```
import pandas as pd
data=pd.read_csv("/content/cirrhosis.csv")
print(data.head())
```

```
   ID  N_Days Status             Drug    Age Sex Ascites Hepatomegaly Spiders  \
0   1     400      D  D-penicillamine  21464   F       Y            Y       Y
1   2    4500      C  D-penicillamine  20617   F       N            Y       Y
2   3    1012      D  D-penicillamine  25594   M       N            N       N
3   4    1925      D  D-penicillamine  19994   F       N            Y       Y
4   5    1504     CL          Placebo  13918   F       N            Y       Y

  Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  \
0     Y       14.5        261.0     2.60   156.0    1718.0  137.95
1     N        1.1        302.0     4.14    54.0    7394.8  113.52
2     S        1.4        176.0     3.48   210.0     516.0   96.10
3     S        1.8        244.0     2.54    64.0    6121.8   60.63
4     N        3.4        279.0     3.53   143.0     671.0  113.15

   Tryglicerides  Platelets  Prothrombin  Stage
0          172.0      190.0         12.2    4.0
1           88.0      221.0         10.6    3.0
2           55.0      151.0         12.0    4.0
3           92.0      183.0         10.3    4.0
4           72.0      136.0         10.9    3.0
```

```
data.rename(columns={'Age':'Age(in days)'},inplace=True)
```

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

```
ID                0
N_Days            0
Status            0
Drug            106
Age(in days)      0
Sex               0
Ascites         106
Hepatomegaly    106
Spiders         106
Edema             0
Bilirubin         0
Cholesterol     134
Albumin           0
Copper          108
Alk_Phos        106
SGOT            106
Tryglicerides   136
Platelets        11
Prothrombin       2
Stage             6
dtype: int64
```

```
data['Cholesterol'].fillna(data['Cholesterol'].mean(),inplace=True)
data['Copper'].fillna(data['Copper'].mean(),inplace=True)
data['Alk_Phos'].fillna(data['Alk_Phos'].mean(),inplace=True)
data['SGOT'].fillna(data['SGOT'].mean(),inplace=True)
data['Tryglicerides'].fillna(data['Tryglicerides'].mean(),inplace=True)
data['Platelets'].fillna(data['Platelets'].mean(),inplace=True)
data['Prothrombin'].fillna(data['Prothrombin'].mean(),inplace=True)
data['Stage'].fillna(data['Stage'].mean(),inplace=True)
```

```
<ipython-input-4-fc90cb943b11>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Cholesterol'].fillna(data['Cholesterol'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Copper'].fillna(data['Copper'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
```

```
  data['Alk_Phos'].fillna(data['Alk_Phos'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['SGOT'].fillna(data['SGOT'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Tryglicerides'].fillna(data['Tryglicerides'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Platelets'].fillna(data['Platelets'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Prothrombin'].fillna(data['Prothrombin'].mean(),inplace=True)
<ipython-input-4-fc90cb943b11>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  data['Stage'].fillna(data['Stage'].mean(),inplace=True)
```

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

```
ID                 0
N_Days             0
Status             0
Drug             106
Age(in days)       0
Sex                0
Ascites          106
Hepatomegaly     106
Spiders          106
Edema              0
Bilirubin          0
Cholesterol        0
Albumin            0
Copper             0
Alk_Phos           0
SGOT               0
Tryglicerides      0
Platelets          0
Prothrombin        0
Stage              0
dtype: int64
```

```
data_encoded=pd.get_dummies(data,columns=['Drug','Ascites','Hepatomegaly','Spiders','Sex','Edema'])
print(data_encoded.head())
```

```
   ID  N_Days Status  Age(in days)  Bilirubin  Cholesterol  Albumin  Copper  \
0   1     400      D         21464       14.5        261.0     2.60   156.0
1   2    4500      C         20617        1.1        302.0     4.14    54.0
2   3    1012      D         25594        1.4        176.0     3.48   210.0
3   4    1925      D         19994        1.8        244.0     2.54    64.0
4   5    1504     CL         13918        3.4        279.0     3.53   143.0

   Alk_Phos    SGOT  ...  Ascites_Y  Hepatomegaly_N  Hepatomegaly_Y  \
0    1718.0  137.95  ...       True           False            True
1    7394.8  113.52  ...      False           False            True
2     516.0   96.10  ...      False            True           False
3    6121.8   60.63  ...      False           False            True
4     671.0  113.15  ...      False           False            True

   Spiders_N  Spiders_Y  Sex_F  Sex_M  Edema_N  Edema_S  Edema_Y
0      False       True   True  False    False    False     True
```

```
1      False       True   True  False     True    False    False
2       True      False  False   True    False     True    False
3      False       True   True  False    False     True    False
4      False       True   True  False     True    False    False

[5 rows x 27 columns]
```

```
print(data_encoded.isnull().sum())
```

```
ID                      0
N_Days                  0
Status                  0
Age(in days)            0
Bilirubin               0
Cholesterol             0
Albumin                 0
Copper                  0
Alk_Phos                0
SGOT                    0
Tryglicerides           0
Platelets               0
Prothrombin             0
Stage                   0
Drug_D-penicillamine    0
Drug_Placebo            0
Ascites_N               0
Ascites_Y               0
Hepatomegaly_N          0
Hepatomegaly_Y          0
Spiders_N               0
Spiders_Y               0
Sex_F                   0
Sex_M                   0
Edema_N                 0
Edema_S                 0
Edema_Y                 0
dtype: int64
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error,r2_score
```

```
x=data_encoded.drop('Status',axis=1)
y=data_encoded['Status']
```

```
x_train_logistic,x_test_logistic,y_train_logistic,y_test_logistic=train_test_split(x,y,test_size=0.3,random_state=23)
```

```
lr=LogisticRegression(random_state=23)
lr.fit(x_train_logistic,y_train_logistic)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾       LogisticRegression
LogisticRegression(random_state=23)
```

```
y_pred=lr.predict(x_test_logistic)
```

```
acc=accuracy_score(y_test_logistic,y_pred)
print(acc*100)
```

```
82.53968253968253
```

```
#Using SVM
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
```

```
SVC
SVC()
```

```
pred=model.predict(x_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           C       0.60      0.97      0.74        66
          CL       0.00      0.00      0.00         9
           D       0.90      0.35      0.51        51

    accuracy                           0.65       126
   macro avg       0.50      0.44      0.42       126
weighted avg       0.68      0.65      0.60       126

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
print(confusion_matrix(y_test,pred))
```

```
[[64  0  2]
 [ 9  0  0]
 [33  0 18]]
```

```
print(accuracy_score(y_test,pred)*100)
```

```
65.07936507936508
```

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred_decison=clf.predict(x_test)
```

```
print(accuracy_score(y_test,y_pred_decison)*100)
```

```
71.42857142857143
```

```
#Using Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb_classifier=GaussianNB()
nb_classifier.fit(x_train,y_train)
```

```
GaussianNB
GaussianNB()
```

```
y_pred_naive=nb_classifier.predict(x_test)
```

```
accuracy=accuracy_score(y_test,y_pred_naive)
print(accuracy*100)
```

```
65.87301587301587
```

```
print(classification_report(y_test,y_pred_naive))
```

```
              precision    recall  f1-score   support

           C       0.67      0.85      0.75        66
          CL       0.17      0.22      0.19         9
           D       0.81      0.49      0.61        51

    accuracy                           0.66       126
   macro avg       0.55      0.52      0.52       126
weighted avg       0.69      0.66      0.65       126
```

```
#Using sample test data for prediction
pred_data=pd.read_csv("/content/sample_test_data.csv")
print(pred_data.head())
```

```
   ID  N_Days             Drug    Age Sex Ascites Hepatomegaly Spiders Edema  \
0   1     400  D-penicillamine  21464   F       Y            Y       Y     Y   
1   2    4500  D-penicillamine  20617   F       N            Y       Y     N   
2   3    1012  D-penicillamine  25594   M       N            N       N     S   
3   4    1925  D-penicillamine  19994   F       N            Y       Y     S   
4   5    1504          Placebo  13918   F       N            Y       Y     N   

   Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  Tryglicerides  \
0       14.5        261.0     2.60     156    1718.0  137.95          172.0   
1        1.1        302.0     4.14      54    7394.8  113.52           88.0   
2        1.4        176.0     3.48     210     516.0   96.10           55.0   
3        1.8        244.0     2.54      64    6121.8   60.63           92.0   
4        3.4        279.0     3.53     143     671.0  113.15           72.0   

   Platelets  Prothrombin  Stage  
0      190.0         12.2      4  
1      221.0         10.6      3  
2      151.0         12.0      4  
3      183.0         10.3      4  
4      136.0         10.9      3  
```

```
pred_data.rename(columns={'Age':'Age(in days)'},inplace=True)
```

```
pred_data['Cholesterol'].fillna(pred_data['Cholesterol'].mean(),inplace=True)
pred_data['Copper'].fillna(pred_data['Copper'].mean(),inplace=True)
pred_data['Alk_Phos'].fillna(pred_data['Alk_Phos'].mean(),inplace=True)
pred_data['SGOT'].fillna(pred_data['SGOT'].mean(),inplace=True)
pred_data['Tryglicerides'].fillna(pred_data['Tryglicerides'].mean(),inplace=True)
pred_data['Platelets'].fillna(pred_data['Platelets'].mean(),inplace=True)
pred_data['Prothrombin'].fillna(pred_data['Prothrombin'].mean(),inplace=True)
pred_data['Stage'].fillna(pred_data['Stage'].mean(),inplace=True)
```

Show hidden output

```
pred_data_encoded=pd.get_dummies(pred_data,columns=['Drug','Ascites','Hepatomegaly','Spiders','Sex','Edema'])
print(pred_data_encoded.head())
```

```
   ID  N_Days  Age(in days)  Bilirubin  Cholesterol  Albumin  Copper  \
0   1     400         21464       14.5        261.0     2.60     156   
1   2    4500         20617        1.1        302.0     4.14      54   
2   3    1012         25594        1.4        176.0     3.48     210   
3   4    1925         19994        1.8        244.0     2.54      64   
4   5    1504         13918        3.4        279.0     3.53     143   

   Alk_Phos    SGOT  Tryglicerides  ...  Ascites_Y  Hepatomegaly_N  \
0    1718.0  137.95          172.0  ...       True           False   
1    7394.8  113.52           88.0  ...      False           False   
2     516.0   96.10           55.0  ...      False            True   
3    6121.8   60.63           92.0  ...      False           False   
4     671.0  113.15           72.0  ...      False           False   

   Hepatomegaly_Y  Spiders_N  Spiders_Y  Sex_F  Sex_M  Edema_N  Edema_S  \
0            True      False       True   True  False    False    False   
1            True      False       True   True  False     True    False   
2           False       True      False  False   True    False     True   
3            True      False       True   True  False    False     True   
4            True      False       True   True  False     True    False   

   Edema_Y  
0     True  
1    False  
2    False  
3    False  
4    False  
```

```
[5 rows x 26 columns]
```

```
y_pred_logistic=lr.predict(pred_data_encoded)
print(y_pred_logistic)
```

```
['D' 'C' 'D' 'D' 'D' 'D' 'D' 'D' 'D' 'D' 'C' 'D' 'C' 'D' 'D' 'C' 'D' 'D'
 'C' 'D' 'C' 'D' 'D' 'D' 'C' 'D' 'D' 'D' 'C']
```