**Liver Disease Prediction**

**Content**

This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age "90".

Columns:

* Age of the patient
* Gender of the patient
* Total Bilirubin
* Direct Bilirubin
* Alkaline Phosphotase
* Alamine Aminotransferase
* Aspartate Aminotransferase
* Total Protiens
* Albumin
* Albumin and Globulin Ratio
* Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

In [1]:

*# Importing Libraries:*

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

In [2]:

*# for displaying all feature from dataset:*

pd**.**pandas**.**set\_option('display.max\_columns', **None**)

In [3]:

*# Reading Dataset:*

dataset **=** pd**.**read\_csv("C:/Users/91739/PycharmProjects/Liver-Disease-Prediction-Project-main/Dataset/Liver\_data.csv")

*# Top 5 records:*

dataset**.**head()

Out[3]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Direct\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 65 | Female | 0.7 | 0.1 | 187 | 16 | 18 | 6.8 | 3.3 | 0.90 | 1 |
| **1** | 62 | Male | 10.9 | 5.5 | 699 | 64 | 100 | 7.5 | 3.2 | 0.74 | 1 |
| **2** | 62 | Male | 7.3 | 4.1 | 490 | 60 | 68 | 7.0 | 3.3 | 0.89 | 1 |
| **3** | 58 | Male | 1.0 | 0.4 | 182 | 14 | 20 | 6.8 | 3.4 | 1.00 | 1 |
| **4** | 72 | Male | 3.9 | 2.0 | 195 | 27 | 59 | 7.3 | 2.4 | 0.40 | 1 |

In [4]:

*# Last 5 records:*

dataset**.**tail()

Out[4]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Direct\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **578** | 60 | Male | 0.5 | 0.1 | 500 | 20 | 34 | 5.9 | 1.6 | 0.37 | 2 |
| **579** | 40 | Male | 0.6 | 0.1 | 98 | 35 | 31 | 6.0 | 3.2 | 1.10 | 1 |
| **580** | 52 | Male | 0.8 | 0.2 | 245 | 48 | 49 | 6.4 | 3.2 | 1.00 | 1 |
| **581** | 31 | Male | 1.3 | 0.5 | 184 | 29 | 32 | 6.8 | 3.4 | 1.00 | 1 |
| **582** | 38 | Male | 1.0 | 0.3 | 216 | 21 | 24 | 7.3 | 4.4 | 1.50 | 2 |

In [5]:

*# Shape of dataset:*

dataset**.**shape

Out[5]:

(583, 11)

In [6]:

*# Cheaking Missing (NaN) Values:*

dataset**.**isnull()**.**sum()

Out[6]:

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

* 'Albumin\_and\_Globulin\_Ratio' feature contain 4 NaN values.

In [7]:

*# Mean & Median of "Albumin\_and\_Globulin\_Ratio" feature:*

print(dataset['Albumin\_and\_Globulin\_Ratio']**.**median())

print(dataset['Albumin\_and\_Globulin\_Ratio']**.**mean())

0.93

0.9470639032815201

In [8]:

*# Filling NaN Values of "Albumin\_and\_Globulin\_Ratio" feature with Median :*

dataset['Albumin\_and\_Globulin\_Ratio'] **=** dataset['Albumin\_and\_Globulin\_Ratio']**.**fillna(dataset['Albumin\_and\_Globulin\_Ratio']**.**median())

In [9]:

*# Datatypes:*

dataset**.**dtypes

Out[9]:

Age int64

Gender object

Total\_Bilirubin float64

Direct\_Bilirubin float64

Alkaline\_Phosphotase int64

Alamine\_Aminotransferase int64

Aspartate\_Aminotransferase int64

Total\_Protiens float64

Albumin float64

Albumin\_and\_Globulin\_Ratio float64

Dataset int64

dtype: object

In [10]:

*# Description:*

dataset**.**describe()

Out[10]:

|  | **Age** | **Total\_Bilirubin** | **Direct\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 |
| **mean** | 44.746141 | 3.298799 | 1.486106 | 290.576329 | 80.713551 | 109.910806 | 6.483190 | 3.141852 | 0.946947 | 1.286449 |
| **std** | 16.189833 | 6.209522 | 2.808498 | 242.937989 | 182.620356 | 288.918529 | 1.085451 | 0.795519 | 0.318495 | 0.452490 |
| **min** | 4.000000 | 0.400000 | 0.100000 | 63.000000 | 10.000000 | 10.000000 | 2.700000 | 0.900000 | 0.300000 | 1.000000 |
| **25%** | 33.000000 | 0.800000 | 0.200000 | 175.500000 | 23.000000 | 25.000000 | 5.800000 | 2.600000 | 0.700000 | 1.000000 |
| **50%** | 45.000000 | 1.000000 | 0.300000 | 208.000000 | 35.000000 | 42.000000 | 6.600000 | 3.100000 | 0.930000 | 1.000000 |
| **75%** | 58.000000 | 2.600000 | 1.300000 | 298.000000 | 60.500000 | 87.000000 | 7.200000 | 3.800000 | 1.100000 | 2.000000 |
| **max** | 90.000000 | 75.000000 | 19.700000 | 2110.000000 | 2000.000000 | 4929.000000 | 9.600000 | 5.500000 | 2.800000 | 2.000000 |

In [11]:

*# Target feature:*

print("Liver Disease Patients :", dataset['Dataset']**.**value\_counts()[1])

print("Non Liver Disease Patients :", dataset['Dataset']**.**value\_counts()[2])

*# Visualization:*

sns**.**countplot(dataset['Dataset'])

plt**.**show()

Liver Disease Patients : 416

Non Liver Disease Patients : 167

C:\Users\91739\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



In [12]:

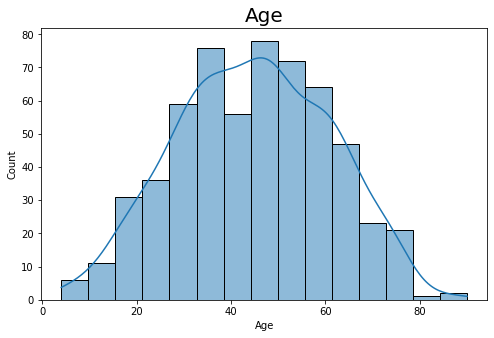
*# Histrogram of Age:*

plt**.**figure(figsize**=**(8,5))

sns**.**histplot(dataset['Age'], kde**=True**)

plt**.**title('Age', fontsize**=**20)

plt**.**show()



In [13]:

dataset**.**head()

Out[13]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Direct\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 65 | Female | 0.7 | 0.1 | 187 | 16 | 18 | 6.8 | 3.3 | 0.90 | 1 |
| **1** | 62 | Male | 10.9 | 5.5 | 699 | 64 | 100 | 7.5 | 3.2 | 0.74 | 1 |
| **2** | 62 | Male | 7.3 | 4.1 | 490 | 60 | 68 | 7.0 | 3.3 | 0.89 | 1 |
| **3** | 58 | Male | 1.0 | 0.4 | 182 | 14 | 20 | 6.8 | 3.4 | 1.00 | 1 |
| **4** | 72 | Male | 3.9 | 2.0 | 195 | 27 | 59 | 7.3 | 2.4 | 0.40 | 1 |

In [14]:

*# Gender feature:*

print("Total Male :", dataset['Gender']**.**value\_counts()[0])

print("Total Female :", dataset['Gender']**.**value\_counts()[1])

*# Visualization:*

sns**.**countplot(dataset['Gender'])

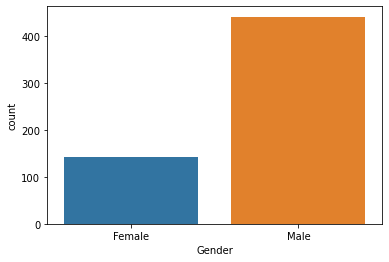
plt**.**show()

Total Male : 441

Total Female : 142

C:\Users\91739\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



In [15]:

*# Printing How many Unique values present in each feature:*

**for** feature **in** dataset**.**columns:

print(feature,":", len(dataset[feature]**.**unique()))

Age : 72

Gender : 2

Total\_Bilirubin : 113

Direct\_Bilirubin : 80

Alkaline\_Phosphotase : 263

Alamine\_Aminotransferase : 152

Aspartate\_Aminotransferase : 177

Total\_Protiens : 58

Albumin : 40

Albumin\_and\_Globulin\_Ratio : 69

Dataset : 2

In [16]:

*# Label Encoding*

dataset['Gender'] **=** np**.**where(dataset['Gender']**==**'Male', 1,0)

In [17]:

dataset**.**head()

Out[17]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Direct\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 65 | 0 | 0.7 | 0.1 | 187 | 16 | 18 | 6.8 | 3.3 | 0.90 | 1 |
| **1** | 62 | 1 | 10.9 | 5.5 | 699 | 64 | 100 | 7.5 | 3.2 | 0.74 | 1 |
| **2** | 62 | 1 | 7.3 | 4.1 | 490 | 60 | 68 | 7.0 | 3.3 | 0.89 | 1 |
| **3** | 58 | 1 | 1.0 | 0.4 | 182 | 14 | 20 | 6.8 | 3.4 | 1.00 | 1 |
| **4** | 72 | 1 | 3.9 | 2.0 | 195 | 27 | 59 | 7.3 | 2.4 | 0.40 | 1 |

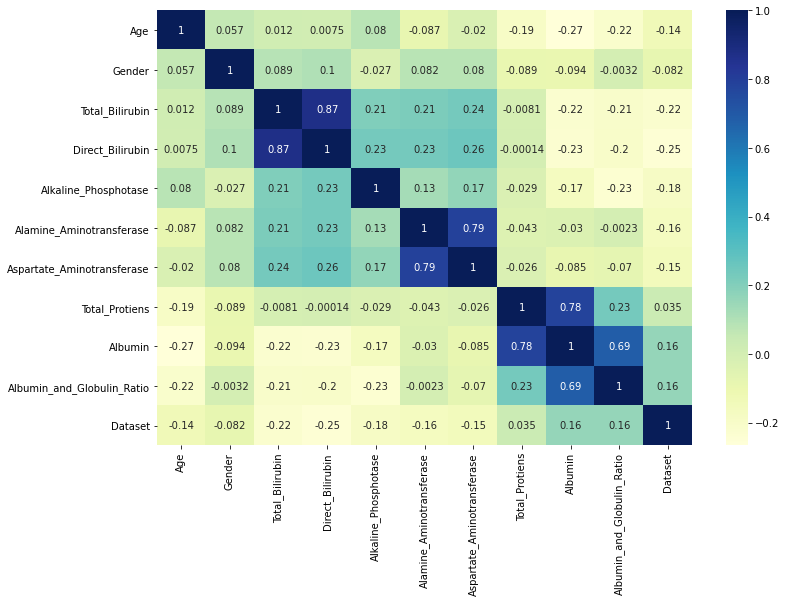
In [18]:

*# Correlation using Heatmap:*

plt**.**figure(figsize**=**(12,8))

sns**.**heatmap(dataset**.**corr(), annot**=True**, cmap**=**'YlGnBu')

plt**.**show()



**There is Multi-Collinearity found on our dataset.**

In [19]:

dataset**.**columns

Out[19]:

Index(['Age', 'Gender', 'Total\_Bilirubin', 'Direct\_Bilirubin',

'Alkaline\_Phosphotase', 'Alamine\_Aminotransferase',

'Aspartate\_Aminotransferase', 'Total\_Protiens', 'Albumin',

'Albumin\_and\_Globulin\_Ratio', 'Dataset'],

dtype='object')

1. Multicollinearity betwwen **'Total\_Bilirubin'** and **'Direct\_Bilirubin'** is **0.87%**
2. Multicollinearity betwwen **'Alamine\_Aminotransferase'** and **'Aspartate\_Aminotransferase'**is **0.79%**
3. Multicollinearity betwwen **'Total\_Protiens'** and **'Albumin'** is **0.78%**
4. Multicollinearity betwwen **'Albumin'** and **'Albumin\_and\_Globulin\_Ratio'** is **0.69%**

Usually we drop that feature which has above 0.85% multicollinearity between two independent feature. Here we have only 'Total\_Bilirubin' and 'Direct\_Bilirubin' feature which has 0.87% mutlicollinearity. So we drop one of the feature from them and other independent feature has less multicollinearity, less than 0.80% So we keep that feature.

In [20]:

*# Droping 'Direct\_Bilirubin' feature:*

dataset **=** dataset**.**drop('Direct\_Bilirubin', axis**=**1)

In [21]:

dataset**.**columns

Out[21]:

Index(['Age', 'Gender', 'Total\_Bilirubin', 'Alkaline\_Phosphotase',

'Alamine\_Aminotransferase', 'Aspartate\_Aminotransferase',

'Total\_Protiens', 'Albumin', 'Albumin\_and\_Globulin\_Ratio', 'Dataset'],

dtype='object')

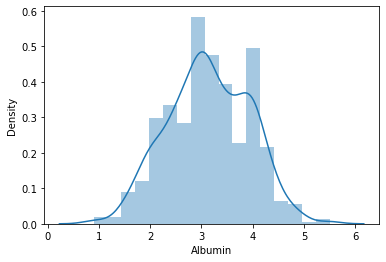
In [22]:

sns**.**distplot(dataset['Albumin'])

C:\Users\91739\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

Out[22]:



In [23]:

*# Calculate the boundaries of Total\_Protiens feature which differentiates the outliers:*

uppper\_boundary**=**dataset['Total\_Protiens']**.**mean() **+** 3**\*** dataset['Total\_Protiens']**.**std()

lower\_boundary**=**dataset['Total\_Protiens']**.**mean() **-** 3**\*** dataset['Total\_Protiens']**.**std()

print(dataset['Total\_Protiens']**.**mean())

print(lower\_boundary)

print(uppper\_boundary)

6.483190394511151

3.2268359424407524

9.73954484658155

In [24]:

*##### Calculate the boundaries of Albumin feature which differentiates the outliers:*

uppper\_boundary**=**dataset['Albumin']**.**mean() **+** 3**\*** dataset['Albumin']**.**std()

lower\_boundary**=**dataset['Albumin']**.**mean() **-** 3**\*** dataset['Albumin']**.**std()

print(dataset['Albumin']**.**mean())

print(lower\_boundary)

print(uppper\_boundary)

3.14185248713551

0.7552960692434287

5.528408905027591

In [25]:

*# Lets compute the Interquantile range of Total\_Bilirubin feature to calculate the boundaries:*

IQR **=** dataset**.**Total\_Bilirubin**.**quantile(0.75)**-**dataset**.**Total\_Bilirubin**.**quantile(0.25)

*# Extreme outliers*

lower\_bridge **=** dataset['Total\_Bilirubin']**.**quantile(0.25) **-** (IQR**\***3)

upper\_bridge **=** dataset['Total\_Bilirubin']**.**quantile(0.75) **+** (IQR**\***3)

print(lower\_bridge)

print(upper\_bridge)

*# if value greater than upper bridge, we replace that value with upper\_bridge value:*

dataset**.**loc[dataset['Total\_Bilirubin'] **>=** upper\_bridge, 'Total\_Bilirubin'] **=** upper\_bridge

-4.6000000000000005

8.0

In [26]:

*# Lets compute the Interquantile range of Alkaline\_Phosphotase feature to calculate the boundaries:*

IQR **=** dataset**.**Alkaline\_Phosphotase**.**quantile(0.75) **-** dataset**.**Alkaline\_Phosphotase**.**quantile(0.25)

*# Extreme outliers*

lower\_bridge **=** dataset['Alkaline\_Phosphotase']**.**quantile(0.25) **-** (IQR**\***3)

upper\_bridge **=** dataset['Alkaline\_Phosphotase']**.**quantile(0.75) **+** (IQR**\***3)

print(lower\_bridge)

print(upper\_bridge)

*# if value greater than upper bridge, we replace that value with upper\_bridge value:*

dataset**.**loc[dataset['Alkaline\_Phosphotase'] **>=** upper\_bridge, 'Alkaline\_Phosphotase'] **=** upper\_bridge

-192.0

665.5

In [27]:

*# Lets compute the Interquantile range of Alamine\_Aminotransferase feature to calculate the boundaries:*

IQR **=** dataset**.**Alamine\_Aminotransferase**.**quantile(0.75) **-** dataset**.**Alamine\_Aminotransferase**.**quantile(0.25)

*# Extreme outliers*

lower\_bridge **=** dataset['Alamine\_Aminotransferase']**.**quantile(0.25) **-** (IQR**\***3)

upper\_bridge **=** dataset['Alamine\_Aminotransferase']**.**quantile(0.75) **+** (IQR**\***3)

print(lower\_bridge)

print(upper\_bridge)

*# if value greater than upper bridge, we replace that value with upper\_bridge value:*

dataset**.**loc[dataset['Alamine\_Aminotransferase'] **>=** upper\_bridge, 'Alamine\_Aminotransferase'] **=** upper\_bridge

-89.5

173.0

In [28]:

*# Lets compute the Interquantile range of Aspartate\_Aminotransferase feature to calculate the boundaries:*

IQR **=** dataset**.**Aspartate\_Aminotransferase**.**quantile(0.75) **-** dataset**.**Aspartate\_Aminotransferase**.**quantile(0.25)

*# Extreme outliers*

lower\_bridge **=** dataset['Aspartate\_Aminotransferase']**.**quantile(0.25) **-** (IQR**\***3)

upper\_bridge **=** dataset['Aspartate\_Aminotransferase']**.**quantile(0.75) **+** (IQR**\***3)

print(lower\_bridge)

print(upper\_bridge)

*# if value greater than upper bridge, we replace that value with upper\_bridge value:*

dataset**.**loc[dataset['Aspartate\_Aminotransferase'] **>=** upper\_bridge, 'Aspartate\_Aminotransferase'] **=** upper\_bridge

-161.0

273.0

In [29]:

*# Lets compute the Interquantile range of Albumin\_and\_Globulin\_Ratio feature to calculate the boundaries*

IQR **=** dataset**.**Albumin\_and\_Globulin\_Ratio**.**quantile(0.75) **-** dataset**.**Albumin\_and\_Globulin\_Ratio**.**quantile(0.25)

*# Extreme outliers*

lower\_bridge **=** dataset['Albumin\_and\_Globulin\_Ratio']**.**quantile(0.25) **-** (IQR**\***3)

upper\_bridge **=** dataset['Albumin\_and\_Globulin\_Ratio']**.**quantile(0.75) **+** (IQR**\***3)

print(lower\_bridge)

print(upper\_bridge)

*# if value greater than upper bridge, we replace that value with upper\_bridge value:*

dataset**.**loc[dataset['Albumin\_and\_Globulin\_Ratio'] **>=** upper\_bridge, 'Albumin\_and\_Globulin\_Ratio'] **=** upper\_bridge

-0.5000000000000004

2.3000000000000007

In [30]:

*# Top 5 records:*

dataset**.**head()

Out[30]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 65 | 0 | 0.7 | 187.0 | 16.0 | 18.0 | 6.8 | 3.3 | 0.90 | 1 |
| **1** | 62 | 1 | 8.0 | 665.5 | 64.0 | 100.0 | 7.5 | 3.2 | 0.74 | 1 |
| **2** | 62 | 1 | 7.3 | 490.0 | 60.0 | 68.0 | 7.0 | 3.3 | 0.89 | 1 |
| **3** | 58 | 1 | 1.0 | 182.0 | 14.0 | 20.0 | 6.8 | 3.4 | 1.00 | 1 |
| **4** | 72 | 1 | 3.9 | 195.0 | 27.0 | 59.0 | 7.3 | 2.4 | 0.40 | 1 |

In [31]:

*# Description after deal with outliers by IQR:*

dataset**.**describe()

Out[31]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** | **Dataset** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 |
| **mean** | 44.746141 | 0.756432 | 2.249400 | 266.389365 | 53.399657 | 73.041166 | 6.483190 | 3.141852 | 0.945403 | 1.286449 |
| **std** | 16.189833 | 0.429603 | 2.382344 | 145.665460 | 46.059536 | 73.549864 | 1.085451 | 0.795519 | 0.310942 | 0.452490 |
| **min** | 4.000000 | 0.000000 | 0.400000 | 63.000000 | 10.000000 | 10.000000 | 2.700000 | 0.900000 | 0.300000 | 1.000000 |
| **25%** | 33.000000 | 1.000000 | 0.800000 | 175.500000 | 23.000000 | 25.000000 | 5.800000 | 2.600000 | 0.700000 | 1.000000 |
| **50%** | 45.000000 | 1.000000 | 1.000000 | 208.000000 | 35.000000 | 42.000000 | 6.600000 | 3.100000 | 0.930000 | 1.000000 |
| **75%** | 58.000000 | 1.000000 | 2.600000 | 298.000000 | 60.500000 | 87.000000 | 7.200000 | 3.800000 | 1.100000 | 2.000000 |
| **max** | 90.000000 | 1.000000 | 8.000000 | 665.500000 | 173.000000 | 273.000000 | 9.600000 | 5.500000 | 2.300000 | 2.000000 |

In [32]:

*# Independent and Dependent Feature:*

X **=** dataset**.**iloc[:, :**-**1]

y **=** dataset**.**iloc[:, **-**1]

In [33]:

*# top 5 records of Independent features:*

X**.**head()

Out[33]:

|  | **Age** | **Gender** | **Total\_Bilirubin** | **Alkaline\_Phosphotase** | **Alamine\_Aminotransferase** | **Aspartate\_Aminotransferase** | **Total\_Protiens** | **Albumin** | **Albumin\_and\_Globulin\_Ratio** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 65 | 0 | 0.7 | 187.0 | 16.0 | 18.0 | 6.8 | 3.3 | 0.90 |
| **1** | 62 | 1 | 8.0 | 665.5 | 64.0 | 100.0 | 7.5 | 3.2 | 0.74 |
| **2** | 62 | 1 | 7.3 | 490.0 | 60.0 | 68.0 | 7.0 | 3.3 | 0.89 |
| **3** | 58 | 1 | 1.0 | 182.0 | 14.0 | 20.0 | 6.8 | 3.4 | 1.00 |
| **4** | 72 | 1 | 3.9 | 195.0 | 27.0 | 59.0 | 7.3 | 2.4 | 0.40 |

In [34]:

*# top 5 records of dependent features:*

y**.**head()

Out[34]:

0 1

1 1

2 1

3 1

4 1

Name: Dataset, dtype: int64

In [36]:

pip install imblearn

Collecting imblearn

Using cached imblearn-0.0-py2.py3-none-any.whl (1.9 kB)

Collecting imbalanced-learn

Using cached imbalanced\_learn-0.9.1-py3-none-any.whl (199 kB)

Requirement already satisfied: numpy>=1.17.3 in c:\users\91739\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.20.1)

Requirement already satisfied: joblib>=1.0.0 in c:\users\91739\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.0.1)

Requirement already satisfied: scipy>=1.3.2 in c:\users\91739\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.6.2)

Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\91739\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (2.1.0)

Collecting scikit-learn>=1.1.0

Downloading scikit\_learn-1.1.2-cp38-cp38-win\_amd64.whl (7.3 MB)

Installing collected packages: scikit-learn, imbalanced-learn, imblearn

Attempting uninstall: scikit-learn

Found existing installation: scikit-learn 0.24.1

Uninstalling scikit-learn-0.24.1:

Successfully uninstalled scikit-learn-0.24.1

Successfully installed imbalanced-learn-0.9.1 imblearn-0.0 scikit-learn-1.1.2

Note: you may need to restart the kernel to use updated packages.

In [37]:

*# SMOTE Technique:*

**from** imblearn.combine **import** SMOTETomek

smote **=** SMOTETomek()

X\_smote, y\_smote **=** smote**.**fit\_resample(X,y)

In [38]:

*# Counting before and after SMOTE:*

**from** collections **import** Counter

print('Before SMOTE : ', Counter(y))

print('After SMOTE : ', Counter(y\_smote))

Before SMOTE : Counter({1: 416, 2: 167})

After SMOTE : Counter({1: 390, 2: 390})

In [39]:

*# Train Test Split:*

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,y\_train,y\_test **=** train\_test\_split(X\_smote,y\_smote, test\_size**=**0.3, random\_state**=**33)

In [40]:

print(X\_train**.**shape)

print(X\_test**.**shape)

(546, 9)

(234, 9)

In [41]:

*# Feature Importance :*

**from** sklearn.feature\_selection **import** SelectKBest

**from** sklearn.feature\_selection **import** chi2

*### Apply SelectKBest Algorithm*

ordered\_rank\_features**=**SelectKBest(score\_func**=**chi2,k**=**9)

ordered\_feature**=**ordered\_rank\_features**.**fit(X,y)

dfscores**=**pd**.**DataFrame(ordered\_feature**.**scores\_,columns**=**["Score"])

dfcolumns**=**pd**.**DataFrame(X**.**columns)

features\_rank**=**pd**.**concat([dfcolumns,dfscores],axis**=**1)

features\_rank**.**columns**=**['Features','Score']

features\_rank**.**nlargest(9, 'Score')

Out[41]:

|  | **Features** | **Score** |
| --- | --- | --- |
| **5** | Aspartate\_Aminotransferase | 3368.743077 |
| **3** | Alkaline\_Phosphotase | 2385.790640 |
| **4** | Alamine\_Aminotransferase | 1717.348297 |
| **2** | Total\_Bilirubin | 127.476411 |
| **0** | Age | 64.315174 |
| **7** | Albumin | 3.053371 |
| **8** | Albumin\_and\_Globulin\_Ratio | 1.704602 |
| **1** | Gender | 0.964518 |
| **6** | Total\_Protiens | 0.129627 |

**There is no need of Standardization and Normalization of our dataset, as we using Ensemble Technique.**

In [42]:

*# Importing Performance Metrics:*

**from** sklearn.metrics **import** accuracy\_score, confusion\_matrix, classification\_report

In [43]:

*# RandomForestClassifier:*

**from** sklearn.ensemble **import** RandomForestClassifier

RandomForest **=** RandomForestClassifier()

RandomForest **=** RandomForest**.**fit(X\_train,y\_train)

*# Predictions:*

y\_pred **=** RandomForest**.**predict(X\_test)

*# Performance:*

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

Accuracy: 0.8589743589743589

[[ 98 17]

[ 16 103]]

precision recall f1-score support

1 0.86 0.85 0.86 115

2 0.86 0.87 0.86 119

accuracy 0.86 234

macro avg 0.86 0.86 0.86 234

weighted avg 0.86 0.86 0.86 234

In [44]:

*# AdaBoostClassifier:*

**from** sklearn.ensemble **import** AdaBoostClassifier

AdaBoost **=** AdaBoostClassifier()

AdaBoost **=** AdaBoost**.**fit(X\_train,y\_train)

*# Predictions:*

y\_pred **=** AdaBoost**.**predict(X\_test)

*# Performance:*

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

Accuracy: 0.7735042735042735

[[88 27]

[26 93]]

precision recall f1-score support

1 0.77 0.77 0.77 115

2 0.78 0.78 0.78 119

accuracy 0.77 234

macro avg 0.77 0.77 0.77 234

weighted avg 0.77 0.77 0.77 234

In [45]:

*# GradientBoostingClassifier:*

**from** sklearn.ensemble **import** GradientBoostingClassifier

GradientBoost **=** GradientBoostingClassifier()

GradientBoost **=** GradientBoost**.**fit(X\_train,y\_train)

*# Predictions:*

y\_pred **=** GradientBoost**.**predict(X\_test)

*# Performance:*

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

Accuracy: 0.7863247863247863

[[94 21]

[29 90]]

precision recall f1-score support

1 0.76 0.82 0.79 115

2 0.81 0.76 0.78 119

accuracy 0.79 234

macro avg 0.79 0.79 0.79 234

weighted avg 0.79 0.79 0.79 234

**RandomizedSearchCV**

In [46]:

*# Importing RandomizedSearchCV:*

**from** sklearn.model\_selection **import** RandomizedSearchCV

In [47]:

*# Number of trees in random forest:*

n\_estimators **=** [int(x) **for** x **in** np**.**linspace(start **=** 100, stop **=** 2000, num **=** 20)]

*# Number of features to consider at every split:*

max\_features **=** ['auto', 'sqrt','log2']

*# Maximum number of levels in tree:*

max\_depth **=** [int(x) **for** x **in** np**.**linspace(100, 100,20)]

*# Minimum number of samples required to split a node:*

min\_samples\_split **=** [1,2,3,4,5,6,7,8,9,10,12,14,16,18,20]

*# Minimum number of samples required at each leaf node:*

min\_samples\_leaf **=** [1,2,3,4,5,6,7,8,9,10,12,14,16,18,20]

In [48]:

*# Create the random grid:*

random\_grid **=** {'n\_estimators': n\_estimators,

'max\_features': max\_features,

'max\_depth': max\_depth,

'min\_samples\_split': min\_samples\_split,

'min\_samples\_leaf': min\_samples\_leaf,

'criterion':['entropy','gini']}

print(random\_grid)

{'n\_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000], 'max\_features': ['auto', 'sqrt', 'log2'], 'max\_depth': [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100], 'min\_samples\_split': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20], 'min\_samples\_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20], 'criterion': ['entropy', 'gini']}

In [49]:

rf **=** RandomForestClassifier()

rf\_randomcv **=** RandomizedSearchCV(estimator **=** rf, param\_distributions **=** random\_grid, n\_iter **=** 100, cv **=** 5, verbose **=** 2,

random\_state **=** 0, n\_jobs **=** **-**1)

*# fit the randomized model:*

rf\_randomcv**.**fit(X\_train,y\_train)

Fitting 5 folds for each of 100 candidates, totalling 500 fits

C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py:378: FitFailedWarning:

40 fits failed out of a total of 500.

The score on these train-test partitions for these parameters will be set to nan.

If these failures are not expected, you can try to debug them by setting error\_score='raise'.

Below are more details about the failures:

--------------------------------------------------------------------------------

40 fits failed with the following error:

Traceback (most recent call last):

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py", line 686, in \_fit\_and\_score

estimator.fit(X\_train, y\_train, \*\*fit\_params)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py", line 476, in fit

trees = Parallel(

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 1041, in \_\_call\_\_

if self.dispatch\_one\_batch(iterator):

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 859, in dispatch\_one\_batch

self.\_dispatch(tasks)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 777, in \_dispatch

job = self.\_backend.apply\_async(batch, callback=cb)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\\_parallel\_backends.py", line 208, in apply\_async

result = ImmediateResult(func)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\\_parallel\_backends.py", line 572, in \_\_init\_\_

self.results = batch()

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in \_\_call\_\_

return [func(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in

return [func(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\utils\fixes.py", line 117, in \_\_call\_\_

return self.function(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py", line 189, in \_parallel\_build\_trees

tree.fit(X, y, sample\_weight=curr\_sample\_weight, check\_input=False)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 969, in fit

super().fit(

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 265, in fit

check\_scalar(

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\utils\validation.py", line 1480, in check\_scalar

raise ValueError(

ValueError: min\_samples\_split == 1, must be >= 2.

warnings.warn(some\_fits\_failed\_message, FitFailedWarning)

C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_search.py:953: UserWarning: One or more of the test scores are non-finite: [ nan 0.71981651 0.75089241 0.7253211 0.71979983 0.72900751

0.75276063 0.72900751 0.74178482 0.75823186 0.71984987 0.72900751

0.72348624 0.71252711 0.72348624 0.7253211 0.71983319 0.75274395

0.73264387 0.76373645 nan 0.72717264 0.76185154 0.72165138

0.72348624 0.71983319 0.73084237 0.74545455 0.74722269 0.73451209

0.72346956 nan 0.73998332 0.74175146 0.72715596 0.73449541

0.72533778 0.75638032 0.76371977 0.74727273 0.72902419 0.71618015

0.72165138 0.71067556 0.71618015 0.73998332 nan 0.72166806

nan 0.72528774 0.73449541 0.71801501 0.73451209 0.74361968

0.72718932 0.743603 0.7581985 0.75274395 0.71983319 0.73447873

0.7235196 0.73633028 0.71618015 0.73262719 0.77284404 0.72533778

0.74176814 0.7235196 0.73446205 0.72166806 nan 0.74905755

0.73082569 0.73079233 0.73629691 0.74178482 0.73816514 0.72717264

0.72717264 0.71983319 0.74356964 0.72717264 0.74363636 nan

0.71434529 0.75454545 0.73634696 0.76737281 0.74178482 0.76005004

0.74543786 0.75641368 nan 0.74545455 0.72165138 0.71801501

0.73267723 0.72718932 0.72899083 0.72715596]

warnings.warn(

C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py:427: FutureWarning: `max\_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max\_features='sqrt'` or remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.

warn(

Out[49]:

RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n\_iter=100,

n\_jobs=-1,

param\_distributions={'criterion': ['entropy', 'gini'],

'max\_depth': [100, 100, 100, 100, 100,

100, 100, 100, 100, 100,

100, 100, 100, 100, 100,

100, 100, 100, 100, 100],

'max\_features': ['auto', 'sqrt',

'log2'],

'min\_samples\_leaf': [1, 2, 3, 4, 5, 6,

7, 8, 9, 10, 12,

14, 16, 18, 20],

'min\_samples\_split': [1, 2, 3, 4, 5, 6,

7, 8, 9, 10, 12,

14, 16, 18, 20],

'n\_estimators': [100, 200, 300, 400,

500, 600, 700, 800,

900, 1000, 1100, 1200,

1300, 1400, 1500, 1600,

1700, 1800, 1900,

2000]},

random\_state=0, verbose=2)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [50]:

*# Best parameter of RandomizedSearchCV:*

rf\_randomcv**.**best\_params\_

Out[50]:

{'n\_estimators': 200,

'min\_samples\_split': 6,

'min\_samples\_leaf': 1,

'max\_features': 'auto',

'max\_depth': 100,

'criterion': 'entropy'}

In [51]:

*# Creating model using best parameter of RandomizedSearchCV:*

RandomForest\_RandomCV **=** RandomForestClassifier(criterion **=** 'entropy', n\_estimators **=** 2000, max\_depth **=** 100, max\_features **=** 'log2',

min\_samples\_split **=** 3, min\_samples\_leaf **=** 2)

RandomForest\_RandomCV **=** RandomForest\_RandomCV**.**fit(X\_train,y\_train)

*# Predictions:*

y\_pred **=** RandomForest\_RandomCV**.**predict(X\_test)

*# Performance:*

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

Accuracy: 0.8461538461538461

[[ 97 18]

[ 18 101]]

precision recall f1-score support

1 0.84 0.84 0.84 115

2 0.85 0.85 0.85 119

accuracy 0.85 234

macro avg 0.85 0.85 0.85 234

weighted avg 0.85 0.85 0.85 234

**GridSearchCV**

In [52]:

*# Importing GridSearchCV:*

**from** sklearn.model\_selection **import** GridSearchCV

In [53]:

*# Best parameter:*

rf\_randomcv**.**best\_params\_

Out[53]:

{'n\_estimators': 200,

'min\_samples\_split': 6,

'min\_samples\_leaf': 1,

'max\_features': 'auto',

'max\_depth': 100,

'criterion': 'entropy'}

In [54]:

param\_grid **=** {

'criterion': [rf\_randomcv**.**best\_params\_['criterion']],

'max\_features': [rf\_randomcv**.**best\_params\_['max\_features']],

'max\_depth': [rf\_randomcv**.**best\_params\_['max\_depth']**-**50,

rf\_randomcv**.**best\_params\_['max\_depth'],

rf\_randomcv**.**best\_params\_['max\_depth']**+**50],

'min\_samples\_leaf': [rf\_randomcv**.**best\_params\_['min\_samples\_leaf']**-**1,

rf\_randomcv**.**best\_params\_['min\_samples\_leaf'],

rf\_randomcv**.**best\_params\_['min\_samples\_leaf']**+**1],

'min\_samples\_split': [rf\_randomcv**.**best\_params\_['min\_samples\_split'] **-** 1,

rf\_randomcv**.**best\_params\_['min\_samples\_split'],

rf\_randomcv**.**best\_params\_['min\_samples\_split'] **+**1],

'n\_estimators': [rf\_randomcv**.**best\_params\_['n\_estimators'] **-** 50,

rf\_randomcv**.**best\_params\_['n\_estimators'],

rf\_randomcv**.**best\_params\_['n\_estimators'] **+** 50]

}

print(param\_grid)

{'criterion': ['entropy'], 'max\_features': ['auto'], 'max\_depth': [50, 100, 150], 'min\_samples\_leaf': [0, 1, 2], 'min\_samples\_split': [5, 6, 7], 'n\_estimators': [150, 200, 250]}

In [55]:

*# Fit the grid\_search to the data:*

rf **=** RandomForestClassifier()

grid\_search **=** GridSearchCV(estimator **=** rf, param\_grid **=** param\_grid, cv**=**5 , n\_jobs **=** **-**1, verbose **=** 2)

grid\_search**.**fit(X\_train,y\_train)

Fitting 5 folds for each of 81 candidates, totalling 405 fits

C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py:378: FitFailedWarning:

135 fits failed out of a total of 405.

The score on these train-test partitions for these parameters will be set to nan.

If these failures are not expected, you can try to debug them by setting error\_score='raise'.

Below are more details about the failures:

--------------------------------------------------------------------------------

135 fits failed with the following error:

Traceback (most recent call last):

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py", line 686, in \_fit\_and\_score

estimator.fit(X\_train, y\_train, \*\*fit\_params)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py", line 476, in fit

trees = Parallel(

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 1041, in \_\_call\_\_

if self.dispatch\_one\_batch(iterator):

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 859, in dispatch\_one\_batch

self.\_dispatch(tasks)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 777, in \_dispatch

job = self.\_backend.apply\_async(batch, callback=cb)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\\_parallel\_backends.py", line 208, in apply\_async

result = ImmediateResult(func)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\\_parallel\_backends.py", line 572, in \_\_init\_\_

self.results = batch()

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in \_\_call\_\_

return [func(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in

return [func(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\utils\fixes.py", line 117, in \_\_call\_\_

return self.function(\*args, \*\*kwargs)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py", line 189, in \_parallel\_build\_trees

tree.fit(X, y, sample\_weight=curr\_sample\_weight, check\_input=False)

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 969, in fit

super().fit(

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 247, in fit

check\_scalar(

File "C:\Users\91739\anaconda3\lib\site-packages\sklearn\utils\validation.py", line 1480, in check\_scalar

raise ValueError(

ValueError: min\_samples\_leaf == 0, must be >= 1.

warnings.warn(some\_fits\_failed\_message, FitFailedWarning)

C:\Users\91739\anaconda3\lib\site-packages\sklearn\model\_selection\\_search.py:953: UserWarning: One or more of the test scores are non-finite: [ nan nan nan nan nan nan

nan nan nan 0.77653044 0.75452877 0.76735613

0.76740617 0.78018349 0.76555463 0.76376981 0.76553795 0.756397

0.76375313 0.75089241 0.76190158 0.7581985 0.75634696 0.76738949

0.75823186 0.76552127 0.75638032 nan nan nan

nan nan nan nan nan nan

0.74904087 0.75452877 0.76003336 0.76553795 0.77653044 0.77100917

0.76553795 0.7618849 0.75821518 0.76001668 0.76001668 0.76003336

0.76370309 0.74904087 0.75818182 0.76555463 0.74907423 0.76001668

nan nan nan nan nan nan

nan nan nan 0.7728774 0.76920767 0.76555463

0.7581985 0.75641368 0.76366972 0.75454545 0.76003336 0.76190158

0.74907423 0.75821518 0.76552127 0.756397 0.75823186 0.7581985

0.76185154 0.75816514 0.75636364]

warnings.warn(

C:\Users\91739\anaconda3\lib\site-packages\sklearn\ensemble\\_forest.py:427: FutureWarning: `max\_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max\_features='sqrt'` or remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.

warn(

Out[55]:

GridSearchCV(cv=5, estimator=RandomForestClassifier(), n\_jobs=-1,

param\_grid={'criterion': ['entropy'], 'max\_depth': [50, 100, 150],

'max\_features': ['auto'],

'min\_samples\_leaf': [0, 1, 2],

'min\_samples\_split': [5, 6, 7],

'n\_estimators': [150, 200, 250]},

verbose=2)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [59]:

*# Best Parameter of GridSearchCV:*

grid\_search**.**best\_params\_

Out[59]:

{'criterion': 'entropy',

'max\_depth': 50,

'max\_features': 'auto',

'min\_samples\_leaf': 1,

'min\_samples\_split': 6,

'n\_estimators': 200}

In [60]:

*# Creating model using best parameter of GridSearchCV:*

RandomForest\_gridCV **=** RandomForestClassifier(criterion**=**'entropy', n\_estimators**=**1950, max\_depth**=**150, max\_features**=**'log2',

min\_samples\_split**=**2, min\_samples\_leaf**=**1)

RandomForest\_gridCv **=** RandomForest\_gridCV**.**fit(X\_train,y\_train)

*# Predictions:*

y\_pred **=** RandomForest\_gridCV**.**predict(X\_test)

*# Performance:*

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

Accuracy: 0.8547008547008547

[[ 98 17]

[ 17 102]]

precision recall f1-score support

1 0.85 0.85 0.85 115

2 0.86 0.86 0.86 119

accuracy 0.85 234

macro avg 0.85 0.85 0.85 234

weighted avg 0.85 0.85 0.85 234

**- We saw that after doing RandomizedSearchCV and GridSearchCV, Our accuracy, Precision, Recall, f1-Score doesn't increase.**

In [61]:

*# Creating a pickle file for the classifier*

**import** pickle

filename **=** 'Liver.pkl'

pickle**.**dump(RandomForestClassifier, open(filename, 'wb'))

In [ ]:

In [ ]: