In [1]:
```
from sklearn.datasets import load_breast_cancer
```

In [2]:
```
import pandas as pd
dataset=load_breast_cancer()
```

In [4]:
```
df=pd.DataFrame(dataset.data,columns=dataset.feature_names)
df
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	
...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	

569 rows × 30 columns

In [5]:
```
df.shape
```

Out[5]:
```
(569, 30)
```

In [6]:
```
df.sample(2)
```

Out[6]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
473	12.27	29.97	77.42	465.4	0.07699	0.03398	0.0000	0.00000	(
287	12.89	13.12	81.89	515.9	0.06955	0.03729	0.0226	0.01171	(

2 rows × 30 columns

In [7]:
```
df["class"]=dataset.target
```

In [8]:
```
df.shape
```

Out[8]:
```
(569, 31)
```

In [9]:
```
df.sample(2)
```

Out[9]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
74	12.31	16.52	79.19	470.9	0.09172	0.06829	0.03372	0.02272	(
537	11.69	24.44	76.37	406.4	0.12360	0.15520	0.04515	0.04531	(

2 rows × 31 columns

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
x=df.iloc[:,:-1]
x
```

Out[11]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	syı
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	
...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	

569 rows × 30 columns

In [12]:
```
x.shape
```

Out[12]:
```
(569, 30)
```

In [13]:
```
y=df.iloc[:,-1]
y
```

Out[13]:
```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: class, Length: 569, dtype: int32
```

In [14]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=1,test_size=0.
```

In [15]:
```
xtrain.shape
```

Out[15]:
```
(398, 30)
```

In [16]:
```
xtest.shape
```

Out[16]:
```
(171, 30)
```

In [17]:
```
ytrain.shape
```

Out[17]:
```
(398,)
```

In [18]:
```
ytest.shape
```

Out[18]:
```
(171,)
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
classifier=LogisticRegression()
```

In [21]:

```
classifier.fit(xtrain,ytrain)
```

```
C:\Users\SAM\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.p
y: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 i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-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-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

Out[21]:

```
▾ LogisticRegression

LogisticRegression()
```

In [25]:

```
predictions=classifier.predict(xtest)
```

In [27]:

```
df["class"].value_counts()
```

Out[27]:

```
class
1    357
0    212
Name: count, dtype: int64
```

In [28]:

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

In [29]:

```
accuracy_score(ytest,predictions)
```

Out[29]:

```
0.9298245614035088
```

In [30]:

```
confusion_matrix(ytest,predictions)
```

Out[30]:

```
array([[ 57,   6],
       [  6, 102]], dtype=int64)
```

In [ ]: