In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [62]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction (1).csv")
a
```

0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030
...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100
565	926682	M	20.13	28.25	131.20	1261.0	0.09780
566	926954	M	16.60	28.08	108.30	858.1	0.08455
567	927241	M	20.60	29.33	140.10	1265.0	0.11780
568	92751	B	7.76	24.54	47.92	181.0	0.05263

569 rows × 33 columns

In [63]:
```
a=a.head(10)
a
```

Out[63]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	
5	843786	M	12.45	15.70	82.57	477.1	0.12780	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	
8	844981	M	13.00	21.82	87.50	519.8	0.12730	
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	

10 rows × 33 columns

In [64]:
```
a.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 [65]:
```
a.columns
```

Out[65]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

In [66]:

```
d=a[['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean','fractal_dimension_mean']]
d
```

Out[66]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavi
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	
5	12.45	15.70	82.57	477.1	0.12780	0.17000	
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	
7	13.71	20.83	90.20	577.9	0.11890	0.16450	
8	13.00	21.82	87.50	519.8	0.12730	0.19320	
9	12.46	24.04	83.97	475.9	0.11860	0.23960	

In [72]:

```
a.describe()
```

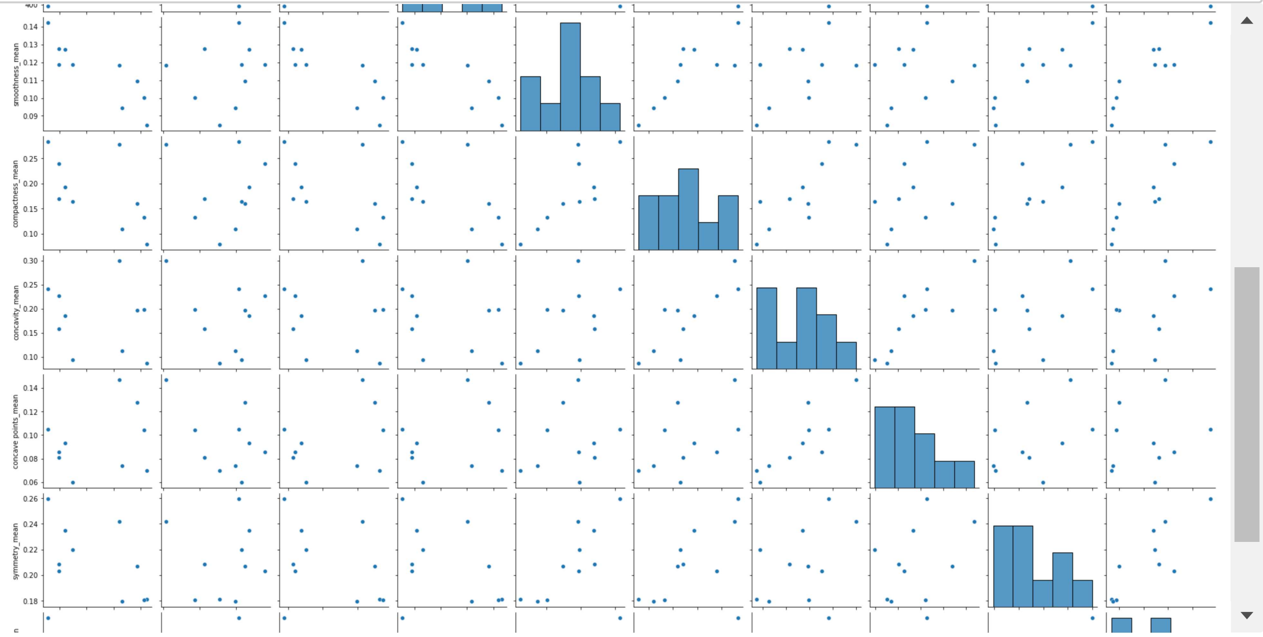
Out[72]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compact
count	1.000000e+01	10.000000	10.00000	10.000000	10.000000	10.000000	
mean	4.261848e+07	15.983000	18.64900	106.222000	830.380000	0.114277	
std	4.403463e+07	3.686001	4.10719	23.680745	377.613035	0.017262	
min	8.423020e+05	11.420000	10.38000	77.580000	386.100000	0.084740	
25%	8.439292e+05	12.595000	16.21750	84.852500	487.775000	0.102625	
50%	4.257294e+07	15.850000	20.18000	104.900000	789.450000	0.118500	
75%	8.435588e+07	19.330000	21.14500	128.200000	1162.250000	0.125200	
max	8.450100e+07	20.570000	24.04000	135.100000	1326.000000	0.142500	

8 rows × 32 columns

In [68]:

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



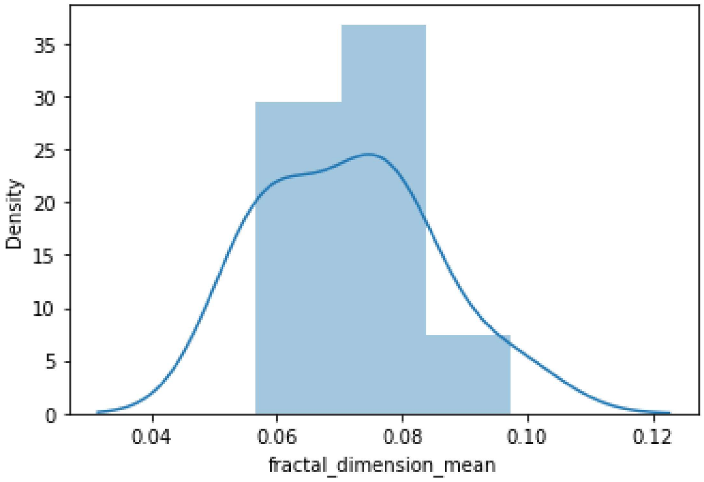
In [73]:

```
sns.distplot(a['fractal_dimension_mean'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[73]:

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

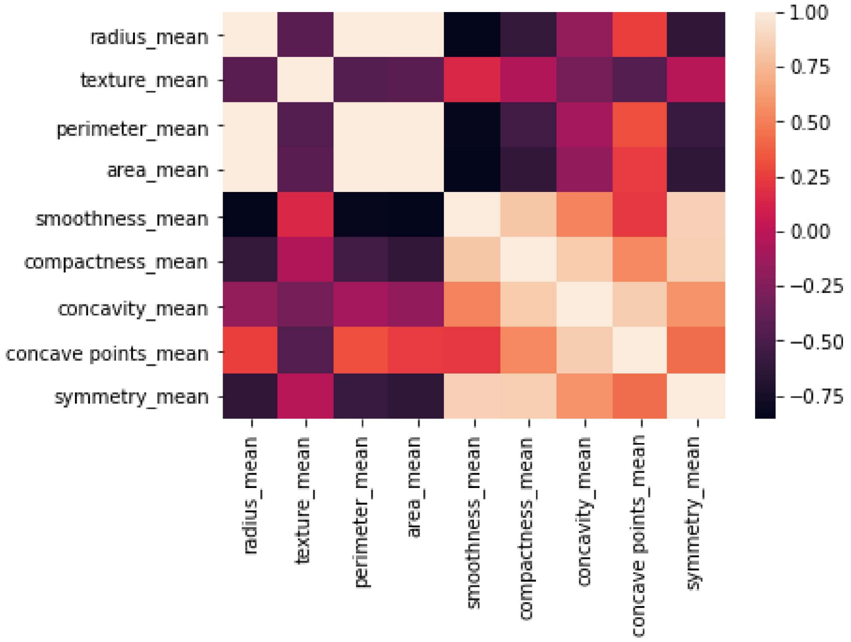


In [74]:

```
x1=a[['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean']]
```

In [75]:
```
sns.heatmap(x1.corr())
```

Out[75]:
```
<AxesSubplot:>
```



In [77]:
```
x=a[['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean']]
y=a['fractal_dimension_mean']
```

In [78]:
```
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 [79]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[79]:
```
LinearRegression()
```

In [80]:
```
print(lr.intercept_)
```

```
0.12629427211999972
```

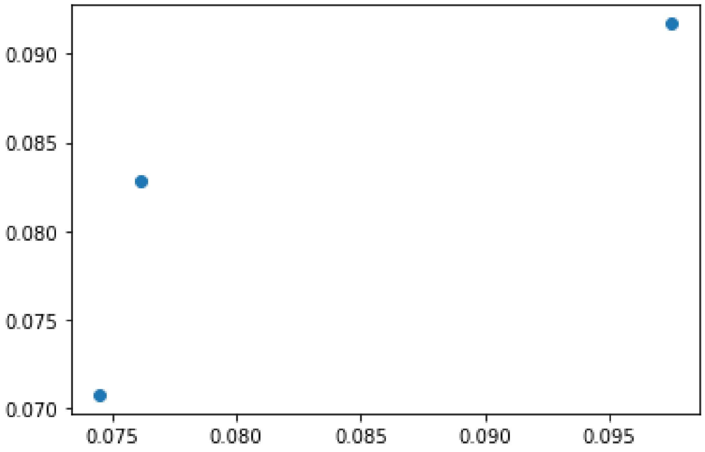
In [81]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[81]:

	Co-efficient
radius_mean	0.021708
texture_mean	-0.000980
perimeter_mean	-0.004010
area_mean	0.000018
smoothness_mean	-0.015172
compactness_mean	0.109463
concavity_mean	0.076163
concave points_mean	0.002074
symmetry_mean	-0.023103

In [82]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[82]:
```
<matplotlib.collections.PathCollection at 0x190acd60f40>
```



In [83]:
```
print(lr.score(x_test,y_test))
```

```
0.7197225917744946
```

In [84]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [85]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[85]:
```
Ridge(alpha=10)
```

In [86]:
```
rr.score(x_test,y_test)
```

Out[86]:
```
0.16415218142979626
```

```
In [87]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[87]: Lasso(alpha=10)
```

```
In [88]: la.score(x_test,y_test)

Out[88]: -2.2996945609419837
```

```
In [89]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()
         en.fit(x_train,y_train)

Out[89]: ElasticNet()
```

```
In [90]: print(en.coef_)

         [-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -2.09559085e-05
           0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
           0.00000000e+00]
```

```
In [91]: print(en.intercept_)

         0.08739344476152233
```

```
In [92]: print(en.predict(x_test))

         [0.07528303 0.07739538 0.07930237]
```

```
In [93]: en.score(x_test,y_test)

Out[93]: -0.011187395926438137
```

```
In [94]: from sklearn import metrics
```

```
In [95]: print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))

         Mean Absolute Error 0.005393903833292972
```

```
In [96]: print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))

         Mean Squared Error 3.059774140308087e-05
```

```
In [97]: print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))

          Root Mean Squared Error 0.005531522521248636
```