Importing dataset using scikit learn which is sklearn

In [1]:
```
from sklearn.datasets import load_breast_cancer
```

loading into variable

In [2]:
```
loading = load_breast_cancer()
```

spiliting it into 2 types that is data and target

In [3]:
```
x= loading.data
y= loading.target
```

printing x(data) and y(target)

In [4]:
```
print(x)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

In [5]:
```
print(y)
```

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

Printing shape to analyze the data (using .shape method)

In [6]:
```
print(x.shape) # there are 569 rows and 30 columns in data
print("Data rows :- ",x.shape[0])
print("Data columns :- ",x.shape[1])
```

```
(569, 30)
Data rows :-  569
Data columns :-  30
```

In [7]:
```
print(y.shape)  # there are 569 rows but 1 columns which represents the target.
print("Target rows :- ",y.shape[0])
```

```
(569,)
Target rows :-  569
```

importing pandas for converting arrays into tabular form

In [8]:
```
import pandas as pd
```

creating a dataframe and giving column names

In [9]:
```
loading.feature_names # these are the names of the columns
```

Out[9]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [10]:
```
read = pd.DataFrame(x,columns=[loading.feature_names]) # creating a dataframe wit
```

In [11]:
```
read['Class'] = y # Adding target variable with the name Class to the dataframe w
```

In [12]:
```
read.head(5) # displaying 1st 5 rows  of the dataframe
```

Out[12]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	d
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	

5 rows × 31 columns

In [13]:
```
read.tail(5) # displaying last 5 rows of the dataframe
```

Out[13]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

5 rows × 31 columns

In [14]:
```
read.describe() # detailed statistics summary that includes mean,standard deviati
```

Out[14]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	

8 rows × 31 columns

Displaying Value Counts

In [15]:
```
# the target variable is in categorical format and hence we are able to see how m
# the value_counts() is an inbuilt function of dataframe that shows the appearanc
read['Class'].value_counts()
```

Out[15]:
```
(Class,)
1           357
0           212
dtype: int64
```

In [16]:
```
loading.target_names # these are the 0's and 1's
```

Out[16]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [17]:
```
read
```

Out[17]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 31 columns

In [18]:
```
read.shape
```

Out[18]:
```
(569, 31)
```

In [19]:
```
type(read['Class'])
```

Out[19]:
```
pandas.core.frame.DataFrame
```

In [20]:
```
# read.groupby(by='Class',).mean()
```

Dividing the data into 2 parts i.e Train and test using scikit learn

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [22]:
```
# but before that we have to split the data into 2 parts that is independent vari
x=read.iloc[:,:-1] # Selecting entire columns without considering the last columr
y=read["Class"] # selecting the last column
```

In [23]:
```
# printing the shapes of x and y again
print(x.shape)
print(y.shape)
```

```
(569, 30)
(569, 1)
```

using train_test_split with test_size as 0.1

In [24]:
```
# the first argument that is passed the independent variables
# the second argument that is passed the dependent variable
# the last argument that is mandatory to be passed is the test_size which is in t
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.1)
```

In [25]:
```
# printing the difference in the shape of y
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (512, 1)
y test shape :- (57, 1)
```

In [26]:
```
# printing difference in all the shapes (in xtrain,xtest,ytrain,ytest)
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (512, 30)
shape of ytrain :- (512, 1)
shape of xtest :- (57, 30)
shape of ytest :- (57, 1)
```

using train_test_split with test_size as 0.1 and stratify as y

In [27]:
```
# Shape after using stratify
# stratify = data is split according the mean of the data

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.1,stratify = y)
```

In [28]:
```
# printing the difference in the shape of y
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (512, 1)
y test shape :- (57, 1)
```

In [29]:
```
# printing y mean
print("mean of y :- ",y.mean(),"\n\n")
print("mean of ytrain :- ",ytrain.mean(),"\n\n")
print("mean of ytest :- ",ytest.mean(),"\n\n")
```

```
mean of y :-  Class    0.627417
dtype: float64


mean of ytrain :-  Class    0.626953
dtype: float64


mean of ytest :-  Class    0.631579
dtype: float64
```

In [30]:
```
#printing mean of x, xtrain,xtest
print("mean of x :- ",x.mean(),"\n\n")
print("mean of xtrain :- ",xtrain.mean(),"\n\n")
print("mean of xtest :- ",xtest.mean(),"\n\n")
```

```
mean of x :-  mean radius                14.127292
mean texture                 19.289649
mean perimeter               91.969033
mean area                   654.889104
mean smoothness               0.096360
mean compactness              0.104341
mean concavity                0.088799
mean concave points           0.048919
mean symmetry                 0.181162
mean fractal dimension        0.062798
radius error                  0.405172
texture error                 1.216853
perimeter error               2.866059
area error                   40.337079
smoothness error              0.007041
compactness error             0.025478
concavity error               0.031894
concave points error          0.011796
symmetry error                0.020542
fractal dimension error       0.003795
worst radius                 16.269190
worst texture                25.677223
worst perimeter             107.261213
worst area                  880.583128
worst smoothness              0.132369
worst compactness             0.254265
worst concavity               0.272188
worst concave points          0.114606
worst symmetry                0.290076
worst fractal dimension       0.083946
dtype: float64


mean of xtrain :-  mean radius                14.200830
mean texture                 19.296387
mean perimeter               92.441855
mean area                   661.691211
mean smoothness               0.095980
mean compactness              0.103686
mean concavity                0.088367
mean concave points           0.048884
mean symmetry                 0.180329
mean fractal dimension        0.062549
radius error                  0.410757
texture error                 1.212309
perimeter error               2.906081
area error                   41.180102
smoothness error              0.007050
compactness error             0.025413
concavity error               0.031940
concave points error          0.011887
symmetry error                0.020432
```

```
fractal dimension error      0.003773
worst radius                16.348223
worst texture               25.599863
worst perimeter            107.804863
worst area                 889.589258
worst smoothness             0.131570
worst compactness            0.251292
worst concavity              0.269311
worst concave points         0.114163
worst symmetry               0.287329
worst fractal dimension      0.083273
dtype: float64


mean of xtest :-  mean radius                13.466737
mean texture                19.229123
mean perimeter              87.721930
mean area                  593.789474
mean smoothness              0.099772
mean compactness             0.110224
mean concavity               0.092687
mean concave points          0.049236
mean symmetry                0.188646
mean fractal dimension       0.065028
radius error                 0.355005
texture error                1.257677
perimeter error              2.506561
area error                  32.764667
smoothness error             0.006964
compactness error            0.026060
concavity error              0.031482
concave points error         0.010980
symmetry error               0.021534
fractal dimension error      0.003988
worst radius                15.559281
worst texture               26.372105
worst perimeter            102.377895
worst area                 799.685965
worst smoothness             0.139539
worst compactness            0.280974
worst concavity              0.298036
worst concave points         0.118591
worst symmetry               0.314746
worst fractal dimension      0.089985
dtype: float64
```

In [31]:
```
# printing difference in all the shapes (in xtrain,xtest,ytrain,ytest)
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (512, 30)
shape of ytrain :- (512, 1)
shape of xtest :- (57, 30)
shape of ytest :- (57, 1)
```

using test_size as 0.2

In [32]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2)
```

In [33]:
```
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (455, 1)
y test shape :- (114, 1)
```

In [34]:
```
print("mean of y :- ",y.mean(),"\n\n")
print("mean of ytrain :- ",ytrain.mean(),"\n\n")
print("mean of ytest :- ",ytest.mean(),"\n\n")
```

```
mean of y :-  Class    0.627417
dtype: float64


mean of ytrain :-  Class    0.637363
dtype: float64


mean of ytest :-  Class    0.587719
dtype: float64
```

In [35]:
```
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (455, 30)
shape of ytrain :- (455, 1)
shape of xtest :- (114, 30)
shape of ytest :- (114, 1)
```

using train_test_split with test_size as 0.2 and stratify as y

In [36]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y)
```

In [37]:
```
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (455, 1)
y test shape :- (114, 1)
```

In [38]:
```
print("mean of y :- ",y.mean(),"\n\n")
print("mean of ytrain :- ",ytrain.mean(),"\n\n")
print("mean of ytest :- ",ytest.mean(),"\n\n")
```

```
mean of y :-  Class    0.627417
dtype: float64


mean of ytrain :-  Class    0.626374
dtype: float64


mean of ytest :-  Class    0.631579
dtype: float64
```

In [39]:
```
#printing mean of x, xtrain,xtest
print("mean of x :- ",x.mean(),"\n\n")
print("mean of xtrain :- ",xtrain.mean(),"\n\n")
print("mean of xtest :- ",xtest.mean(),"\n\n")
```

```
mean of x :-  mean radius                14.127292
mean texture                19.289649
mean perimeter              91.969033
mean area                  654.889104
mean smoothness              0.096360
mean compactness             0.104341
mean concavity               0.088799
mean concave points          0.048919
mean symmetry                0.181162
mean fractal dimension       0.062798
radius error                 0.405172
texture error                1.216853
perimeter error              2.866059
area error                  40.337079
smoothness error             0.007041
compactness error            0.025478
concavity error              0.031894
concave points error         0.011796
symmetry error               0.020542
fractal dimension error      0.003795
worst radius                16.269190
worst texture               25.677223
worst perimeter            107.261213
worst area                 880.583128
worst smoothness             0.132369
worst compactness            0.254265
worst concavity              0.272188
worst concave points         0.114606
worst symmetry               0.290076
worst fractal dimension      0.083946
dtype: float64


mean of xtrain :-  mean radius                14.098519
mean texture                19.323429
mean perimeter              91.776879
mean area                  651.322198
mean smoothness              0.096057
mean compactness             0.104121
mean concavity               0.088862
mean concave points          0.048627
mean symmetry                0.181073
mean fractal dimension       0.062777
radius error                 0.403644
texture error                1.226099
perimeter error              2.855849
area error                  39.864462
smoothness error             0.007077
compactness error            0.025690
concavity error              0.032432
concave points error         0.011841
symmetry error               0.020752
```

```
fractal dimension error      0.003828
worst radius                16.204923
worst texture               25.744110
worst perimeter            106.820330
worst area                 870.073626
worst smoothness             0.131903
worst compactness            0.253356
worst concavity              0.271686
worst concave points         0.113924
worst symmetry               0.289317
worst fractal dimension      0.083833
dtype: float64


mean of xtest :-  mean radius                 14.242132
mean texture                19.154825
mean perimeter              92.735965
mean area                  669.125439
mean smoothness              0.097570
mean compactness             0.105218
mean concavity               0.088550
mean concave points          0.050084
mean symmetry                0.181515
mean fractal dimension       0.062879
radius error                 0.411269
texture error                1.179951
perimeter error              2.906809
area error                  42.223404
smoothness error             0.006899
compactness error            0.024634
concavity error              0.029744
concave points error         0.011616
symmetry error               0.019705
fractal dimension error      0.003663
worst radius                16.525693
worst texture               25.410263
worst perimeter            109.020877
worst area                 922.528947
worst smoothness             0.134229
worst compactness            0.257894
worst concavity              0.274192
worst concave points         0.117329
worst symmetry               0.293102
worst fractal dimension      0.084395
dtype: float64
```

In [40]:
```
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (455, 30)
shape of ytrain :- (455, 1)
shape of xtest :- (114, 30)
shape of ytest :- (114, 1)
```

using random_state in train_test_split

In [41]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y)
```

In [42]:
```
xtrain.head(5)
```

Out[42]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
135	12.77	22.47	81.72	506.3	0.09055	0.05761	0.04711	0.02704	0.1585
529	12.07	13.44	77.83	445.2	0.11000	0.09009	0.03781	0.02798	0.1657
87	19.02	24.59	122.00	1076.0	0.09029	0.12060	0.14680	0.08271	0.1953
372	21.37	15.10	141.30	1386.0	0.10010	0.15150	0.19320	0.12550	0.1973
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	0.2030

5 rows × 30 columns

In [43]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y,rando
```

In [44]:
```
xtrain.head(5)
```

Out[44]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
54	15.100	22.02	97.26	712.8	0.09056	0.07081	0.05253	0.03334	0.1616
114	8.726	15.83	55.84	230.9	0.11500	0.08201	0.04132	0.01924	0.1649
256	19.550	28.77	133.60	1207.0	0.09260	0.20630	0.17840	0.11440	0.1893
176	9.904	18.06	64.60	302.4	0.09699	0.12940	0.13070	0.03716	0.1669
52	11.940	18.24	75.71	437.6	0.08261	0.04751	0.01972	0.01349	0.1868

5 rows × 30 columns

In [45]:
```
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (455, 30)
shape of ytrain :- (455, 1)
shape of xtest :- (114, 30)
shape of ytest :- (114, 1)
```

In [46]:
```
print("x orignal shape :-",x.shape)
print("x training shape :-",xtrain.shape)
print("x test shape :-",xtest.shape)
```

```
x orignal shape :- (569, 30)
x training shape :- (455, 30)
x test shape :- (114, 30)
```

In [47]:
```
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (455, 1)
y test shape :- (114, 1)
```

In [48]:
```
print("mean of y :- ",y.mean(),"\n\n")
print("mean of ytrain :- ",ytrain.mean(),"\n\n")
print("mean of ytest :- ",ytest.mean(),"\n\n")
```

```
mean of y :-  Class    0.627417
dtype: float64


mean of ytrain :-  Class    0.626374
dtype: float64


mean of ytest :-  Class    0.631579
dtype: float64
```

In [49]:
```
#printing mean of x, xtrain,xtest
print("mean of x :- ",x.mean(),"\n\n")
print("mean of xtrain :- ",xtrain.mean(),"\n\n")
print("mean of xtest :- ",xtest.mean(),"\n\n")
```

```
mean of x :-  mean radius                14.127292
mean texture                19.289649
mean perimeter              91.969033
mean area                  654.889104
mean smoothness              0.096360
mean compactness             0.104341
mean concavity               0.088799
mean concave points          0.048919
mean symmetry                0.181162
mean fractal dimension       0.062798
radius error                 0.405172
texture error                1.216853
perimeter error              2.866059
area error                  40.337079
smoothness error             0.007041
compactness error            0.025478
concavity error              0.031894
concave points error         0.011796
symmetry error               0.020542
fractal dimension error      0.003795
worst radius                16.269190
worst texture               25.677223
worst perimeter            107.261213
worst area                 880.583128
worst smoothness             0.132369
worst compactness            0.254265
worst concavity              0.272188
worst concave points         0.114606
worst symmetry               0.290076
worst fractal dimension      0.083946
dtype: float64


mean of xtrain :-  mean radius                14.140697
mean texture                19.228791
mean perimeter              92.049341
mean area                  657.659121
mean smoothness              0.096481
mean compactness             0.104119
mean concavity               0.087923
mean concave points          0.049103
mean symmetry                0.181072
mean fractal dimension       0.062719
radius error                 0.404835
texture error                1.214218
perimeter error              2.876167
area error                  40.688165
smoothness error             0.007084
compactness error            0.025323
concavity error              0.031098
concave points error         0.011803
symmetry error               0.020547
```

```
fractal dimension error      0.003742
worst radius                16.277292
worst texture               25.591780
worst perimeter            107.365055
worst area                 883.720440
worst smoothness             0.132465
worst compactness            0.254187
worst concavity              0.269816
worst concave points         0.114925
worst symmetry               0.289765
worst fractal dimension      0.083953
dtype: float64


mean of xtest :-  mean radius                 14.073789
mean texture                19.532544
mean perimeter              91.648509
mean area                  643.833333
mean smoothness              0.095877
mean compactness             0.105227
mean concavity               0.092296
mean concave points          0.048186
mean symmetry                0.181520
mean fractal dimension       0.063111
radius error                 0.406517
texture error                1.227374
perimeter error              2.825717
area error                  38.935816
smoothness error             0.006871
compactness error            0.026098
concavity error              0.035070
concave points error         0.011769
symmetry error               0.020524
fractal dimension error      0.004006
worst radius                16.236851
worst texture               26.018246
worst perimeter            106.846754
worst area                 868.061404
worst smoothness             0.131984
worst compactness            0.254576
worst concavity              0.281656
worst concave points         0.113333
worst symmetry               0.291313
worst fractal dimension      0.083919
dtype: float64
```

In [50]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y)
```

In [51]:
```
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (455, 30)
shape of ytrain :- (455, 1)
shape of xtest :- (114, 30)
shape of ytest :- (114, 1)
```

In [52]:
```
xtrain.head(5)
```

Out[52]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
146	11.800	16.58	78.99	432.0	0.10910	0.17000	0.16590	0.074150	0.2678
358	8.878	15.49	56.74	241.0	0.08293	0.07698	0.04721	0.023810	0.1930
467	9.668	18.10	61.06	286.3	0.08311	0.05428	0.01479	0.005769	0.1680
112	14.260	19.65	97.83	629.9	0.07837	0.22330	0.30030	0.077980	0.1704
35	16.740	21.59	110.10	869.5	0.09610	0.13360	0.13480	0.060180	0.1896

5 rows × 30 columns

In [53]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y,randc
```

In [54]:
```
print("shape of xtrain :-" ,xtrain.shape)
print("shape of ytrain :-" ,ytrain.shape)
print("shape of xtest :-" ,xtest.shape)
print("shape of ytest :-" ,ytest.shape)
```

```
shape of xtrain :- (455, 30)
shape of ytrain :- (455, 1)
shape of xtest :- (114, 30)
shape of ytest :- (114, 1)
```

In [55]:
```
xtrain.head(5)
```

Out[55]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
159	10.90	12.96	68.69	366.8	0.07515	0.03718	0.00309	0.006588	0.1442
113	10.51	20.19	68.64	334.2	0.11220	0.13030	0.06476	0.030680	0.1922
295	13.77	13.27	88.06	582.7	0.09198	0.06221	0.01063	0.019170	0.1592
495	14.87	20.21	96.12	680.9	0.09587	0.08345	0.06824	0.049510	0.1487
398	11.06	14.83	70.31	378.2	0.07741	0.04768	0.02712	0.007246	0.1535

5 rows × 30 columns

In [56]:
```
print("x orignal shape :-",x.shape)
print("x training shape :-",xtrain.shape)
print("x test shape :-",xtest.shape)
```

```
x orignal shape :- (569, 30)
x training shape :- (455, 30)
x test shape :- (114, 30)
```

In [57]:
```
print("y orignal shape :-",y.shape)
print("y training shape :-",ytrain.shape)
print("y test shape :-",ytest.shape)
```

```
y orignal shape :- (569, 1)
y training shape :- (455, 1)
y test shape :- (114, 1)
```

In [58]:
```
print("mean of y :- ",y.mean(),"\n\n")
print("mean of ytrain :- ",ytrain.mean(),"\n\n")
print("mean of ytest :- ",ytest.mean(),"\n\n")
```

```
mean of y :-  Class    0.627417
dtype: float64


mean of ytrain :-  Class    0.626374
dtype: float64


mean of ytest :-  Class    0.631579
dtype: float64
```

In [59]:
```
#printing mean of x, xtrain,xtest
print("mean of x :- ",x.mean(),"\n\n")
print("mean of xtrain :- ",xtrain.mean(),"\n\n")
print("mean of xtest :- ",xtest.mean(),"\n\n")
```

```
mean of x :-  mean radius                14.127292
mean texture                 19.289649
mean perimeter               91.969033
mean area                   654.889104
mean smoothness               0.096360
mean compactness              0.104341
mean concavity                0.088799
mean concave points           0.048919
mean symmetry                 0.181162
mean fractal dimension        0.062798
radius error                  0.405172
texture error                 1.216853
perimeter error               2.866059
area error                   40.337079
smoothness error              0.007041
compactness error             0.025478
concavity error               0.031894
concave points error          0.011796
symmetry error                0.020542
fractal dimension error       0.003795
worst radius                 16.269190
worst texture                25.677223
worst perimeter             107.261213
worst area                  880.583128
worst smoothness              0.132369
worst compactness             0.254265
worst concavity               0.272188
worst concave points          0.114606
worst symmetry                0.290076
worst fractal dimension       0.083946
dtype: float64


mean of xtrain :-  mean radius                14.078587
mean texture                 19.392088
mean perimeter               91.588967
mean area                   650.297802
mean smoothness               0.096464
mean compactness              0.103368
mean concavity                0.087287
mean concave points           0.048441
mean symmetry                 0.181587
mean fractal dimension        0.062740
radius error                  0.407198
texture error                 1.216565
perimeter error               2.875628
area error                   40.641215
smoothness error              0.007080
compactness error             0.025462
concavity error               0.031560
concave points error          0.011765
symmetry error                0.020632
```

```
fractal dimension error      0.003826
worst radius                16.213347
worst texture               25.746132
worst perimeter            106.838484
worst area                 874.624615
worst smoothness             0.132285
worst compactness            0.252776
worst concavity              0.270154
worst concave points         0.113799
worst symmetry               0.290939
worst fractal dimension      0.083790
dtype: float64


mean of xtest :-  mean radius                14.321684
mean texture                18.880789
mean perimeter              93.485965
mean area                  673.214035
mean smoothness              0.095948
mean compactness             0.108226
mean concavity               0.094835
mean concave points          0.050827
mean symmetry                0.179463
mean fractal dimension       0.063028
radius error                 0.397086
texture error                1.218006
perimeter error              2.827868
area error                  39.123202
smoothness error             0.006886
compactness error            0.025541
concavity error              0.033227
concave points error         0.011922
symmetry error               0.020185
fractal dimension error      0.003671
worst radius                16.492070
worst texture               25.402193
worst perimeter            108.948421
worst area                 904.364912
worst smoothness             0.132701
worst compactness            0.260208
worst concavity              0.280308
worst concave points         0.117828
worst symmetry               0.286630
worst fractal dimension      0.084567
dtype: float64
```

In [60]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y,randc
```

In [61]:
```
xtrain.head(5)
```

Out[61]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
326	14.110	12.88	90.03	616.5	0.09309	0.05306	0.01765	0.02733	0.1373
91	15.370	22.76	100.20	728.2	0.09200	0.10360	0.11220	0.07483	0.1717
168	17.470	24.68	116.10	984.6	0.10490	0.16030	0.21590	0.10430	0.1538
400	17.910	21.02	124.40	994.0	0.12300	0.25760	0.31890	0.11980	0.2113
358	8.878	15.49	56.74	241.0	0.08293	0.07698	0.04721	0.02381	0.1930

5 rows × 30 columns

In [62]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y,randc
```

In [63]:
```
xtrain.head(5)
```

Out[63]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
223	15.75	20.25	102.60	761.3	0.10250	0.12040	0.11470	0.06462	0.1935
461	27.42	26.27	186.90	2501.0	0.10840	0.19880	0.36350	0.16890	0.2061
413	14.99	22.11	97.53	693.7	0.08515	0.10250	0.06859	0.03876	0.1944
189	12.30	15.90	78.83	463.7	0.08080	0.07253	0.03844	0.01654	0.1667
554	12.88	28.92	82.50	514.3	0.08123	0.05824	0.06195	0.02343	0.1566

5 rows × 30 columns

In [64]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,stratify = y,randc
```

In [65]:
```
xtrain.head()
```

Out[65]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
261	17.35	23.06	111.00	933.1	0.08662	0.06290	0.02891	0.02837	0.1564
157	16.84	19.46	108.40	880.2	0.07445	0.07223	0.05150	0.02771	0.1844
342	11.06	14.96	71.49	373.9	0.10330	0.09097	0.05397	0.03341	0.1776
26	14.58	21.53	97.41	644.8	0.10540	0.18680	0.14250	0.08783	0.2252
528	13.94	13.17	90.31	594.2	0.12480	0.09755	0.10100	0.06615	0.1976

5 rows × 30 columns

making variables into binary format using pd.cut

In [66]:
```
# cut method is used to segment and sort data values into bins
# it also convert continous data to categorical data
xtrain_binarised = xtrain.apply(pd.cut, bins=2, labels =[1,0])
xtest_binarised = xtest.apply(pd.cut, bins=2, labels =[1,0])
print("xtrain_binarised type :- ",type(xtrain_binarised))
print("xtest_binarised type:- ",type(xtest_binarised))
```

```
xtrain_binarised type :-  <class 'pandas.core.frame.DataFrame'>
xtest_binarised type:-  <class 'pandas.core.frame.DataFrame'>
```

In [67]:
```
xtrain_binarised.head(2)
```

Out[67]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
261	1	1	1	1	1	1	1	1	1
157	1	1	1	1	1	1	1	1	1

2 rows × 30 columns

In [68]:
```
xtest_binarised.head(2)
```

Out[68]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
367	1	1	1	1	1	1	1	1	1
108	0	1	0	0	0	0	0	0	0

2 rows × 30 columns

In [69]:
```
# converting the binarised values to array
xtrain_binarised = xtrain_binarised.values
xtest_binarised = xtest_binarised.values
print('xtrain_binarised :- ',type(xtrain_binarised))
print('xtest_binarised  :- ',type(xtest_binarised))
```

```
xtrain_binarised :-  <class 'numpy.ndarray'>
xtest_binarised  :-  <class 'numpy.ndarray'>
```

In [70]:
```
print("shape of xtrain_binarised :- ",xtrain_binarised.shape)
print("shape of xtest_binarised  :- ",xtest_binarised.shape)
```

```
shape of xtrain_binarised :-  (455, 30)
shape of xtest_binarised  :-  (114, 30)
```

In [71]:
```
# possible values that b can take...
import numpy as np
b,i=10,100
if(np.sum(xtrain_binarised[i,:])>=b):     # calculating sum of the i rows and res
    print('Model Prediction is Malignant')
else:
    print('Model prediction is Benign')

if(list(ytrain['Class'])==1):                    # if the ytrain of is equal to 1
    print('Actual Outcome is Malignant')
else:
    print('Actual Outcome is Benign')
```

```
Model Prediction is Malignant
Actual Outcome is Benign
```

In [72]:
```
from random import randint
b,i=10,randint(0,xtrain_binarised.shape[0])
# using randint for getting any random value from the range of 0 to xtrain_binari

print("The number it came is  :- ",i)
# printing the number that came using random


if(np.sum(xtrain_binarised[i,:])>=b):     # calculating sum of the i rows and res
    print('Model Prediction is Malignant')
else:
    print('Model prediction is Benign')

if(list(ytrain['Class'])==1):                    # if the ytrain of is equal to 1
    print('Actual Outcome is Malignant')
else:
    print('Actual Outcome is Benign')
```

```
The number it came is  :-  144
Model Prediction is Malignant
Actual Outcome is Benign
```

In [73]:
```
b=10
yarray=ytrain.values
from random import randint
#i=randint(0,X_binarised_train.shape[0])

accurate_rows= 0
for i,x in zip(xtrain_binarised,yarray):
#     print(i,x[0])
    ypred=(np.sum(i)>=b)
    accurate_rows += (x[0] == ypred)

print("Number of accurate rows :- ",accurate_rows)
print("Accuracy score :- ",accurate_rows/xtrain_binarised.shape[0])
```

```
Number of accurate rows :-  285
Accuracy score :-  0.6263736263736264
```

In [74]:
```
i=100
max_accuracy=[]
for b in range(xtrain.shape[1]+1):
#     print(b)
    accurate_rows= 0
    for x,y in zip(xtrain_binarised,yarray):
        y_pred=(np.sum(x)>=b)
        accurate_rows += (y == y_pred)
    print("For b = ",str(b)+"\n"+"Number of accurate rows",str(accurate_rows[0]),
    max_accuracy.append([accurate_rows[0],(accurate_rows/xtrain_binarised.shape[0
```

```
For b =  0
Number of accurate rows 285 Accuracy Score :-  0.6263736263736264


For b =  1
Number of accurate rows 285 Accuracy Score :-  0.6263736263736264


For b =  2
Number of accurate rows 285 Accuracy Score :-  0.6263736263736264


For b =  3
Number of accurate rows 285 Accuracy Score :-  0.6263736263736264


For b =  4
Number of accurate rows 285 Accuracy Score :-  0.6263736263736264
```

In [76]:
```
max_row,max_acc=0,0
for i in max_accuracy:
    if i[1]>max_acc:
        max_acc = i[1]
        max_row=i[0]
    else:
        pass
```

In [77]:
```
print("The maximum accuracy that we gained is :-",max_acc,"and the rows it contai
```

```
The maximum accuracy that we gained is :- 0.8747252747252747 and the rows it co
ntains is :- 398
```