Problem statement

Data collection ¶

In [1]:

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

In [50]:

```
df=pd.read_csv(r"E:\Dataset\8_BreastCancerPrediction.csv")[0:500]
df
```

Out[50]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...	...	...
495	914333	B	14.87	20.21	96.12	680.9	0.09587	0.08345
496	914366	B	12.65	18.17	82.69	485.6	0.10760	0.13340
497	914580	B	12.47	17.31	80.45	480.1	0.08928	0.07630
498	914769	M	18.49	17.52	121.30	1068.0	0.10120	0.13170
499	91485	M	20.59	21.24	137.80	1320.0	0.10850	0.16440

500 rows × 33 columns

In [51]:

```
df.head()
```

Out[51]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	co
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	

5 rows × 33 columns

In [52]:

```
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 [53]:

```
#to display summary of statistics
df.describe()
```

Out[53]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	con
count	5.000000e+02	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	
mean	3.263049e+07	14.224206	19.086320	92.606620	662.844800	0.095978	0.103948	
std	1.326933e+08	3.476809	4.164842	23.983476	349.357241	0.013666	0.053096	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.062510	0.019380	
25%	8.667040e+05	11.807500	16.070000	75.995000	430.550000	0.085992	0.063622	
50%	9.014320e+05	13.435000	18.680000	86.735000	556.150000	0.095825	0.091280	
75%	8.910808e+06	16.115000	21.562500	106.225000	800.775000	0.105100	0.130500	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.144700	0.345400	

8 rows × 32 columns

In [54]:

```
#to display cloumn heading
df.columns
```

Out[54]:

```
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')
```

EDA and VISUALIZATION

In [58]:

```
df1=df[['id', 'radius_mean', 'texture_mean', 'perimeter_mean','area_mean', 'smoothness_mean', 'compactness_mean']]
df1
```

Out[58]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...	...
495	914333	14.87	20.21	96.12	680.9	0.09587	0.08345
496	914366	12.65	18.17	82.69	485.6	0.10760	0.13340
497	914580	12.47	17.31	80.45	480.1	0.08928	0.07630
498	914769	18.49	17.52	121.30	1068.0	0.10120	0.13170
499	91485	20.59	21.24	137.80	1320.0	0.10850	0.16440

500 rows × 7 columns

In [59]:

```
df1.fillna(1)
```

Out[59]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...	...
495	914333	14.87	20.21	96.12	680.9	0.09587	0.08345
496	914366	12.65	18.17	82.69	485.6	0.10760	0.13340
497	914580	12.47	17.31	80.45	480.1	0.08928	0.07630
498	914769	18.49	17.52	121.30	1068.0	0.10120	0.13170
499	91485	20.59	21.24	137.80	1320.0	0.10850	0.16440

500 rows × 7 columns

In [60]:

```
df1.info()
```

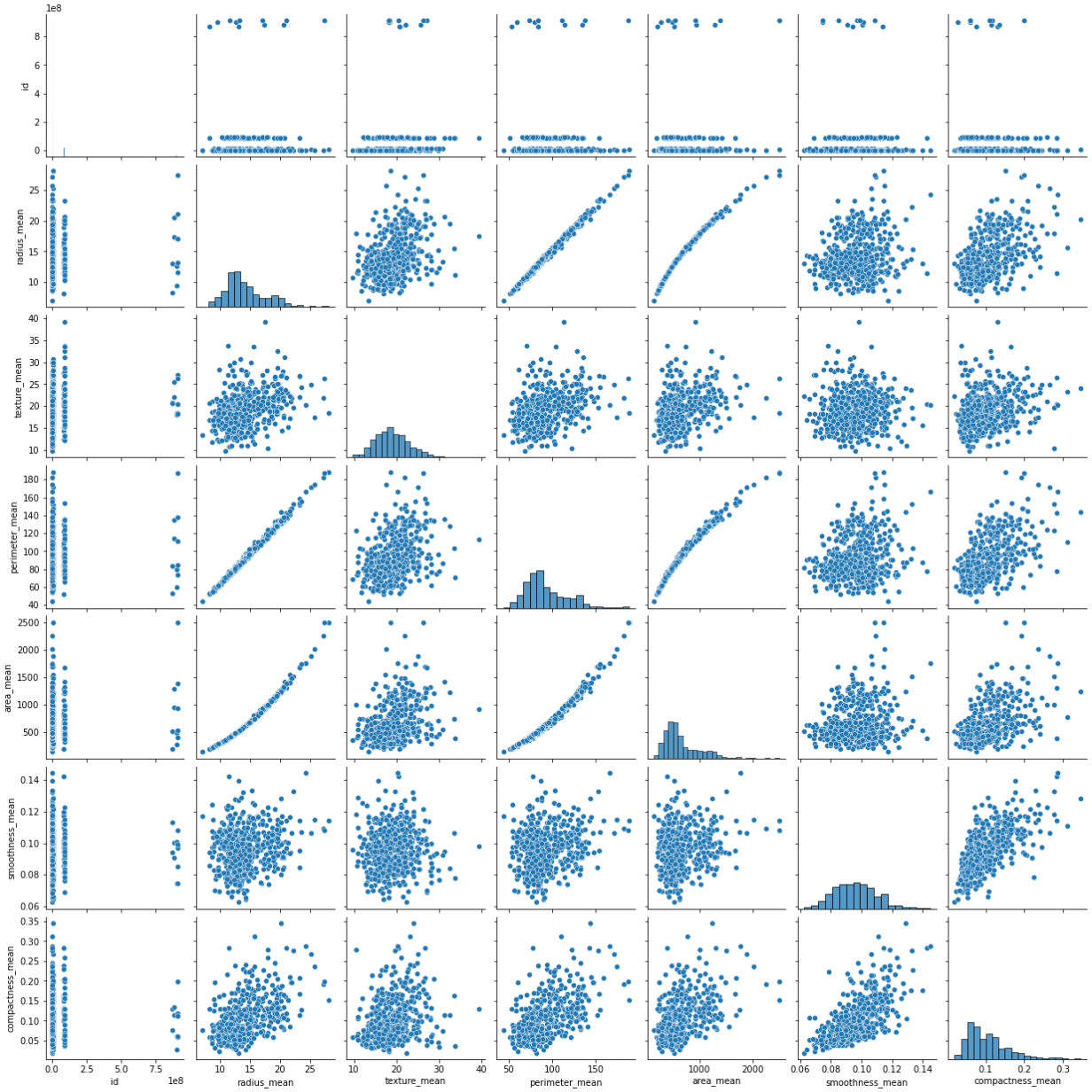
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   id                500 non-null    int64  
 1   radius_mean       500 non-null    float64
 2   texture_mean      500 non-null    float64
 3   perimeter_mean    500 non-null    float64
 4   area_mean         500 non-null    float64
 5   smoothness_mean   500 non-null    float64
 6   compactness_mean  500 non-null    float64
dtypes: float64(6), int64(1)
memory usage: 27.5 KB
```

In [61]:

```
sns.pairplot(df1)
```

Out[61]:

```
<seaborn.axisgrid.PairGrid at 0x165bbeeb5b0>
```



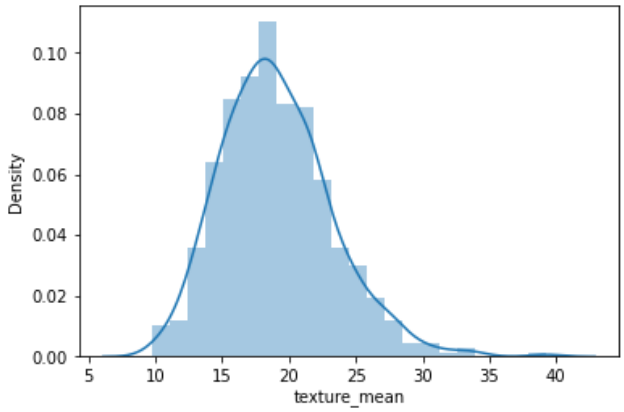
In [63]:

```
sns.distplot(df['texture_mean'])
```

```
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 adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[63]:

```
<AxesSubplot:xlabel='texture_mean', ylabel='Density'>
```

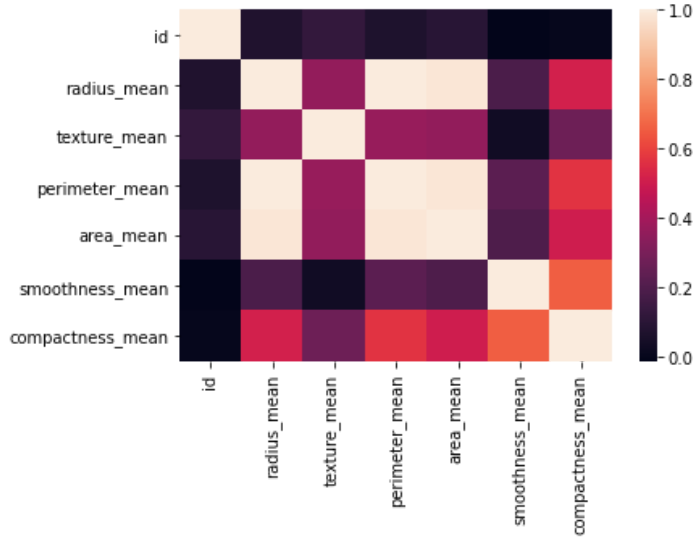


In [64]:

```
data=df1[['id', 'radius_mean', 'texture_mean', 'perimeter_mean','area_mean', 'smoothness_mean', 'compactness_mean
sns.heatmap(data.corr())
```

Out[64]:

```
<AxesSubplot:>
```



to Train the model-Model buliding

we are going to split our data into two variable where x is a independent and y is dependent on x

In [66]:

```
x=data[['id', 'radius_mean', 'texture_mean', 'perimeter_mean','area_mean', 'smoothness_mean', 'compactness_mean']]
y=data['texture_mean']
```

In [67]:

```
# to split my dataset into test and train 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 [68]:

```
from sklearn.linear_model import LinearRegression

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

Out[68]:

```
LinearRegression()
```

In [69]:

```
print(lr.intercept_)
```

```
-2.6290081223123707e-13
```

In [70]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[70]:

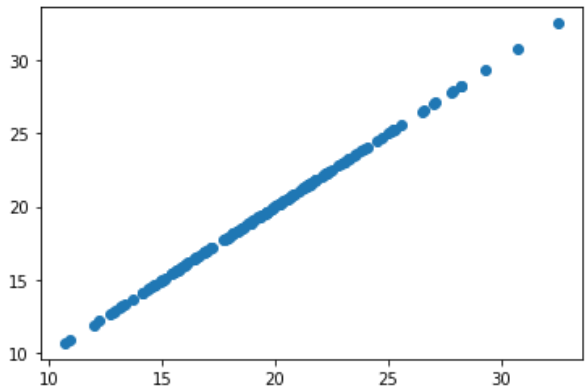
	Co-effecient
id	6.634004e-21
radius_mean	2.124866e-15
texture_mean	1.000000e+00
perimeter_mean	1.192560e-16
area_mean	-1.941193e-17
smoothness_mean	-4.796421e-16
compactness_mean	1.085649e-15

In [71]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[71]:

```
<matplotlib.collections.PathCollection at 0x165be437d90>
```



In [72]:

```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [ ]:

```
```