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("8_BreastCancerPrediction.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
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 [3]:
```
df.head()
```

Out[3]:

	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 × 33 columns

DATA CLEANING AND DATA PREPROCESSING

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [5]:
```
df.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	

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 32 columns

In [6]:
```
df.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', 'Unnamed: 32'],
      dtype='object')
```

In [7]:
```
df1=df.dropna(axis=1)
df1
```

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
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 [8]:
```
df1.columns
```

Out[8]:
```
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',
```

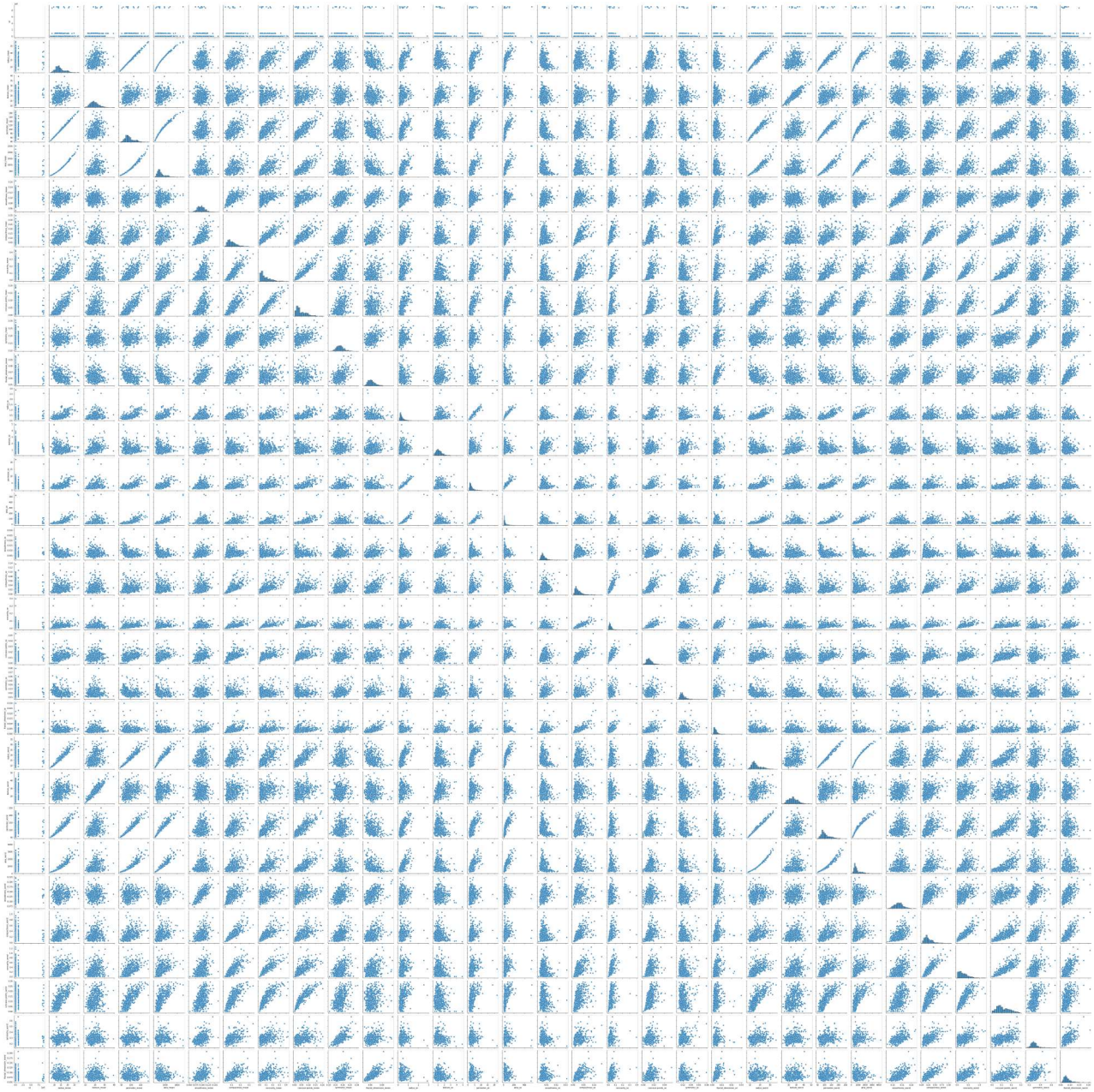
```
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

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

EDA AND VISUALIZATION

In [10]:
```
sns.pairplot(df1)
```

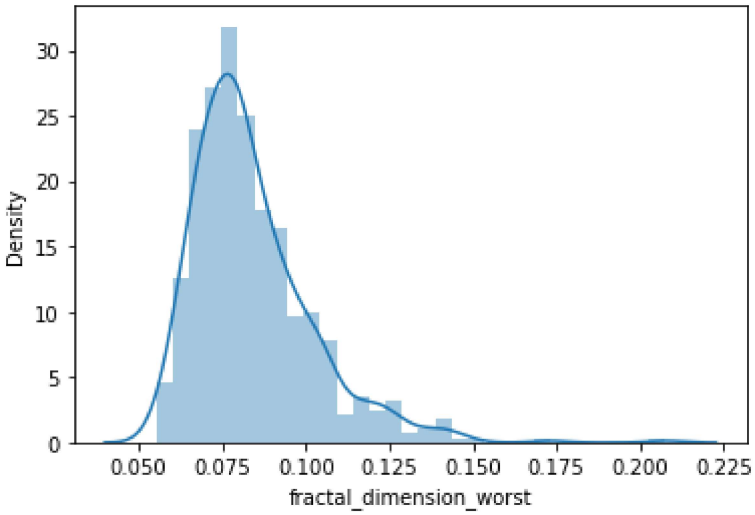
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x2196332aaf0>
```



In [11]:
```
sns.distplot(df1['fractal_dimension_worst'])
```

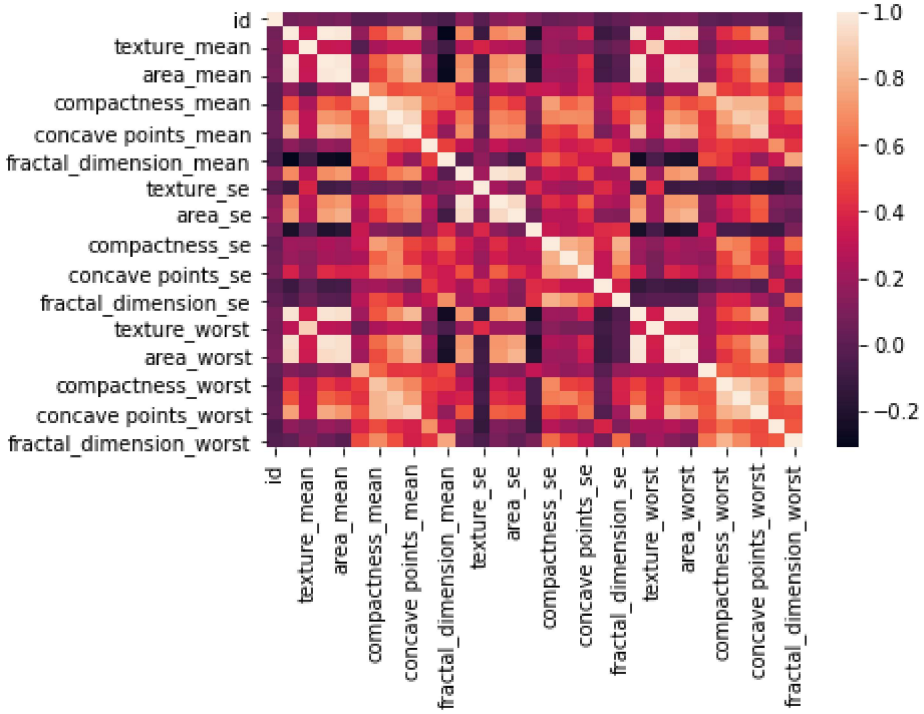
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function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

Out[11]: <AxesSubplot:xlabel='fractal_dimension_worst', ylabel='Density'>



In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [13]:
```
x=df[['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']]
y=df['fractal_dimension_worst']
```

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

In [16]:
```
lr.intercept_
```

Out[16]:
```
-0.019376606470375562
```

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

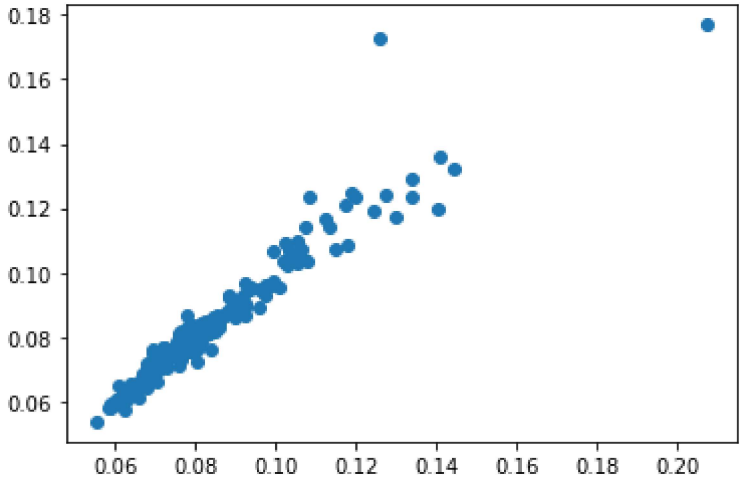
Out[17]:

	Co-efficient
id	-1.189407e-12
radius_mean	-1.893442e-03
texture_mean	1.639736e-04
perimeter_mean	6.698185e-05
area_mean	8.100353e-06
smoothness_mean	-1.187122e-01
compactness_mean	-1.106323e-01
concavity_mean	-1.131393e-02
concave points_mean	3.272328e-02
symmetry_mean	1.793499e-03
fractal_dimension_mean	1.254875e+00
radius_se	-1.662931e-02
texture_se	8.889301e-04
perimeter_se	1.179974e-03
area_se	5.073787e-05
smoothness_se	-2.391696e-01
compactness_se	-3.797623e-01
concavity_se	3.546400e-02
concave points_se	8.942628e-02
symmetry_se	-1.769514e-01
fractal_dimension_se	2.903695e+00

	Co-efficient
radius_worst	2.781068e-03
texture_worst	-1.465198e-04
perimeter_worst	-1.262191e-04
area_worst	-1.152201e-05
smoothness_worst	1.052709e-01
compactness_worst	9.211353e-02
concavity_worst	3.052297e-03
concave points_worst	-1.748221e-02
symmetry_worst	2.316532e-02

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

Out[18]: <matplotlib.collections.PathCollection at 0x21913aaa520>



ACCURACY

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

Out[19]: 0.9181851862109315

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

Out[20]: 0.9447284850571938

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

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAlgWar
ning: Ill-conditioned matrix (rcond=1.05303e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

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

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

Out[23]:
```
0.6984779912802299
```

In [24]:
```
rr.score(x_train,y_train)
```

Out[24]:
```
0.6821454972510337
```

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:
```
Lasso(alpha=10)
```

In [26]:
```
la.score(x_train,y_train)
```

Out[26]:
```
3.934589584320136e-05
```

In [27]:
```
la.score(x_test,y_test)
```

Out[27]:
```
-0.004846599255716066
```

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[28]:
```
ElasticNet()
```

In [29]:
```
en.coef_
```

Out[29]:
```
array([ 5.26894715e-13, -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,  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,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        6.31731369e-07,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00])
```

In [30]:
```
en.intercept_
```

Out[30]: 0.08298014415858686

In [31]:
```
prediction=en.predict(x_test)
```

In [32]:
```
en.score(x_test,y_test)
```

Out[32]: -0.001657057348537494

In [33]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
0.014671640695918482
0.0004208989700842258 5
0.02051582243255741
```