In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

DATA COLLECTION

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction.csv")
a
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
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	
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 33 columns

In [3]:

```
b=a.head(10)
b
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
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	843786	M	12.45	15.70	82.57	477.1	
6	844359	M	18.25	19.98	119.60	1040.0	
7	84458202	M	13.71	20.83	90.20	577.9	
8	844981	M	13.00	21.82	87.50	519.8	
9	84501001	M	12.46	24.04	83.97	475.9	

10 rows × 33 columns

DATA CLEANING AND PRE-PROCESSING

In [4]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       10 non-null     int64  
 1   diagnosis                10 non-null     object 
 2   radius_mean              10 non-null     float64
 3   texture_mean             10 non-null     float64
 4   perimeter_mean           10 non-null     float64
 5   area_mean                10 non-null     float64
 6   smoothness_mean          10 non-null     float64
 7   compactness_mean         10 non-null     float64
 8   concavity_mean           10 non-null     float64
 9   concave points_mean      10 non-null     float64
 10  symmetry_mean            10 non-null     float64
 11  fractal_dimension_mean   10 non-null     float64
 12  radius_se                10 non-null     float64
 13  texture_se               10 non-null     float64
 14  perimeter_se             10 non-null     float64
 15  area_se                  10 non-null     float64
 16  smoothness_se            10 non-null     float64
 17  compactness_se           10 non-null     float64
 18  concavity_se             10 non-null     float64
 19  concave points_se        10 non-null     float64
 20  symmetry_se              10 non-null     float64
 21  fractal_dimension_se     10 non-null     float64
 22  radius_worst             10 non-null     float64
 23  texture_worst            10 non-null     float64
 24  perimeter_worst          10 non-null     float64
 25  area_worst               10 non-null     float64
 26  smoothness_worst         10 non-null     float64
 27  compactness_worst        10 non-null     float64
 28  concavity_worst          10 non-null     float64
 29  concave points_worst     10 non-null     float64
 30  symmetry_worst           10 non-null     float64
 31  fractal_dimension_worst  10 non-null     float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 2.7+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
count	1.000000e+01	10.000000	10.00000	10.000000	10.000000	10.0(
mean	4.261848e+07	15.983000	18.64900	106.222000	830.380000	0.1'
std	4.403463e+07	3.686001	4.10719	23.680745	377.613035	0.0'
min	8.423020e+05	11.420000	10.38000	77.580000	386.100000	0.08
25%	8.439292e+05	12.595000	16.21750	84.852500	487.775000	0.1(
50%	4.257294e+07	15.850000	20.18000	104.900000	789.450000	0.1'
75%	8.435588e+07	19.330000	21.14500	128.200000	1162.250000	0.12
max	8.450100e+07	20.570000	24.04000	135.100000	1326.000000	0.14

8 rows × 32 columns

In [6]:

```
c=b.dropna(axis=1)
c
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
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	843786	M	12.45	15.70	82.57	477.1	(
6	844359	M	18.25	19.98	119.60	1040.0	(
7	84458202	M	13.71	20.83	90.20	577.9	(
8	844981	M	13.00	21.82	87.50	519.8	(
9	84501001	M	12.46	24.04	83.97	475.9	(

10 rows × 32 columns

In [7]:

```
d=c[['diagnosis','radius_mean','id','concave points_se','symmetry_worst']]
d
```

Out[7]:

	diagnosis	radius_mean	id	concave points_se	symmetry_worst
0	M	17.99	842302	0.01587	0.4601
1	M	20.57	842517	0.01340	0.2750
2	M	19.69	84300903	0.02058	0.3613
3	M	11.42	84348301	0.01867	0.6638
4	M	20.29	84358402	0.01885	0.2364
5	M	12.45	843786	0.01137	0.3985
6	M	18.25	844359	0.01039	0.3063
7	M	13.71	84458202	0.01448	0.3196
8	M	13.00	844981	0.01226	0.4378
9	M	12.46	84501001	0.01432	0.4366

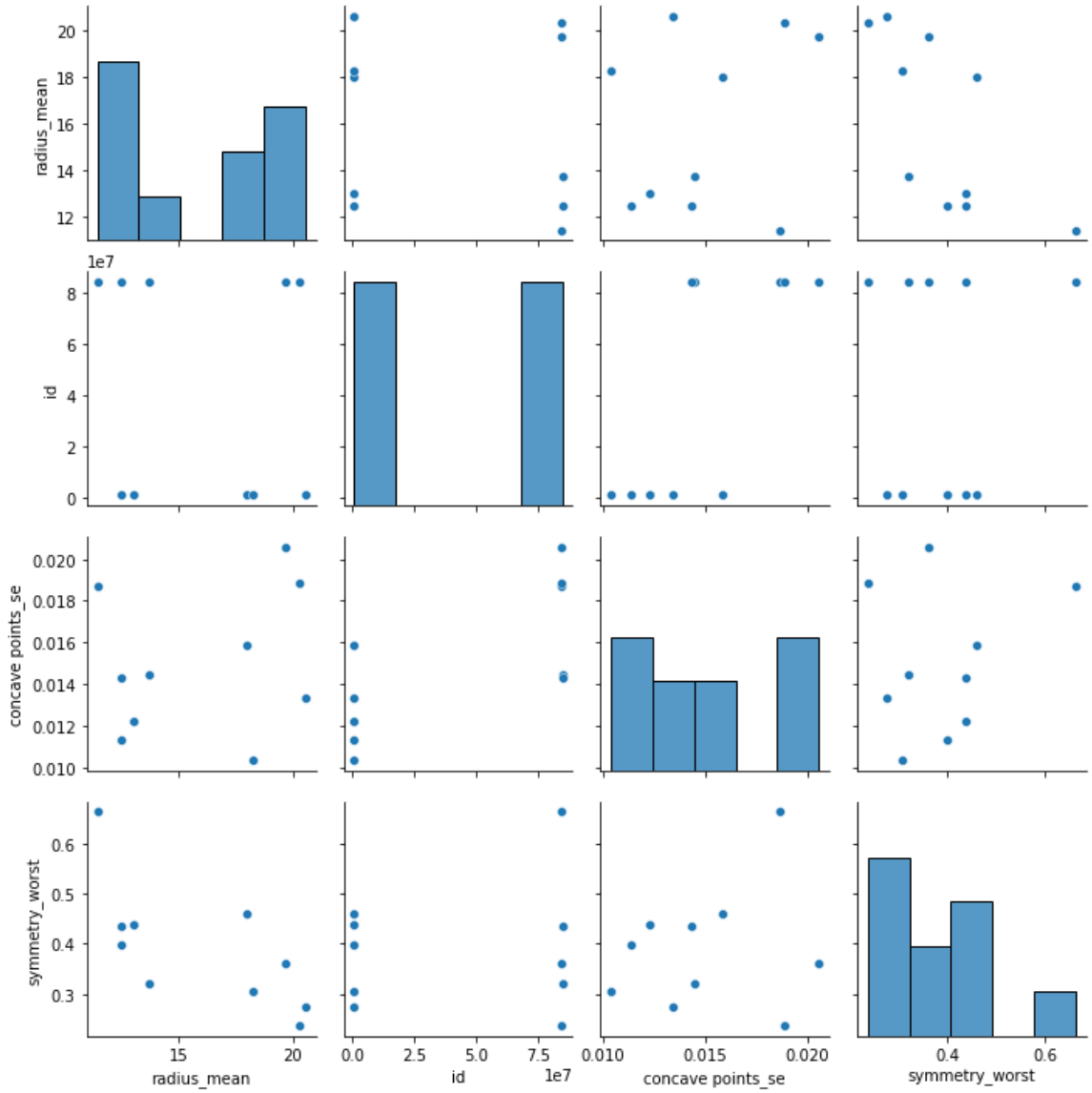
EDA AND VISUALIZATION

In [8]:

```
sns.pairplot(d)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x1c308c3faf0>
```



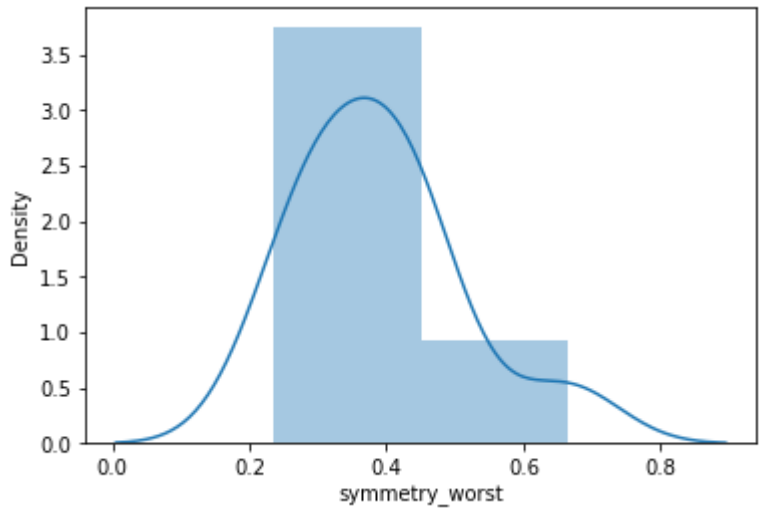
In [9]:

```
sns.distplot(c['symmetry_worst'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='symmetry_worst', ylabel='Density'>
```



In [10]:

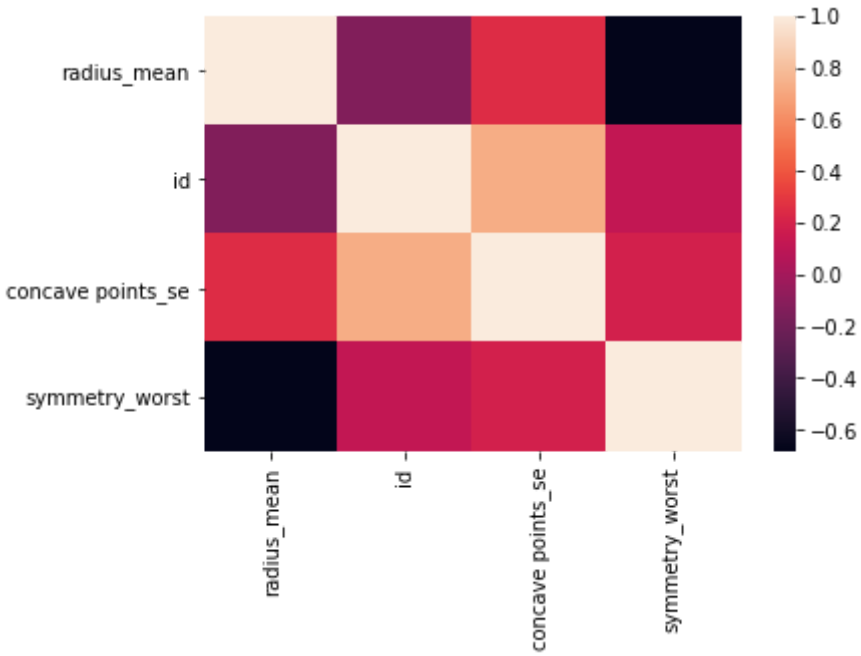
```
f=d[['diagnosis','radius_mean','id','concave points_se','symmetry_worst']]
```

In [11]:

```
sns.heatmap(f.corr())
```

Out[11]:

<AxesSubplot:>



In [12]:

```
x=d[['radius_mean','id','concave points_se']]
y=d['symmetry_worst']
```

In [13]:

```
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)
```

In [14]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:

```
LinearRegression()
```

In [15]:

```
print(lr.intercept_)
```

```
0.6359137078818231
```

In [16]:

```
r=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
r
```

Out[16]:

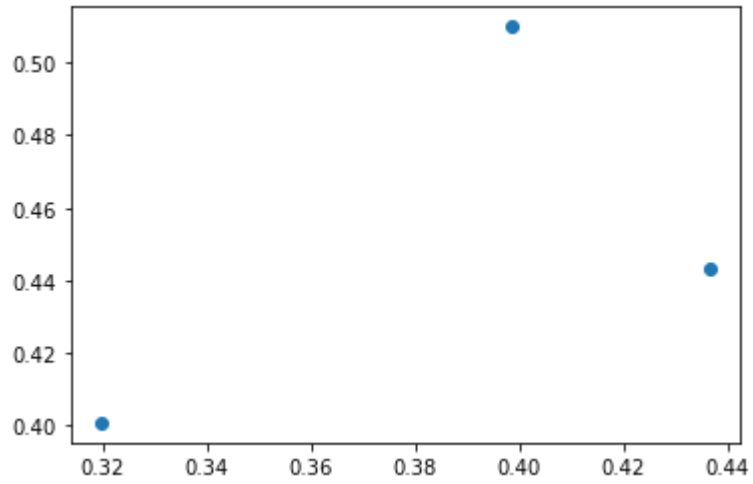
	Co-efficient
radius_mean	-3.782487e-02
id	-1.868782e-09
concave points_se	3.047789e+01

In [17]:

```
u=lr.predict(x_test)
plt.scatter(y_test,u)
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x1c30a6c8e80>
```



In [18]:

```
print(lr.score(x_test,y_test))
```

```
-1.6763702177832105
```

In [19]:

```
lr.score(x_train,y_train)
```

Out[19]:

```
0.8928063885882163
```

RIDGE REGRESSION

In [20]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:

```
Ridge(alpha=10)
```

In [22]:

```
rr.score(x_test,y_test)
```

Out[22]:

```
-9.262542207605945
```

In [23]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:

```
Lasso(alpha=10)
```

In [24]:

```
la.score(x_test,y_test)
```

Out[24]:

```
-0.5833117510651398
```

In [25]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x_train,y_train)
```

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(p.coef_)
```

```
[-0.00000000e+00  6.07322827e-10  0.00000000e+00]
```

In [27]:

```
print(p.intercept_)
```

```
0.3692847859420497
```

In [28]:

```
print(p.predict(x_test))
```

```
[0.42060417 0.42057818 0.36979724]
```

In [29]:

```
prediction=p.predict(x_test)
print(p.score(x_test,y_test))
```

```
-0.5833197071734311
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.04855892358301578
```

In [32]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.003758769318435582
```

In [33]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.06130880294407633
```

In [ ]: