques and visualize the same statistically.

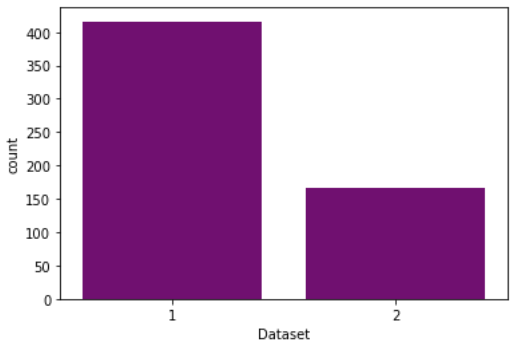
In [13]:

```
# Import libraries for Data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly as px
import cufflinks as cf
```

In [14]:

```
# Plotting the Number of patients with liver disease vs Number of patients with no liver disease
sns.countplot(data=liver,x="Dataset",color="purple")
diagnosed,non_diagnosed=liver["Dataset"].value_counts()
print("Patients record who diagnosed with liver disease:",diagnosed)
print("Patients record who not diagnosed with liver disease:",non_diagnosed)
```

```
Patients record who diagnosed with liver disease: 416
Patients record who not diagnosed with liver disease: 167
```



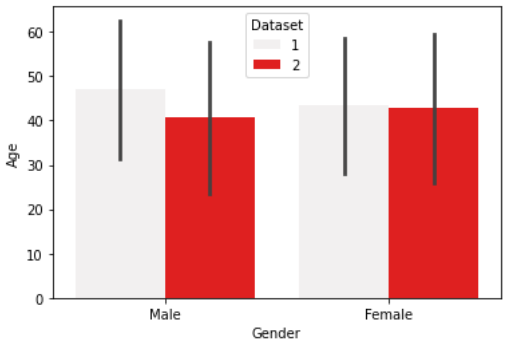
We can clearly see in the output as well as in the graph that, it is an imbalanced dataset, any patients diagnosed (416 RECORDS) with liver disease are higher compared to the ones who are not diagnosed (167 RECORDS).

In [15]:

```
# Plotting Gender against Age with respect to target variable
sns.barplot(x="Gender",y="Age",data=liver,hue="Dataset",order=["Male","Female"],ci="sd",color="red")
```

Out[15]:

```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```



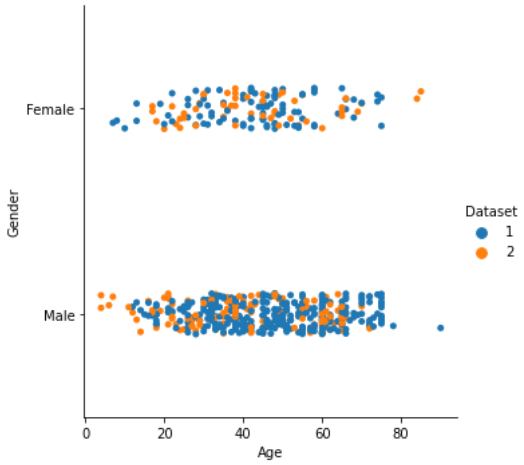
We can clearly see in the output as well as in the graph that, number of patient suffering from liver disease are higher in males than in females.

In [16]:

```
# Plotting patient Age vs Gender
sns.catplot(data=liver,x="Age",y="Gender",hue="Dataset")
```

Out[16]:

```
<seaborn.axisgrid.FacetGrid at 0x1c6ade90f70>
```



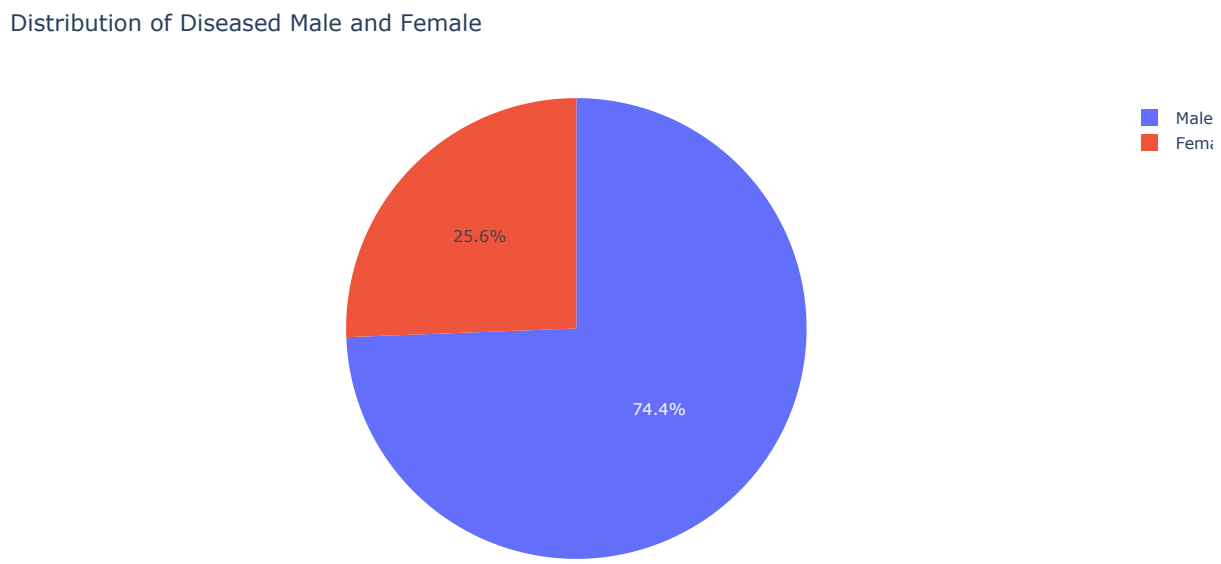
Here is scatter plot that shows, males are at higher risk of chronic liver diseases as compare to females.

In [17]:

```
# Importing plotly and Cufflinks for interactive plot
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
import plotly.express as px
import cufflinks as cf
```

In [18]:

```
fig=px.pie(liver,"Gender",values="Dataset",title="Distribution of Diseased Male and Female")
fig.show()
```



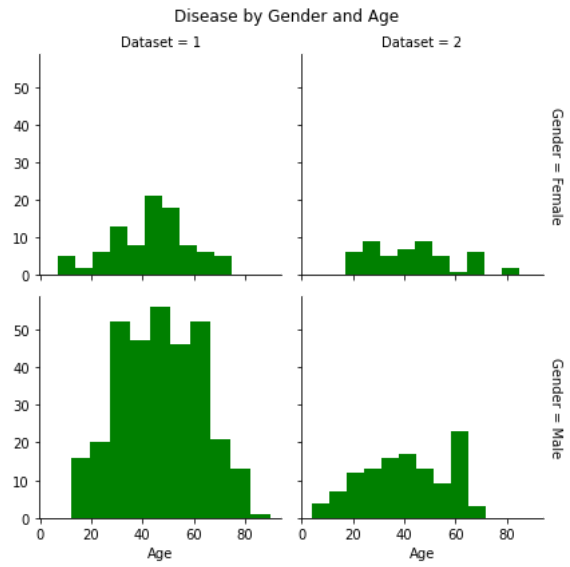
Here is an interactive plot() that shows, males are mostly affected by chronic liver diseases as compare to females. The proportion of males who are diseased is 74% and females who are diseased is 26%.

In [19]:

```
# Plotting Age vs Gender
g = sns.FacetGrid(liver, col="Dataset", row="Gender", margin_titles=True)
g.map(plt.hist,"Age",color="green")
plt.subplots_adjust(top=0.9)
g.fig.suptitle("Disease by Gender and Age")
```

Out[19]:

```
Text(0.5, 0.98, 'Disease by Gender and Age')
```



Before, we have seen some of the visualization based on gender (separately), here in this FacetGrid plot we can track cases according to both Gender and Age

In [20]:

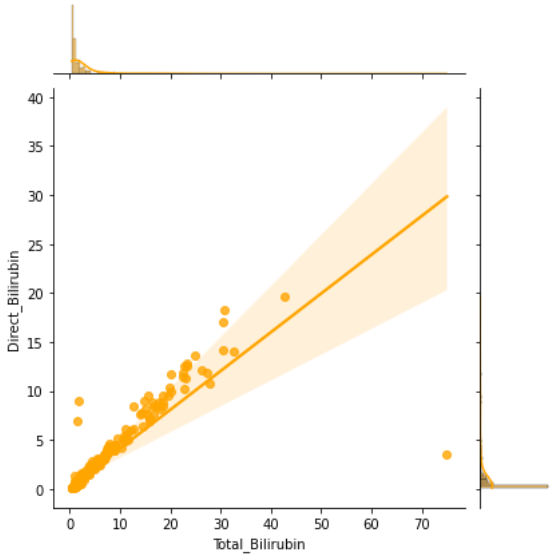
```
# Plotting Total_Bilirubin vs Direct_Bilirubin
sns.jointplot("Total_Bilirubin", "Direct_Bilirubin", data=liver, kind="reg",color="orange")
```

```
C:\Users\rohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, a
nd passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

Out[20]:

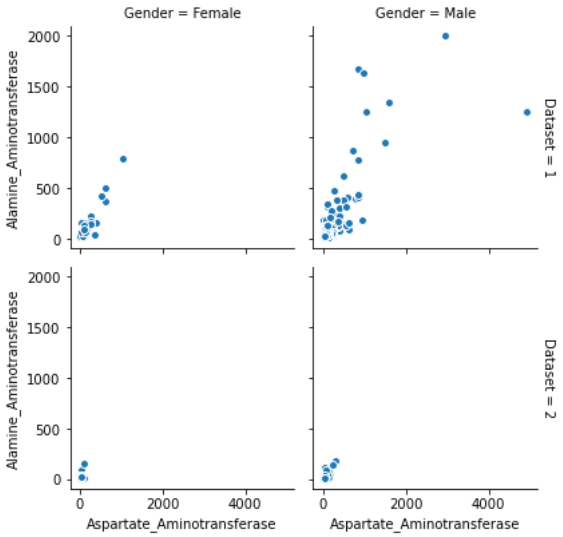
```
<seaborn.axisgrid.JointGrid at 0x1c6b376ddf0>
```



Here in this plot(), we have plotted Total_bilirubin vs Direct_Bilrubin and got the insight that both of the features have a direct relationship with each other.

In [21]:

```
# Plotting Gender along with Aspartate Aminotransferase, Alamine Aminotransferase
g = sns.FacetGrid(liver, col="Gender", row="Dataset", margin_titles=True)
g.map(plt.scatter,"Aspartate_Aminotransferase", "Alamine_Aminotransferase",  edgecolor="w")
plt.subplots_adjust(top=0.9)
```



In this FacetGrid plot we are plotting two significant features(Alamine and Aspartate -Aminotransferase) along with Gender as a form of hue and it clearly shows that males are highly effective concerning these two features the most.

In [22]:

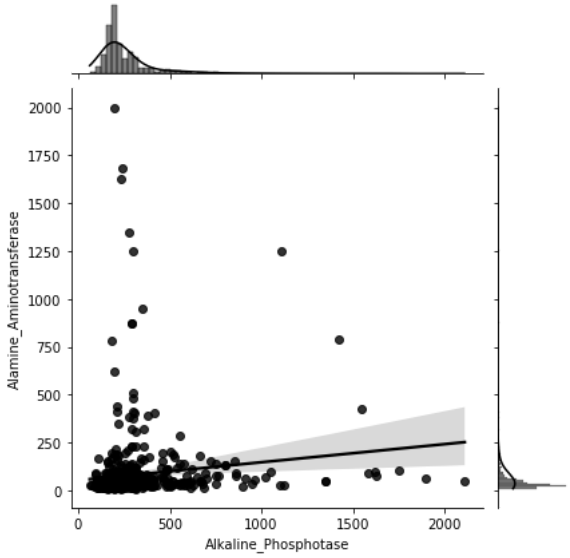
```
# Plotting Alkaline_Phosphotase vs Alamine_Aminotransferase
sns.jointplot("Alkaline_Phosphotase", "Alamine_Aminotransferase", data=liver, kind="reg",color="black")
```

```
C:\Users\rohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, a
nd passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

Out[22]:

```
<seaborn.axisgrid.JointGrid at 0x1c6b4996370>
```



In this plot, we can see that Alkaline _Phosphotase and Alamine_Aminotransferase do have a direct regressive relationship but we can also note that there are a bit “outliers” too from the side of Alamine_Aminotransferase.

In [23]:

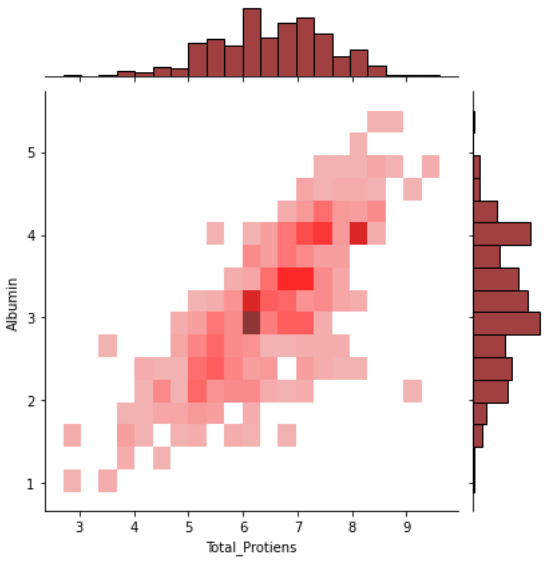
```
# Plotting Total_Protiens vs Albumin
sns.jointplot("Total_Protiens", "Albumin", data=liver, kind="hist",color="maroon")
```

```
C:\Users\rohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, a
nd passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

Out[23]:

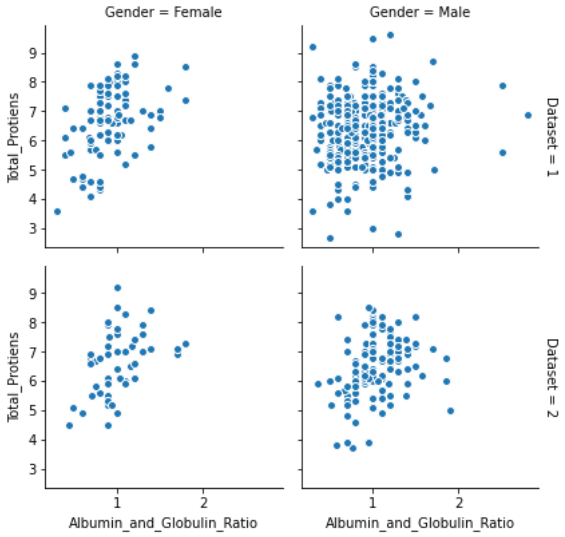
```
<seaborn.axisgrid.JointGrid at 0x1c6b1d4fdc0>
```



Now with the help of the above plot we can find out that, Total_protiens and Albumin features are in positive regressive nature, with some outliers.

In [24]:

```
# Plotting Gender along with Albumin and Globulin Ratio and Total Protiens
g = sns.FacetGrid(liver, col="Gender", row="Dataset", margin_titles=True)
g.map(plt.scatter,"Albumin_and_Globulin_Ratio", "Total_Protiens",  edgecolor="w")
plt.subplots_adjust(top=0.9)
```



Here in this plot, we are trying to show that though Albumin and Globulin Ratio has regressive datapoints yet the most crowded being the male region i.e. they are at high risk in these features too.

In [25]:

```
X=liver.drop(["Dataset"],axis=1)
X.head()
```

Out[25]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	A
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	

In [26]:

```
y=liver["Dataset"]
```

In [27]:

```
#Correlation between all the features
corr=X.corr()
corr
```

Out[27]:

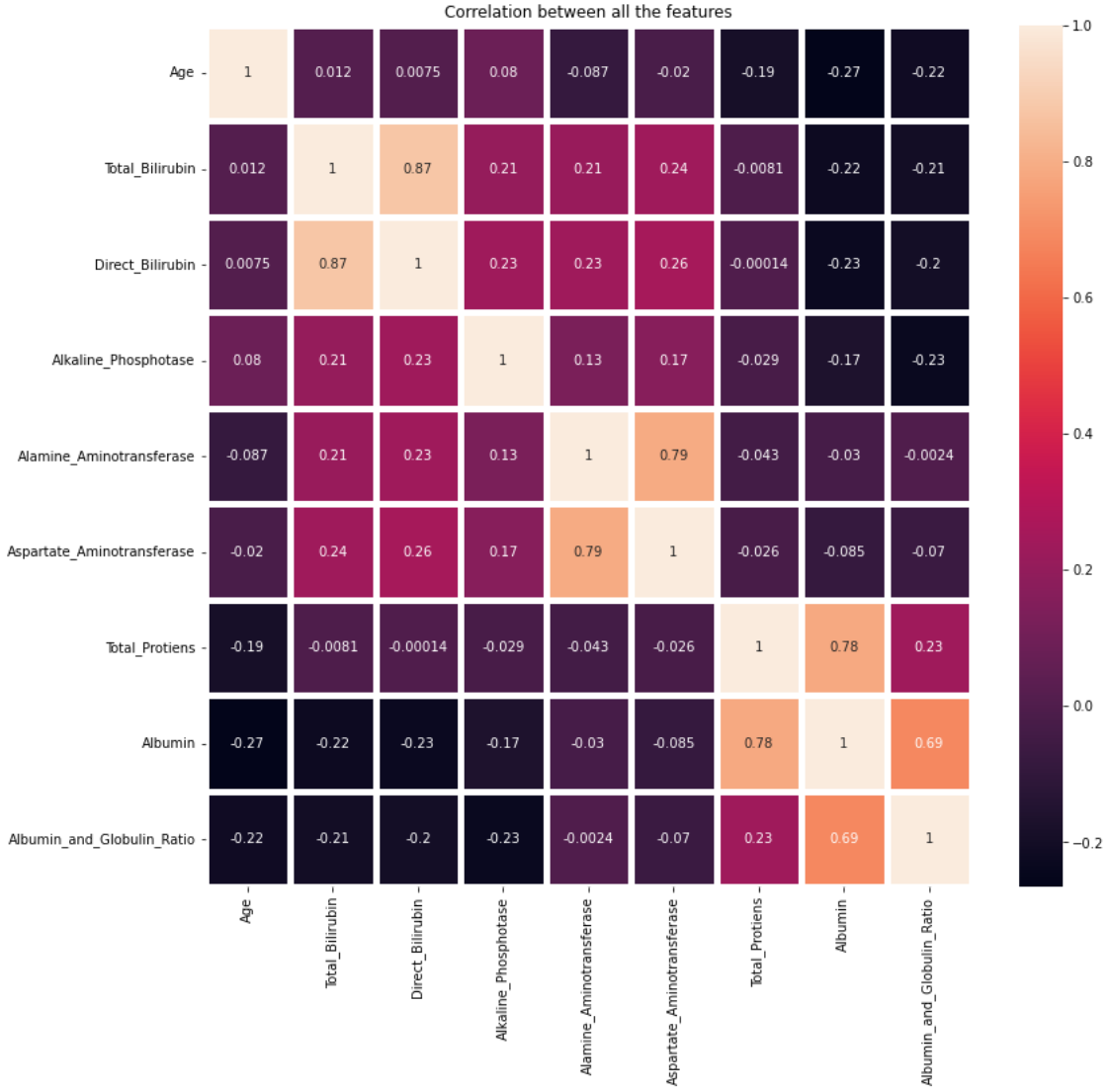
al_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_
0.011763	0.007529	0.080425	-0.086883	-0.019910	-0.187461	-0.265924	-0.21
1.000000	0.874618	0.206669	0.214065	0.237831	-0.008099	-0.222250	-0.20
0.874618	1.000000	0.234939	0.233894	0.257544	-0.000139	-0.228531	-0.20
0.206669	0.234939	1.000000	0.125680	0.167196	-0.028514	-0.165453	-0.23
0.214065	0.233894	0.125680	1.000000	0.791966	-0.042518	-0.029742	-0.00
0.237831	0.257544	0.167196	0.791966	1.000000	-0.025645	-0.085290	-0.07
-0.008099	-0.000139	-0.028514	-0.042518	-0.025645	1.000000	0.784053	0.23
-0.222250	-0.228531	-0.165453	-0.029742	-0.085290	0.784053	1.000000	0.68
-0.206159	-0.200004	-0.233960	-0.002374	-0.070024	0.233904	0.686322	1.00

In [28]:

```
plt.figure(figsize=(13,12))
sns.heatmap(corr,cbar=True,annot=True,linewidth=5)
plt.title("Correlation between all the features")
```

Out[28]:

```
Text(0.5, 1.0, 'Correlation between all the features')
```



The 1.00 part of the heatmap signifies that data is positively correlated.The correlation between the (Total_Bilirubin and Direct_Bilirubin),(Alamine_Aminotransferase and Aspartate_Aminotransferase),(Total_Protiens and Albumin),(Albumin and Albumin_and_Globulin_Ratio) columns are very much good with positive correlation whereas, correlation between (Age and albumin) ,(Albumin_and_Globulin_Ratio and Age) and some are not that much correlated with each other.

In [29]:

```
#converting the values of gender to zeros and ones
def converter(Gender):
    if Gender=='Male':
        return 0
    else:
        return 1
```

In [30]:

```
#converted male as 0 and female as 1
liver['Gender']=liver['Gender'].apply(converter)
```

In [31]:

```
liver.head()
```

Out[31]:

bin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Da
0.7	0.1	187	16	18	6.8	3.3	0.90	
0.9	5.5	699	64	100	7.5	3.2	0.74	
7.3	4.1	490	60	68	7.0	3.3	0.89	
1.0	0.4	182	14	20	6.8	3.4	1.00	
3.9	2.0	195	27	59	7.3	2.4	0.40	

2.For the problem statement in Question 1, build the relevant machine learning model(any 3) and compare its performance metrics. Justify the performance of the best model.

In [32]:

```
#Splitting the data into Train and Test
X=liver.drop(["Dataset"],axis=1)
y=liver["Dataset"]
```

Since the target variable ("Dataset") is categorical where it is categorized into diseased as 1 and non diseased as 2 records of patients it is a CLASSIFICATION MODEL

LOGISTIC REGRESSION

In [33]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
```

In [34]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
```

In [35]:

```
X_train.shape
```

Out[35]:

```
(390, 10)
```

In [36]:

```
X_test.shape
```

Out[36]:

```
(193, 10)
```

In [37]:

```
y_train.shape
```

Out[37]:

```
(390,)
```

In [38]:

```
y_test.shape
```

Out[38]:

```
(193,)
```

In [39]:

```
log=LogisticRegression()
```

In [40]:

```
# Train the model using the training sets
log.fit(X_train,y_train)
```

```
C:\Users\rohit\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/
linear_model.html#logistic-regression)
```

Out[40]:

```
LogisticRegression()
```

In [41]:

```
log_pred=log.predict(X_test)
```

In [42]:

```
log_pred
```

Out[42]:

```
array([2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1], dtype=int64)
```

In [43]:

```
print(classification_report(y_test,log_pred))
```

```
              precision    recall  f1-score   support

           1       0.75      0.93      0.83       141
           2       0.44      0.15      0.23        52

    accuracy                           0.72       193
   macro avg       0.60      0.54      0.53       193
weighted avg       0.67      0.72      0.67       193
```

The accuracy score of logistic regression is 72%

In [44]:

```
confusion_matrix(y_test,log_pred)
```

Out[44]:

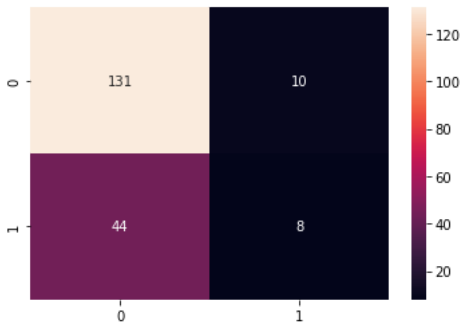
```
array([[131,  10],
       [ 44,   8]], dtype=int64)
```

In [45]:

```
#Confision Matrix
sns.heatmap(confusion_matrix(y_test,log_pred),annot=True,fmt="d")
```

Out[45]:

```
<AxesSubplot:>
```



True positive =131 patients are truly diseased, True negative=8 patients are truly not diseased, False positive=44 patients are wrongly recorded as diseased, False negative=10 patients are wrongly recorded as non diseased

DECISION TREE

In [46]:

```
x=liver.drop(["Dataset"],axis=1)
Y=liver["Dataset"]
```

In [47]:

```
x_train,x_test,Y_train,Y_test=train_test_split(x,Y,test_size=0.33,random_state=42)
```

In [48]:

```
x_train.shape
```

Out[48]:

```
(390, 10)
```

In [49]:

```
x_test.shape
```

Out[49]:

```
(193, 10)
```

In [50]:

```
Y_train.shape
```

Out[50]:

```
(390,)
```

In [51]:

```
Y_test.shape
```

Out[51]:

```
(193,)
```

In [52]:

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,Y_train)
deci=dt.predict(x_test)
deci
```

Out[52]:

```
array([1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 1,
       2, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 2,
       1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1,
       2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2], dtype=int64)
```

In [53]:

```
print(classification_report(Y_test,deci))
```

```
              precision    recall  f1-score   support

           1       0.82      0.79      0.81       141
           2       0.48      0.52      0.50        52

    accuracy                           0.72       193
   macro avg       0.65      0.66      0.65       193
weighted avg       0.73      0.72      0.72       193
```

```
The accuracy score of Decision Tree is 72%
```

In [54]:

```
confusion_matrix(Y_test,deci)
```

Out[54]:

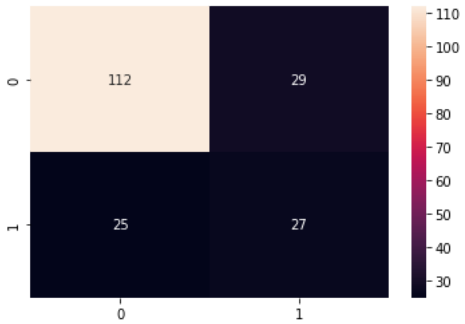
```
array([[112,  29],
       [ 25,  27]], dtype=int64)
```

In [55]:

```
sns.heatmap(confusion_matrix(Y_test,deci),annot=True,fmt="d")
```

Out[55]:

```
<AxesSubplot:>
```



True positive=114,True Negative=27,False positive=25,False negative=27

In [56]:

```
from sklearn import tree
```

In [57]:

```
tree.plot_tree(dt,feature_names=x_train.columns)
```

Out[57]:

```
[Text(0.6129032258064516, 0.9736842105263158, 'Direct_Bilirubin <= 0.65\ngini = 0.416\nsamples = 390\nvalue = [275, 11
5]'),
 Text(0.38911290322580644, 0.9210526315789473, 'Alkaline_Phosphotase <= 212.0\ngini = 0.481\nsamples = 248\nvalue = [148,
100]'),
 Text(0.21370967741935484, 0.868421052631579, 'Alamine_Aminotransferase <= 19.5\ngini = 0.5\nsamples = 167\nvalue = [85,
82]'),
 Text(0.0967741935483871, 0.8157894736842105, 'Alamine_Aminotransferase <= 15.5\ngini = 0.404\nsamples = 32\nvalue = [9,
23]'),
 Text(0.06451612903225806, 0.7631578947368421, 'Albumin_and_Globulin_Ratio <= 1.15\ngini = 0.498\nsamples = 17\nvalue =
[8, 9]'),
 Text(0.04838709677419355, 0.7105263157894737, 'Aspartate_Aminotransferase <= 11.5\ngini = 0.459\nsamples = 14\nvalue =
[5, 9]'),
 Text(0.03225806451612903, 0.6578947368421053, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.06451612903225806, 0.6578947368421053, 'Albumin <= 3.45\ngini = 0.426\nsamples = 13\nvalue = [4, 9]'),
 Text(0.04838709677419355, 0.6052631578947368, 'Albumin <= 3.35\ngini = 0.494\nsamples = 9\nvalue = [4, 5]'),
 Text(0.03225806451612903, 0.5526315789473685, 'Aspartate_Aminotransferase <= 31.0\ngini = 0.278\nsamples = 6\nvalue =
[1, 5]'),
 Text(0.016129032258064516, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [0, 5]'),
```

SUPPORT VECTOR MACHINE

In [58]:

```
from sklearn.svm import SVC
svc=SVC()
svc.fit(X_train,y_train)
```

Out[58]:

```
SVC()
```

In [59]:

```
svc_pred=svc.predict(X_test)
```

In [60]:

```
confusion_matrix(y_test,svc_pred)
```

Out[60]:

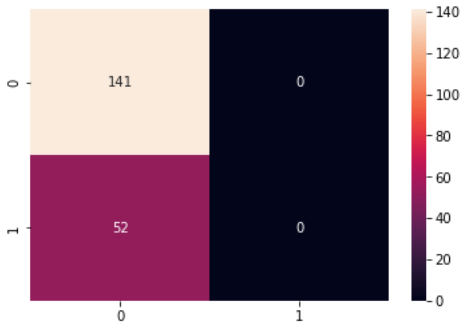
```
array([[141,   0],
       [ 52,   0]], dtype=int64)
```

In [61]:

```
sns.heatmap(confusion_matrix(y_test,svc_pred),annot=True,fmt="d")
```

Out[61]:

<AxesSubplot:>



True positive=141,True negative and False negative=0,True positive=52

In [62]:

```
print(classification_report(y_test,svc_pred))
```

```
              precision    recall  f1-score   support

           1       0.73      1.00      0.84       141
           2       0.00      0.00      0.00        52

    accuracy                           0.73       193
   macro avg       0.37      0.50      0.42       193
weighted avg       0.53      0.73      0.62       193


C:\Users\rohit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.

C:\Users\rohit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.

C:\Users\rohit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
```

The accuracy score of Support vector machine is 73%

The accuracy score of Logistic regression = 72% The accuracy score of Decision tree = 72% The accuracy score of Support vector machine = 73%

Since the accuracy of SUPPORT VECTOR MACHINE (SVM) model is (73%) which is higher than the other two models it is said to be the best model among the three machine learning algorithms

QUESTION 3

3.A. Apply AutoML concept on the dataset based on the research problem and provide the leaderboard of the best algorithms with accuracy score.Justify your results.

B. Perform the following mathematical operations using tensorflow. i) Addition ii) Subtraction iii) Mutliplication

In [63]:

```
#pip install tensorflow
```

In [64]:

```
import tensorflow as tf
```

In [65]:

```
with tf.compat.v1.Session() as sess:
    a=tf.constant(50)
    b=tf.constant(12)
    c=tf.add(a,b)
    ADDITION=sess.run(c)
    d=tf.subtract(a,b)
    SUBTRACTION=sess.run(d)
    s=tf.multiply(a,b)
    MULTIPLICATION=sess.run(s)
    print("The addition of the given 2 values is:",ADDITION)
    print("The subtraction of the given 2 values is:",SUBTRACTION)
    print("The multiplication of the given 2 values is:",MULTIPLICATION)
```

```
The addition of the given 2 values is: 62
The subtraction of the given 2 values is: 38
The multiplication of the given 2 values is: 600
```