

Problem statement

Data collection

Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\bs.csv")
data
```

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

head

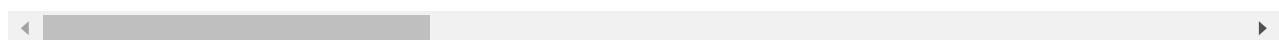
In [3]:

```
# to display first 8 dataset values
da=data.head(8)
da
```

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	843786	M	12.45	15.70	82.57	477.1	0.12780	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	

8 rows × 32 columns



info

In [4]:

```
# to identify missing values
data.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

```

describe

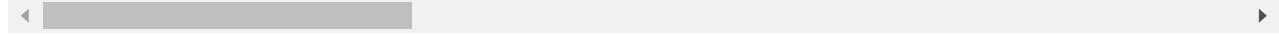
In [5]:

```
# to display summary of the dataset
data.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



columns

In [6]:

```
# to display headings of the dataset
data.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]:

```
a=data.dropna(axis=1)
a
```

Out[7]:

	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

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cor
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]:

`a.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]:

`df=data[0:5]`
`df`

Out[9]:

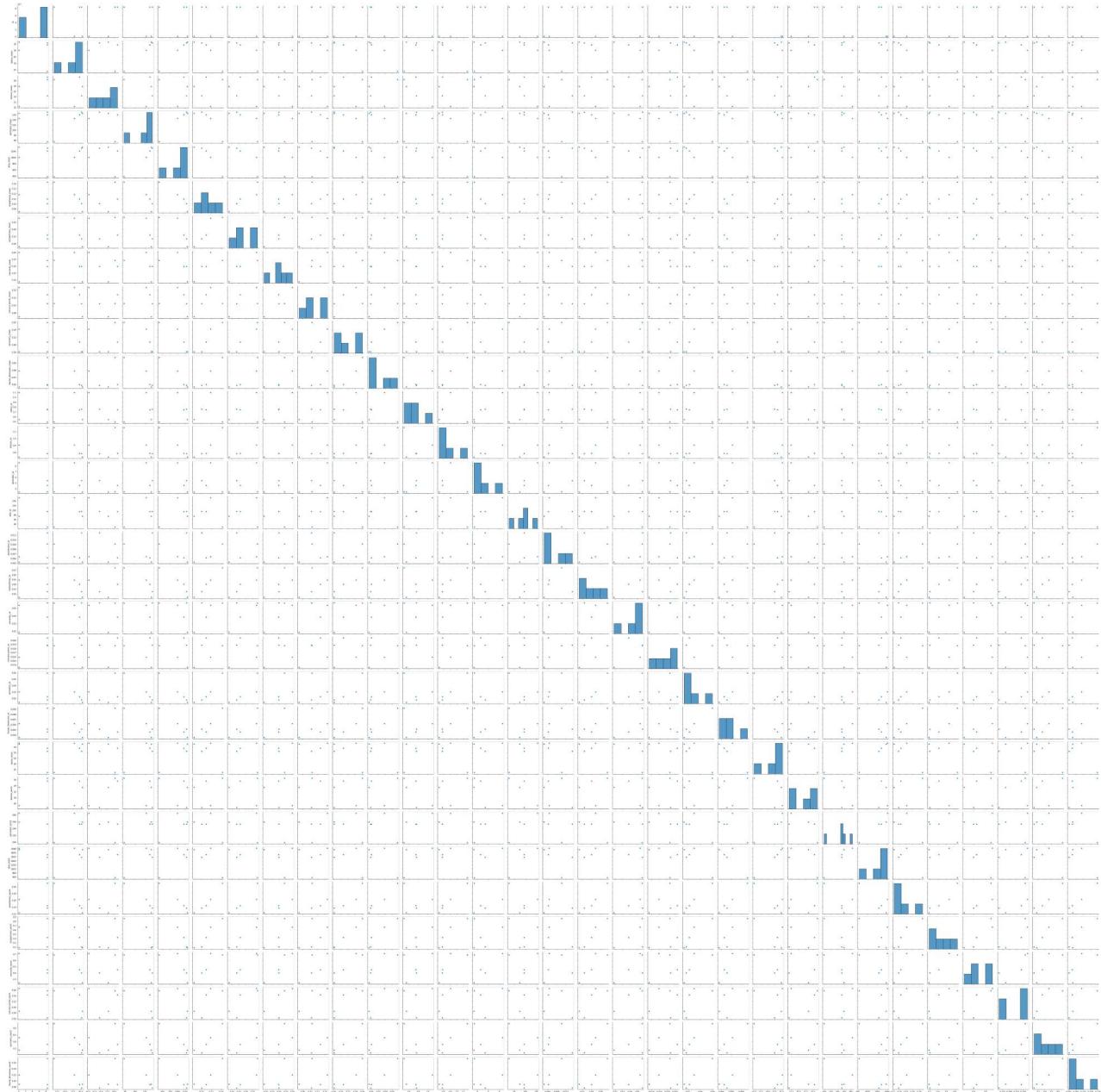
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cor
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 32 columns

EDA and Visualization

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

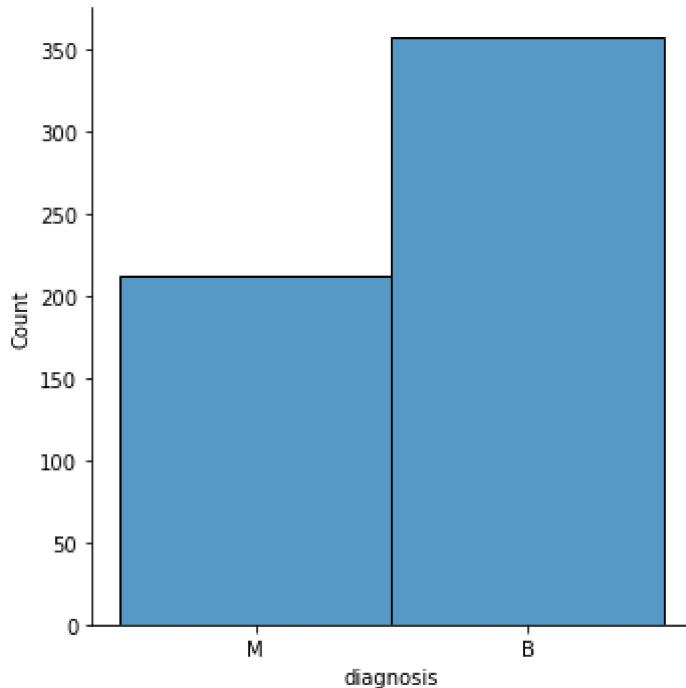
```
Out[10]: <seaborn.axisgrid.PairGrid at 0x25fc039b580>
```



distribution plot

```
In [11]: sns.displot(a["diagnosis"])
```

```
Out[11]: <seaborn.axisgrid.FacetGrid at 0x25fdf2262b0>
```

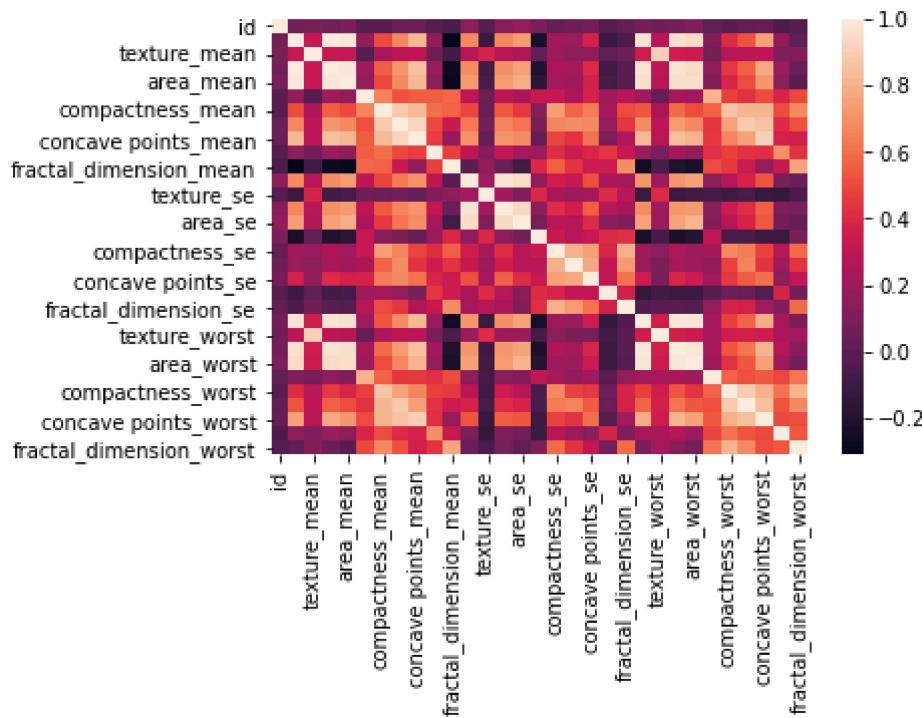


correlation

In [12]:

```
dat=data[['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']]  
sns.heatmap(dat.corr())
```

Out[12]: <AxesSubplot:>



To train the model-Model Building

```
In [13]:  
x=a[['id']]  
y=a['id']
```

```
In [14]:  
# to split my dataset into training 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 [15]:  
from sklearn.linear_model import LinearRegression  
lr= LinearRegression()  
lr.fit(x_train,y_train)
```

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

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

```
0.0
```

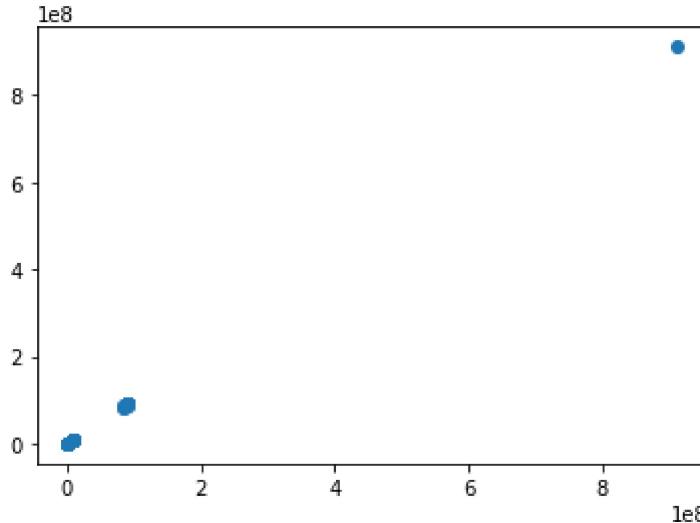
```
In [17]:  
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[17]: Co-efficient
```

id	1.0
-----------	-----

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

```
Out[18]: <matplotlib.collections.PathCollection at 0x25fed8a49d0>
```



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

```
1.0
```

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

```
Out[20]: 1.0
```

Ridge regression

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

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

```
Out[22]: 1.0
```

```
In [23]: rr.score(x_train,y_train)
```

```
Out[23]: 1.0
```

Lasso regression

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

```
la.score(x_train,y_train)
```

Out[24]: 1.0

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

Out[25]: 1.0