In [1]:
```
import pandas as pd
df=pd.read_csv("C:\\Users\\HP\\Downloads\\archive (7)\\blood_samples_datas
df
```

Out[1]:

	Glucose	Cholesterol	Hemoglobin	Platelets	White Blood Cells	Red Blood Cells	Hematocrit	M Corpusc Vol
0	0.001827	0.033693	0.114755	0.997927	0.562604	0.866499	0.578042	0.91
1	0.436679	0.972653	0.084998	0.180909	0.675736	0.563889	0.798382	0.67
2	0.545697	0.324815	0.584467	0.475748	0.558596	0.661007	0.934056	0.38
3	0.172994	0.050351	0.736000	0.782022	0.069435	0.085219	0.032907	0.46
4	0.758534	0.739968	0.597868	0.772683	0.875720	0.860265	0.486189	0.48
...	...	...	...	...	...	...	...	
481	0.985163	0.412960	0.529993	0.263765	0.431288	0.198882	0.581289	0.70
482	0.581914	0.629325	0.491644	0.901473	0.347797	0.633286	0.698114	0.51
483	0.066669	0.404558	0.591041	0.228401	0.127461	0.026670	0.847444	0.27
484	0.901444	0.430680	0.243853	0.825551	0.493884	0.726299	0.660930	0.44
485	0.877912	0.597809	0.730440	0.462307	0.498438	0.792822	0.976056	0.88

486 rows × 25 columns

In [2]:
```
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

In [3]:
```
x=df.drop("Disease", axis=1)
x
```

Out[3]:

	Glucose	Cholesterol	Hemoglobin	Platelets	White Blood Cells	Red Blood Cells	Hematocrit	M Corpusc Vol
0	0.001827	0.033693	0.114755	0.997927	0.562604	0.866499	0.578042	0.91
1	0.436679	0.972653	0.084998	0.180909	0.675736	0.563889	0.798382	0.67
2	0.545697	0.324815	0.584467	0.475748	0.558596	0.661007	0.934056	0.38
3	0.172994	0.050351	0.736000	0.782022	0.069435	0.085219	0.032907	0.46
4	0.758534	0.739968	0.597868	0.772683	0.875720	0.860265	0.486189	0.48
...	...	...	...	...	...	...	...	
481	0.985163	0.412960	0.529993	0.263765	0.431288	0.198882	0.581289	0.70
482	0.581914	0.629325	0.491644	0.901473	0.347797	0.633286	0.698114	0.51
483	0.066669	0.404558	0.591041	0.228401	0.127461	0.026670	0.847444	0.27
484	0.901444	0.430680	0.243853	0.825551	0.493884	0.726299	0.660930	0.44
485	0.877912	0.597809	0.730440	0.462307	0.498438	0.792822	0.976056	0.88

486 rows × 24 columns

In [4]:
```
y=df['Disease']
y
```

Out[4]:
```
0      Thalasse
1      Diabetes
2      Heart Di
3      Diabetes
4      Heart Di
         ...
481    Diabetes
482    Heart Di
483      Anemia
484    Diabetes
485    Diabetes
Name: Disease, Length: 486, dtype: object
```

In [5]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
label_encoder=LabelEncoder()
label_encoder.fit(y)
label_encoder.transform(y)
```

Out[7]:
```
array([4, 1, 3, 1, 3, 3, 1, 1, 3, 1, 0, 4, 1, 1, 1, 4, 1, 1, 1, 0, 3, 4,
       4, 1, 1, 4, 1, 1, 1, 1, 4, 3, 1, 1, 1, 1, 1, 1, 3, 4, 1, 4, 1, 5,
       1, 0, 3, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 5, 1, 5,
       0, 1, 0, 0, 1, 5, 1, 1, 1, 1, 1, 1, 1, 0, 1, 3, 0, 1, 3, 1, 3, 0,
       1, 4, 0, 0, 0, 1, 5, 1, 1, 0, 4, 1, 1, 1, 4, 0, 5, 1, 1, 3, 4, 0,
       3, 1, 1, 1, 0, 0, 4, 1, 1, 1, 0, 3, 1, 0, 1, 1, 0, 1, 4, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 4, 0, 1, 1, 3, 4, 1, 1, 1, 0, 3, 1, 4, 1, 1,
       1, 4, 0, 5, 1, 3, 0, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 4, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 4, 0, 0, 1, 1, 3, 1, 0,
       0, 1, 1, 1, 4, 3, 4, 1, 0, 1, 4, 1, 1, 1, 0, 1, 1, 1, 1, 1, 4, 1,
       1, 3, 1, 3, 1, 1, 1, 4, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       0, 1, 1, 4, 1, 1, 1, 1, 4, 1, 4, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 4, 4, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 5, 2, 1, 4, 1, 1, 1, 4, 0,
       4, 1, 1, 1, 1, 0, 3, 0, 4, 1, 1, 1, 1, 3, 1, 3, 3, 4, 1, 4, 1, 0,
       0, 5, 4, 4, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 2, 1, 1, 1, 0, 3, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 1, 1, 0, 4, 1,
       0, 1, 0, 3, 1, 4, 1, 0, 0, 1, 1, 5, 1, 1, 3, 1, 1, 1, 1, 3, 1, 1,
       0, 1, 1, 1, 5, 1, 5, 1, 0, 1, 3, 1, 0, 2, 1, 4, 1, 0, 0, 0, 1, 1,
       5, 1, 1, 1, 5, 4, 1, 4, 1, 1, 0, 1, 0, 1, 3, 1, 3, 1, 3, 1, 2, 0,
       1, 1, 0, 1, 1, 0, 1, 3, 0, 1, 0, 0, 1, 3, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 4, 1, 1, 3, 1, 4, 1, 1, 1, 1, 1, 0, 1, 5, 4, 3, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 4, 0, 1, 1, 1, 3, 0,
       1, 1])
```

In [8]:
```

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
```

In [9]:
```
model= LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\HP\desktop\anaconda3\Lib\site-packages\sklearn\linear_model\_lo
gistic.py:460: 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 (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-r
egression (https://scikit-learn.org/stable/modules/linear_model.html#log
istic-regression)
  n_iter_i = _check_optimize_result(
```

Out[9]:
```
▾ LogisticRegression
LogisticRegression()
```

In [10]:
```
y_pred =model.predict(x_test)
y_pred
```

Out[10]:
```
array(['Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Anemia', 'Diabetes', 'Diabetes', 'Diabetes', 'Anemi
a',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse',
       'Thalasse', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Anemia', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Anemia', 'Diabetes', 'Diabetes', 'Thalasse', 'Diabetes',
       'Thalasse', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Heart Di', 'Diabetes', 'Thromboc', 'Anemia', 'Diabetes',
       'Diabetes', 'Heart Di', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Anemia',
       'Diabetes', 'Diabetes', 'Anemia', 'Anemia', 'Diabetes', 'Diabete
s',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes'], dtype=object)
```

In [11]:
```
from sklearn.metrics import accuracy_score,recall_score,precision_score,f1
```

In [12]:
```
accuracy_score=accuracy_score(y_test,y_pred)
accuracy_score
```

Out[12]:
```
0.6530612244897959
```

MODEL OPTIMIZATION

In [44]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
```

In [45]:
```
model=LogisticRegression()
```

In [46]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [47]:
```
#creating the parameters
parameters={
    "max_iter":[100,200,500,1000],
    "C":[0.1,2,5,10],
    "solver":["saga","sag","liblinear","lbfgs","newton-cg"],
    "penalty":[None,'l2']
}
```

In [48]:
```
#using the gridsearch
grid=GridSearchCV(model,parameters,cv=5)
```

In [49]:
```
# Fitting the GridSearchCV object to the data
grid.fit(x_train, y_train)
```

Out[49]:
```
▸        GridSearchCV
▸ estimator: LogisticRegression
      ▸ LogisticRegression
```

In [50]:
```
#getting the best parameters
best_parameters=grid.best_params_
best_parameters
```

Out[50]:
```
{'C': 2, 'max_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
```

In [51]:
```
#fitting the best parameters in the logistic regression model
```

In [52]:
```
model=LogisticRegression(**best_parameters)
model.fit(x_train,y_train)
model
```

Out[52]:
```
▾  LogisticRegression
LogisticRegression(C=2)
```

In [53]:
```
y_pred = model.predict(x_test)
y_pred
```

Out[53]:
```
array(['Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Thromboc', 'Diabetes',
       'Diabetes', 'Anemia', 'Diabetes', 'Diabetes', 'Diabetes', 'Anemi
a',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse',
       'Thalasse', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Anemia', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Anemia', 'Diabetes', 'Diabetes', 'Thalasse', 'Diabetes',
       'Thalasse', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Anemia', 'Diabetes',
       'Heart Di', 'Diabetes', 'Thromboc', 'Anemia', 'Diabetes',
       'Diabetes', 'Heart Di', 'Diabetes', 'Anemia', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse', 'Diabetes',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Thalasse',
       'Diabetes', 'Anemia', 'Anemia', 'Diabetes', 'Thalasse', 'Diabete
s',
       'Diabetes', 'Anemia', 'Anemia', 'Diabetes', 'Diabetes', 'Diabete
s',
       'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes', 'Diabetes'],
      dtype=object)
```

In [55]:
```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[55]:
```
0.6530612244897959
```

In [ ]: