

Cancer Prediction

Dataset Information:

Target Variable (y):

- Diagnosis (M = malignant, B = benign)

Ten features (X) are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness ($\text{perimeter}^2 / \text{area} - 1.0$)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (coastline approximation - 1)

For each characteristic three measures are given:

- a. Mean
- b. Standard error
- c. Largest/ Worst

```
In [1]: # Step 1 : import library
import pandas as pd
```

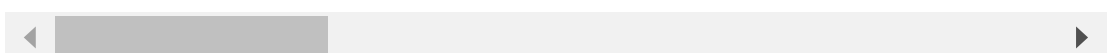
```
In [2]: # Step 2 : import data
cancer = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/main')
```

```
In [3]: cancer.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoo
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns



```
In [4]: cancer.info()
```

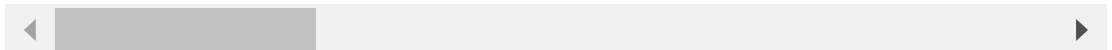
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                     569 non-null    int64
1   diagnosis                             569 non-null    object
2   radius_mean                           569 non-null    float64
3   texture_mean                           569 non-null    float64
4   perimeter_mean                         569 non-null    float64
5   area_mean                             569 non-null    float64
6   smoothness_mean                       569 non-null    float64
7   compactness_mean                      569 non-null    float64
8   concavity_mean                        569 non-null    float64
9   concave points_mean                   569 non-null    float64
10  symmetry_mean                         569 non-null    float64
11  fractal_dimension_mean                 569 non-null    float64
12  radius_se                              569 non-null    float64
13  texture_se                             569 non-null    float64
14  perimeter_se                           569 non-null    float64
15  area_se                                569 non-null    float64
16  smoothness_se                          569 non-null    float64
17  compactness_se                         569 non-null    float64
18  concavity_se                           569 non-null    float64
19  concave points_se                      569 non-null    float64
20  symmetry_se                            569 non-null    float64
21  fractal_dimension_se                   569 non-null    float64
22  radius_worst                           569 non-null    float64
23  texture_worst                           569 non-null    float64
24  perimeter_worst                        569 non-null    float64
25  area_worst                             569 non-null    float64
26  smoothness_worst                       569 non-null    float64
27  compactness_worst                      569 non-null    float64
28  concavity_worst                        569 non-null    float64
29  concave points_worst                   569 non-null    float64
30  symmetry_worst                         569 non-null    float64
31  fractal_dimension_worst                 569 non-null    float64
32  Unnamed: 32                             0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

```
In [5]: cancer.describe()
```

Out[5]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smooth
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	

8 rows × 32 columns



```
In [6]: # Step 3 : define target (y) and features (X)
```

```
In [7]: cancer.columns
```

Out[7]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'], dtype='object')

```
In [8]: y = cancer['diagnosis']
```

```
In [9]: X = cancer.drop(['id', 'diagnosis', 'Unnamed: 32'], axis=1)
```

```
In [10]: # Step 4 : train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)
```

```
In [11]: ▶ # check shape of train and test sample
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
Out[11]: ((398, 30), (171, 30), (398,), (171,))
```

```
In [12]: ▶ # Step 5 : select model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=5000)
```

```
In [13]: ▶ # Step 6 : train or fit model
model.fit(X_train,y_train)
```

```
Out[13]: LogisticRegression(max_iter=5000)
```

```
In [14]: ▶ model.intercept_
```

```
Out[14]: array([-29.49840967])
```

```
In [15]: ▶ model.coef_
```

```
Out[15]: array([[ -9.04309380e-01,  -1.88693401e-01,   2.36465162e-01,
        -2.39701052e-02,   1.57767489e-01,   1.81651624e-01,
         4.29904515e-01,   2.49073653e-01,   1.87042677e-01,
         3.12701310e-02,  -8.14808210e-04,  -1.27998720e+00,
        -2.15082640e-01,   1.27945786e-01,   2.75531281e-02,
        -8.45325440e-02,  -1.02460154e-02,   3.18227925e-02,
         2.66934912e-02,  -2.05532877e-02,  -2.25545003e-01,
         4.49709783e-01,   1.79244553e-01,   1.09126700e-02,
         3.27078006e-01,   5.51684050e-01,   1.06887289e+00,
         5.06025040e-01,   5.64121521e-01,   6.82532635e-02]])
```

```
In [16]: ▶ # Step 7 : predict model
y_pred = model.predict(X_test)
```

In [17]: `y_pred`

```
Out[17]: array(['B', 'M', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B', 'B', 'M',
                'B',
                'M', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B',
                'M',
                'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'B',
                'B',
                'M', 'M', 'M', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B',
                'B',
                'B', 'B', 'B', 'B', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'M',
                'M',
                'M', 'M', 'B', 'M', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B',
                'B',
                'M', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B', 'B',
                'M',
                'B', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B',
                'B',
                'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'B',
                'B',
                'M', 'B', 'B', 'M', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'B',
                'M',
                'M', 'B', 'M', 'M', 'M', 'B', 'B', 'M', 'B', 'M', 'B', 'M',
                'B',
                'M', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'B',
                'B',
                'B', 'M', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'B',
                'B',
                'B', 'B'], dtype=object)
```

In [18]: `# Step 8 : model accuracy`
`from sklearn.metrics import confusion_matrix, accuracy_score, classification_report`

In [19]: `confusion_matrix(y_test,y_pred)`

```
Out[19]: array([[97,  5],
                [ 2, 67]], dtype=int64)
```

In [20]: `accuracy_score(y_test,y_pred)`

```
Out[20]: 0.9590643274853801
```

In [21]: `print(classification_report(y_test,y_pred))`

	precision	recall	f1-score	support
B	0.98	0.95	0.97	102
M	0.93	0.97	0.95	69
accuracy			0.96	171
macro avg	0.96	0.96	0.96	171
weighted avg	0.96	0.96	0.96	171

In []: ▶

In []: ▶