DAY-9

CANCER

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\user\Downloads\cancer.csv")[0:500]
df
```

Out[2]:

	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.·
...	...	...	...	...	...	...	
495	914333	B	14.87	20.21	96.12	680.9	0.(
496	914366	B	12.65	18.17	82.69	485.6	0.·
497	914580	B	12.47	17.31	80.45	480.1	0.(
498	914769	M	18.49	17.52	121.30	1068.0	0.·
499	91485	M	20.59	21.24	137.80	1320.0	0.·

500 rows × 33 columns

In [3]:
```
df.head(10)
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100
5	843786	M	12.45	15.70	82.57	477.1	0.127
6	844359	M	18.25	19.98	119.60	1040.0	0.094
7	84458202	M	13.71	20.83	90.20	577.9	0.118
8	844981	M	13.00	21.82	87.50	519.8	0.127
9	84501001	M	12.46	24.04	83.97	475.9	0.118

10 rows × 33 columns

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

Out[4]:

worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	poin
00000	500.000000	500.000000	500.000000	500.000000	500.000000	50
08500	108.258320	896.003200	0.131972	0.256324	0.276420	
53133	33.312706	571.074422	0.022739	0.159147	0.209012	
20000	50.410000	185.200000	0.071170	0.027290	0.000000	
17500	84.567500	522.600000	0.116200	0.145925	0.114475	
40000	97.980000	691.750000	0.131250	0.214850	0.231400	
50000	127.150000	1150.750000	0.146000	0.343525	0.389450	
40000	251.200000	4254.000000	0.222600	1.058000	1.252000	

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       500 non-null    int64  
 1   diagnosis                500 non-null    object 
 2   radius_mean              500 non-null    float64
 3   texture_mean             500 non-null    float64
 4   perimeter_mean           500 non-null    float64
 5   area_mean                500 non-null    float64
 6   smoothness_mean          500 non-null    float64
 7   compactness_mean         500 non-null    float64
 8   concavity_mean           500 non-null    float64
 9   concave points_mean      500 non-null    float64
 10  symmetry_mean            500 non-null    float64
 11  fractal_dimension_mean   500 non-null    float64
 12  radius_se                500 non-null    float64
 13  texture_se               500 non-null    float64
 14  perimeter_se             500 non-null    float64
 15  area_se                  500 non-null    float64
 16  smoothness_se            500 non-null    float64
 17  compactness_se           500 non-null    float64
 18  concavity_se             500 non-null    float64
 19  concave points_se        500 non-null    float64
 20  symmetry_se              500 non-null    float64
 21  fractal_dimension_se     500 non-null    float64
 22  radius_worst             500 non-null    float64
 23  texture_worst            500 non-null    float64
 24  perimeter_worst          500 non-null    float64
 25  area_worst               500 non-null    float64
 26  smoothness_worst         500 non-null    float64
 27  compactness_worst        500 non-null    float64
 28  concavity_worst          500 non-null    float64
 29  concave points_worst     500 non-null    float64
 30  symmetry_worst           500 non-null    float64
 31  fractal_dimension_worst  500 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 129.0+ KB
```

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]:
```
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', 'fractal_dimension_worst']]
y=df['fractal_dimension_worst']
```

In [8]:
```
#to split my dataset into traning and test data

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 [9]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[9]:
```
LinearRegression()
```

In [10]:
```
print(lr.intercept_)
```

```
7.018245845591409e-09
```

In [11]:
```
print(lr.score(x_test,y_test))
```

```
0.999999999999998
```

In [12]:
```
lr.score(x_train,y_train)
```

Out[12]:
```
0.9999999999999982
```

Ridge Regression

In [13]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [14]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:14
7: LinAlgWarning: Ill-conditioned matrix (rcond=1.06619e-18): result may not
be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

Out[14]:
```
Ridge(alpha=10)
```

In [15]:
```
rr.score(x_test,y_test)
```

Out[15]:
```
0.6890197630171238
```

Lasso Regression

In [16]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[16]:
```
Lasso(alpha=10)
```

In [17]:
```
la.score(x_test,y_test)
```

Out[17]:
```
0.0014969047264388191
```

In [ ]:
```

```