In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as py
import seaborn as sns
```

In [2]:
```
d=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction - 8_BreastCancerPredic
d
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
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 × 32 columns

In [3]:
```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
```

```
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [4]:
```
d.head()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cor
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 rows × 32 columns

In [5]:
```
d.describe()
```

Out[5]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 31 columns

In [6]:
```
d.columns
```

Out[6]:
```
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'],
      dtype='object')
```

In [7]:
```
d.index
```

Out[7]:
```
RangeIndex(start=0, stop=569, step=1)
```

In [8]:
```
d=d.head(100)
d
```

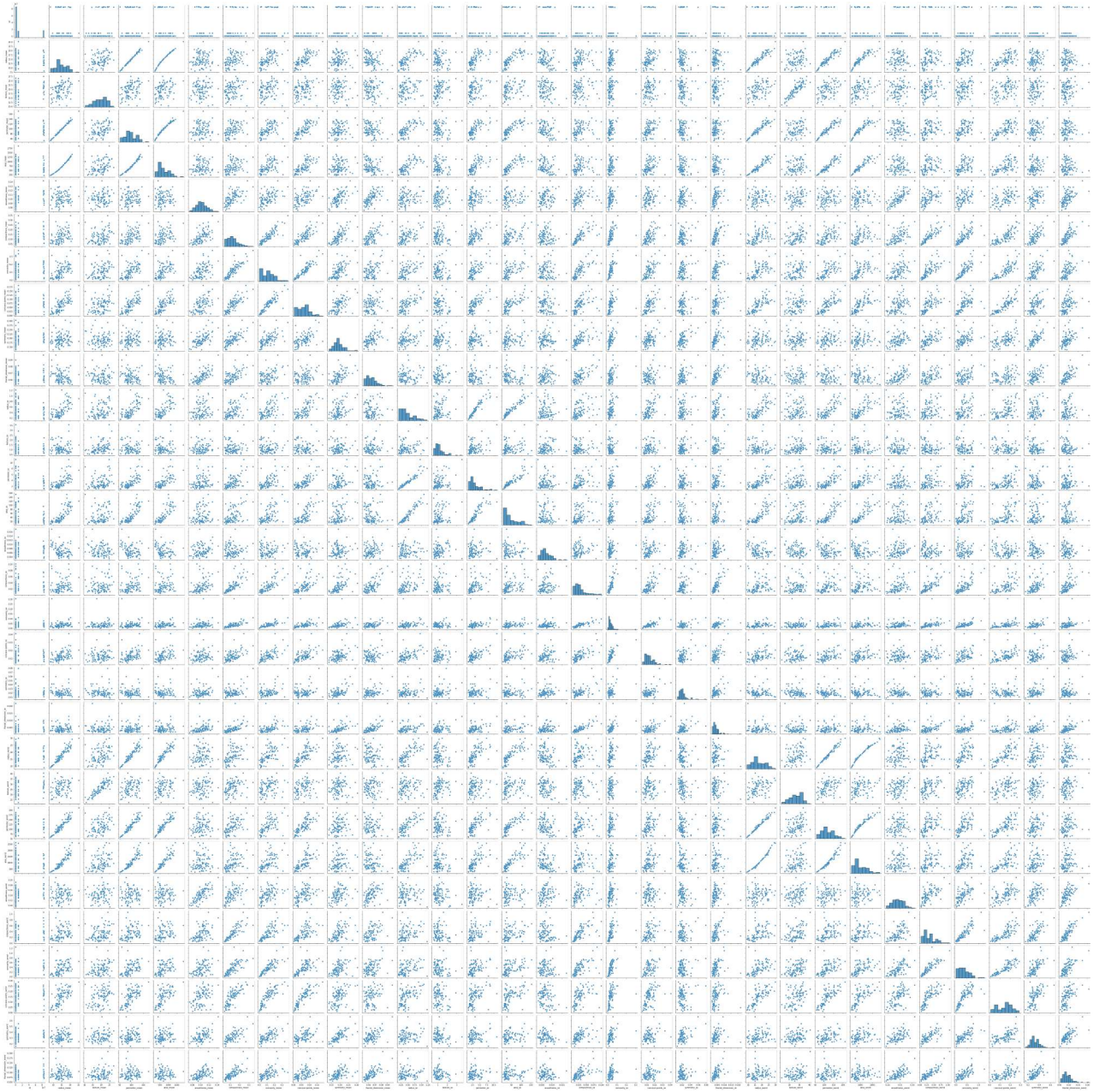
Out[8]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cc
0	842302	M	17.990	10.38	122.80	1001.0	0.11840	
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.690	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.420	20.38	77.58	386.1	0.14250	
4	84358402	M	20.290	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	...	
95	86208	M	20.260	23.03	132.40	1264.0	0.09078	
96	86211	B	12.180	17.84	77.79	451.1	0.10450	
97	862261	B	9.787	19.94	62.11	294.5	0.10240	
98	862485	B	11.600	12.84	74.34	412.6	0.08983	
99	862548	M	14.420	19.77	94.48	642.5	0.09752	

100 rows × 32 columns

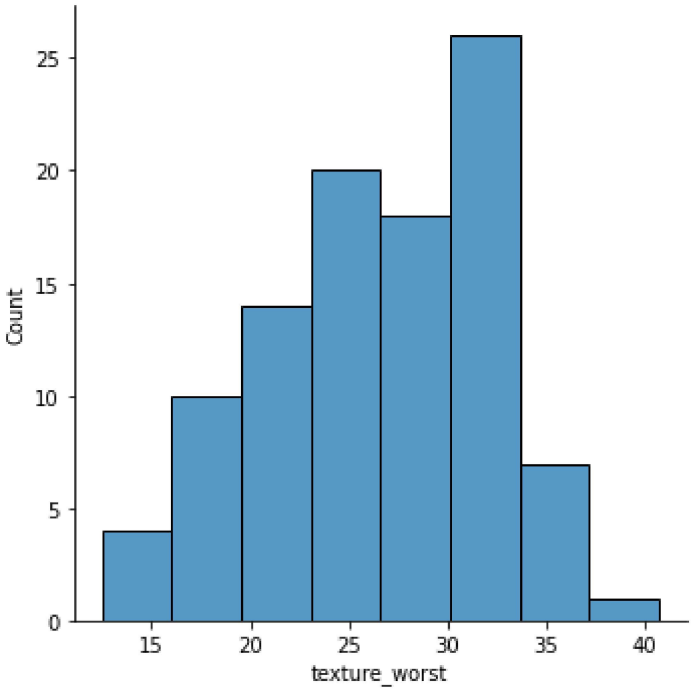
In [9]:
```
sns.pairplot(d)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1c611a02ac0>
```



In [10]:
```
sns.displot(d['texture_worst'])
```

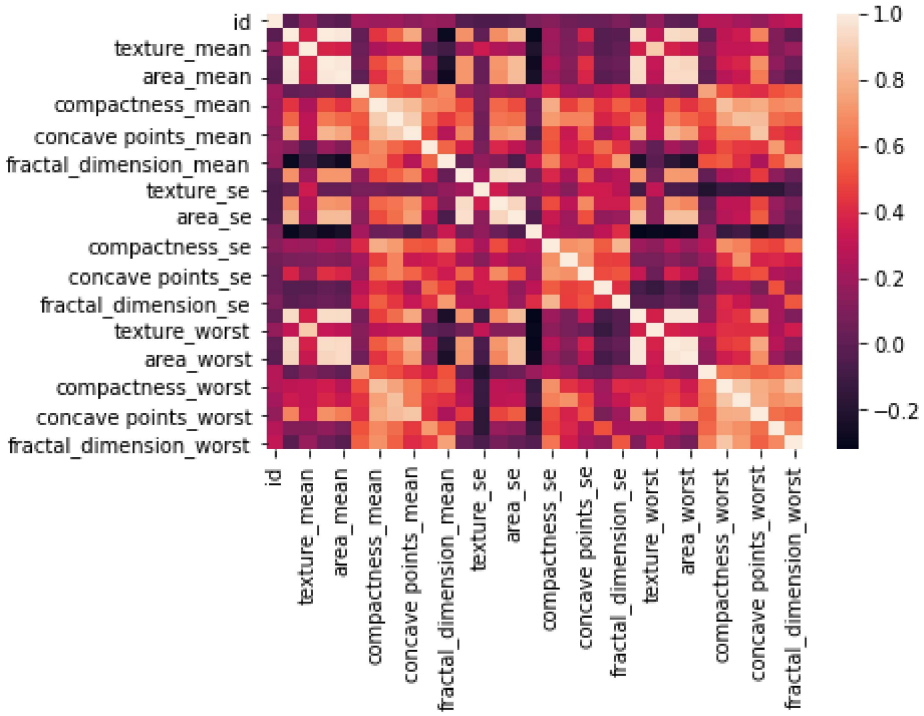
Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1c62f4c9a90>
```



In [11]:

```
d1=d[['id','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']]
sns.heatmap(d1.corr())
```

Out[11]: <AxesSubplot:>



In [12]:
```
x=d1[['id','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',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']]
y=d1['texture_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
```

In [15]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
7.768370725512021
```

In [17]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

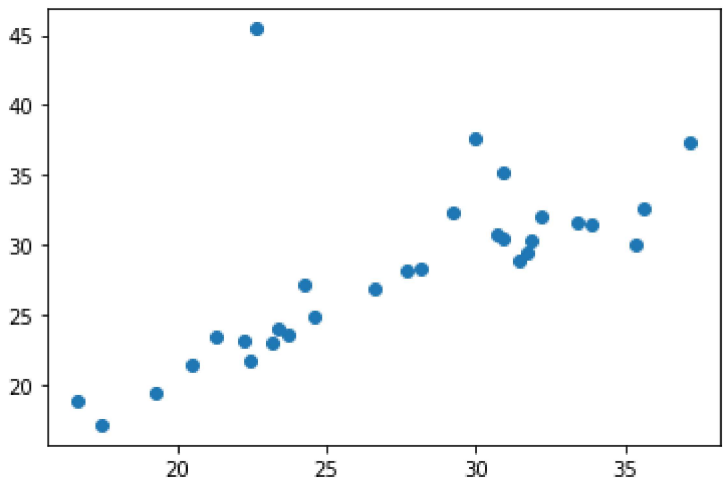
Out[17]:

	Co-efficient
id	-6.167724e-09
radius_mean	-4.762474e+00
texture_mean	9.983675e-01
perimeter_mean	5.037197e-01
area_mean	6.770820e-03
smoothness_mean	-4.229362e+00
compactness_mean	-1.684254e+01
concavity_mean	5.776701e-01
concave points_mean	-1.678392e+01
symmetry_mean	5.073842e+00
fractal_dimension_mean	-1.138669e+02
radius_se	-1.358313e+01
texture_se	4.267078e+00

	Co-efficient
perimeter_se	8.906163e-01
area_se	2.749287e-02
smoothness_se	3.100938e+00
compactness_se	-9.982537e+01
concavity_se	1.586933e+02
concave points_se	-2.532030e+02
symmetry_se	-1.788652e+02
fractal_dimension_se	3.714711e+02
radius_worst	1.862902e+00
perimeter_worst	-9.672876e-02
area_worst	-4.829339e-03
smoothness_worst	7.298086e+00
compactness_worst	1.179993e+01
concavity_worst	-1.256368e+01
concave points_worst	2.084799e+01
symmetry_worst	1.771838e+01
fractal_dimension_worst	4.577291e+00

In [18]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1c63e6b1640>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.26020604163599714
```

In [20]:
```
print(lr.score(x_train,y_train))
```

```
0.979974494921878
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[22]:
```
Ridge(alpha=10)
```

In [23]:
```
rr.score(x_test,y_test)
```

Out[23]:
```
0.8526287338234598
```

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.3758727320834674
```

In [26]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[ 5.39367002e-09 -0.00000000e+00  1.16721822e+00  0.00000000e+00
 -8.64161965e-03  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -5.68378395e-02 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  7.99481821e-02  4.33108567e-03
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
```

In [28]:
```
print(en.intercept_)
```

```
-1.3079187673180286
```

In [29]:
```
print(en.predict(x_test))
```

```
[28.56418411 24.7240092  22.41132316 32.03050919 24.70105322 30.55899175
 28.64462008 32.59379398 34.47040646 22.16050009 18.29861885 27.33660677
```

```
 30.41000412 29.93776065 28.29758086 29.69104229 22.83219592 27.3787687
 25.69795334 24.90453323 29.98359871 24.9361333  31.50434211 22.39846774
 18.58349231 20.31749044 29.23170681 36.43265101 37.75812747 29.67103041]
```

In [30]:
```
print(en.score(x_test,y_test))
```

```
0.8005361031884479
```

In [31]:
```
from sklearn import metrics
```

In [32]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 2.319831500938191
```

In [33]:
```
print(en.coef_)
```

```
[ 5.39367002e-09 -0.00000000e+00  1.16721822e+00  0.00000000e+00
 -8.64161965e-03  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -5.68378395e-02 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  7.99481821e-02  4.33108567e-03
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
```

In [34]:
```
print(en.intercept_)
```

```
-1.3079187673180286
```

In [35]:
```
print(en.predict(x_test))
```

```
[28.56418411 24.7240092  22.41132316 32.03050919 24.70105322 30.55899175
 28.64462008 32.59379398 34.47040646 22.16050009 18.29861885 27.33660677
 30.41000412 29.93776065 28.29758086 29.69104229 22.83219592 27.3787687
 25.69795334 24.90453323 29.98359871 24.9361333  31.50434211 22.39846774
 18.58349231 20.31749044 29.23170681 36.43265101 37.75812747 29.67103041]
```

In [36]:
```
print(en.score(x_test,y_test))
```

```
0.8005361031884479
```

In [37]:
```
from sklearn import metrics
```

In [38]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 2.319831500938191
```

In [39]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 22.924561248004174
```

In [40]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 4.787960029908789
```

In [ ]: