Sanket Gore

Internship ID : UMIP27591

In [51]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
```

In [52]:
```
#Load the dataset
df = pd.read_csv(r"dataset.csv")
```

EDA

In [53]:
```
df.head(3)
```

Out[53]:

	Age	Gender	Smoking	Hx Smoking	Hx Radiothreapy	Thyroid Function	Physical Examination	Adenopathy	Pat
0	27	F	No	No	No	Euthyroid	Single nodular goiter-left	No	Microp
1	34	F	No	Yes	No	Euthyroid	Multinodular goiter	No	Microp
2	30	F	No	No	No	Euthyroid	Single nodular goiter-right	No	Microp

In [54]:
```
df.shape
```

Out[54]:
```
(383, 17)
```

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Age                   383 non-null    int64 
 1   Gender                383 non-null    object
 2   Smoking               383 non-null    object
 3   Hx Smoking            383 non-null    object
 4   Hx Radiothreapy       383 non-null    object
 5   Thyroid Function      383 non-null    object
 6   Physical Examination  383 non-null    object
 7   Adenopathy            383 non-null    object
 8   Pathology             383 non-null    object
 9   Focality              383 non-null    object
 10  Risk                  383 non-null    object
 11  T                     383 non-null    object
 12  N                     383 non-null    object
 13  M                     383 non-null    object
 14  Stage                 383 non-null    object
 15  Response              383 non-null    object
 16  Recurred              383 non-null    object
dtypes: int64(1), object(16)
memory usage: 51.0+ KB
```

In [56]:
```
df["Thyroid Function"].unique()
```

Out[56]:
```
array(['Euthyroid', 'Clinical Hyperthyroidism', 'Clinical Hypothyroidism',
       'Subclinical Hyperthyroidism', 'Subclinical Hypothyroidism'],
      dtype=object)
```

In [57]:
```
df["Physical Examination"].unique()
```

Out[57]:
```
array(['Single nodular goiter-left', 'Multinodular goiter',
       'Single nodular goiter-right', 'Normal', 'Diffuse goiter'],
      dtype=object)
```

In [58]:
```
df["Pathology"].unique()
```

Out[58]:
```
array(['Micropapillary', 'Papillary', 'Follicular', 'Hurthel cell'],
      dtype=object)
```

In [59]:
```
df["Adenopathy"].unique()
```

Out[59]:
```
array(['No', 'Right', 'Extensive', 'Left', 'Bilateral', 'Posterior'],
      dtype=object)
```

In [60]:
```
df["Focality"].unique()
```

Out[60]:
```
array(['Uni-Focal', 'Multi-Focal'], dtype=object)
```

In [61]:
```
df["Risk"].unique()
```

Out[61]:
```
array(['Low', 'Intermediate', 'High'], dtype=object)
```

In [62]:
```
df.nunique()
```

Out[62]:
```
Age                     65
Gender                   2
Smoking                  2
Hx Smoking               2
Hx Radiothreapy          2
Thyroid Function         5
Physical Examination     5
Adenopathy               6
Pathology                4
Focality                 2
Risk                     3
T                        7
N                        3
M                        2
Stage                    5
Response                 4
Recurred                 2
dtype: int64
```

In [63]:
```
df["T"].unique()
```

Out[63]:
```
array(['T1a', 'T1b', 'T2', 'T3a', 'T3b', 'T4a', 'T4b'], dtype=object)
```

In [64]:
```
df["N"].unique()
```

Out[64]:
```
array(['N0', 'N1b', 'N1a'], dtype=object)
```

In [65]:
```
df["M"].unique()
```

Out[65]:
```
array(['M0', 'M1'], dtype=object)
```

In [66]:
```
df["Stage"].unique()
```

Out[66]:
```
array(['I', 'II', 'IVB', 'III', 'IVA'], dtype=object)
```

In [67]:
```
df["Response"].unique()
```

Out[67]:
```
array(['Indeterminate', 'Excellent', 'Structural Incomplete',
       'Biochemical Incomplete'], dtype=object)
```

In [68]:
```
df["Recurred"].unique()
```

Out[68]:
```
array(['No', 'Yes'], dtype=object)
```

In [69]:
```
l_enc = LabelEncoder()
```

In [70]:
```
for col in df.columns:
    if col != 'Age':
        df[col] = l_enc.fit_transform(df[col])
```

In [71]:
```
df.head(5)
```

Out[71]:

	Age	Gender	Smoking	Hx Smoking	Hx Radiothreapy	Thyroid Function	Physical Examination	Adenopathy	Patho
0	27	0	0	0	0	2	3	3	
1	34	0	0	1	0	2	1	3	
2	30	0	0	0	0	2	4	3	
3	62	0	0	0	0	2	4	3	
4	62	0	0	0	0	2	1	3	

In [72]:
```
df["Response"].unique()
```

Out[72]:
```
array([2, 1, 3, 0])
```

In [73]:
```
df["T"].unique()
```

Out[73]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

In [74]:
```
X = df.drop("Recurred", axis=1)
```

In [75]:
```
X.shape
```

Out[75]:
```
(383, 16)
```

In [76]:
```
Y = df["Recurred"]
```

In [78]:
```
scaler = StandardScaler()
```

Train-test split

In [80]:
```
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2, rando
m_state = 42)
```

In [81]:
```
x_train_std = scaler.fit_transform(x_train)
```

In [82]:
```
x_test_std = scaler.transform(x_test)
```

In [84]:
```
log_reg = LogisticRegression(random_state = 42, max_iter = 1000)
```

In [86]:
```
log_reg.fit(x_train_std, y_train)
```

Out[86]:
```
▾ LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

In [87]:
```
y_pred = log_reg.predict(x_test)
```

```
c:\Users\sanke\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\base.p
y:432: UserWarning: X has feature names, but LogisticRegression was fitted wit
hout feature names
  warnings.warn(
```

In [88]:
```
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_s
core
```

In [89]:
```
y_prob = log_reg.predict_proba(x_test)[:, 1]
```

```
c:\Users\sanke\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\base.p
y:432: UserWarning: X has feature names, but LogisticRegression was fitted wit
hout feature names
  warnings.warn(
```

In [91]:
```
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[ 0 58]
 [ 0 19]]
```

In [92]:
```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        58
           1       0.25      1.00      0.40        19

    accuracy                           0.25        77
   macro avg       0.12      0.50      0.20        77
weighted avg       0.06      0.25      0.10        77


c:\Users\sanke\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\metrics
\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_
division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\sanke\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\metrics
\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_
division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\sanke\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\metrics
\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_
division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [93]:
```
print("\nAUC-ROC Score:", roc_auc_score(y_test, y_prob))
```

```
AUC-ROC Score: 0.9156079854809438
```

In [102]:
```
def predict_recurrence_log(input_data):
    i_df = pd.DataFrame([input_data])
    for col in i_df.columns:
        if col != 'Age':
            i_df[col] = l_enc.fit_transform(i_df[col])
    i_df_scaled = scaler.transform(i_df)
    prediction = log_reg.predict(i_df_scaled)[0]
    probability = log_reg.predict_proba(i_df_scaled)[0][1]
    return "Yes" if prediction == 1 else "No", probability
```

In [103]:
```
new_patient = {
    "Age": 29,
    "Gender": "F",
    "Smoking": "No",
    "Hx Smoking": "No",
    "Hx Radiothreapy": "Yes",
    "Thyroid Function": "Euthyroid",
    "Physical Examination": "Single nodular goiter-left",
    "Adenopathy": "No",
    "Pathology": "Micropapillary",
    "Focality": "Uni-Focal",
    "Risk": "Low",
    "T": "T1a",
    "N": "N0",
    "M": "M0",
    "Stage": "I",
    "Response": "Excellent"
}
```

In [104]:
```
result, prob = predict_recurrence_log(new_patient)
print(f"Prediction: {result}, Probability: {prob:.2f}")
```

```
Prediction: No, Probability: 0.03
```

In [ ]: