Machine Learning(22AIE213)

Project Report on Research Paper

Course Instructor- Dr. Madhusudan Rao

Name of Project- A systematic method for diagnosis of hepatitis disease using machine learning.

By-Ravi Kumar Sachdeva Priyanka Bathla, Pooja Rani, Vikas Solanki, Rakesh Ahuja

Done By-

Anna Charith -AV.EN.U4AIE22001

Adithya Santhosh -AV.EN.U4AIE22002

Akshar Samudrala -AV.EN.U4AIE22003

B. Natraj Koushik -AV.EN.U4AIE22004

Description

Hepatitis stands as one of the deadliest diseases worldwide, wreaking havoc on liver cells known as hepatocytes, resulting in inflammation and severe damage, disrupting the organ's vital functions. This ailment manifests in two forms: acute and chronic, with causes ranging from excessive alcohol intake, adverse drug reactions, to viral and bacterial infections.

In a research endeavor, various classifiers—ranging from support vector machines to *logistic regression*, *K-nearest neighbor*, and *random forest*—were harnessed to forecast hepatitis*. LR, Kernel SVM,* and *KNN* surfaced as top performers, boasting an accuracy rate of *90.32%*, whereas the random forest classifier clinched the lead with an impressive *92.88%* accuracy rate on the UCI hepatitis dataset. Furthermore, the study incorporated a medley of techniques including *SVM, Gaussian NB, LR, decision trees, KNN, and MLP,* yielding a peak accuracy of *87%* with MLP and LR classifiers.

Within this analytical framework, the SMOTE algorithm takes center stage, offering a sophisticated means of augmenting dataset diversity by generating synthetic examples through an intricate process of blending existing instances. This methodological approach aims to enrich the dataset, particularly amplifying representation for underrepresented classes, thus bolstering the efficacy of machine learning models in diagnosing hepatitis accurately.

Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression (LR), and Random Forest (RF) are commonly used classification algorithms in machine learning for disease prediction tasks such as hepatitis. Together, these algorithms play a crucial role in hepatitis prediction, facilitating early detection and effective management through analysis and classification of patient data.

Diagnosis of Hepatitis Disease using Machine Learning

Required Packages

These are the packages that we are going to use in this file.

```
import pandas as pd
import numpy as np
import seaborn as sns
import  matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn import linear_model, metrics
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,
classification_report, roc_curve, accuracy_score,precision_score,
recall_score, f1_score
from sklearn.preprocessing import MinMaxScaler,Normalizer, OneHotEncoder
from imblearn.over_sampling import SMOTE
import warnings
warnings.filterwarnings("ignore")
#warnings.filterwarnings("default")
# from google.colab import drive

# drive.mount('/content/drive')
# !ls /content/drive/
```

File path of the Dataset

```
#file_path =
"/content/drive/MyDrive/ColabNotebook/heapatitis_dataset_modified.csv"
file_path = "heapatitis_dataset_modified.csv"
data = pd.read_csv(file_path)
```

Data Information

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155 entries, 0 to 154
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Unnamed: 0       155 non-null    int64
 1   Class            155 non-null    int64
 2   Age              155 non-null    int64
 3   Sex              155 non-null    int64
 4   Steroid          154 non-null    float64
 5   Antivirals       155 non-null    int64
 6   Fatigue          154 non-null    float64
 7   Malaise          154 non-null    float64
 8   Anorexia         154 non-null    float64
 9   Liver Big        145 non-null    float64
 10  Liver Firm       144 non-null    float64
 11  Spleen Palpable  150 non-null    float64
 12  Spiders          150 non-null    float64
 13  Ascities         150 non-null    float64
 14  Varices          150 non-null    float64
 15  Bilirubin        149 non-null    float64
 16  ALK Poshphate    126 non-null    float64
 17  SGOT             151 non-null    float64
 18  Albumin          139 non-null    float64
 19  Protime          88 non-null     float64
 20  Histology        155 non-null    int64
dtypes: float64(15), int64(6)
memory usage: 25.6 KB
```

Display the Dataset

```
# print the head i.e, top 5 row of the file
data.head()
```

```
    Unnamed: 0  Class  Age  Sex  Steroid  Antivirals  Fatigue  Malaise  \
5            5      2   34    1      2.0           2      2.0      2.0
10          10      2   39    1      1.0           1      2.0      2.0
11          11      2   32    1      2.0           1      1.0      2.0
12          12      2   41    1      2.0           1      1.0      2.0
13          13      2   30    1      2.0           2      1.0      2.0

    Anorexia  Liver Big  ...  Spleen Palpable  Spiders  Ascities  Varices  \
5        2.0        2.0  ...              2.0      2.0       2.0      2.0
10       2.0        1.0  ...              2.0      2.0       2.0      2.0
11       2.0        2.0  ...              2.0      1.0       2.0      2.0
12       2.0        2.0  ...              2.0      2.0       2.0      2.0
13       2.0        2.0  ...              2.0      2.0       2.0      2.0

    Bilirubin  ALK Poshphate   SGOT  Albumin  Protime  Histology
5         0.9       0.271654   28.0      4.0     0.75          1
10        1.3       0.204724   30.0      4.4     0.85          1
11        1.0       0.129921  249.0      3.7     0.54          1
12        0.9       0.216535   60.0      3.9     0.52          1
13        2.2       0.122047  144.0      4.9     0.78          1

[5 rows x 21 columns]
```

```
# print the tail i.e, last 5 row of the file
print(data.tail())
```

```
     Unnamed: 0  Class  Age  Sex  Steroid  Antivirals  Fatigue  Malaise  \
139         139      2   45    1      2.0           1      2.0      2.0
143         143      1   49    1      1.0           2      1.0      1.0
145         145      2   31    1      1.0           2      1.0      2.0
153         153      2   53    2      1.0           2      1.0      2.0
154         154      1   43    1      2.0           2      1.0      2.0

     Anorexia  Liver Big  ...  Spleen Palpable  Spiders  Ascities  Varices  \
139       2.0        2.0  ...              2.0      2.0       2.0      2.0
143       2.0        2.0  ...              1.0      1.0       2.0      2.0
145       2.0        2.0  ...              2.0      2.0       2.0      2.0
153       2.0        2.0  ...              1.0      1.0       2.0      1.0
154       2.0        2.0  ...              1.0      1.0       1.0      2.0

     Bilirubin  ALK Poshphate   SGOT  Albumin  Protime  Histology
139        1.3       0.232283   44.0      4.2     0.85          2
143        1.4       0.232283   70.0      3.5     0.35          2
145        1.2       0.192913  173.0      4.2     0.54          2
153        1.5       0.216535   19.0      4.1     0.48          2
154        1.2       0.291339   19.0      3.1     0.42          2

[5 rows x 21 columns]
```

Prints all the basic mathematic static of dataset

```
data.describe()
```

```
       Unnamed: 0       Class         Age         Sex     Steroid  Antivirals
\
count  155.000000  155.000000  155.000000  155.000000  154.000000  155.000000
mean    77.000000    1.793548   41.200000    1.103226    1.506494    1.845161
std     44.888751    0.406070   12.565878    0.305240    0.501589    0.362923
min      0.000000    1.000000    7.000000    1.000000    1.000000    1.000000
25%     38.500000    2.000000   32.000000    1.000000    1.000000    2.000000
50%     77.000000    2.000000   39.000000    1.000000    2.000000    2.000000
75%    115.500000    2.000000   50.000000    1.000000    2.000000    2.000000
max    154.000000    2.000000   78.000000    2.000000    2.000000    2.000000

          Fatigue     Malaise    Anorexia   Liver Big  ...  Spleen Palpable
\
count  154.000000  154.000000  154.000000  145.000000  ...        150.00000
mean     1.350649    1.603896    1.792208    1.827586  ...          1.80000
std      0.478730    0.490682    0.407051    0.379049  ...          0.40134
min      1.000000    1.000000    1.000000    1.000000  ...          1.00000
25%      1.000000    1.000000    2.000000    2.000000  ...          2.00000
50%      1.000000    2.000000    2.000000    2.000000  ...          2.00000
75%      2.000000    2.000000    2.000000    2.000000  ...          2.00000
max      2.000000    2.000000    2.000000    2.000000  ...          2.00000

          Spiders    Ascities    Varices   Bilirubin  ALK Poshphate  \
count  150.000000  150.000000  150.00000  149.000000     126.000000
```

```
mean     1.660000    1.866667    1.88000    1.427517     105.325397
std      0.475296    0.341073    0.32605    1.212149      51.508109
min      1.000000    1.000000    1.00000    0.300000      26.000000
25%      1.000000    2.000000    2.00000    0.700000      74.250000
50%      2.000000    2.000000    2.00000    1.000000      85.000000
75%      2.000000    2.000000    2.00000    1.500000     132.250000
max      2.000000    2.000000    2.00000    8.000000     295.000000

            SGOT     Albumin     Protime   Histology
count  151.00000  139.000000   88.000000  155.000000
mean    85.89404    3.817266   61.852273    1.451613
std     89.65089    0.651523   22.875244    0.499266
min     14.00000    2.100000    0.000000    1.000000
25%     31.50000    3.400000   46.000000    1.000000
50%     58.00000    4.000000   61.000000    1.000000
75%    100.50000    4.200000   76.250000    2.000000
max    648.00000    6.400000  100.000000    2.000000

[8 rows x 21 columns]
```

Here we going to show the size of our dataset

```
print(data.shape)
```

```
(155, 21)
```

Here we are showing the number of missing values in our dataset.

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

```
Unnamed: 0          0
Class               0
Age                 0
Sex                 0
Steroid             1
Antivirals          0
Fatigue             1
Malaise             1
Anorexia            1
Liver Big          10
Liver Firm         11
Spleen Palpable     5
Spiders             5
Ascities            5
Varices             5
Bilirubin           6
ALK Poshphate      29
SGOT                4
Albumin            16
Protime            67
Histology           0
dtype: int64
```

Here we are showing the unique number of values in given feature.

```
class_unique=data['Class'].nunique()
print(f'unique values in class are:{class_unique}')

Age_unique=data['Age'].nunique()
print(f'unique values in Age are:{Age_unique}')

Sex_unique=data['Sex'].nunique()
print(f'unique values in Sex are:{Sex_unique}')

Steroid_unique=data['Steroid'].nunique()
print(f'unique values in Steroid are:{Steroid_unique}')

Fatigue_unique=data['Fatigue'].nunique()
print(f'unique values in Fatigue are:{Fatigue_unique}')

Malaise_unique=data['Malaise'].nunique()
print(f'unique values in Malaise are:{Malaise_unique}')

Anorexia_unique=data['Anorexia'].nunique()
print(f'unique values in Anorexia are:{Anorexia_unique}')

Liver_Big_unique=data['Liver Big'].nunique()
print(f'unique values in Liver big are:{Liver_Big_unique}')
```

```
unique values in class are:2
unique values in Age are:49
unique values in Sex are:2
unique values in Steroid are:2
unique values in Fatigue are:2
unique values in Malaise are:2
unique values in Anorexia are:2
unique values in Liver big are:2
```

Data Preprocessing

We're going to remove the rows with NaN Values and do some Class Balancing using SMOTE

Remove the rows with null value.

We're droping the row with null value using dropna.This will help clean the data so that the model can learn better.

```
modified_data = data
modified_data.dropna(inplace=True)
modified_data
```

```
     Unnamed: 0  Class  Age  Sex  Steroid  Antivirals  Fatigue  Malaise  \
5             5      2   34    1      2.0           2      2.0      2.0
10           10      2   39    1      1.0           1      2.0      2.0
11           11      2   32    1      2.0           1      1.0      2.0
12           12      2   41    1      2.0           1      1.0      2.0
13           13      2   30    1      2.0           2      1.0      2.0
..          ...    ...  ...  ...      ...         ...      ...      ...
139         139      2   45    1      2.0           1      2.0      2.0
143         143      1   49    1      1.0           2      1.0      1.0
145         145      2   31    1      1.0           2      1.0      2.0
153         153      2   53    2      1.0           2      1.0      2.0
154         154      1   43    1      2.0           2      1.0      2.0

     Anorexia  Liver Big  ...  Spleen Palpable  Spiders  Ascities  Varices  \
5         2.0        2.0  ...              2.0      2.0       2.0      2.0
10        2.0        1.0  ...              2.0      2.0       2.0      2.0
11        2.0        2.0  ...              2.0      1.0       2.0      2.0
12        2.0        2.0  ...              2.0      2.0       2.0      2.0
13        2.0        2.0  ...              2.0      2.0       2.0      2.0
..        ...        ...  ...              ...      ...       ...      ...
139       2.0        2.0  ...              2.0      2.0       2.0      2.0
143       2.0        2.0  ...              1.0      1.0       2.0      2.0
145       2.0        2.0  ...              2.0      2.0       2.0      2.0
153       2.0        2.0  ...              1.0      1.0       2.0      1.0
154       2.0        2.0  ...              1.0      1.0       1.0      2.0

     Bilirubin  ALK Poshphate   SGOT  Albumin  Protime  Histology
5          0.9           95.0   28.0      4.0     75.0          1
10         1.3           78.0   30.0      4.4     85.0          1
11         1.0           59.0  249.0      3.7     54.0          1
12         0.9           81.0   60.0      3.9     52.0          1
13         2.2           57.0  144.0      4.9     78.0          1
..         ...            ...    ...      ...      ...        ...
139        1.3           85.0   44.0      4.2     85.0          2
143        1.4           85.0   70.0      3.5     35.0          2
145        1.2           75.0  173.0      4.2     54.0          2
153        1.5           81.0   19.0      4.1     48.0          2
154        1.2          100.0   19.0      3.1     42.0          2

[80 rows x 21 columns]
```

```
# to confirm that no Null values are there.
modified_data.isnull().sum()
```

```
Unnamed: 0         0
Class              0
Age                0
Sex                0
Steroid            0
Antivirals         0
```

```
Fatigue            0
Malaise            0
Anorexia           0
Liver Big          0
Liver Firm         0
Spleen Palpable    0
Spiders            0
Ascities           0
Varices            0
Bilirubin          0
ALK Poshphate      0
SGOT               0
Albumin            0
Protime            0
Histology          0
dtype: int64
```

Number of Male and Female after filtering.

```
modified_data["Class"].value_counts()
```

```
Class
2    67
1    13
Name: count, dtype: int64
```

From above result we can see that the class is imbalace, so we're going to do some class balancing technique and other preprocessing method

Here we're going to use SMOTE(Synthetic Minority Oversampling Technique) for Class Balancing.

Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way.

```
x = modified_data.drop(['Class'], axis=1)
y = modified_data['Class']
sm = SMOTE(random_state=42, k_neighbors=5)
X_res, y_res = sm.fit_resample(x,y)
X_res["Class"]=y_res
data_smote = X_res.drop("Unnamed: 0", axis = 1)
```

```
data_smote["Class"].value_counts()
```

```
Class
2    67
1    67
Name: count, dtype: int64
```

Here we are using Min Max Scalar

We are using Minmaxscalar on following features:

- ALK Poshphate
- Protime

```
# Feature scaling
```

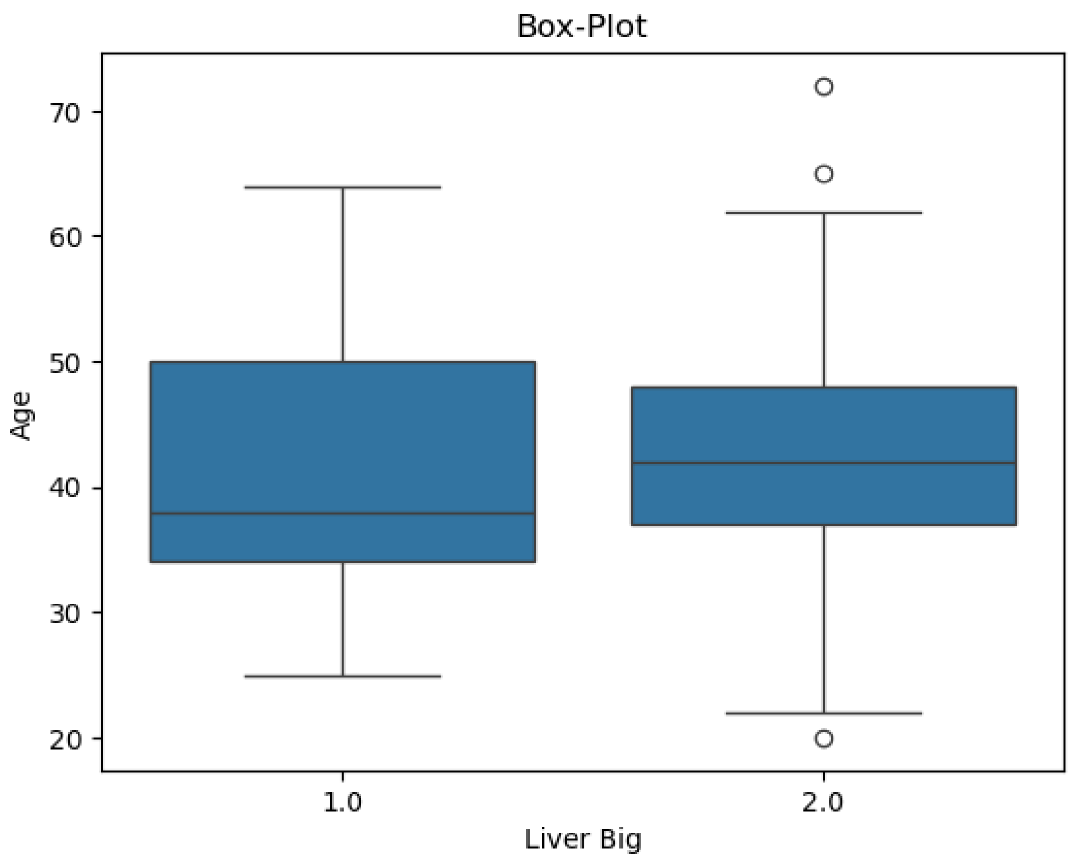
Here we are using MinMaxScalar on `ALK Poshphate, Protime feature to range of (0,1)`

```
minmax = MinMaxScaler(feature_range=(0,1))
data_smote["ALK Poshphate"]= minmax.fit_transform(data_smote["ALK Poshphate"].values.reshape(-1,1))
data_smote["Protime"]=
minmax.fit_transform(data_smote["Protime"].values.reshape(-1,1))
data_smote.head()

   Age  Sex  Steroid  Antivirals  Fatigue  Malaise  Anorexia  Liver Big  \
0   34    1      2.0           2      2.0      2.0       2.0        2.0
1   39    1      1.0           1      2.0      2.0       2.0        1.0
2   32    1      2.0           1      1.0      2.0       2.0        2.0
3   41    1      2.0           1      1.0      2.0       2.0        2.0
4   30    1      2.0           2      1.0      2.0       2.0        2.0

   Liver Firm  Spleen Palpable  Spiders  Ascities  Varices  Bilirubin  \
0         2.0              2.0      2.0       2.0      2.0        0.9
1         1.0              2.0      2.0       2.0      2.0        1.3
2         1.0              2.0      1.0       2.0      2.0        1.0
3         1.0              2.0      2.0       2.0      2.0        0.9
4         1.0              2.0      2.0       2.0      2.0        2.2

   ALK Poshphate   SGOT  Albumin  Protime  Histology  Class
0       0.271654   28.0      4.0     0.75          1      2
1       0.204724   30.0      4.4     0.85          1      2
2       0.129921  249.0      3.7     0.54          1      2
3       0.216535   60.0      3.9     0.52          1      2
4       0.122047  144.0      4.9     0.78          1      2

# Displaying the Data after Preprocessing.
print(data_smote)

     Age  Sex  Steroid  Antivirals  Fatigue  Malaise  Anorexia  Liver Big  \
0     34    1      2.0           2      2.0      2.0       2.0        2.0
1     39    1      1.0           1      2.0      2.0       2.0        1.0
2     32    1      2.0           1      1.0      2.0       2.0        2.0
3     41    1      2.0           1      1.0      2.0       2.0        2.0
4     30    1      2.0           2      1.0      2.0       2.0        2.0
..   ...  ...      ...         ...      ...      ...       ...        ...
129   39    1      1.0           1      1.0      1.0       2.0        2.0
130   39    1      1.0           1      1.0      1.0       2.0        2.0
131   58    1      1.0           2      1.0      1.0       2.0        2.0
```

```
132   47    1      2.0           2      1.0      1.0       2.0        2.0
133   41    1      1.0           1      1.0      1.0       2.0        2.0

     Liver Firm  Spleen Palpable   Spiders  Ascities   Varices  Bilirubin  \
0      2.000000         2.000000  2.000000  2.000000  2.000000   0.900000
1      1.000000         2.000000  2.000000  2.000000  2.000000   1.300000
2      1.000000         2.000000  1.000000  2.000000  2.000000   1.000000
3      1.000000         2.000000  2.000000  2.000000  2.000000   0.900000
4      1.000000         2.000000  2.000000  2.000000  2.000000   2.200000
..          ...              ...       ...       ...       ...        ...
129    1.000000         2.000000  1.974581  1.974581  1.974581   2.363548
130    1.107891         2.000000  1.892109  2.000000  2.000000   2.105795
131    1.000000         1.031429  1.000000  1.968571  1.968571   1.603716
132    1.000000         2.000000  1.636410  1.000000  1.000000   1.445487
133    1.000000         2.000000  1.685644  1.685644  1.685644   3.085890

     ALK Poshphate        SGOT   Albumin   Protime  Histology  Class
0         0.271654   28.000000  4.000000  0.750000          1      2
1         0.204724   30.000000  4.400000  0.850000          1      2
2         0.129921  249.000000  3.700000  0.540000          1      2
3         0.216535   60.000000  3.900000  0.520000          1      2
4         0.122047  144.000000  4.900000  0.780000          1      2
..             ...         ...       ...       ...        ...    ...
129       0.984288   99.499728  3.772039  0.397712          1      1
130       0.907400   94.763257  3.800000  0.388132          1      1
131       0.320877  157.000000  3.571714  0.377800          2      1
132       0.350737   23.635896  2.281795  0.405462          2      1
133       0.805693  116.547003  3.454208  0.371708          1      1

[134 rows x 20 columns]
```

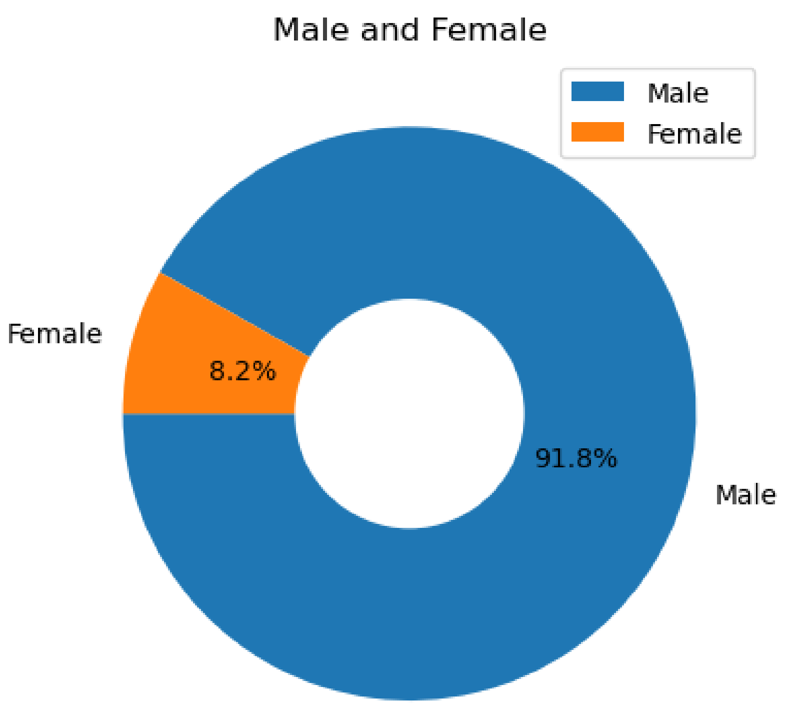
Data Visualization

```
# Boxplot of both Fatigue and Age

sns.boxplot(x="Liver Big", y="Age", data=data_smote);
plt.title("Box-Plot");
```



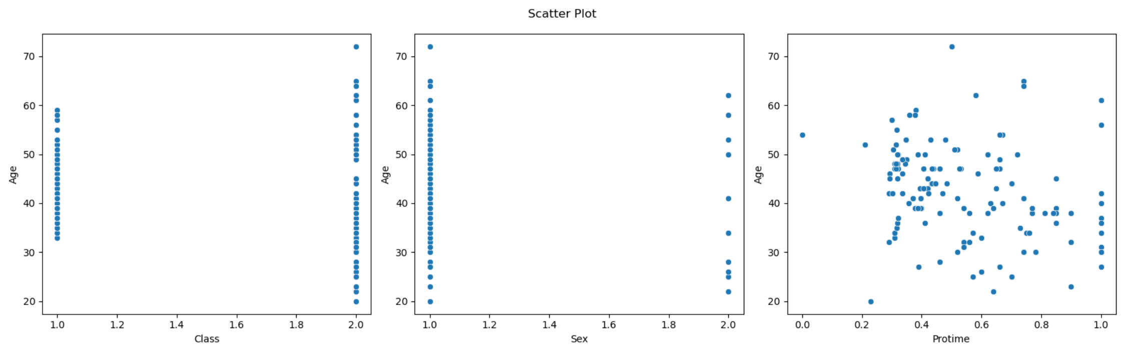
The below visualisation is pie chart of Male and female in our dataset.

```
# Pie Chart of Sex Distribution

plt.pie(data_smote["Sex"].value_counts(), labels=["Male", "Female"],
autopct="%1.1f%%", startangle=180)
center_circle=plt.Circle((0,0),0.40,fc="white")
fig = plt.gcf()
fig.gca().add_artist(center_circle)
plt.title("Male and Female")
plt.legend()
plt.show()
```

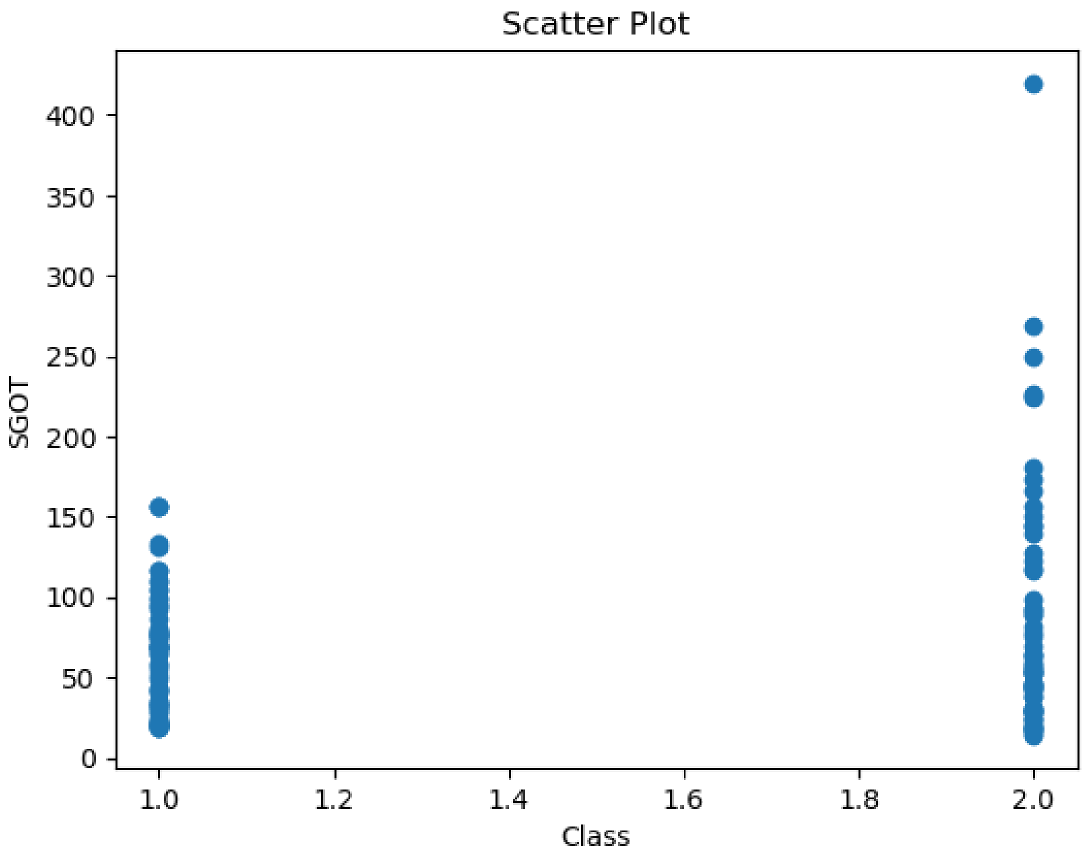


The next visulaisation is Scatter plot on diffeerent features.

```
# assign thee required values

_ , axes = plt.subplots(nrows = 1, ncols = 3, figsize=(16, 5))

scatterplot1 = sns.scatterplot(x='Class', y='Age', data=data_smote,
ax=axes[0])
scatterplot2 = sns.scatterplot(x='Sex', y='Age', data=data_smote, ax=axes[1])
scatterplot3 = sns.scatterplot(x='Protime', y='Age', data=data_smote,
ax=axes[2])

plt.suptitle("Scatter Plot");

# Adjust layout
plt.tight_layout()
```

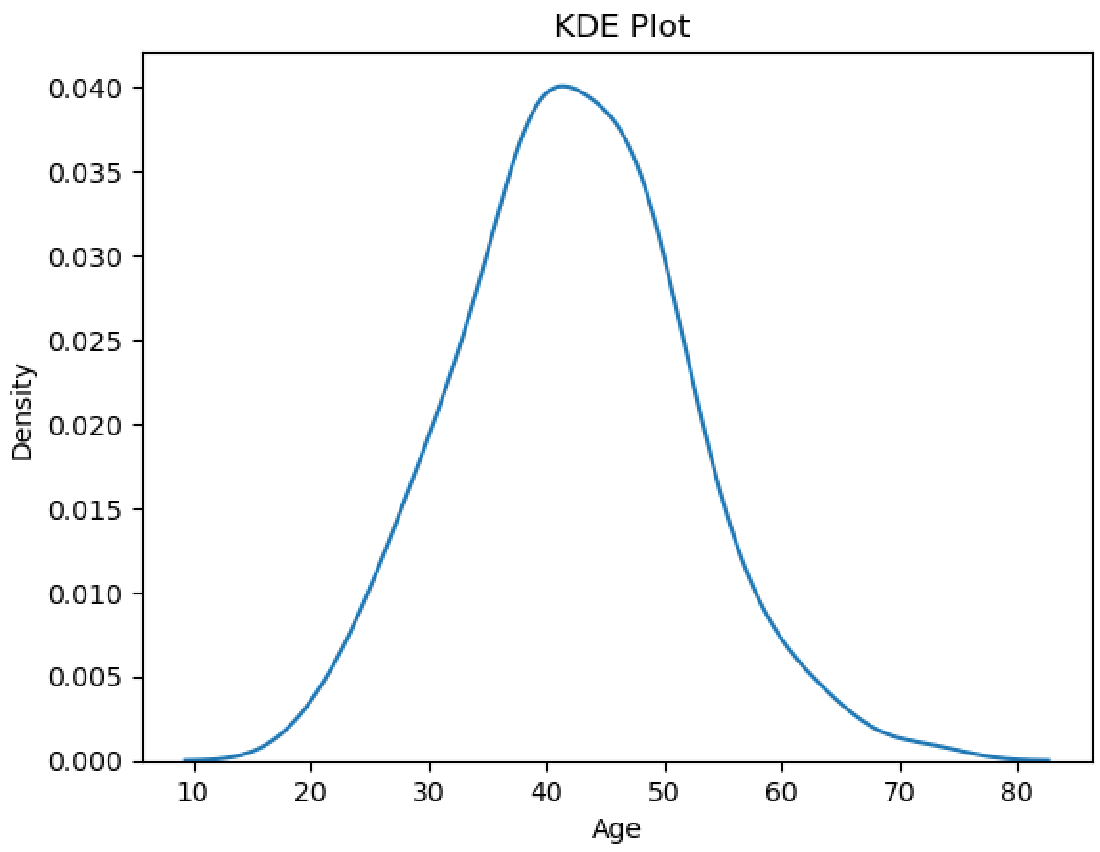


```
# Using Scatter Plot

plt.scatter(data_smote["Class"],data_smote["SGOT"]);
plt.xlabel("Class")
plt.ylabel("SGOT")
plt.title("Scatter Plot");
```

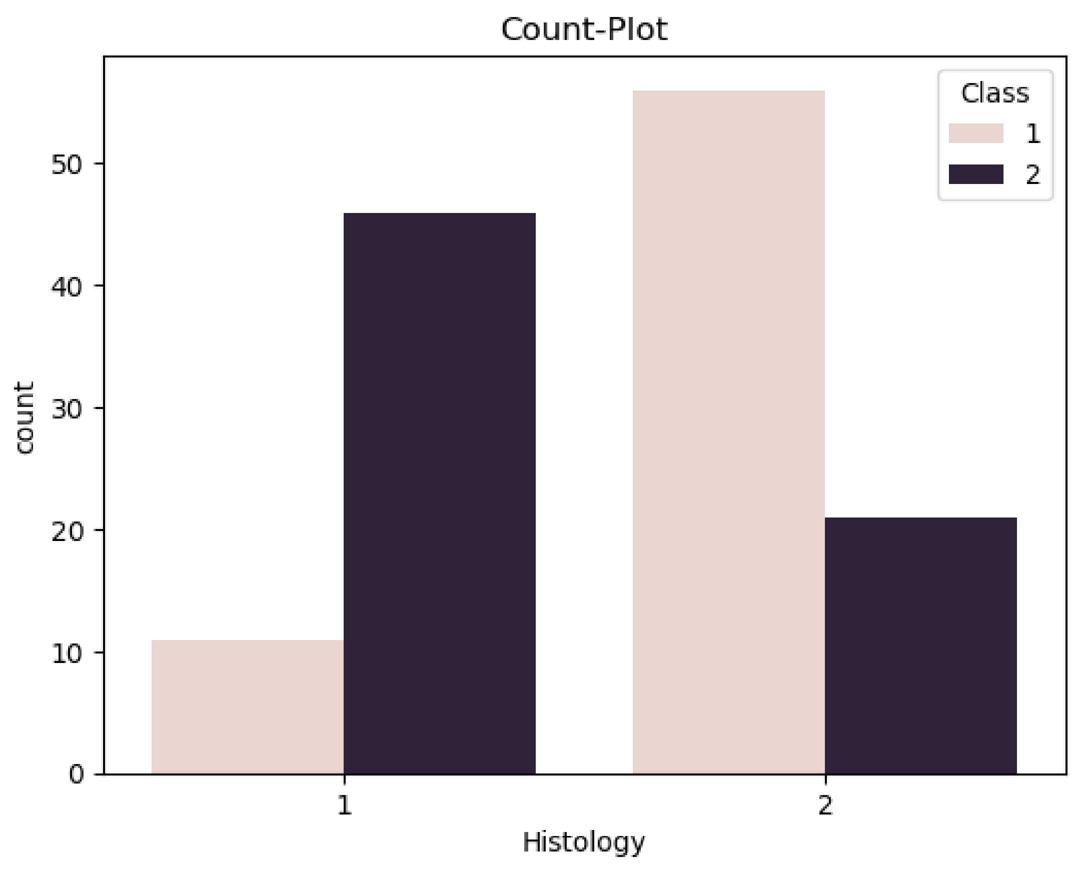


```
# Kernel Density Estimation Plot on Age
sns.kdeplot(data_smote["Age"]);
plt.title(" KDE Plot")
```

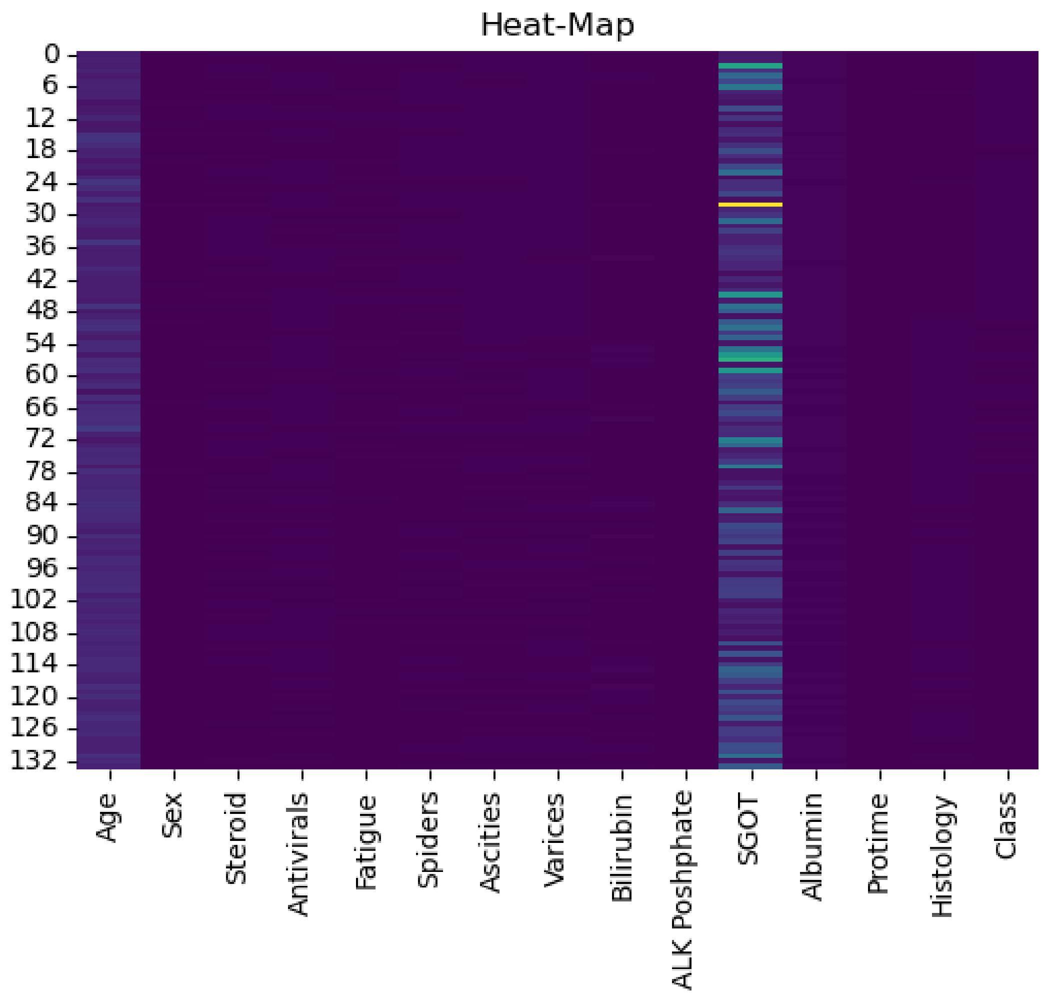
```
Text(0.5, 1.0, ' KDE Plot')
```



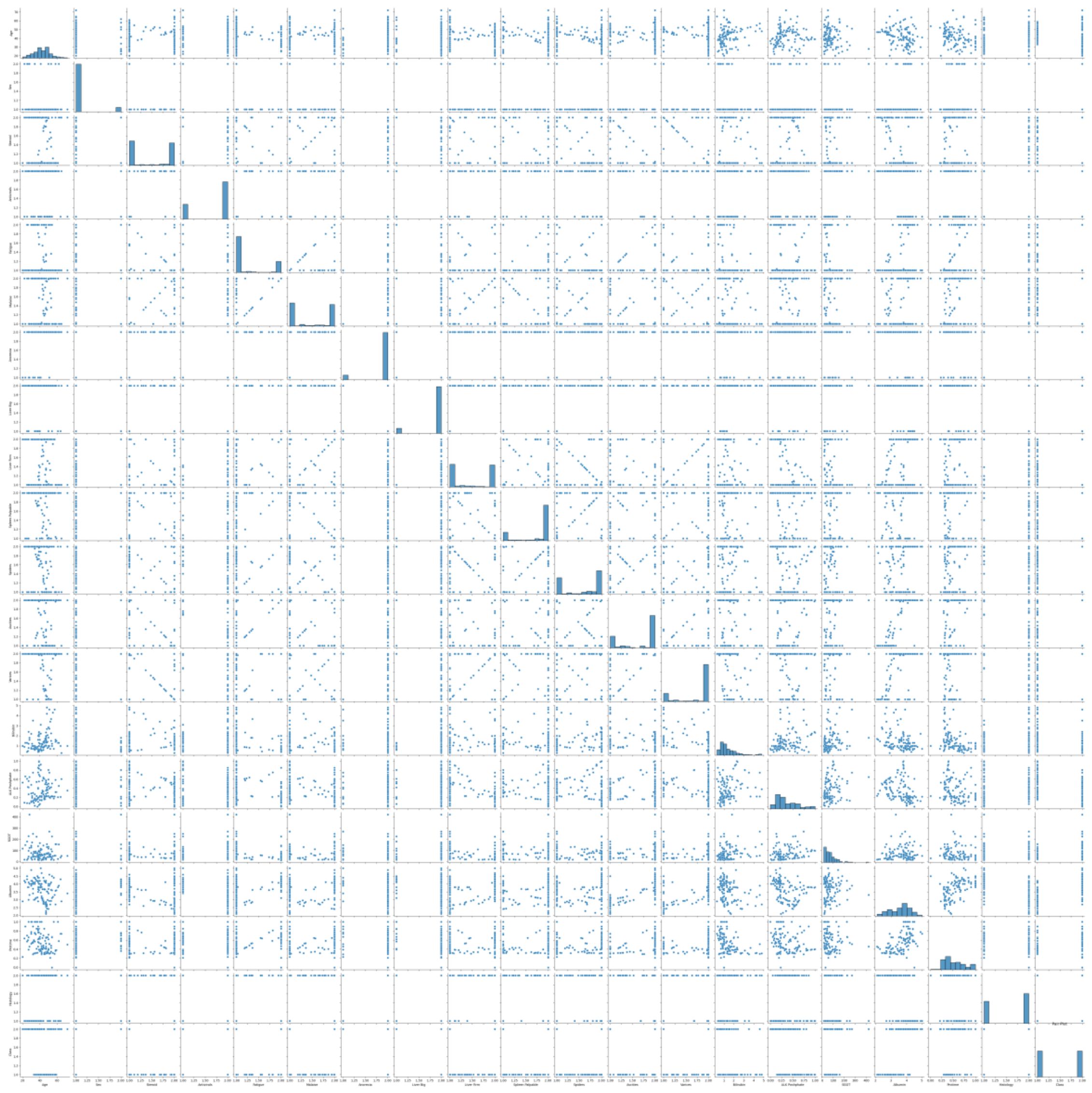
```
# Count plot on Fatigue
sns.countplot(x="Histology",data=data_smote,hue ="Class" );
plt.title("Count-Plot");
```



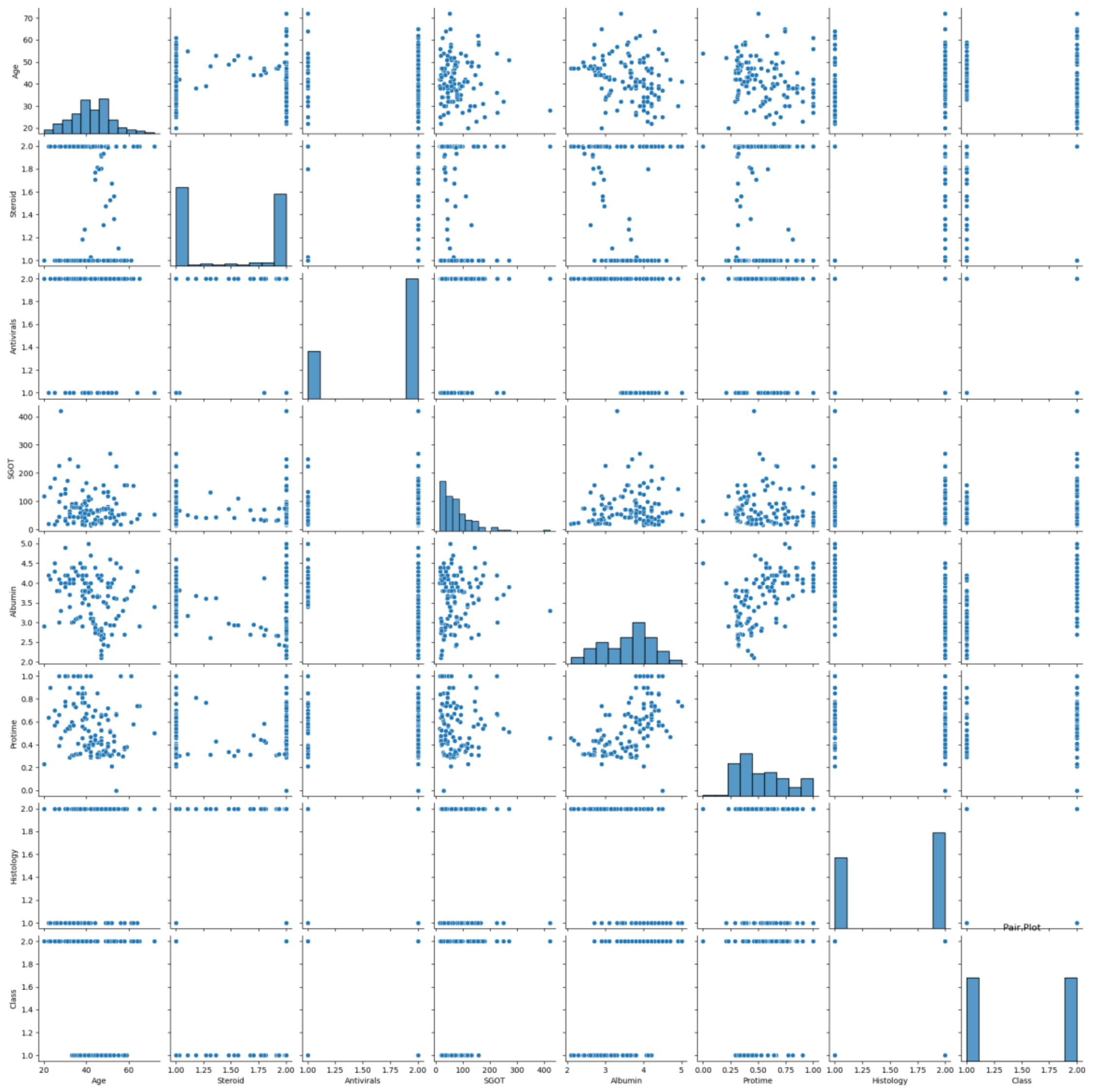
```
# Here we droped the data and did heatmap on the modified dataset.

mod = data_smote.drop(data_smote.columns[[5,6,7,8,9]],axis=1)
sns.heatmap(data=mod,cbar=False,cmap="viridis");
plt.title("Heat-Map");
```



```
# Orginal Pair Plot with every Feature
sns.pairplot(data_smote);
plt.title("Pair Plot");
```



```
# ReducedPair Plot with every Feature
modified_plot =
data_smote.drop(data_smote.columns[[1,4,5,6,7,8,9,10,11,12,13,14]],axis=1)
sns.pairplot(modified_plot);
plt.title("Pair Plot");
```



Spliting the data and Fitting the Model

```
# Here we split the dataset into two data and target
X = data_smote.drop(columns=[ "Class"])
y = data_smote["Class"]
X

     Age  Sex  Steroid  Antivirals  Fatigue  Malaise  Anorexia  Liver Big  \
0     34    1      2.0           2      2.0      2.0       2.0        2.0
1     39    1      1.0           1      2.0      2.0       2.0        1.0
2     32    1      2.0           1      1.0      2.0       2.0        2.0
3     41    1      2.0           1      1.0      2.0       2.0        2.0
4     30    1      2.0           2      1.0      2.0       2.0        2.0
..   ...  ...      ...         ...      ...      ...       ...        ...
129   39    1      1.0           1      1.0      1.0       2.0        2.0
130   39    1      1.0           1      1.0      1.0       2.0        2.0
131   58    1      1.0           2      1.0      1.0       2.0        2.0
```

```
132   47    1      2.0           2      1.0      1.0       2.0        2.0
133   41    1      1.0           1      1.0      1.0       2.0        2.0

     Liver Firm  Spleen Palpable   Spiders  Ascities   Varices  Bilirubin  \
0      2.000000         2.000000  2.000000  2.000000  2.000000   0.900000
1      1.000000         2.000000  2.000000  2.000000  2.000000   1.300000
2      1.000000         2.000000  1.000000  2.000000  2.000000   1.000000
3      1.000000         2.000000  2.000000  2.000000  2.000000   0.900000
4      1.000000         2.000000  2.000000  2.000000  2.000000   2.200000
..          ...              ...       ...       ...       ...        ...
129    1.000000         2.000000  1.974581  1.974581  1.974581   2.363548
130    1.107891         2.000000  1.892109  2.000000  2.000000   2.105795
131    1.000000         1.031429  1.000000  1.968571  1.968571   1.603716
132    1.000000         2.000000  1.636410  1.000000  1.000000   1.445487
133    1.000000         2.000000  1.685644  1.685644  1.685644   3.085890

     ALK Poshphate        SGOT   Albumin   Protime  Histology
0         0.271654   28.000000  4.000000  0.750000          1
1         0.204724   30.000000  4.400000  0.850000          1
2         0.129921  249.000000  3.700000  0.540000          1
3         0.216535   60.000000  3.900000  0.520000          1
4         0.122047  144.000000  4.900000  0.780000          1
..             ...         ...       ...       ...        ...
129       0.984288   99.499728  3.772039  0.397712          1
130       0.907400   94.763257  3.800000  0.388132          1
131       0.320877  157.000000  3.571714  0.377800          2
132       0.350737   23.635896  2.281795  0.405462          2
133       0.805693  116.547003  3.454208  0.371708          1

[134 rows x 19 columns]
```

```
def evaluate_preds(y_true, y_preds):
    """
    performs evaluations comparison on y+true labels vs y_preds labels.
    on a Classifications
    """
    accuracy = accuracy_score(y_true, y_preds)
    precision = precision_score(y_true, y_preds)
    recall = recall_score(y_true, y_preds)
    f1 = f1_score(y_true, y_preds)
    metric_dict = {"accuracy": round(accuracy, 2),
                  "precison": round(precision, 2),
                  "recall": round(recall, 2),
                  "f1": round(f1, 2)}

    return metric_dict
```

KNN Classifier(K Neighbours)

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using KNeighborClassifier with certain parameter

knn = KNeighborsClassifier(n_neighbors=1,weights="distance",p=1)

# Fitting(Training) the model with training dataset (x_train, y_train)

knn.fit(x_train,y_train)

# Predicting the output of x_test
y_pred1 = knn.predict(x_test.values)

# Evaluating

accuracy_knn = accuracy_score(y_test, y_pred1)
report_knn = evaluate_preds(y_test, y_pred1)
print(classification_report(y_test, y_pred1))

print(f"The Accuracy of the model is : {accuracy_knn:.2f}")
              precision    recall  f1-score   support

           1       0.82      0.95      0.88        19
           2       0.95      0.82      0.88        22

    accuracy                           0.88        41
   macro avg       0.88      0.88      0.88        41
weighted avg       0.89      0.88      0.88        41

The Accuracy of the model is : 0.88
```

SVM (Support vector machines)

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using SVC with certain parameter
clf_svm = svm.SVC(kernel='linear', C = 100, gamma = 0.001)

# Fitting(Training) the model with training dataset (x_train, y_train)
clf_svm.fit(x_train, y_train)

# Predicting the output of x_test
y_pred2 = clf_svm.predict(x_test)

# Evaluating

accuracy_svm = accuracy_score(y_test, y_pred2)
```

```
report_svm = evaluate_preds(y_test, y_pred2)
print(classification_report(y_test, y_pred2))
print(f"The Accuracy of the model is : {accuracy_svm:.2f}")
```

```
              precision    recall  f1-score   support

           1       0.90      1.00      0.95        19
           2       1.00      0.91      0.95        22

    accuracy                           0.95        41
   macro avg       0.95      0.95      0.95        41
weighted avg       0.96      0.95      0.95        41

The Accuracy of the model is : 0.95
```

Linear Regression

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using linaer Regression with certain parameter
LR = linear_model.LogisticRegression(max_iter=1000, solver="lbfgs",
verbose=0)

# Fitting(Training) the model with training dataset (x_train, y_train)
LR.fit(x_train, y_train)

# Predicting the output of x_test
y_pred3 = LR.predict(x_test)

# Evaluating

accuracy_lr = accuracy_score(y_test, y_pred3)
report_lr = evaluate_preds(y_test, y_pred3)
print(classification_report(y_test, y_pred3))
print(f"The Accuracy of the model is : {accuracy_lr:.2f}")
```

```
              precision    recall  f1-score   support

           1       0.90      1.00      0.95        19
           2       1.00      0.91      0.95        22

    accuracy                           0.95        41
   macro avg       0.95      0.95      0.95        41
weighted avg       0.96      0.95      0.95        41

The Accuracy of the model is : 0.95
```

Random Forest Classifier

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using Random Forest Regressor with certain parameter
clf = RandomForestClassifier(n_estimators=10, random_state=42,
oob_score=True)

# Fitting(Training) the model with training dataset (x_train, y_train)
clf.fit(x_train, y_train)

# Predicting the output of x_test
y_pred4 = clf.predict(x_test)

# Evaluating
accuracy_clf = accuracy_score(y_test, y_pred4)
report_clf = evaluate_preds(y_test, y_pred4)
print(classification_report(y_test, y_pred4))
print(f"The Accuracy of the model is : {accuracy_clf:.2f}")
```

```
              precision    recall  f1-score   support

           1       0.86      1.00      0.93        19
           2       1.00      0.86      0.93        22

    accuracy                           0.93        41
   macro avg       0.93      0.93      0.93        41
weighted avg       0.94      0.93      0.93        41

The Accuracy of the model is : 0.93
```

Naive Bayes Classification (Gaussian)

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using Naive Bayes with certain parameter
naiveB = GaussianNB()

# Fitting(Training) the model with training dataset (x_train, y_train)
naiveB.fit(x_train, y_train)

# Predicting the output of x_test
y_pred5 = regressor.predict(x_test)

# Evaluating
accuracy_naive = accuracy_score(y_test, y_pred5)
report_naive = evaluate_preds(y_test, y_pred5)
```

```
print(classification_report(y_test, y_pred5))
print(f"The Accuracy of the model is : {accuracy_naive:.2f}")
```

```
              precision    recall  f1-score   support

           1       0.86      1.00      0.93        19
           2       1.00      0.86      0.93        22

    accuracy                           0.93        41
   macro avg       0.93      0.93      0.93        41
weighted avg       0.94      0.93      0.93        41

The Accuracy of the model is : 0.93
```

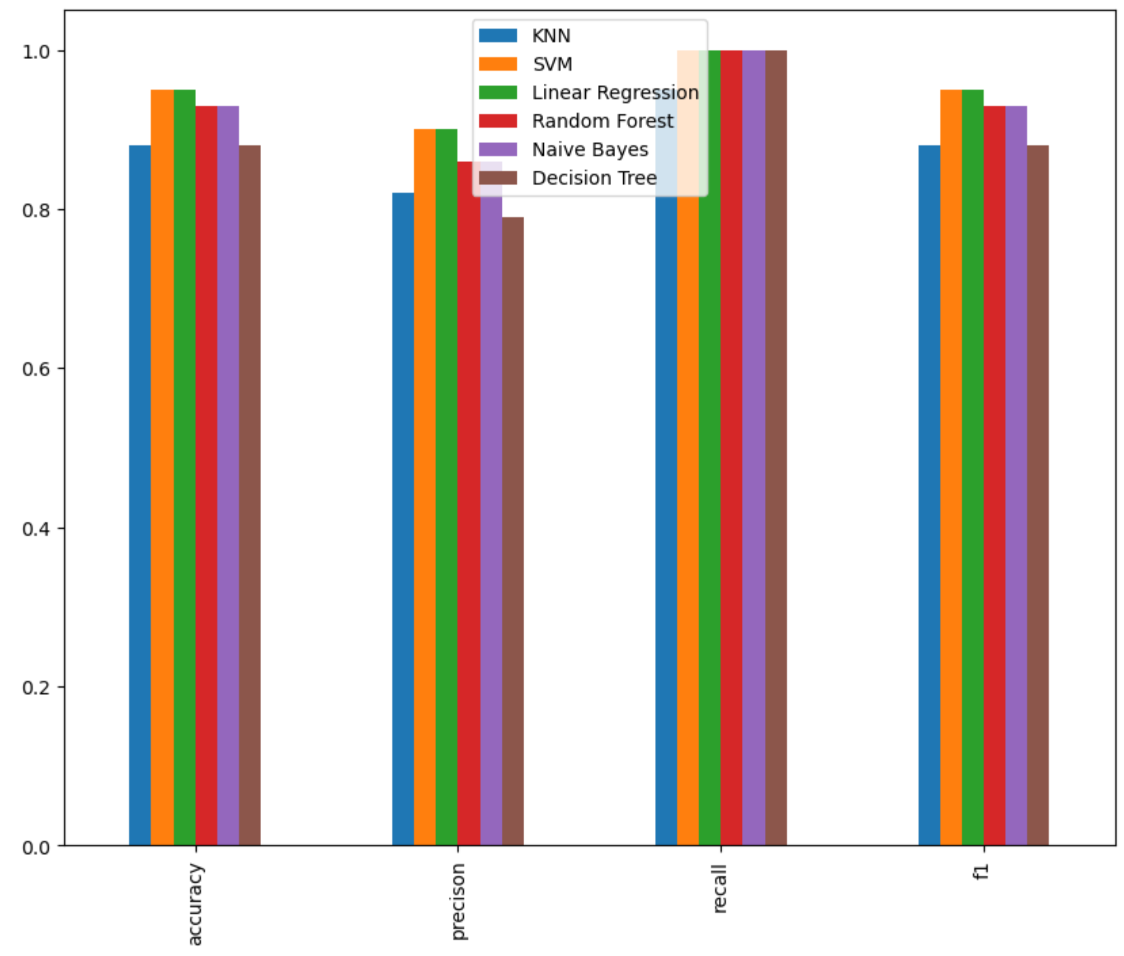
Decision Tree Classification

```
# Splitting Of Dataset into training and testing
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size =
0.3,random_state=42)

# Using Decision Tree with certain parameter
clf_tree = DecisionTreeClassifier(max_depth=11)

# Fitting(Training) the model with training dataset (x_train, y_train)
clf_tree.fit(x_train, y_train)

# Predicting the output of x_test
y_pred6 = clf_tree.predict(x_test)

# Evaluating
accuracy_tree = accuracy_score(y_test, y_pred6)
report_tree = evaluate_preds(y_test, y_pred6)
print(classification_report(y_test, y_pred6))
print(f"The Accuracy of the model is : {accuracy_tree:.2f}")
```

```
              precision    recall  f1-score   support

           1       0.79      1.00      0.88        19
           2       1.00      0.77      0.87        22

    accuracy                           0.88        41
   macro avg       0.90      0.89      0.88        41
weighted avg       0.90      0.88      0.88        41

The Accuracy of the model is : 0.88
```

Evaluating the Model

```
compare_metrics = pd.DataFrame({"KNN": report_knn,
                               "SVM": report_svm,
                               "Linear Regression":report_lr,
                               "Random Forest":report_clf,
                               "Naive Bayes":report_naive,
                               "Decision Tree":report_tree})
compare_metrics.plot.bar(figsize =(10,8));
```



```
accuracy_Knn = accuracy_score(y_test, y_pred1)
accuracy_naive = accuracy_score(y_test, y_pred5)
print(f"The Accuracy for KNN Model is {accuracy_Knn:.2f}%\nThe accuracy for
naive Bayes is {accuracy_naive:.2f}% ")
```

```
The Accuracy for KNN Model is 0.74%
The accuracy for naive Bayes is 0.79%
```

```
# Classification Report on KNN
report_knn = classification_report(y_test, y_pred1)
print(report_knn)
```

```
              precision    recall  f1-score   support

           1       0.25      0.60      0.35         5
           2       0.94      0.76      0.84        38
```

```
    accuracy                           0.74        43
   macro avg       0.59      0.68      0.60        43
weighted avg       0.86      0.74      0.78        43
```

```
# Classification Report on Naive Bayes
report_naive = classification_report(y_test, y_pred5)
print(report_naive)
```

```
              precision    recall  f1-score   support

           1       0.33      0.80      0.47         5
           2       0.97      0.79      0.87        38

    accuracy                           0.79        43
   macro avg       0.65      0.79      0.67        43
weighted avg       0.89      0.79      0.82        43
```

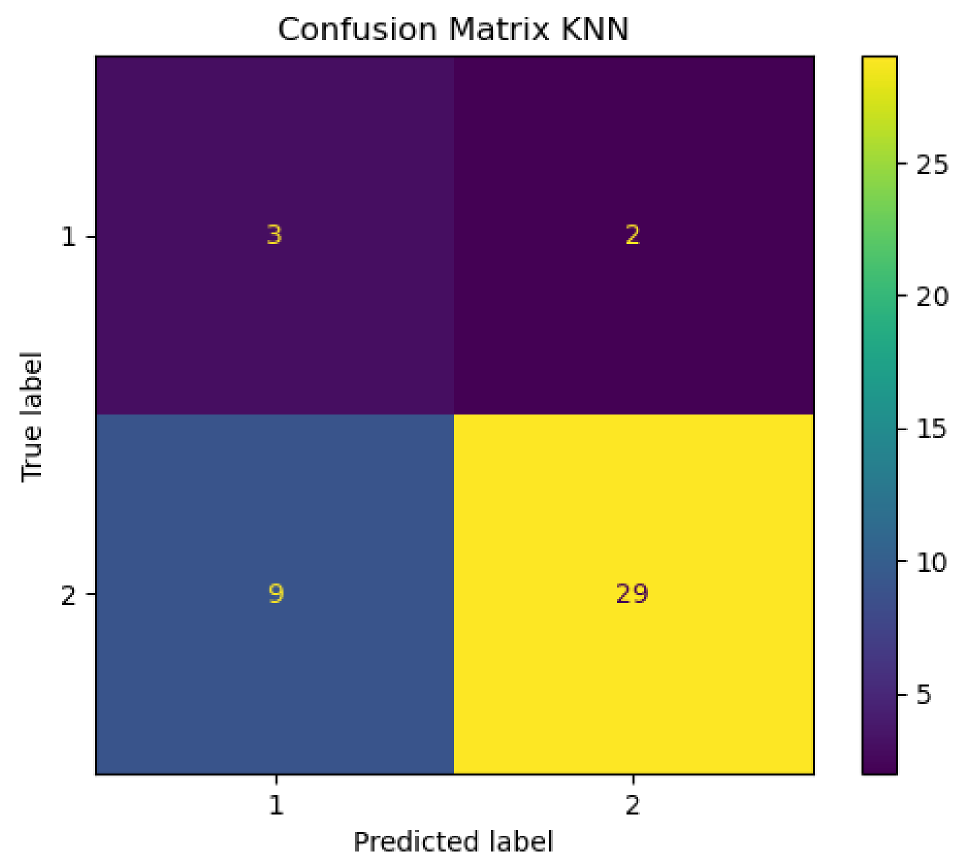
```
# Confusion Matrix for Knn
confusion_matrix(y_test, y_pred1)
```

```
array([[ 3,  2],
       [ 9, 29]], dtype=int64)
```

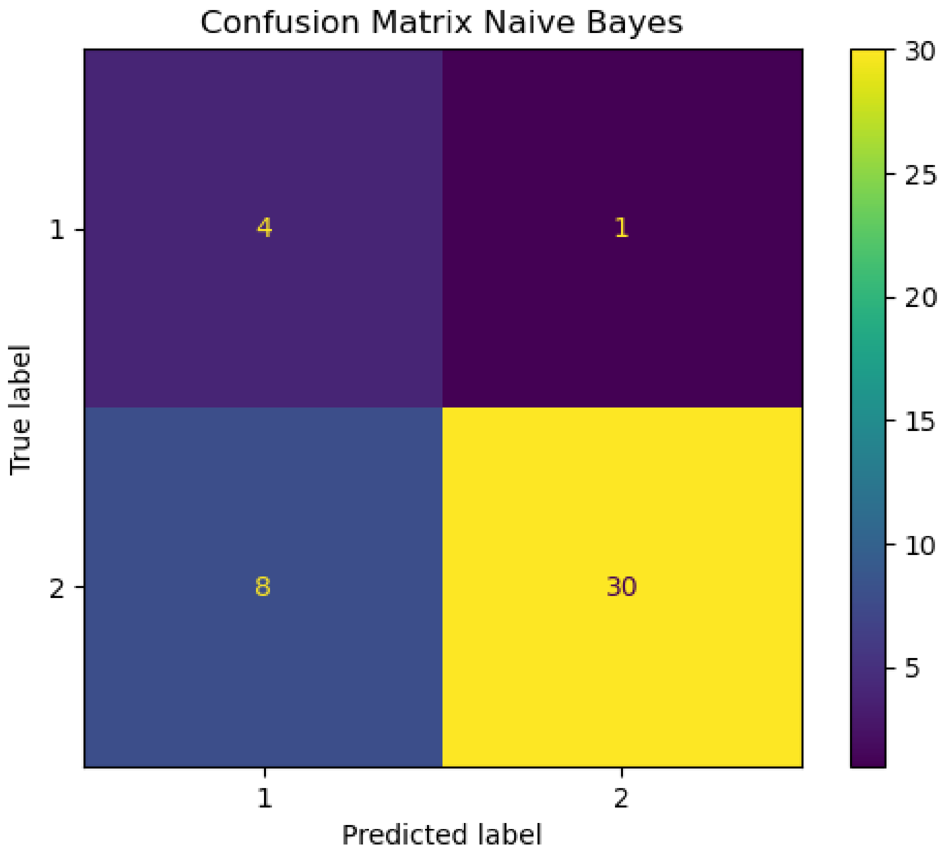
```
confusion_matrix(y_test, y_pred5)
```

```
array([[ 4,  1],
       [ 8, 30]], dtype=int64)
```

```
ConfusionMatrixDisplay.from_predictions(y_true = y_test,
                                      y_pred = y_pred1);
plt.title("Confusion Matrix KNN");
```



```
ConfusionMatrixDisplay.from_predictions(y_true = y_test,
                                        y_pred = y_pred5);
plt.title("Confusion Matrix Naive Bayes");
```



Confusion Matrix Naive Bayes
True label
Predicted label
1
2
1
2
4
1
8
30
30
25
20
15
10
5