In [60]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/data.csv')
df
```

Out [60]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10
...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00

569 rows × 33 columns

In [61]:
```
df.tail()
```

Out [61]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	co points_
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.1389
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.0979
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.0530
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.1520
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.0000

5 rows × 33 columns

In [62]:
```
df.head()
```

Out [62]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	co points_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.1471
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.0701
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.1279
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.1052
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.1043

5 rows × 33 columns

In [63]:
```
df.describe()
```

Out [63]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_m
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.04891
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.03880
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.00000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.02031
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.03350
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.07400
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.20120

8 rows × 32 columns

In [64]:
```
df.shape
```

Out [64]:
```
(569, 33)
```

to find the missing vales

In [65]:
```
df.isna().sum()
```

Out [65]:
```
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
```

In [66]:
```
df.dtypes
```

Out [66]:
```
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
Unnamed: 32                float64
dtype: object
```

DEALING WITH THE MISSING VALUES

In [67]:
```
df = df.drop('Unnamed: 32',axis=1)
```

Transforming

In [68]:
```
df = df.drop("id",axis=1)
```

In [69]:
```
df['diagnosis']=df['diagnosis'].replace({'B':0,'M':1})
df
```

Out [69]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sy
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sy
...	...	...	...	...	...	...	...	...	...	...
564	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.
565	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.
567	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.
568	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.

569 rows × 31 columns

In [70]:
```
df.dtypes
```

Out [70]:
```
diagnosis                  int64
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
```

In [71]:
```
df.isna().sum()
```

Out [71]:
```
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
dtype: int64
```

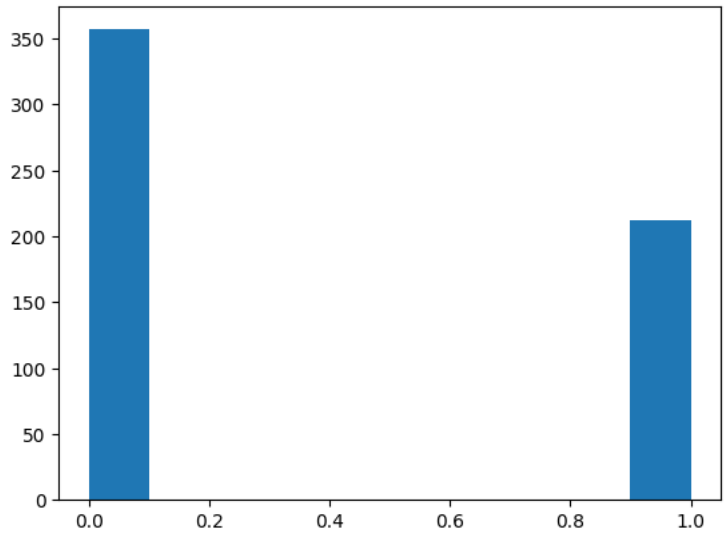
In [72]:
```
newdf=df['diagnosis'].value_counts()
newdf
```

Out [72]:
```
0    357
1    212
Name: diagnosis, dtype: int64
```

visualization

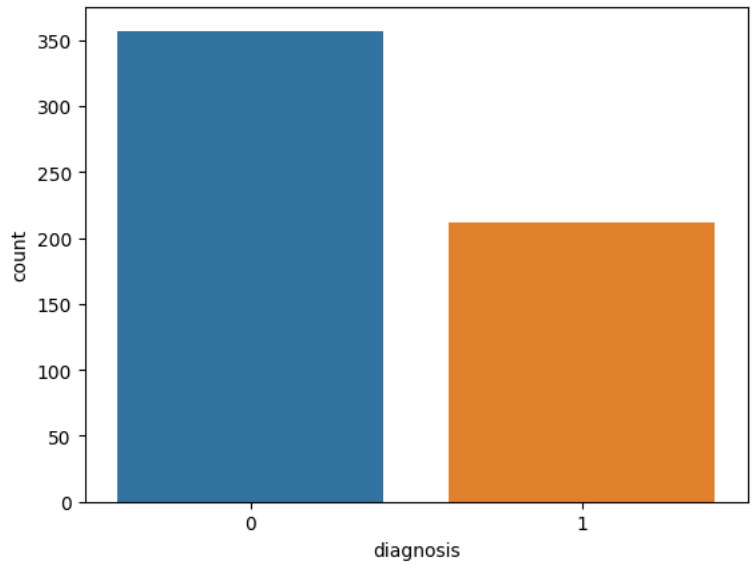
In [73]:
```
plt.hist(df['diagnosis'])
```

Out [73]:
```
(array([357.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 212.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



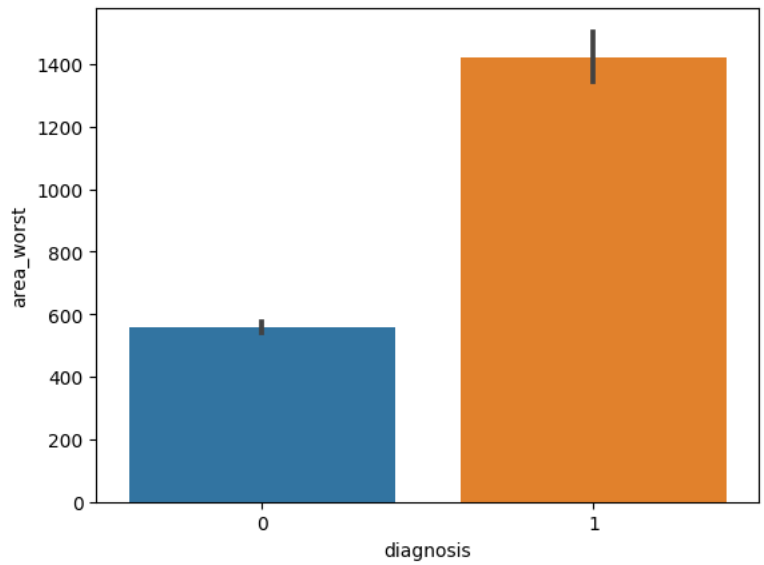
In [74]:
```
sns.countplot(x='diagnosis',data=df)
```

Out [74]: <Axes: xlabel='diagnosis', ylabel='count'>



In [75]:
```
sns.barplot(x='diagnosis',y='area_worst',data=df)
```

Out [75]: <Axes: xlabel='diagnosis', ylabel='area_worst'>



In [76]:
```
df.corr()
```

Out [76]:

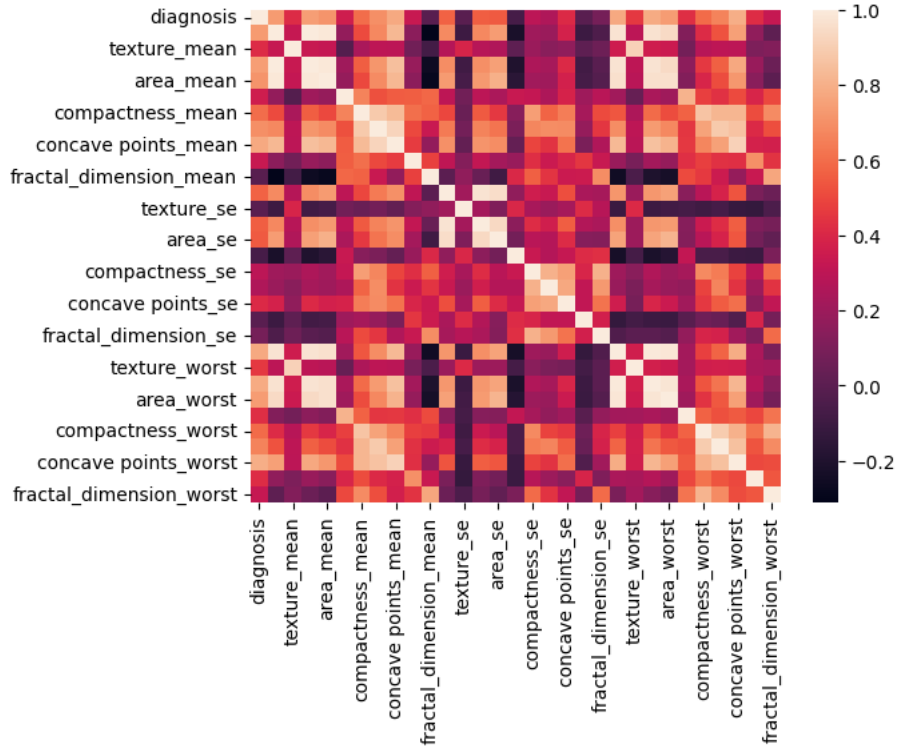
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mea
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mea
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391
area_se	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564
compactness_se	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279
concavity_se	0.253730	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270
concave points_se	0.408042	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260
symmetry_se	-0.006522	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.178009
fractal_dimension_se	0.077972	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.449301
radius_worst	0.776454	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.688236
texture_worst	0.456903	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.299879
perimeter_worst	0.782914	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.729565
area_worst	0.733825	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.675987
smoothness_worst	0.421465	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.448822
compactness_worst	0.590998	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.754968
concavity_worst	0.659610	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.884103
concave points_worst	0.793566	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861323
symmetry_worst	0.416294	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.409464
fractal_dimension_worst	0.323872	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.514930

31 rows × 31 columns

In [77]:
```
sns.heatmap(df.corr())
```

Out [77]: <Axes: >



x and y split

In [78]:
```
x = df.drop("diagnosis",axis=1)
y = df['diagnosis']
```

**

In [79]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split( x, y, test_size=0.3, random_state=42)
```

standardization

In [80]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

model-1 logistic regression

In [81]:
```
from sklearn.metrics import accuracy_score, mean_absolute_error,mean_squared_error,confusion_matrix,classifica
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

Out [81]:
```
array([0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```

In [82]:
```
from sklearn.metrics import mean_absolute_error
print('mae',mean_absolute_error(y_test,y_pred))
```

```
mae 0.017543859649122806
```

In [83]:
```
print(f"Confusion Matrix is: \n {confusion_matrix(y_test,y_pred)}\n\n\n")
print(f"Classification Report is \n: {classification_report(y_test,y_pred)}\n\n\n")
print(f"Accuraccy Precent is: \n {accuracy_score(y_test,y_pred)*100}\n\n\n")
```

```
Confusion Matrix is:
 [[106   2]
 [  1  62]]


Classification Report is
:               precision    recall  f1-score   support

           0       0.99      0.98      0.99       108
           1       0.97      0.98      0.98        63

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171



Accuraccy Precent is:
 98.24561403508771
```

In [84]:
```
from sklearn.metrics import r2_score
lrr2=r2_score(y_test,y_pred)
print('r2 score is',slr2)
```

```
r2 score is 0.7108399944964165
```

In [85]:
```
from sklearn.metrics import mean_squared_error
print("mse is ",mean_squared_error(y_test,y_pred))
```

```
mse is  0.017543859649122806
```

MODEL - 2 SIMPLE LINEAR REGRESSION

In [86]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

```
model.fit(x_train,y_train)
y_pred2=model.predict(x_test)
y_pred2
```

Out [86]:
```
array([ 0.36358932,  0.80106844,  0.74006996, -0.1312271 , -0.12862197,
        1.71286467,  1.08514511,  0.59684934,  0.74723758,  0.04163769,
        0.21293642,  0.6708076 ,  0.08974953,  0.60213481,  0.07201118,
        0.79437703,  0.09248481, -0.23008286, -0.44747534,  0.94547665,
        0.32239982,  0.12794482,  1.44700585, -0.11123994,  0.004409  ,
        0.24397509,  0.03690529,  0.09661851,  0.03557176,  1.21520709,
       -0.02948211, -0.04195904,  0.18886379,  0.04333356, -0.09031912,
        0.15376584,  0.50542208,  0.36738624,  0.71099741,  0.22084575,
       -0.17316721,  0.95019854,  0.03892061, -0.00872505,  0.50740828,
        0.31972561,  0.09250146, -0.09317963,  0.2204641 , -0.02925057,
        0.75303218,  1.13530507,  0.45224857,  0.40514208,  0.00608183,
        0.1768771 ,  0.04557794,  1.35732314,  0.3936453 , -0.06108539,
        0.06380164,  1.20635563,  1.38279505,  0.18752245,  0.08389196,
        0.41737961,  1.17844536,  1.12493952,  0.00802302,  0.18883743,
        0.9376689 ,  0.94166216,  0.12904441,  0.78515976, -0.02580277,
        0.14557101,  0.17893342,  0.41910481, -0.18525044,  0.26593108,
        0.59569318, -0.06625195,  0.44861506,  1.18567009,  0.67074975,
        0.86529039,  1.53076653,  0.93812542,  0.00963358,  0.06945396,
        0.18967739,  0.17771406,  0.20129955, -0.23149213, -0.01637157,
        0.03881138,  0.99717782,  1.21651795, -0.03525015,  0.87019294,
        0.78235617, -0.28896054,  0.80619033,  0.82940747,  0.18835074,
        0.27091328,  0.17512412,  1.10189   ,  0.30858924,  0.28686638,
        0.84397364,  0.0363618 ,  0.44613186,  0.91219975,  0.23138271,
        1.30819076, -0.16270978,  0.16538741, -0.10908774,  1.17694223,
        0.1280301 , -0.01156446,  0.18599272,  0.91999902,  0.31722312,
        1.10500694,  0.84940248,  0.05839869,  0.02969132,  1.08755584,
        1.05083659,  1.52148092,  0.201217  ,  0.03671483,  0.27390341,
        0.80261865,  0.28959389,  0.3342848 ,  0.38806159,  0.88886231,
        0.034615  ,  1.01294041, -0.17290353, -0.11109993,  0.77637878,
       -0.01119869,  1.07339631,  0.95318911,  0.46303091, -0.04877638,
        0.4545745 ,  0.02753639, -0.25536473,  0.15591261, -0.0049611 ,
        1.54344386,  0.80454223, -0.20289741,  0.20781442, -0.14934294,
       -0.73657741,  0.08677702,  0.08611334,  0.11477152,  0.54587315,
        0.05559592,  0.07840024,  0.31662588,  0.00410417,  0.69843006,
        0.35758428])
```

In [87]:
```
from sklearn.metrics import mean_absolute_error
print('mae',mean_absolute_error(y_test,y_pred2))
```

```
mae 0.20007348217357535
```

In [88]:
```
from sklearn.metrics import mean_squared_error
print("mse is ",mean_squared_error(y_test,y_pred2))
```

```
mse is  0.0672837685936316
```

In [89]:
```
numbu=mean_squared_error(y_test,y_pred2)
np.sqrt(numbu)
```

Out [89]:
```
0.2593911497981988
```

In [90]:
```
from sklearn.metrics import r2_score
slr2=r2_score(y_test,y_pred2)
print('r2 score is',slr2)
```

```
r2 score is 0.7108399944964165
```

MODEL 3 RANDOM FOREST

In [91]:
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor()
reg.fit(x_train, y_train)
```

Out [91]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [92]:
```
reg.score(x_test, y_test)
```

Out [92]:
```
0.8683973544973544
```

In [93]:
```
reg.score(x_train,y_train)
```

Out [93]:
```
0.9787735478828063
```

In [94]:
```
ypred3= reg.predict(x_test)
ypred3
```

Out [94]:
```
array([0.  , 1.  , 1.  , 0.  , 0.  , 1.  , 1.  , 0.88, 0.71, 0.1 , 0.24,
       1.  , 0.05, 0.91, 0.01, 1.  , 0.04, 0.  , 0.  , 1.  , 0.14, 0.  ,
       1.  , 0.  , 0.  , 0.02, 0.  , 0.17, 0.  , 1.  , 0.01, 0.  , 0.29,
       0.03, 0.  , 0.  , 0.71, 0.  , 1.  , 0.12, 0.  , 1.  , 0.  , 0.  ,
       0.32, 0.  , 0.16, 0.19, 0.  , 0.  , 1.  , 1.  , 0.24, 0.07, 0.  ,
       0.  , 0.  , 1.  , 0.93, 0.  , 0.  , 1.  , 1.  , 0.08, 0.  , 0.09,
       1.  , 1.  , 0.  , 0.02, 0.98, 1.  , 0.  , 1.  , 0.01, 0.04, 0.01,
       0.47, 0.  , 0.12, 0.98, 0.  , 0.11, 1.  , 0.8 , 1.  , 0.97, 1.  ,
       0.04, 0.  , 0.  , 0.17, 0.31, 0.05, 0.  , 0.  , 1.  , 1.  , 0.  ,
       1.  , 0.91, 0.  , 0.92, 1.  , 0.01, 0.  , 0.  , 1.  , 0.72, 0.1 ,
       1.  , 0.  , 0.25, 1.  , 0.26, 1.  , 0.01, 0.16, 0.  , 1.  , 0.44,
       0.  , 0.  , 1.  , 0.04, 1.  , 1.  , 0.  , 0.  , 1.  , 0.85, 0.99,
       0.08, 0.01, 0.26, 0.89, 0.64, 0.01, 0.4 , 0.94, 0.  , 1.  , 0.  ,
       0.  , 0.99, 0.  , 1.  , 1.  , 0.76, 0.  , 0.83, 0.  , 0.  , 0.  ,
```

```
       0.19, 0.99, 1.  , 0.  , 0.  , 0.  , 0.08, 0.  , 0.  , 0.  , 0.44,
       0.  , 0.  , 0.21, 0.01, 0.98, 0.34])
```

In [95]:
```
from sklearn.metrics import mean_absolute_error
print('mae',mean_absolute_error(y_test,ypred3))
```

```
mae 0.07345029239766082
```

In [98]:
```
from sklearn.metrics import mean_squared_error
print("mse is ",mean_squared_error(y_test,ypred3))
```

```
mse is  0.030622222222222224
```

In [99]:
```
numbu=mean_squared_error(y_test,ypred3)
np.sqrt(numbu)
```

Out [99]:
```
0.17499206331208916
```

In [100]:
```
from sklearn.metrics import r2_score
rfr2=r2_score(y_test,ypred3)
print('r2 score is',rfr2)
```

```
r2 score is 0.8683973544973544
```

In [110]:
```
print(f"Accuraccy Percent is: \n {accuracy_score(y_test,ypred3)*100}\n\n\n")
```

```
Accuraccy Precent is:
 92.98245614035088
```

MODEL 4 Decision tree

In [101]:
```
from sklearn.tree import DecisionTreeRegressor
dtr= DecisionTreeRegressor()
dtr.fit(x_train,y_train)
```

Out [101]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [102]:
```
dtr.score(x_test, y_test)
```

Out [102]:
```
0.6984126984126984
```

In [105]:
```
ypred4=dtr.predict(x_test)
ypred4
```

Out [105]:
```
array([0., 1., 1., 0., 0., 1., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1.,
       1., 0., 0., 0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 0., 1., 1.,
       0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1.,
       1., 1., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0.,
       1., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0.,
       1., 1., 0., 0., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0.,
       1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0., 0.,
       0., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       0.])
```

In [106]:
```
from sklearn.metrics import mean_absolute_error
print('mae',mean_absolute_error(y_test,ypred4))
```

```
mae 0.07017543859649122
```

In [107]:
```
from sklearn.metrics import mean_squared_error
print("mse is ",mean_squared_error(y_test,ypred4))
```

```
mse is  0.07017543859649122
```

In [108]:
```
from sklearn.metrics import r2_score
dtr2=r2_score(y_test,ypred4)
print('r2 score is',dtr2)
```

```
r2 score is 0.6984126984126984
```

In [109]:
```
print(f"Accuraccy Percent is: \n {accuracy_score(y_test,ypred4)*100}\n\n\n")
```

```
Accuraccy Precent is:
 92.98245614035088
```

MODEL 5 - KNN

In [113]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [114]:
```
model = KNeighborsClassifier(n_neighbors=7)
model.fit(x_train,y_train)
```

Out [114]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

In [116]:
```
y_pred5 = model.predict(x_test)
y_pred5
```

Out [116]:
```
array([0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
```

In [118]:
```
print(f"Confusion Matrix is: \n {confusion_matrix(y_test,y_pred5)}\n\n\n")
print(f"Classification Report is \n: {classification_report(y_test,y_pred5)}\n\n\n")
print(f"Accuraccy Precent is: \n {accuracy_score(y_test,y_pred5)*100}\n\n\n")
```

```
Confusion Matrix is:
 [[105   3]
 [  4  59]]


Classification Report is
:               precision    recall  f1-score   support

           0       0.96      0.97      0.97       108
           1       0.95      0.94      0.94        63

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171



Accuraccy Precent is:
 95.90643274853801
```