Import Libraries

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

Download dataset from Kaggle

```
#set kaggle API credentials
import os
os.environ['KAGGLE_USERNAME']='ridhimakharia'
os.environ['KAGGLE_KEY']='4a94042aa120deb4da38cdc4a03dae0d'
```

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

```
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, 26.1MB/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	sr
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	

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

```
(569, 33)
```

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

```
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
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

```
# Drop the column with null values
df.dropna(axis=1,inplace=True)
#axis=0 for row & 1 for column
```

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

```
(569, 32)
```

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

```
B    357
M    212
Name: diagnosis, dtype: int64
```

Label Encoding

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

```
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
fractal_dimension_worst  float64
dtype: object
```

```
#Encode the diagnosis values
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df.iloc[:,1] = labelencoder.fit_transform(df.iloc[:,1].values)
#M becomes 1 and B becomes 0
```

```
#display df
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
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
#diagnosis column depends on all the other columns
X = df.iloc[:,2:].values #independent
Y = df.iloc[:,1].values #dependent
```

```
#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)
#75% train 25% test data
```

```
#Scaling the data(feature scaling)
#all the data must have the same range
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([[2.811e+01, 1.847e+01, 1.885e+02, ..., 1.595e-01, 1.648e-01,
        5.525e-02],
       [1.513e+01, 2.981e+01, 9.671e+01, ..., 6.575e-02, 3.233e-01,
        6.165e-02],
       [1.805e+01, 1.615e+01, 1.202e+02, ..., 2.102e-01, 3.751e-01,
        1.108e-01],
       ...,
       [2.171e+01, 1.725e+01, 1.409e+02, ..., 1.820e-01, 2.510e-01,
        6.494e-02],
       [1.246e+01, 1.283e+01, 7.883e+01, ..., 2.680e-02, 2.280e-01,
        7.028e-02],
       [1.373e+01, 2.261e+01, 9.360e+01, ..., 2.208e-01, 3.596e-01,
        1.431e-01]])
```

Build a Logistic Regression Model

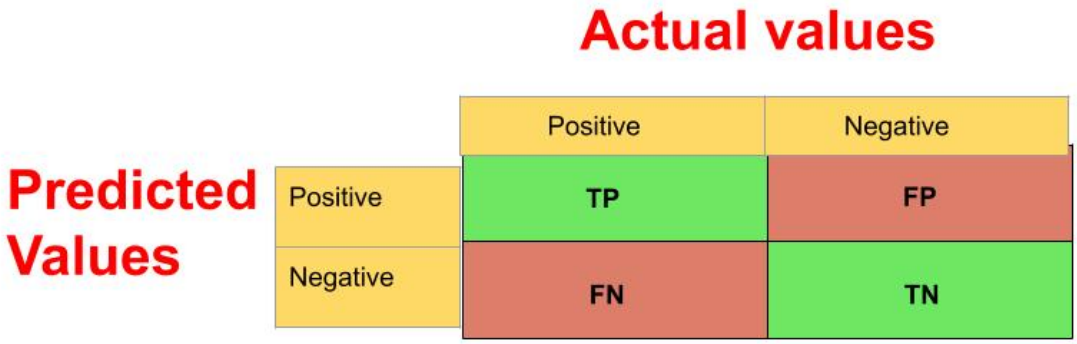
```
#build a logistic regression classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818:
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-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-regress
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

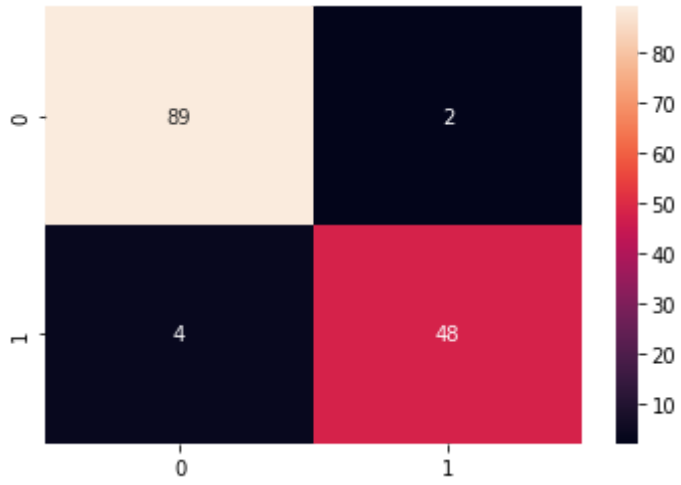
```
#make use of trained model to make predictions on test data
predictions = classifier.predict(X_test)
```

Performance Evaluation




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

```
[[89  2]
 [ 4 48]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f341e23fb90>
```




```
#get accuracy score for model
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test, predictions))
```

```
0.958041958041958
```

```
print(Y_test)
```

```
[0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1
 1 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 0
 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0]
```

```
print(predictions)
```

```
[0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1
 1 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0
 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0]
```