In [76]:
```
# import libaries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [110]:
```
x=pd.read_csv(r"C:\Users\user\Downloads\\8_BreastCancerPrediction.csv")
```

Out[110]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0.
2	84300903	M	19.69	21.25	130.00	1203.0	0.
3	84348301	M	11.42	20.38	77.58	386.1	0.
4	84358402	M	20.29	14.34	135.10	1297.0	0.
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0
565	926682	M	20.13	28.25	131.20	1261.0	0.
566	926954	M	16.60	28.08	108.30	858.1	0.
567	927241	M	20.60	29.33	140.10	1265.0	0
568	92751	B	7.76	24.54	47.92	181.0	0.

569 rows × 33 columns

In [111]:
```
x=x.head(10)
```

Out[111]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.10
5	843786	M	12.45	15.70	82.57	477.1	0.12
6	844359	M	18.25	19.98	119.60	1040.0	0.09
7	84458202	M	13.71	20.83	90.20	577.9	0.11
8	844981	M	13.00	21.82	87.50	519.8	0.12
9	84501001	M	12.46	24.04	83.97	475.9	0.11

10 rows × 33 columns

In [112]:

```
<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 [113]:

Out[113]:

```
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 [114]:
```
d=x[['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean']]
```

Out[114]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean
0	842302	M	17.99	10.38	122.80
1	842517	M	20.57	17.77	132.90
2	84300903	M	19.69	21.25	130.00
3	84348301	M	11.42	20.38	77.58
4	84358402	M	20.29	14.34	135.10
5	843786	M	12.45	15.70	82.57
6	844359	M	18.25	19.98	119.60
7	84458202	M	13.71	20.83	90.20
8	844981	M	13.00	21.82	87.50
9	84501001	M	12.46	24.04	83.97

In [115]:

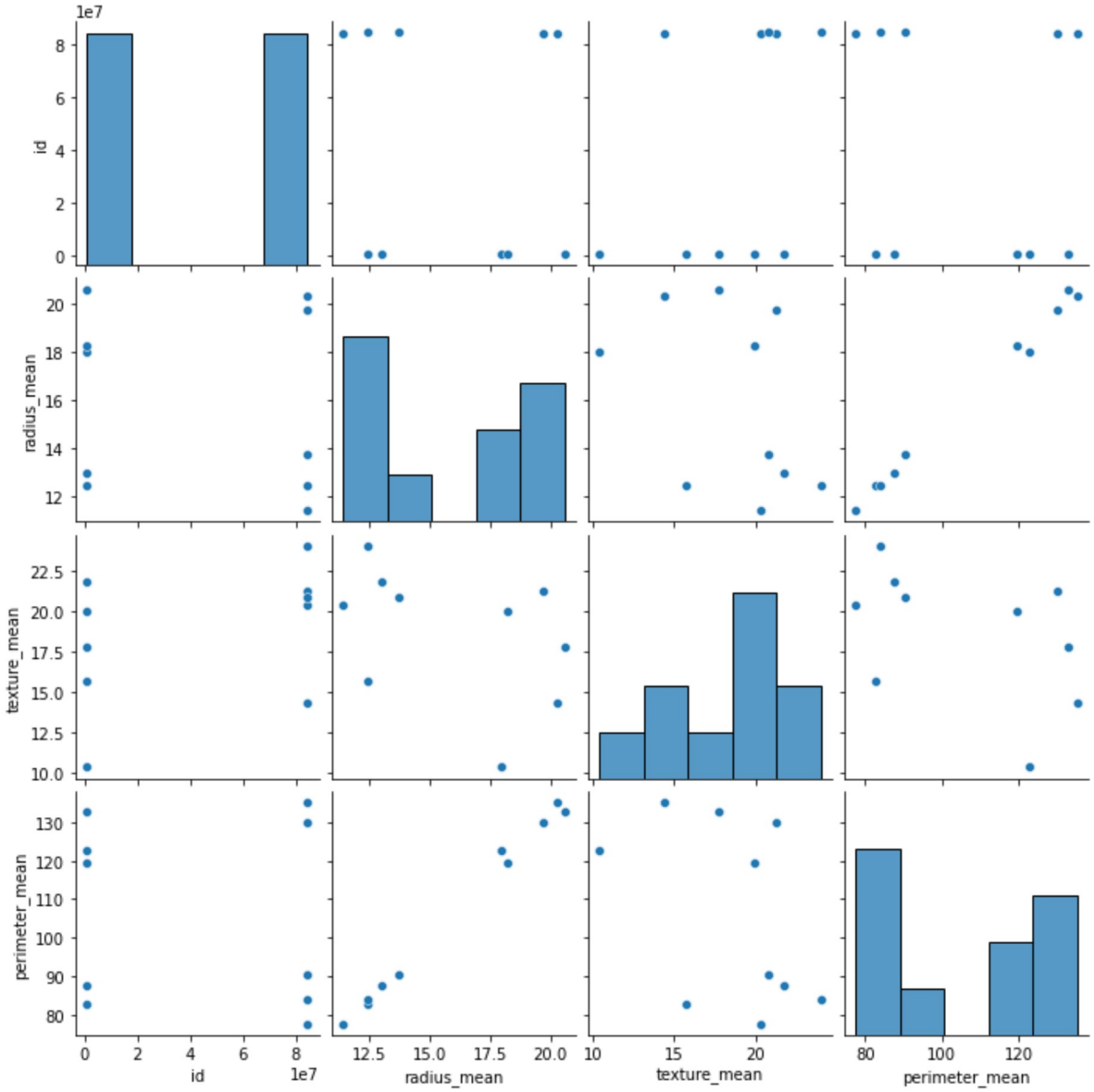
Out[115]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	1.000000e+01	10.000000	10.00000	10.000000	10.000000	10.0000(
mean	4.261848e+07	15.983000	18.64900	106.222000	830.380000	0.1142
std	4.403463e+07	3.686001	4.10719	23.680745	377.613035	0.0172(
min	8.423020e+05	11.420000	10.38000	77.580000	386.100000	0.0847
25%	8.439292e+05	12.595000	16.21750	84.852500	487.775000	0.1026
50%	4.257294e+07	15.850000	20.18000	104.900000	789.450000	0.1185(
75%	8.435588e+07	19.330000	21.14500	128.200000	1162.250000	0.1252(
max	8.450100e+07	20.570000	24.04000	135.100000	1326.000000	0.1425(

8 rows × 32 columns

In [116]:

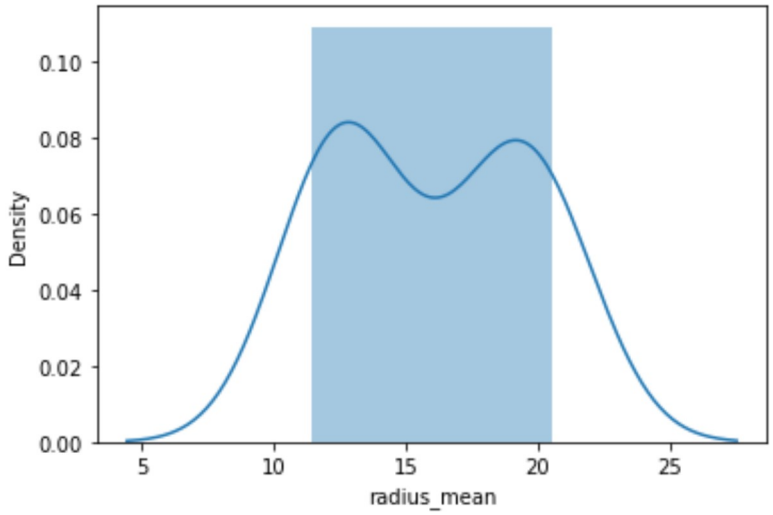
Out[116]: <seaborn.axisgrid.PairGrid at 0x190c2ffc700>



In [117]:

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

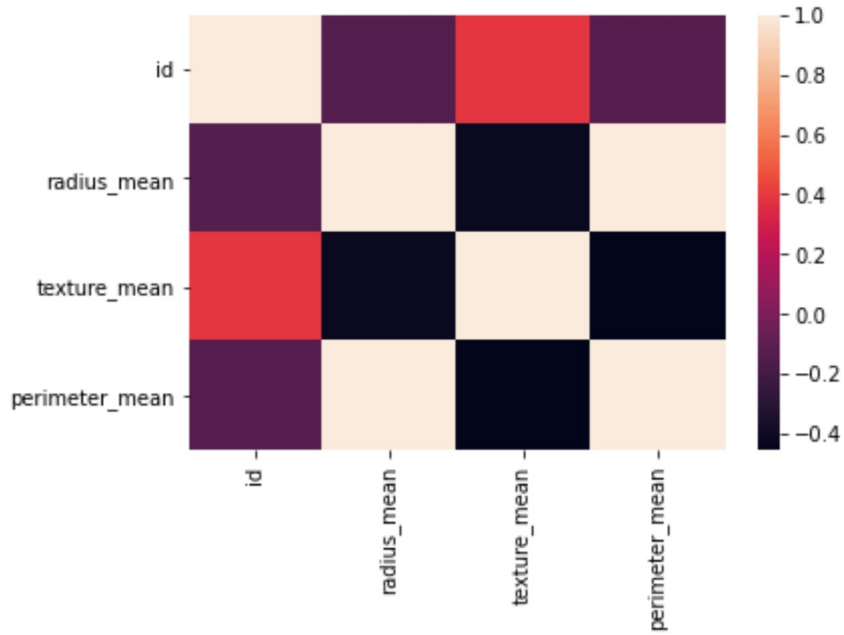
Out[117]: <AxesSubplot:xlabel='radius_mean', ylabel='Density'>



In [119]:

In [120]:

Out[120]: <AxesSubplot:>



In [122]:

```
x=x1[['radius_mean']]
y=x1['radius_mean']
```

In [123]:
```
# to split my dataset into traning and test date

from sklearn.model_selection import train_test_split
```

In [124]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
```

Out[124]:
```
LinearRegression()
```

In [125]:

```
0.0
```

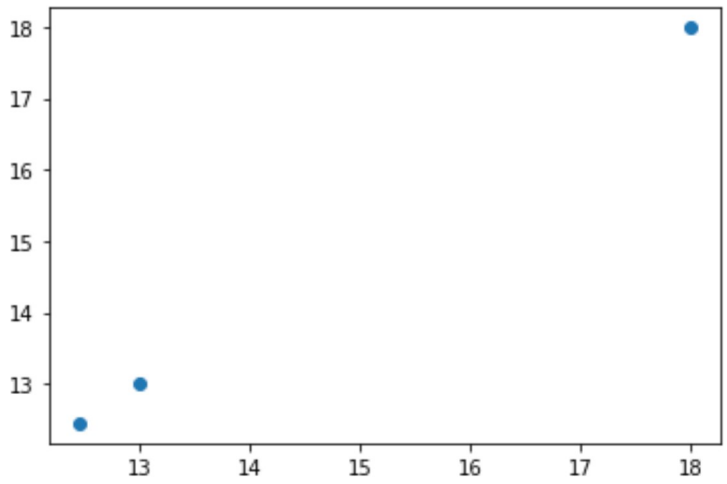
In [126]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[126]:

	Co-efficient
radius_mean	1.0

In [127]:
```
prediction=lr.predict(x_test)
```

Out[127]:
```
<matplotlib.collections.PathCollection at 0x190c29f27f0>
```



In [128]:

Out[128]:
```
1.0
```

In [129]:

Out[129]:
```
1.0
```

In [130]:

In [131]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[131]: 0.9838808001860283

In [132]:
```
la=Lasso(alpha=10)
```

Out[132]: Lasso(alpha=10)

In [133]:

Out[133]: 0.03310287982290383

In [134]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
```

Out[134]: ElasticNet()

In [135]:

Out[135]: array([0.92818048])

In [136]:

Out[136]: array([12.75000038, 13.26049965, 17.89212026])

In [137]:

Out[137]: 1.1941533636890487

In [138]:

Out[138]: 0.9910129813944396

In [139]:

In [140]:

Mean Absolute Error 0.0

In [141]:

Mean Squared Error 0.0

In [142]:

Root Mean Squared Error 0.0

In [ ]: