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

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\8_BreastCancerPrediction - 8_BreastCa
```

In [3]:
```
df.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]:
```
df.describe()
```

Out[4]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.09636
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.01406
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.05263
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.08637
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.09587
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.10530
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.16340

8 rows × 31 columns

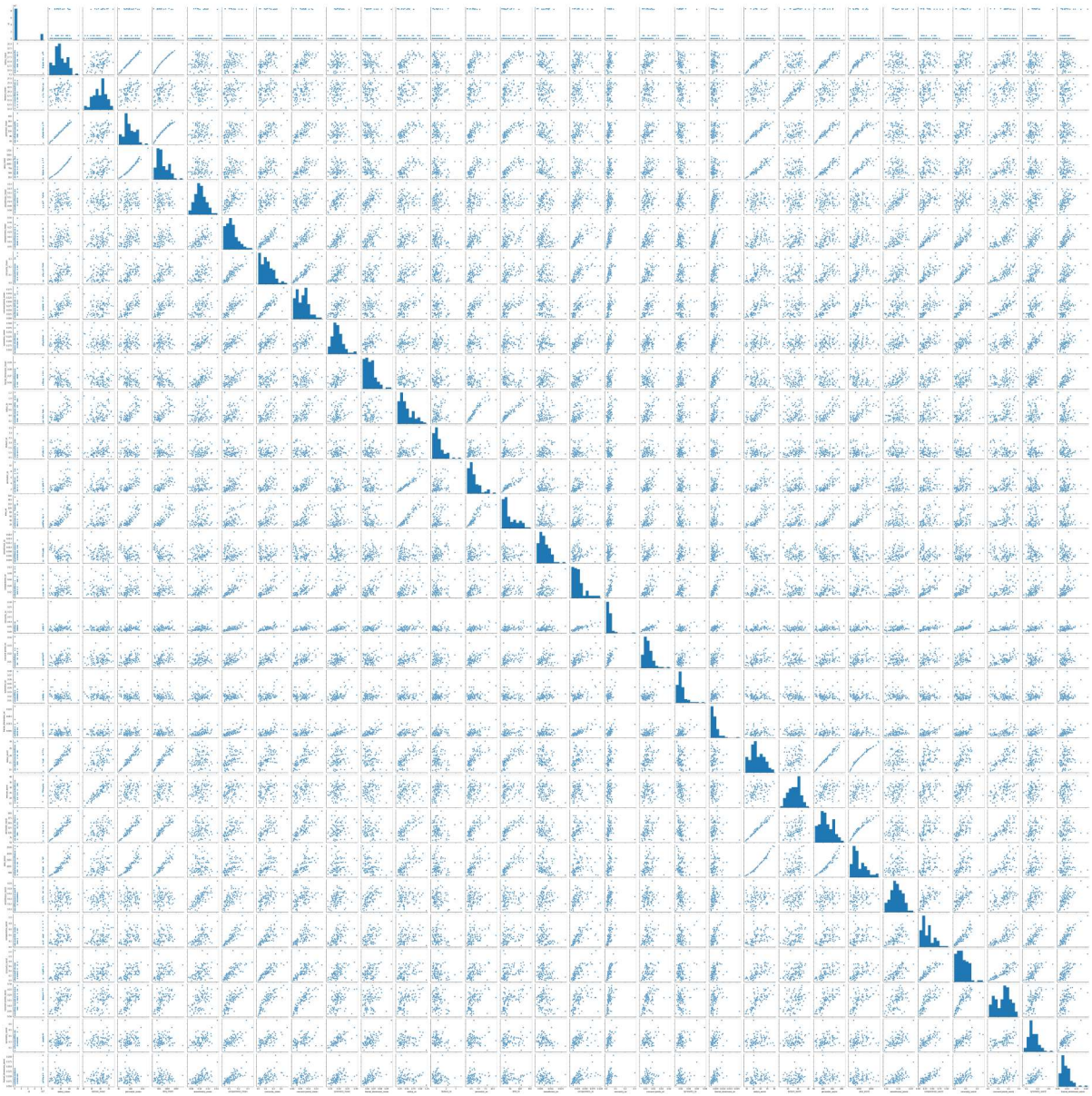
In [5]:
```
df1=df.head(100)
```

In [6]:
```
df1.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')
```

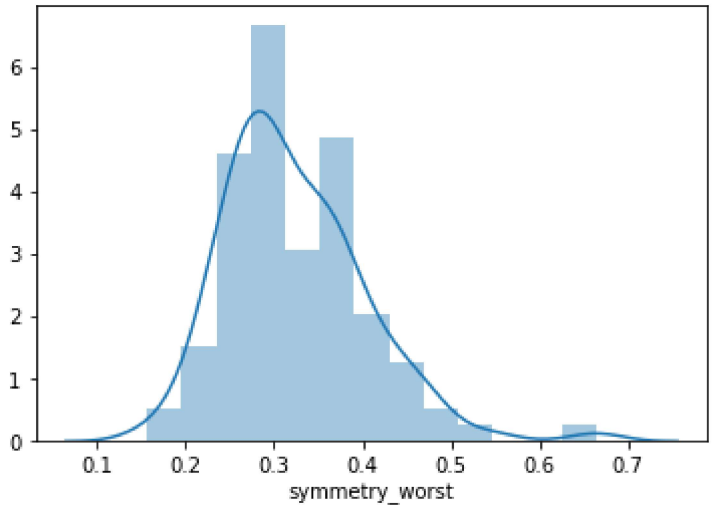
In [7]:
```
sns.pairplot(df1)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x267e52f71f0>
```



In [8]:
```
sns.distplot(df1['symmetry_worst'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2678944e490>
```



In [9]:
```
df2=df1[[ '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']]
df2
```

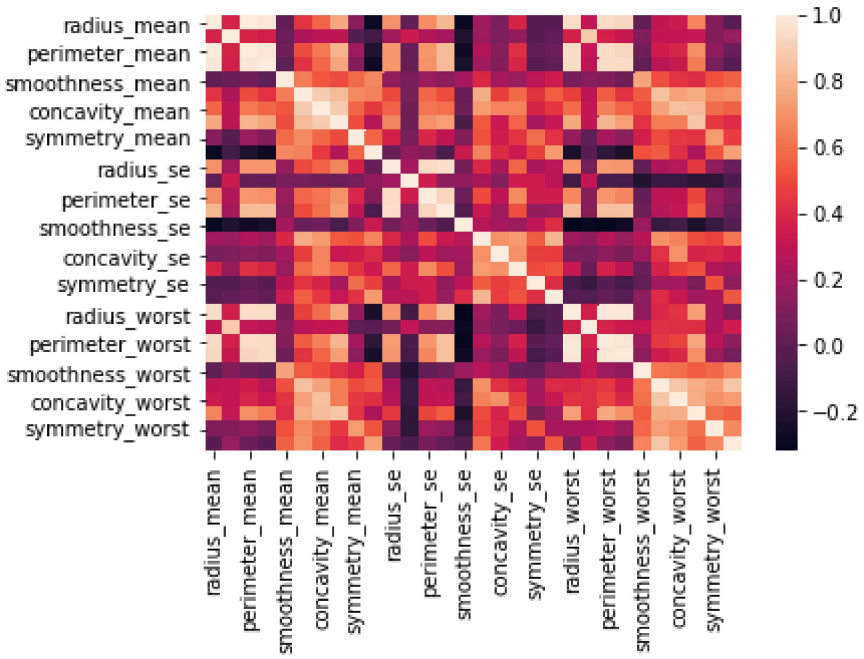
Out[9]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
0	17.990	10.38	122.80	1001.0	0.11840	0.277
1	20.570	17.77	132.90	1326.0	0.08474	0.078
2	19.690	21.25	130.00	1203.0	0.10960	0.159
3	11.420	20.38	77.58	386.1	0.14250	0.283
4	20.290	14.34	135.10	1297.0	0.10030	0.132
...	...	...	...	...	...	
95	20.260	23.03	132.40	1264.0	0.09078	0.131
96	12.180	17.84	77.79	451.1	0.10450	0.070
97	9.787	19.94	62.11	294.5	0.10240	0.053
98	11.600	12.84	74.34	412.6	0.08983	0.075
99	14.420	19.77	94.48	642.5	0.09752	0.114

100 rows × 30 columns

In [10]:
```
sns.heatmap(df2.corr())
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x26787ed0e50>



In [11]:
```
x=df2[['perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst']]
y=df2[['symmetry_worst']]
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

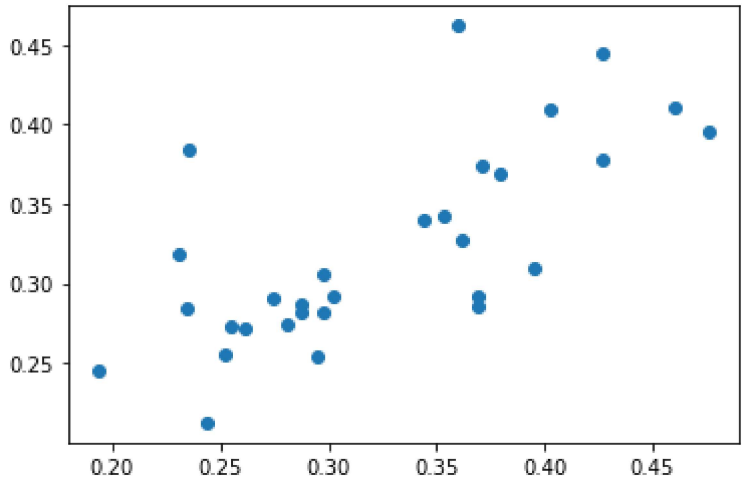
Out[13]: LinearRegression()

In [14]:
```
print(lr.intercept_)
```

```
[0.13189189]
```

In [15]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]: <matplotlib.collections.PathCollection at 0x267920dfdf0>



In [16]:
```
print(lr.score(x_test,y_test))
```

```
0.47886937827817955
```

In [17]:
```
print(lr.score(x_train,y_train))
```

```
0.6172360577900443
```

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[18]: Ridge(alpha=10)

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

Out[19]: 0.3986561963810217

In [20]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[20]: Lasso(alpha=10)

In [21]:
```
la.score(x_test,y_test)
```

Out[21]: -0.00010571697392713908

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]: ElasticNet()

In [23]:
```
print(en.intercept_)
```

```
[0.30179545]
```

In [24]:
```
print(en.predict(x_test))
```

```
[0.33072776 0.36020725 0.32224414 0.32205719 0.32177904 0.3522959
 0.34782723 0.33987028 0.34433894 0.33138894 0.33834273 0.31489821
 0.31602449 0.33879871 0.31770252 0.32801464 0.30933518 0.33243771
 0.32011925 0.30804018 0.34169422 0.3202378  0.33177653 0.33378287
 0.31507832 0.33054537 0.33141174 0.3177276  0.31014227 0.3157281 ]
```

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

```
0.06112963704830321
```

Evaluation

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.03730801599854435
```

In [28]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.002758770989881218
```

In [29]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean Squared Error: 0.05252400393992463
```

In [ ]:
```

```