In [151]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [152]:

```
df=pd.read_csv('C:/Users/aksha/OneDrive/Documents/result7.csv')
df.head(7)
```

Out[152]:

	SNo.	CIE	SEE	Result
0	1	205	284	Fail
1	2	209	278	Fail
2	3	181	235	Fail
3	4	206	287	Fail
4	5	189	242	Fail
5	6	185	252	Fail
6	7	233	365	Distinction

In [153]:

```
df.isnull().sum()
```

Out[153]:

```
SNo.      0
CIE       0
SEE       0
Result    0
dtype: int64
```

In [154]:

```
x=df.drop('Result',axis=1)
y=df. Result
```

In [155]:

```
x.shape
```

Out[155]:

```
(60, 3)
```

In [156]:

```
y.shape
```

Out[156]:

```
(60,)
```

In [157]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(x, y, test_size=0.2,random_state=10)
```

In [158]:

```
X_train.shape
```

Out[158]:

```
(48, 3)
```

In [159]:

```
y_train.shape
```

Out[159]:

```
(48,)
```

In [160]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: 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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[160]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [161]:

```
y_pred=model.predict(X_test)
```

In [129]:

```
y_pred
```

Out[129]:

```
array(['Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail', 'Fail',
       'Fail', 'Distinction', 'Fail', 'Distinction'], dtype=object)
```

In [130]:

```
y_test
```

Out[130]:

```
31           Fail
3            Fail
38           Fail
27    First Class
21           Fail
17           Fail
46    First Class
2            Fail
23           Fail
26    Distinction
35           Fail
39    Distinction
Name: Result, dtype: object
```

In [131]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix

report=classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

 Distinction       1.00      1.00      1.00         2
        Fail       0.80      1.00      0.89         8
 First Class       0.00      0.00      0.00         2

    accuracy                           0.83        12
   macro avg       0.60      0.67      0.63        12
weighted avg       0.70      0.83      0.76        12


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricW
arning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
es. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [132]:

```
accuracy=accuracy_score(y_test,y_pred)
print(accuracy)
```

```
0.8333333333333334
```

In [133]:

```
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[2 0 0]
 [0 8 0]
 [0 2 0]]
```

In [150]:

```

```

Out[150]:

```
204    B
70     M
131    M
431    B
540    B
      ..
486    B
75     M
249    B
238    B
265    M
Name: diagnosis, Length: 114, dtype: object
```

In [149]:

```
df.shape
```

Out[149]:

```
(60, 4)
```

In [135]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [136]:

```
cancer=pd.read_csv('C:/Users/aksha/OneDrive/Documents/cancer.csv')
cancer.head(7)
```

Out[136]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_me
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11

7 rows × 32 columns

In [137]:

```
cancer.isnull().sum()
```

Out[137]:

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

In [139]:

```
x=cancer.drop('diagnosis',axis=1)
y=cancer.diagnosis
```

In [140]:

```
x.head(2)
```

Out[140]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
0	842302	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14
1	842517	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07

2 rows × 31 columns

In [141]:

```
y.head()
```

Out[141]:

```
0    M
1    M
2    M
3    M
4    M
Name: diagnosis, dtype: object
```

In [142]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(x, y, test_size=0.2, random_state=42)
```

In [143]:

```
X_train.shape
```

Out[143]:

```
(455, 31)
```

In [144]:

```
y_train.shape
```

Out[144]:

```
(455,)
```

In [145]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
model.fit(X_train,y_train)
```

Out[145]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [146]:

```
y_pred=model.predict(X_test)
```

In [147]:

```
y_pred
```

Out[147]:

```
array(['B', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'M', 'B', 'B', 'M', 'B',
       'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
       'B', 'B', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'M',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'M', 'M', 'B', 'B',
       'B', 'M', 'M', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B',
       'B', 'B', 'M', 'B', 'B', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'M', 'B', 'M', 'M',
       'B', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'M'], dtype=object)
```

In [148]:

```
y_test
```

Out[148]:

```
204    B
70     M
131    M
431    B
540    B
      ..
486    B
75     M
249    B
238    B
265    M
Name: diagnosis, Length: 114, dtype: object
```

In [81]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix

report=classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           B       0.94      0.94      0.94        71
           M       0.91      0.91      0.91        43

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114
```

In [82]:

```
accuracy=accuracy_score(y_test,y_pred)
print(accuracy)
```

```
0.9298245614035088
```

In [83]:

```
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[67  4]
 [ 4 39]]
```

In [ ]:

```

```