A real estate agent want help to predict the house price for regions in Usa.he gave us the dataset to work on to use linear Regression model.Create a model that helps him to estimate

Data Collection

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

In [160]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\8_BreastCancerPrediction.csv")[
```

In [161]:
```
#to display top 10 rows
data.head()
```

Out[161]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.10

5 rows × 33 columns

In [162]:
```
#to display null values
data.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 [163]:
```
data.shape
```

Out[163]: (500, 33)

In [164]:
```
#to display summary of statistics
data.describe()
```

Out[164]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.000000e+02	500.000000	500.000000	500.000000	500.000000	500.0000(
mean	3.263049e+07	14.224206	19.086320	92.606620	662.844800	0.09597
std	1.326933e+08	3.476809	4.164842	23.983476	349.357241	0.01366
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.0625'
25%	8.667040e+05	11.807500	16.070000	75.995000	430.550000	0.0859(
50%	9.014320e+05	13.435000	18.680000	86.735000	556.150000	0.09582
75%	8.910808e+06	16.115000	21.562500	106.225000	800.775000	0.1051(
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.1447(

8 rows × 32 columns

In [165]:
```
#to display columns name
data.columns
```

Out[165]:
```
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 [166]:
```
data.fillna(value=5)
```

Out[166]:

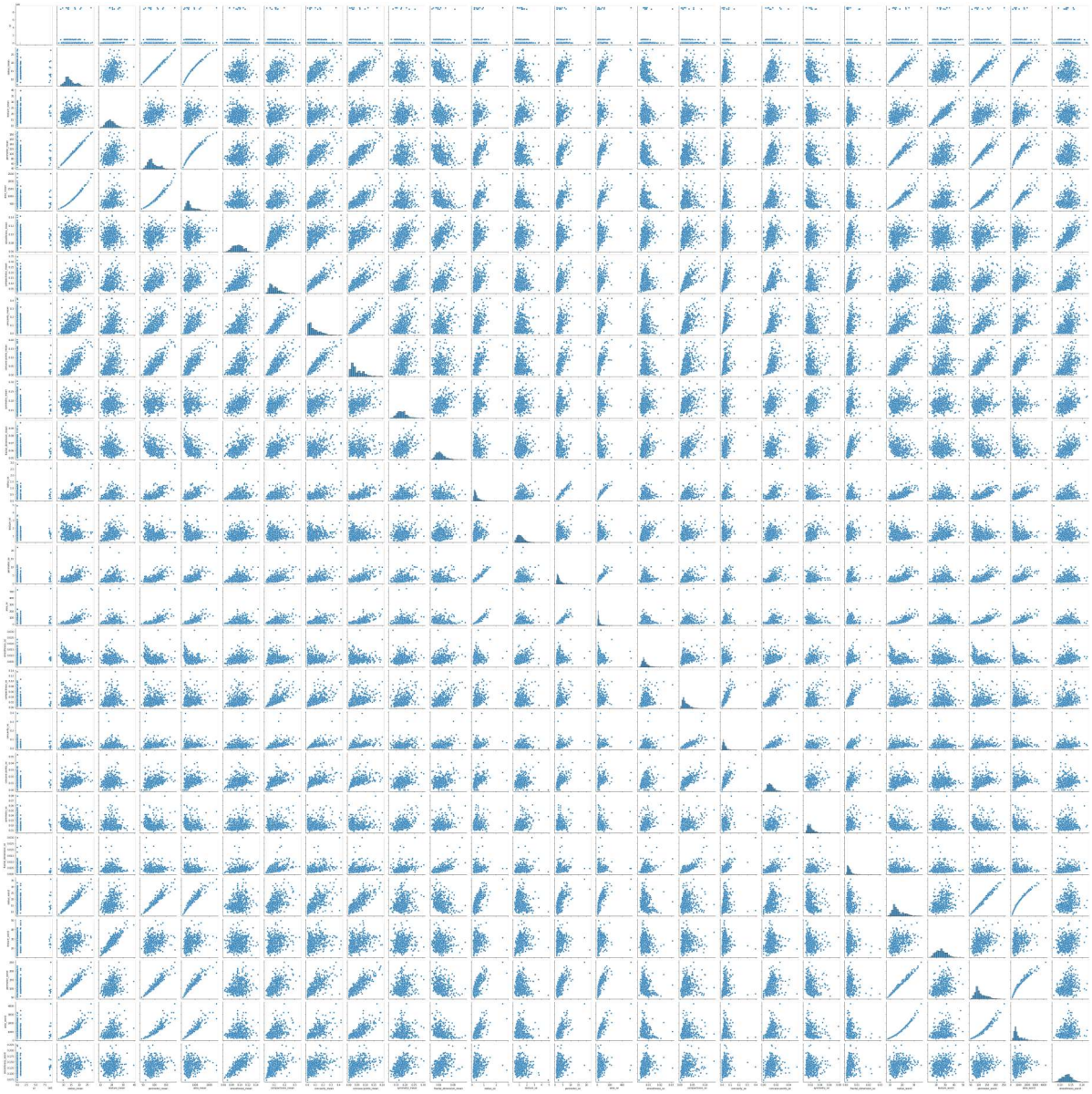
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.
2	84300903	M	19.69	21.25	130.00	1203.0	0.
3	84348301	M	11.42	20.38	77.58	386.1	0.
4	84358402	M	20.29	14.34	135.10	1297.0	0.
...	...	...	...	...	...	...	
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 [167]:
```
data1=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']]
```

In [168]:
```
sns.pairplot(data1)
```

Out[168]:
```
<seaborn.axisgrid.PairGrid at 0x1250c0afa00>
```

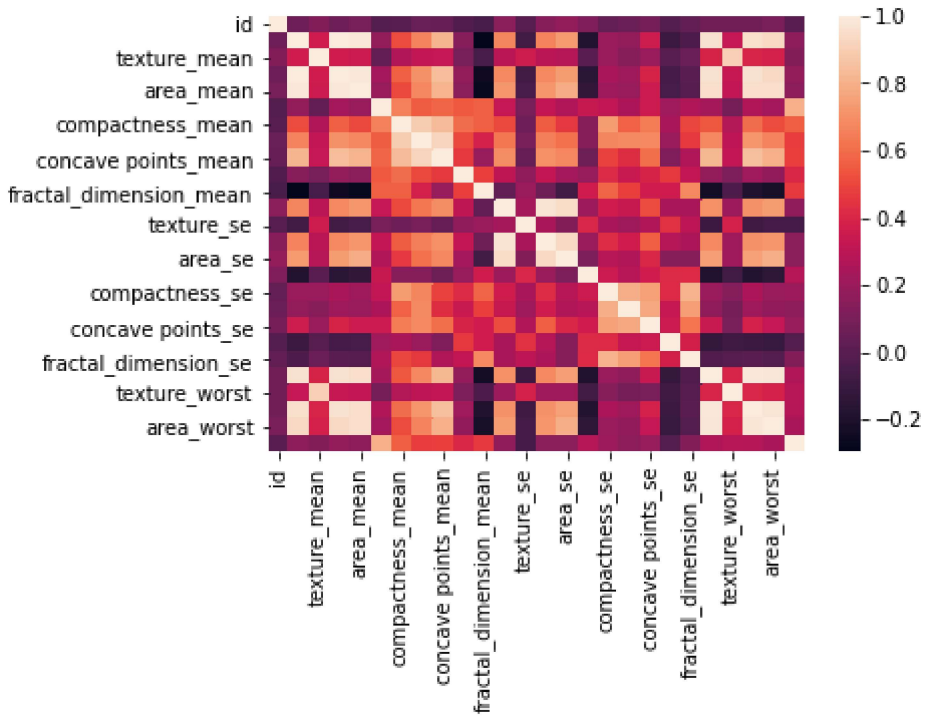


EDA and Visualization

In [169]:
```
#sns.distplot(data['Co2-Emissions'])
```

In [170]:
```
sns.heatmap(data1.corr())
```

Out[170]: <AxesSubplot:>



In [174]:
```
data1.fillna(value=5)
```

Out[174]:

	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 × 27 columns

To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore

In [183]:
```
x=data[['compactness_worst', 'concavity_worst',
       ]]

y=data1['id']
```

In [184]:
```
#To split 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=0.1)
```

In [185]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[185]:
```
LinearRegression()
```

In [186]:
```
lr.intercept_
```

Out[186]:
```
38835336.15720841
```

In [187]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

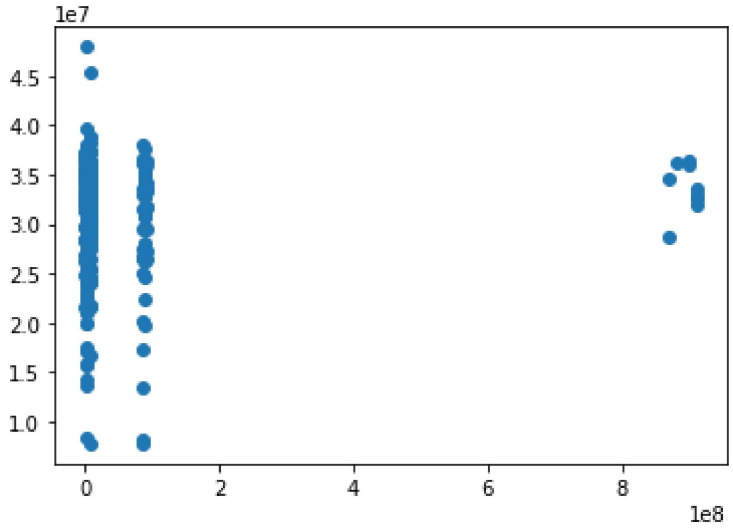
Out[187]:

	Co-efficient
compactness_worst	-5.751864e+07
concavity_worst	2.734553e+07

In [ ]:

In [188]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[188]:
```
<matplotlib.collections.PathCollection at 0x1252332bdf0>
```



In [189]:
```
lr.score(x_test,y_test)
```

Out[189]:
```
-0.004073209588378202
```

In [190]:
```
lr.score(x_train,y_train)
```

Out[190]:
```
0.0015226742955708472
```

In [191]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [192]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[192]:
```
-0.009582880189396903
```

In [193]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[193]:
```
-0.0040737802210928376
```

In [194]:
```
from sklearn.linear_model import ElasticNet
en= ElasticNet()
en.fit(x_train,y_train)
```

Out[194]:
```
ElasticNet()
```

In [195]:
```
print(en.coef_)
```

```
[-1241747.15166715  -916017.96199218]
```

In [196]:
```
print(en.intercept_)
```

```
32089776.5923149
```

In [197]:
```
prediction = en.predict(x_test)
prediction
```

Out[197]:
```
array([31393740.13355368, 31088477.23434921, 31253514.61656909,
       31542670.42800784, 31365755.09598579, 31882295.855237  ,
       31913180.90717359, 31534008.63013534, 31710076.15057814,
       31780477.09711697, 31991828.95134632, 31496373.89934802,
       31741856.92519902, 31944222.95010149, 31769883.86167683,
       31373511.88890291, 31168714.43934447, 31044309.26365758,
       31874848.90308605, 31368514.77333577, 31781323.68778142,
       31934699.37516704, 31284579.70265878, 31228892.23275906,
       31627703.65606768, 31960955.92049683, 31496299.88949108,
       31547727.64488157, 31849175.58418468, 30784912.09903306,
       31439275.55225564, 31609501.3545823 , 31918403.4321233 ,
       31493836.73418531, 31848553.36347367, 31356747.691898  ,
       31664347.58357762, 31226737.5240613 , 30935373.9311361 ,
       31996899.68451488, 31771755.96513853, 32002412.79677851,
       31819043.74666099, 31906922.55968469, 31666012.16821847,
       31571940.18819558, 31242794.29848333, 31353086.52385158,
       31404467.02644319, 31614317.40728684])
```

In [198]:
```
print(en.score(x_test,y_train))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-198-48d1f0543252> in <module>
----> 1 print(en.score(x_test,y_train))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in score(self, X,
y, sample_weight)
    552         from .metrics import r2_score
    553         y_pred = self.predict(X)
--> 554         return r2_score(y, y_pred, sample_weight=sample_weight)
    555
    556     def _more_tags(self):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inn
er_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in
r2_score(y_true, y_pred, sample_weight, multioutput)
    674     -3.0
    675     """
--> 676     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    677         y_true, y_pred, multioutput)
    678     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in
_check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in che
ck_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent num
bers of"
    263                          " samples: %r" % [int(l) for l in lengths])
    264

ValueError: Found input variables with inconsistent numbers of samples: [450,
50]
```

In [199]:
```
from sklearn import metrics
```

In [200]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error: 62586502.49745239
```

In [201]:
```
print("Mean Absolute Square error:",metrics.mean_squared_error(y_test,predicti
```

```
Mean Absolute Square error: 3.080001673006421e+16
```

In [202]:
```
print("Root mean Square error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root mean Square error: 175499335.41203
```

In [ ]:

In [ ]: