Table of contents ✕ •••

+ Section

```
import pandas as pd
```

```
cancer = pd.read_csv('https://github.com/YBIFoundation/Dataset/ra
```

```
cancer.head()
```

	id	diagnosis	radius_mean	texture_mean	perimete
0	842302	M	17.99	10.38	
1	842517	M	20.57	17.77	
2	84300903	M	19.69	21.25	
3	84348301	M	11.42	20.38	
4	84358402	M	20.29	14.34	

5 rows × 33 columns

```
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
```

```
cancer.describe()
```

	id	radius_mean	texture_mean	peri
count	5.690000e+02	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	
std	1.250206e+08	3.524049	4.301036	
min	8.670000e+03	6.981000	9.710000	
25%	8.692180e+05	11.700000	16.170000	
50%	9.060240e+05	13.370000	18.840000	
75%	8.813129e+06	15.780000	21.800000	
max	9.113205e+08	28.110000	39.280000	

8 rows × 32 columns

```
cancer.columns
```

```
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')
```

```
y = cancer['diagnosis']
```

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

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((398, 30), (171, 30), (398,), (171,))
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=5000)
```

```
model.fit(X_train,y_train)
```

```
▾       LogisticRegression
LogisticRegression(max_iter=5000)
```

```
LogisticRegression(max_iter=5000)
```

```
▾       LogisticRegression
LogisticRegression(max_iter=5000)
```

```
model.intercept_
```

```
array([-30.20269391])
```

```
model.coef_
```

```
array([[-0.8644508 , -0.1823121 ,  0.26510852, -0.02688942,
0.13284582,
         0.19445151,  0.40918278,  0.20206338,  0.17199488,
0.03798515,
         0.0192444 , -1.13284188, -0.13597054,  0.11911954,
0.02266663,
        -0.03006638,  0.04691738,  0.02805721,  0.03329433,
-0.00980702,
        -0.27140621,  0.44034405,  0.16566196,  0.01286379,
0.2719812 ,
         0.59704539,  1.06177846,  0.40903862,  0.51193487,
0.08436947]])
```

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

```
y_pred
```

```
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)
```

```
#model accuracy
```

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

```
confusion_matrix(y_test,y_pred)
```

```
array([[97,  5],
       [ 2, 67]])
```

```
accuracy_score(y_test,y_pred)
```

```
0.9590643274853801
```

```
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
```