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']='rapidhunter'
os.environ['KAGGLE_KEY']='b871e133a65ec7b2acce5f4473'
```

```
#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, 33.4MB/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	smo
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)
```

```
# 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)
```

```
#display df
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sr
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([[ 1.22253093, -0.20195429,  1.19244588, ...,  0.7312142 ,
         1.28253854,  0.08513246],
       [ 1.23099424, -0.43729877,  1.20063333, ...,  0.76242591,
        -0.61478878,  0.55062419],
       [-0.28675916,  0.74413052, -0.20269504, ..., -0.03570195,
         1.97582355,  1.25746208],
       ...,
       [ 0.45801203, -0.03721316,  0.61932461, ...,  1.46394611,
         2.1631546 ,  1.11770706],
       [ 0.07716313,  1.79141346,  0.01140669, ..., -0.8824309 ,
        -0.87897359, -1.05172089],
       [-1.0907735 , -1.09861676, -1.05500825, ..., -1.10373674,
        -0.7364739 ,  0.0362182 ]])
```

Build a Logistic Regression Model

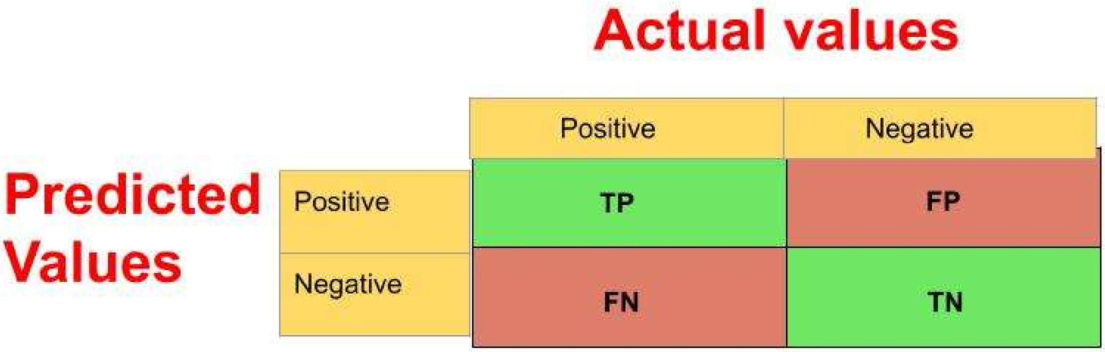
```
#build a logistic regression classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train,Y_train)

LogisticRegression(C=1.0, class_weight= None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
LogisticRegression()
```

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

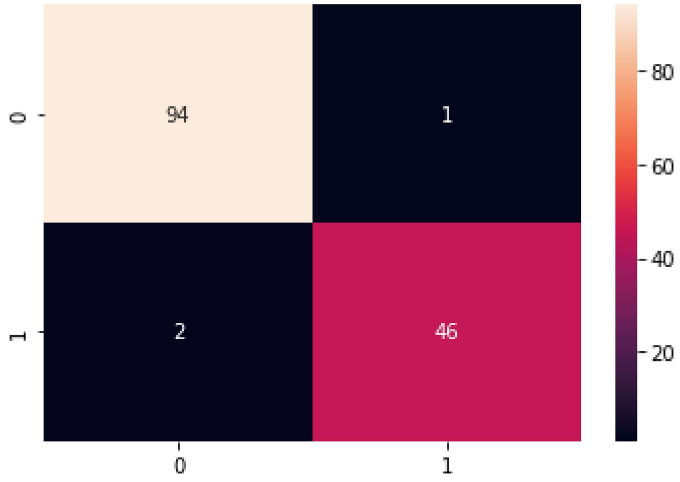
Performance Evaluation



		Actual values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

```
#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)
```

```
[[94  1]
 [ 2 46]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f8536b92110>
```



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

```
0.9790209790209791
```