```
import pandas as pd
import numpy as np

df = pd.read_csv('hepatitis_csv.csv')

df.head()

   age     sex steroid  antivirals fatigue malaise anorexia liver_big
\
0   30    male   False       False   False   False    False     False

1   50  female   False       False    True   False    False     False

2   78  female    True       False    True   False    False      True

3   31  female     NaN        True   False   False    False      True

4   34  female    True       False   False   False    False      True


  liver_firm spleen_palpable spiders ascites varices  bilirubin  \
0      False           False   False   False   False        1.0
1      False           False   False   False   False        0.9
2      False           False   False   False   False        0.7
3      False           False   False   False   False        0.7
4      False           False   False   False   False        1.0

   alk_phosphate   sgot  albumin  protime  histology class
0           85.0   18.0      4.0      NaN      False  live
1          135.0   42.0      3.5      NaN      False  live
2           96.0   32.0      4.0      NaN      False  live
3           46.0   52.0      4.0     80.0      False  live
4            NaN  200.0      4.0      NaN      False  live

df.shape

(155, 20)

df.replace('?', np.nan, inplace=True)

# Convert all applicable columns to numeric (ignore errors from
bool/categorical columns)
for col in ['bilirubin', 'alk_phosphate', 'sgot', 'albumin',
'protime']:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Remove rows with negative values in numeric columns
numeric_cols = ['bilirubin', 'alk_phosphate', 'sgot', 'albumin',
'protime']
df = df[(df[numeric_cols] >= 0).all(axis=1)]
```

```
# Drop or impute missing values (drop for simplicity)
df_cleaned = df.dropna()

df_cleaned.shape
```

```
(80, 20)
```

```
def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

df_no_outliers = remove_outliers_iqr(df_cleaned, ['bilirubin',
'alk_phosphate', 'sgot', 'albumin', 'protime'])

df_no_outliers.shape
```

```
(54, 20)
```

```
from sklearn.preprocessing import LabelEncoder

# Encode 'sex', 'class' and other binary/categorical columns
label_cols = ['sex', 'steroid', 'antivirals', 'fatigue', 'malaise',
'anorexia', 'liver_big',
              'liver_firm', 'spleen_palpable', 'spiders', 'ascites',
'varices', 'histology', 'class']

for col in label_cols:
    df_no_outliers[col] =
LabelEncoder().fit_transform(df_no_outliers[col].astype(str))

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

# Feature selection
X = df_no_outliers.drop('class', axis=1)
y = df_no_outliers['class']  # 0 = die, 1 = live after encoding

# Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```python
# Logistic Regression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
logreg_acc = accuracy_score(y_test, y_pred_logreg)

# Naive Bayes
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
nb_acc = accuracy_score(y_test, y_pred_nb)

# Compare
print("Logistic Regression Accuracy:", logreg_acc)
print("Naive Bayes Accuracy:", nb_acc)
print("\nClassification Report (LogReg):\n",
classification_report(y_test, y_pred_logreg))
print("\nClassification Report (Naive Bayes):\n",
classification_report(y_test, y_pred_nb))
```

```
Logistic Regression Accuracy: 0.8181818181818182
Naive Bayes Accuracy: 0.8181818181818182

Classification Report (LogReg):
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.90      0.90      0.90        10

    accuracy                           0.82        11
   macro avg       0.45      0.45      0.45        11
weighted avg       0.82      0.82      0.82        11


Classification Report (Naive Bayes):
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.90      0.90      0.90        10

    accuracy                           0.82        11
   macro avg       0.45      0.45      0.45        11
weighted avg       0.82      0.82      0.82        11
```