

## Importing the Dependencies

```
import numpy as np
import pandas as pd
import sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score
import warnings
warnings.filterwarnings("ignore")
```

## Data Collection & Processing

```
# Loading the data from sklearn
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()
```

```
print(breast_cancer_dataset)
```

[illegible]

```
# Loading the data to a data frame
data_frame = pd.DataFrame(breast_cancer_dataset.data, columns = breast_cancer_dataset.feature_names)
```

```
# Printing the first 5 rows
data_frame.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152

5 rows × 30 columns

```
# Adding the 'target' column to the data frame
data_frame['label'] = breast_cancer_dataset.target
```

```
# Printing last 5 rows
data_frame.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	21
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	11
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	1

5 rows × 31 columns

```
# Number of rows and columns in the dataset
data_frame.shape
```

(569, 31)

```
data_frame.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   mean radius                           569 non-null    float64
1   mean texture                           569 non-null    float64
2   mean perimeter                         569 non-null    float64
3   mean area                             569 non-null    float64
4   mean smoothness                       569 non-null    float64
5   mean compactness                      569 non-null    float64
6   mean concavity                        569 non-null    float64
7   mean concave points                   569 non-null    float64
8   mean symmetry                         569 non-null    float64
9   mean fractal dimension                569 non-null    float64
10  radius error                          569 non-null    float64
11  texture error                         569 non-null    float64
12  perimeter error                      569 non-null    float64
13  area error                           569 non-null    float64
14  smoothness error                     569 non-null    float64
15  compactness error                    569 non-null    float64
16  concavity error                      569 non-null    float64
17  concave points error                 569 non-null    float64
18  symmetry error                       569 non-null    float64
19  fractal dimension error              569 non-null    float64
20  worst radius                         569 non-null    float64
21  worst texture                        569 non-null    float64
22  worst perimeter                      569 non-null    float64
23  worst area                           569 non-null    float64
24  worst smoothness                     569 non-null    float64
25  worst compactness                    569 non-null    float64
26  worst concavity                      569 non-null    float64
27  worst concave points                 569 non-null    float64
28  worst symmetry                       569 non-null    float64
29  worst fractal dimension              569 non-null    float64
30  label                                569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
# Checking for missing values
data_frame.isnull().sum()

mean radius      0
mean texture      0
mean perimeter    0
mean area         0
mean smoothness   0
mean compactness  0
mean concavity    0
mean concave points 0
mean symmetry     0
mean fractal dimension 0
radius error      0
texture error     0
perimeter error   0
area error        0
smoothness error  0
compactness error 0
concavity error   0
concave points error 0
symmetry error    0
fractal dimension error 0
worst radius      0
worst texture     0
worst perimeter   0
worst area        0
worst smoothness  0
worst compactness 0
worst concavity   0
worst concave points 0
worst symmetry    0
worst fractal dimension 0
label            0
dtype: int64
```

```
# Statistical Measures
data_frame.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	2
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	1
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	2
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	2
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	2
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	4

8 rows × 31 columns

```
# Checking the distribution of Target Varibale
data_frame['label'].value_counts()

1    357
0    212
Name: label, dtype: int64

1 --> Benign
0 --> Malignant

data_frame.groupby('label').mean()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius
label												
0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.087990	0.192909	0.062680	...	21.134811
1	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.025717	0.174186	0.062867	...	13.379801

2 rows × 30 columns

### Separating the features and target

```
X = data_frame.drop(columns = 'label', axis = 1)
Y = data_frame['label']
```

```
print(X)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	
..	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	
565	20.13	28.25	131.20	1261.0	0.09780	
566	16.60	28.08	108.30	858.1	0.08455	
567	20.60	29.33	140.10	1265.0	0.11780	
568	7.76	24.54	47.92	181.0	0.05263	

  

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	...	...	...	...	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	
567	0.27700	0.35140	0.15200	0.2397	
568	0.04362	0.00000	0.00000	0.1587	

  

	mean fractal dimension	...	worst radius	worst texture	\
0	0.07871	...	25.380	17.33	
1	0.05667	...	24.990	23.41	
2	0.05999	...	23.570	25.53	
3	0.09744	...	14.910	26.50	
4	0.05883	...	22.540	16.67	
..	...	...	...	...	
564	0.05623	...	25.450	26.40	
565	0.05533	...	23.690	38.25	
566	0.05648	...	18.980	34.12	
567	0.07016	...	25.740	39.42	
568	0.05884	...	9.456	30.37	

  

	worst perimeter	worst area	worst smoothness	worst compactness	\
0	184.60	2019.0	0.16220	0.66560	
1	158.80	1956.0	0.12380	0.18660	
2	152.50	1709.0	0.14440	0.42450	
3	98.87	567.7	0.20980	0.86630	
4	152.20	1575.0	0.13740	0.20500	
..	...	...	...	...	
564	166.10	2027.0	0.14100	0.21130	
565	155.00	1731.0	0.11660	0.19220	
566	126.70	1124.0	0.11390	0.30940	
567	184.60	1821.0	0.16500	0.86810	
568	59.16	268.6	0.08996	0.06444	

  

	worst concavity	worst concave points	worst symmetry	\
0	0.7119	0.2654	0.4601	
1	0.2416	0.1860	0.2750	
2	0.4504	0.2430	0.3613	
3	0.6869	0.2575	0.6638	
4	0.4000	0.1625	0.2364	

```
print(Y)
```

```
0    0
1    0
2    0
3    0
4    0
```

```

..
564 0
565 0
566 0
567 0
568 1
Name: label, Length: 569, dtype: int64

```

### Splitting the data into training data & Testing data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

### Model Training

#### Logistic Regression

```
model = LogisticRegression()
```

```
# Training the Logistic Regression model using Training data
```

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

```

▼ LogisticRegression
LogisticRegression()

```

### Model Evaluation

#### Accuracy Score

```

# Accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)

```

```
print('Accuracy on training data = ', training_data_accuracy * 100)
```

```
Accuracy on training data = 94.72527472527472
```

```

# Accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

```

```
print('Accuracy on test data = ', test_data_accuracy * 100)
```

```
Accuracy on test data = 92.98245614035088
```

```

X_train_prediction = model.predict(X_train)
test_predictions = model.predict(X_test)
# Calculate precision for training and test data
train_precision = precision_score(Y_train, X_train_prediction)
test_precision = precision_score(Y_test, X_test_prediction)

```

```

print('Precision on training data:', train_precision * 100)
print('Precision on test data:', test_precision * 100)

```

```

Precision on training data: 95.2054794520548
Precision on test data: 94.20289855072464

```

Model's Accuracy: 92.98 % Model's Precision: 94.20 %

### Building a Predictive System

```
input_data = (13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.
```

```

# Change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

```

```
# Reshape the numpy array as we are predicting for one datapoint
input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_resaped)
print(prediction)

if (prediction[0] == 0):
    print('The Breast cancer is Malignant')

else:
    print('The Breast Cancer is Benign')
```

```
[1]
The Breast Cancer is Benign
```