Import Libraries

```
#import libraries
import pandas as pd
import seaborn as sns
```

Download dataset from Kaggle

```
#set kaggle API credentials
import os
os.environ['KAGGLE_USERNAME']='padalahanish'
os.environ['KAGGLE_KEY']='97a734194d11249c1f7abe66b750f78'
```

```
#download dataset
! kaggle datasets download -d uciml/breast-cancer-wisconsin-data
```

```
Dataset URL: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
License(s): CC-BY-NC-SA-4.0
Downloading breast-cancer-wisconsin-data.zip to /content
  0% 0.00/48.6k [00:00<?, ?B/s]
100% 48.6k/48.6k [00:00<00:00, 54.5MB/s]
```

```
#unzip file
! unzip /content/breast-cancer-wisconsin-data.zip
```

```
Archive:  /content/breast-cancer-wisconsin-data.zip
  inflating: data.csv
```

Load & Explore Data

```
#load data on dataframe
df=pd.read_csv('/content/data.csv')
```

```
#display dataframe
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
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

```
#count of rows and columns
df.shape
```

```
(569, 33)
```

```
#count number of null(empty) values
df.isna().sum()
```

	0
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

```
# Drop the column with null values
df.dropna(axis=1,inplace=True)
```

dtype: int64

```
# count of rows and columns
df.shape
```

```
(569, 32)
```

```
#Get count of number of M or B cells in diagnosis
df['diagnosis'].value_counts()
```

diagnosis	count
B	357
M	212

dtype: int64

Label Encoding

```
#Get Datatypes of each column in our dataset
df.dtypes
```

	0
id	int64
diagnosis	object
radius_mean	float64
texture_mean	float64
perimeter_mean	float64
area_mean	float64
smoothness_mean	float64
compactness_mean	float64
concavity_mean	float64
concave points_mean	float64
symmetry_mean	float64
fractal_dimension_mean	float64
radius_se	float64
texture_se	float64
perimeter_se	float64
area_se	float64
smoothness_se	float64
compactness_se	float64
concavity_se	float64
concave points_se	float64
symmetry_se	float64
fractal_dimension_se	float64
radius_worst	float64
texture_worst	float64
perimeter_worst	float64
area_worst	float64
smoothness_worst	float64
compactness_worst	float64
concavity_worst	float64
concave points_worst	float64
symmetry_worst	float64

```
#Encode the diagnosis values
from sklearn.preprocessing import LabelEncoder
```

```
labelencoder=LabelEncoder()
df.iloc[:,1]=labelencoder.fit_transform(df.iloc[:,1].values)


#display df
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoot
0	842302	1	17.99	10.38	122.80	1001.0	
1	842517	1	20.57	17.77	132.90	1326.0	
2	84300903	1	19.69	21.25	130.00	1203.0	
3	84348301	1	11.42	20.38	77.58	386.1	
4	84358402	1	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	
564	926424	1	21.56	22.39	142.00	1479.0	
565	926682	1	20.13	28.25	131.20	1261.0	
566	926954	1	16.60	28.08	108.30	858.1	
567	927241	1	20.60	29.33	140.10	1265.0	
568	92751	0	7.76	24.54	47.92	181.0	

569 rows × 32 columns

Split Dataset & Feature Scaling

```
#Splitting the dataset into independent and dependent datasets
X=df.iloc[:,2:].values
Y=df.iloc[:,1].values


#Splitting datasets into training(75%) and testing(25%)
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25)


#Scaling the data(feature scaling)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)


#print data
X_train
```

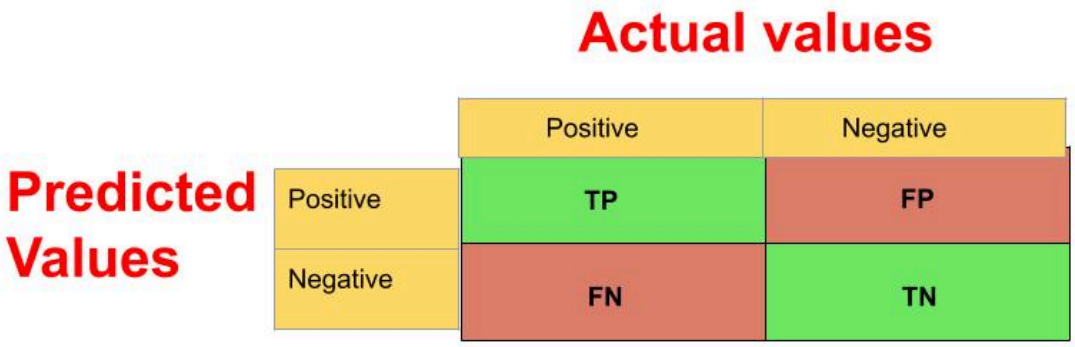
```
array([[ 0.02133391,  0.26794497,  0.01916583, ...,  0.30504437,
        -0.59209558, -0.26979352],
       [-0.30626796, -0.01876183, -0.38625318, ..., -1.55742243,
        -0.74247153, -1.16086724],
       [ 0.82182196,  0.20449347,  0.73444079, ..., -0.06353567,
         0.58083689, -1.05864081],
       ...,
       [ 1.10669315,  0.27499514,  1.09435358, ...,  0.53256291,
         0.54284717, -0.40211995],
       [-0.73642346, -2.03745972, -0.75361244, ..., -1.01077121,
         0.76287094, -0.51059356],
       [-0.41451901, -0.49112304, -0.44251541, ..., -0.87623191,
        -0.34832824, -0.82068039]])
```

Build a Logistic Regression Model

```
#build a logistic regression classifier
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()

Y_train = Y_train.astype(int)       #Y_train to numeric

Y_test = Y_test.astype(int)             #Y_test to numeric

classifier.fit(X_train,Y_train)

classifier.get_params()
```

```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 100,
 'multi_class': 'deprecated',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

```
#make use of trained model to make predictions on test
predictions=classifier.predict(X_test)
```




```
#plot confusion matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm=confusion_matrix(Y_test,predictions)
print(cm)
sns.heatmap(cm,annot=True)
```

```
[[89  2]
 [ 3 49]]
```