Liver Disease Dataset

```
import numpy as np

import pandas as pd

import seaborn as sns

df = pd.read_csv("indian_liver_patient.csv")
df.head()

   Age  Gender  Total_Bilirubin  Direct_Bilirubin
Alkaline_Phosphotase  \
0   65  Female              0.7               0.1
187
1   62    Male             10.9               5.5
699
2   62    Male              7.3               4.1
490
3   58    Male              1.0               0.4
182
4   72    Male              3.9               2.0
195

   Alamine_Aminotransferase  Aspartate_Aminotransferase
Total_Protiens  \
0                        16                          18
6.8
1                        64                         100
7.5
2                        60                          68
7.0
3                        14                          20
6.8
4                        27                          59
7.3

   Albumin  Albumin_and_Globulin_Ratio  Dataset
0      3.3                        0.90        1
1      3.2                        0.74        1
2      3.3                        0.89        1
3      3.4                        1.00        1
4      2.4                        0.40        1

df

     Age  Gender  Total_Bilirubin  Direct_Bilirubin
Alkaline_Phosphotase  \
0     65  Female              0.7               0.1
187
```

```
1     62    Male             10.9               5.5
699
2     62    Male              7.3               4.1
490
3     58    Male              1.0               0.4
182
4     72    Male              3.9               2.0
195
..   ...     ...              ...               ...
...
578   60    Male              0.5               0.1
500
579   40    Male              0.6               0.1
98
580   52    Male              0.8               0.2
245
581   31    Male              1.3               0.5
184
582   38    Male              1.0               0.3
216

     Alamine_Aminotransferase  Aspartate_Aminotransferase
Total_Protiens  \
0                          16                          18
6.8
1                          64                         100
7.5
2                          60                          68
7.0
3                          14                          20
6.8
4                          27                          59
7.3
..                        ...                         ...
...
578                        20                          34
5.9
579                        35                          31
6.0
580                        48                          49
6.4
581                        29                          32
6.8
582                        21                          24
7.3

     Albumin  Albumin_and_Globulin_Ratio  Dataset
0        3.3                        0.90        1
1        3.2                        0.74        1
```

```
2        3.3                        0.89        1
3        3.4                        1.00        1
4        2.4                        0.40        1
..       ...                         ...      ...
578      1.6                        0.37        2
579      3.2                        1.10        1
580      3.2                        1.00        1
581      3.4                        1.00        1
582      4.4                        1.50        2

[583 rows x 11 columns]
```

```
df.tail(5)
```

```
     Age Gender  Total_Bilirubin  Direct_Bilirubin
Alkaline_Phosphotase  \
578   60   Male              0.5               0.1
500
579   40   Male              0.6               0.1
98
580   52   Male              0.8               0.2
245
581   31   Male              1.3               0.5
184
582   38   Male              1.0               0.3
216

     Alamine_Aminotransferase  Aspartate_Aminotransferase
Total_Protiens  \
578                        20                          34
5.9
579                        35                          31
6.0
580                        48                          49
6.4
581                        29                          32
6.8
582                        21                          24
7.3

     Albumin  Albumin_and_Globulin_Ratio  Dataset
578      1.6                        0.37        2
579      3.2                        1.10        1
580      3.2                        1.00        1
581      3.4                        1.00        1
582      4.4                        1.50        2
```

```
#Predict if a patient has liver disease (1) or not (0) using medical attributes
```

**********This is the following step on next **********

#Data Preprocessing

#Exploring Dataset (Missing Values)

#Encoding

#Scalling

#Balancing Classes

#Feature Selection

#Train Test Split

```
#Exploring Dataset (Missing Values)

df.isnull().sum()

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

df.dropna(inplace=True)

#df.dropna(inplace=True) Pandas ka ek statement hai jo missing values (NaN) wale rows ko remove karta hai.

df.shape

(579, 11)
```

```
#df.duplicated().sum() Pandas me duplicate rows count karne ke liye
use hota hai.

df.duplicated().sum()

13

#df.drop_duplicates(inplace=True) Pandas ka use duplicate rows ko
DataFrame se permanently remove karne ke liye hota hai.

df.drop_duplicates(inplace=True)




#Encoding

#Gender is a categorical feature with values Male and Female.
#Convert it into numeric form using Label Encoding or One-Hot Encoding

#Example male--1(Encoded_Value) female--0(Encoded_Value)

#□ Encoding helps ML models process non-numeric (categorical) data
effectively.

#using sklearn

from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

encoder=LabelEncoder()
df['Gender']=encoder.fit_transform(df['Gender'])
df.head(2)

   Age  Gender  Total_Bilirubin  Direct_Bilirubin
Alkaline_Phosphotase  \
0   65       0              0.7               0.1
187
1   62       0             10.9               5.5
699

   Alamine_Aminotransferase  Aspartate_Aminotransferase
Total_Protiens  \
0                        16                          18
6.8
1                        64                         100
7.5

   Albumin  Albumin_and_Globulin_Ratio  Dataset
0      3.3                        0.90        1
1      3.2                        0.74        1

df.tail
```

```
<bound method NDFrame.tail of      Age  Gender  Total_Bilirubin  
Direct_Bilirubin  Alkaline_Phosphotase  \
0     65       0              0.7               0.1
187
1     62       0             10.9               5.5
699
2     62       0              7.3               4.1
490
3     58       0              1.0               0.4
182
4     72       0              3.9               2.0
195
..   ...     ...              ...               ...
...
578   60       0              0.5               0.1
500
579   40       0              0.6               0.1
98
580   52       0              0.8               0.2
245
581   31       0              1.3               0.5
184
582   38       0              1.0               0.3
216

     Alamine_Aminotransferase  Aspartate_Aminotransferase  
Total_Protiens  \
0                          16                          18
6.8
1                          64                         100
7.5
2                          60                          68
7.0
3                          14                          20
6.8
4                          27                          59
7.3
..                        ...                         ...
...
578                        20                          34
5.9
579                        35                          31
6.0
580                        48                          49
6.4
581                        29                          32
6.8
582                        21                          24
7.3
```

```
     Albumin  Albumin_and_Globulin_Ratio  Dataset
0        3.3                        0.90        1
1        3.2                        0.74        1
2        3.3                        0.89        1
3        3.4                        1.00        1
4        2.4                        0.40        1
..       ...                         ...      ...
578      1.6                        0.37        2
579      3.2                        1.10        1
580      3.2                        1.00        1
581      3.4                        1.00        1
582      4.4                        1.50        2

[566 rows x 11 columns]>

# manually
# df['Gender'] = df['Gender'].map({"Male":1, "Female":0})
```

⚙ Feature Scaling

**Scaling ensures all numerical features are on a similar range for better model performance.

**Common methods: StandardScaler: Scales data to mean = 0 and std = 1 MinMaxScaler: Scales values between 0 and 1

Example:

Feature Before After (MinMax)

Age 65 0.78

Total_Bilirubin 10.9 0.85

```
#□ Apply scaling to all numeric columns (except target) before model
training.

from sklearn.preprocessing import StandardScaler ,MinMaxScaler


features = ['Age', 'Total_Bilirubin', 'Direct_Bilirubin',
       'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
       'Aspartate_Aminotransferase', 'Total_Protiens', 'Albumin',
       'Albumin_and_Globulin_Ratio']
```

```
scaler=StandardScaler()
df[features]=scaler.fit_transform(df[features])
df.head(5)

        Age  Gender  Total_Bilirubin  Direct_Bilirubin  
Alkaline_Phosphotase  \
0  1.236928       0        -0.420124         -0.495190              -
0.429625   
1  1.052432       0         1.203777          1.406906              
1.654054   
2  1.052432       0         0.630636          0.913770              
0.803490   
3  0.806437       0        -0.372362         -0.389518              -
0.449974   
4  1.667418       0         0.089335          0.174066              -
0.397068   

   Alamine_Aminotransferase  Aspartate_Aminotransferase  
Total_Protiens  \
0                 -0.352659                   -0.315148        
0.280819   
1                 -0.088755                   -0.033926        
0.925059   
2                 -0.110747                   -0.143671        
0.464887   
3                 -0.363655                   -0.308289        
0.280819   
4                 -0.292181                   -0.174537        
0.740991   

    Albumin  Albumin_and_Globulin_Ratio  Dataset  
0  0.194225                   -0.150315        1  
1  0.068445                   -0.651328        1  
2  0.194225                   -0.181628        1  
3  0.320004                    0.162818        1  
4 -0.937791                   -1.715981        1  




#Data Balancing
```

Class Balance

```
* Check if target classes (0 = No Disease, 1 = Liver Disease) are 
balanced.
```

```
* Imbalanced data can bias the model toward the majority class.

* Use techniques like:
    -SMOTE / Oversampling → add minority samples

    -Undersampling → reduce majority samples

!pip install -q imbalanced-learn


from imblearn.over_sampling import SMOTE

X = df.drop('Dataset',axis=1)
y = df['Dataset']

smote = SMOTE(random_state=42)

X_resampled, y_resampled = smote.fit_resample(X, y)

y_resampled.value_counts()


Dataset
1    404
2    404
Name: count, dtype: int64

#Now i move  Train -Test Split

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(X_resampled,y_resampled
,test_size=0.2,random_state=42)







#Train and Eval Models
```

logistic regression -SVC -Decision Tree -Random Forest

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

model=LogisticRegression()
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
```

```
print("\n\n",classification_report(y_test, y_pred))

print("\n\n", confusion_matrix(y_test, y_pred))



               precision    recall  f1-score   support

           1       0.80      0.57      0.66        76
           2       0.69      0.87      0.77        86

    accuracy                           0.73       162
   macro avg       0.75      0.72      0.72       162
weighted avg       0.74      0.73      0.72       162



 [[43 33]
 [11 75]]

#Random Forest

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,classification_report


model_r=RandomForestClassifier()
model_r.fit(x_train,y_train)

y_pred=model_r.predict(x_test)


print("\n\n",classification_report(y_test, y_pred))

print("\n\n", confusion_matrix(y_test, y_pred))



               precision    recall  f1-score   support

           1       0.83      0.76      0.79        76
           2       0.80      0.86      0.83        86

    accuracy                           0.81       162
   macro avg       0.82      0.81      0.81       162
weighted avg       0.82      0.81      0.81       162



 [[58 18]
 [12 74]]
```

```
#SVC

from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report

model = SVC()

model.fit(x_train,y_train)

y_pred = model.predict(x_test)

print("\n\n",classification_report(y_test, y_pred))

print("\n\n", confusion_matrix(y_test, y_pred))
```

```
               precision    recall  f1-score   support

           1       0.84      0.61      0.70        76
           2       0.72      0.90      0.80        86

    accuracy                           0.76       162
   macro avg       0.78      0.75      0.75       162
weighted avg       0.77      0.76      0.75       162



 [[46 30]
 [ 9 77]]
```

```
#now i find best accuracy  using Random Forest



# test 1
import numpy as np

pred = model_r.predict(np.array([df.iloc[0,:-1]]))

if pred[0] == 1:
    print("Liver Disease")
else:
    print("No Liver Disease")
```

```
Liver Disease

D:\narayan\Python\new\Lib\site-packages\sklearn\base.py:493:
UserWarning: X does not have valid feature names, but
RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```python
# test 2

import numpy as np

pred = model_r.predict(np.array([df.iloc[23,:-1]]))

if pred[0] == 1:
    print("Liver Disease")
else:
    print("No Liver Disease")
```

```
No Liver Disease

D:\narayan\Python\new\Lib\site-packages\sklearn\base.py:493:
UserWarning: X does not have valid feature names, but
RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```python
#Test

actual = df.iloc[0, -1]
print("Actual Value:", actual)
```

```
Actual Value: 1
```

```python
x_test_row = df.iloc[[0], :-1]   # double brackets
pred = model_r.predict(x_test_row)

print("Predicted Value:", pred[0])
```

```
Predicted Value: 1
```

```python
if pred[0] == actual:
    print("□ Model prediction is CORRECT")
else:
    print("□ Model prediction is WRONG")
```

```
□ Model prediction is CORRECT
```

```python
proba = model_r.predict_proba(X_test_row)
print("Prediction Probabilities:", proba)
```

```
Prediction Probabilities: [[0.87 0.13]]
```