

Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

```
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("bottle")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (47,
73) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

To display top 10 rows

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

Out[3]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1	DIC
0	1	1	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0000A-3	0	10.50	33.440	NaN	25.649	NaN	...	NaN	0	NaN	NaN	Na
1	1	2	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0008A-3	8	10.46	33.440	NaN	25.656	NaN	...	NaN	8	NaN	NaN	Na
2	1	3	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0010A-7	10	10.46	33.437	NaN	25.654	NaN	...	NaN	10	NaN	NaN	Na
3	1	4	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0019A-3	19	10.45	33.420	NaN	25.643	NaN	...	NaN	19	NaN	NaN	Na
4	1	5	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0020A-7	20	10.45	33.421	NaN	25.643	NaN	...	NaN	20	NaN	NaN	Na
5	1	6	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0030A-7	30	10.45	33.431	NaN	25.651	NaN	...	NaN	30	NaN	NaN	Na

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1	DIC
6	1	7	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0039A-3	39	10.45	33.440	NaN	25.658	NaN	...	NaN	39	NaN	NaN	Na
7	1	8	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0050A-7	50	10.24	33.424	NaN	25.682	NaN	...	NaN	50	NaN	NaN	Na
8	1	9	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0058A-3	58	10.06	33.420	NaN	25.710	NaN	...	NaN	58	NaN	NaN	Na
9	1	10	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0075A-7	75	9.86	33.494	NaN	25.801	NaN	...	NaN	75	NaN	NaN	Na

10 rows × 74 columns

Data Cleaning And Pre-Processing

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

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 864863 entries, 0 to 864862

Data columns (total 74 columns):

#	Column	Non-Null Count	Dtype
0	Cst_Cnt	864863 non-null	int64
1	Btl_Cnt	864863 non-null	int64
2	Sta_ID	864863 non-null	object
3	Depth_ID	864863 non-null	object
4	Depthm	864863 non-null	int64
5	T_degC	853900 non-null	float64
6	Salnty	817509 non-null	float64
7	O2ml_L	696201 non-null	float64
8	STheta	812174 non-null	float64
9	O2Sat	661274 non-null	float64
10	Oxy_μmol/Kg	661268 non-null	float64
11	BtlNum	118667 non-null	float64
12	RecInd	864863 non-null	int64
13	T_prec	853900 non-null	float64
14	T_qual	23127 non-null	float64
15	S_prec	817509 non-null	float64
16	S_qual	74914 non-null	float64
17	P_qual	673755 non-null	float64
18	O_qual	184676 non-null	float64
19	SThta	65823 non-null	float64
20	O2Satq	217797 non-null	float64
21	ChlorA	225272 non-null	float64
22	Chlqua	639166 non-null	float64
23	Phaeop	225271 non-null	float64
24	Phaqua	639170 non-null	float64
25	PO4uM	413317 non-null	float64
26	PO4q	451786 non-null	float64
27	SiO3uM	354091 non-null	float64
28	SiO3qu	510866 non-null	float64
29	NO2uM	337576 non-null	float64
30	NO2q	529474 non-null	float64
31	NO3uM	337403 non-null	float64
32	NO3q	529933 non-null	float64
33	NH3uM	64962 non-null	float64
34	NH3q	808299 non-null	float64
35	C14As1	14432 non-null	float64
36	C14A1p	12760 non-null	float64
37	C14A1q	848605 non-null	float64

38	C14As2	14414 non-null	float64
39	C14A2p	12742 non-null	float64
40	C14A2q	848623 non-null	float64
41	DarkAs	22649 non-null	float64
42	DarkAp	20457 non-null	float64
43	DarkAq	840440 non-null	float64
44	MeanAs	22650 non-null	float64
45	MeanAp	20457 non-null	float64
46	MeanAq	840439 non-null	float64
47	IncTim	14437 non-null	object
48	LightP	18651 non-null	float64
49	R_Depth	864863 non-null	float64
50	R_TEMP	853900 non-null	float64
51	R_POTEMP	818816 non-null	float64
52	R_SALINITY	817509 non-null	float64
53	R_SIGMA	812007 non-null	float64
54	R_SVA	812092 non-null	float64
55	R_DYNHT	818206 non-null	float64
56	R_O2	696201 non-null	float64
57	R_O2Sat	666448 non-null	float64
58	R_SI03	354099 non-null	float64
59	R_PO4	413325 non-null	float64
60	R_NO3	337411 non-null	float64
61	R_NO2	337584 non-null	float64
62	R_NH4	64982 non-null	float64
63	R_CHLA	225276 non-null	float64
64	R_PHAEO	225275 non-null	float64
65	R_PRES	864863 non-null	int64
66	R_SAMP	122006 non-null	float64
67	DIC1	1999 non-null	float64
68	DIC2	224 non-null	float64
69	TA1	2084 non-null	float64
70	TA2	234 non-null	float64
71	pH2	10 non-null	float64
72	pH1	84 non-null	float64
73	DIC Quality Comment	55 non-null	object

dtypes: float64(65), int64(5), object(4)

memory usage: 488.3+ MB

```
In [5]: # Display the statistical summary
df.describe()
```

Out[5]:

	Cst_Cnt	Btl_Cnt	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	Oxy.
count	864863.000000	864863.000000	864863.000000	853900.000000	817509.000000	696201.000000	812174.000000	661274.000000	66126
mean	17138.790958	432432.000000	226.831951	10.799677	33.840350	3.392468	25.819394	57.103779	14
std	10240.949817	249664.587267	316.050259	4.243825	0.461843	2.073256	1.167787	37.094137	9
min	1.000000	1.000000	0.000000	1.440000	28.431000	-0.010000	20.934000	-0.100000	.
25%	8269.000000	216216.500000	46.000000	7.680000	33.488000	1.360000	24.965000	21.100000	6
50%	16848.000000	432432.000000	125.000000	10.060000	33.863000	3.440000	25.996000	54.400000	15
75%	26557.000000	648647.500000	300.000000	13.880000	34.196900	5.500000	26.646000	97.600000	24
max	34404.000000	864863.000000	5351.000000	31.140000	37.034000	11.130000	250.784000	214.100000	48

8 rows × 10 columns



```
In [6]: # To display the col headings
df.columns
```

Out[6]: Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC',
'Salnty', 'O2ml_L', 'STheta', 'O2Sat', 'Oxy_μmol/Kg', 'BtlNum',
'RecInd', 'T_prec', 'T_qual', 'S_prec', 'S_qual', 'P_qual', 'O_qual',
'SThtaq', 'O2Satq', 'ChlorA', 'Chlqua', 'Phaeop', 'Phaqua', 'PO4uM',
'PO4q', 'SiO3uM', 'SiO3qu', 'NO2uM', 'NO2q', 'NO3uM', 'NO3q', 'NH3uM',
'NH3q', 'C14As1', 'C14A1p', 'C14A1q', 'C14As2', 'C14A2p', 'C14A2q',
'DarkAs', 'DarkAp', 'DarkAq', 'MeanAs', 'MeanAp', 'MeanAq', 'IncTim',
'LightP', 'R_Depth', 'R_TEMP', 'R_POTEMP', 'R_SALINITY', 'R_SIGMA',
'R_SVA', 'R_DYNHT', 'R_O2', 'R_O2Sat', 'R_SIO3', 'R_PO4', 'R_NO3',
'R_NO2', 'R_NH4', 'R_CHLA', 'R_PHAEO', 'R_PRES', 'R_SAMP', 'DIC1',
'DIC2', 'TA1', 'TA2', 'pH2', 'pH1', 'DIC Quality Comment'],
dtype='object')

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



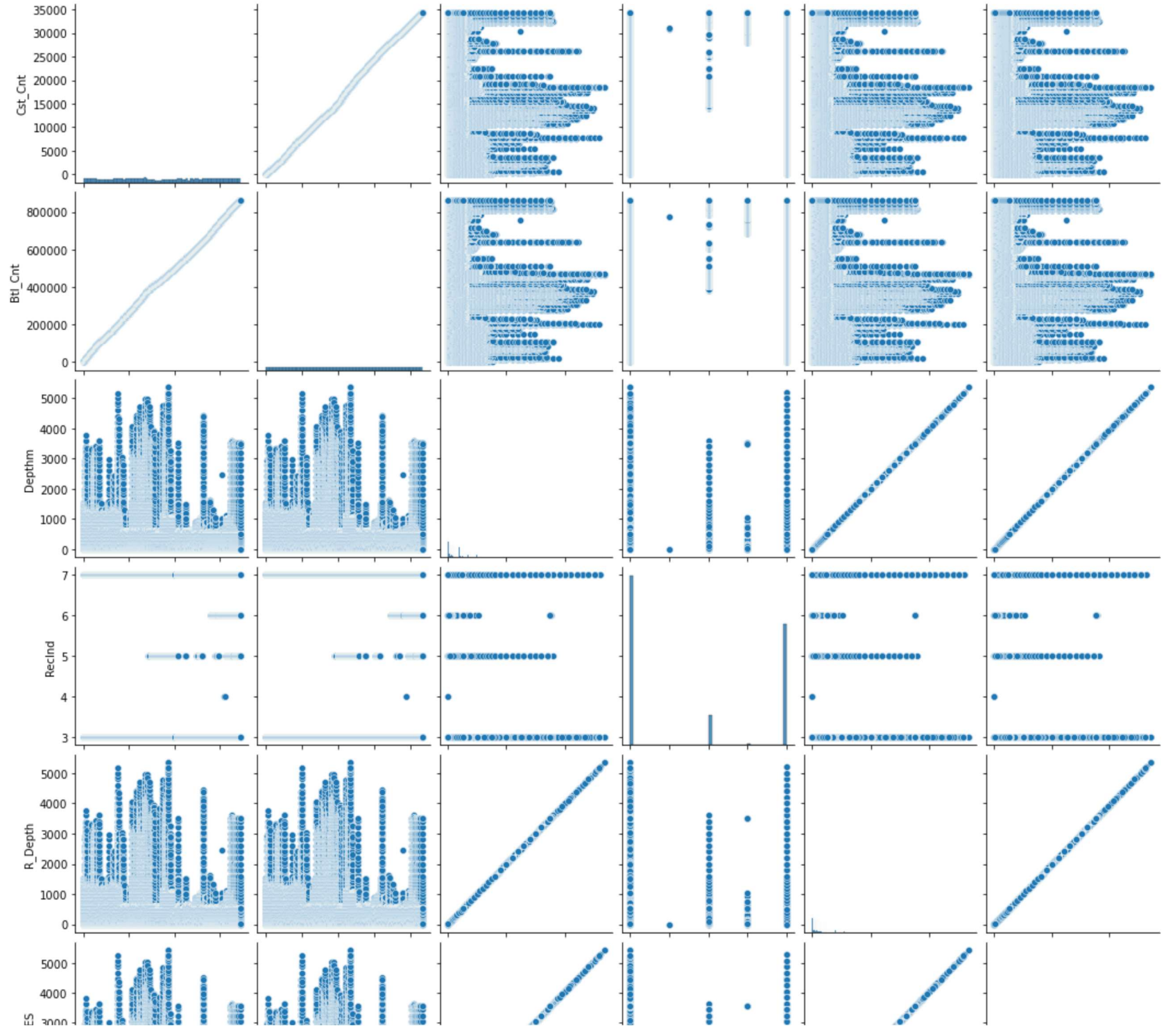
```
In [8]: cols.columns
```

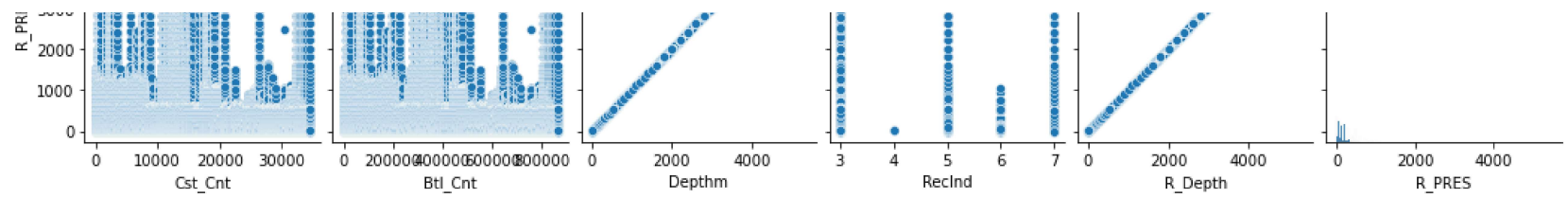
```
Out[8]: Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'RecInd',  
              'R_Depth', 'R_PRES'],  
              dtype='object')
```

EDA and Visualization

```
In [9]: sns.pairplot(cols)
```

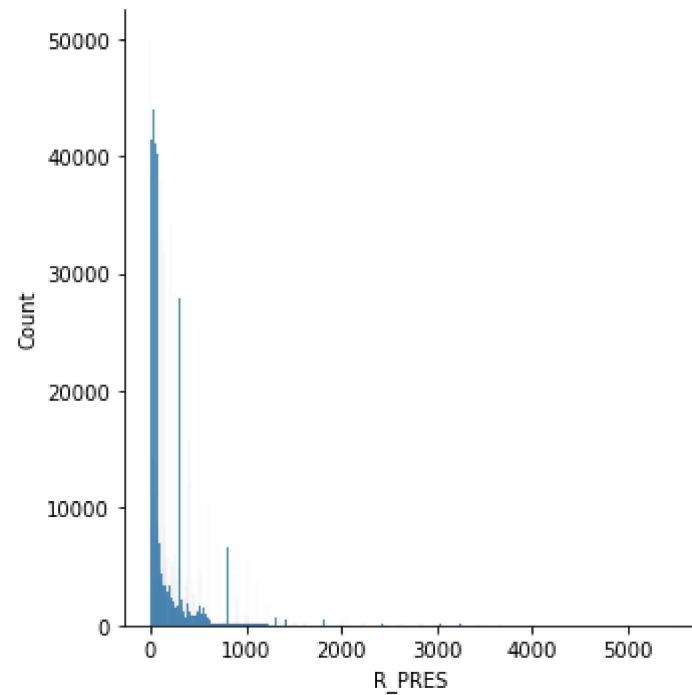
```
Out[9]: <seaborn.axisgrid.PairGrid at 0x220a00f3a90>
```



```
In [10]: sns.displot(df['R_PRES'])
```

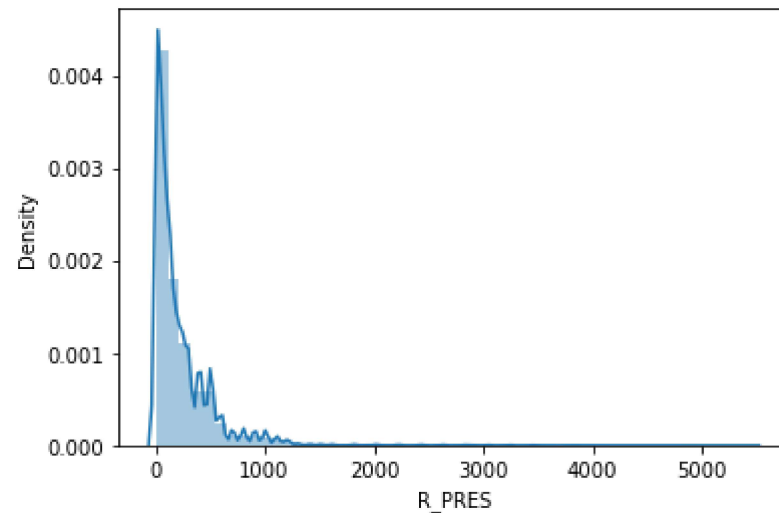
```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x220c99d8dc0>
```



```
In [11]: # We use displot in older version we get distplot use displot
sns.distplot(df['R_PRES'])
```

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 either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

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



```
In [23]: df1=cols[['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depthm', 'RecInd', 'R_PRES']]
df1
```

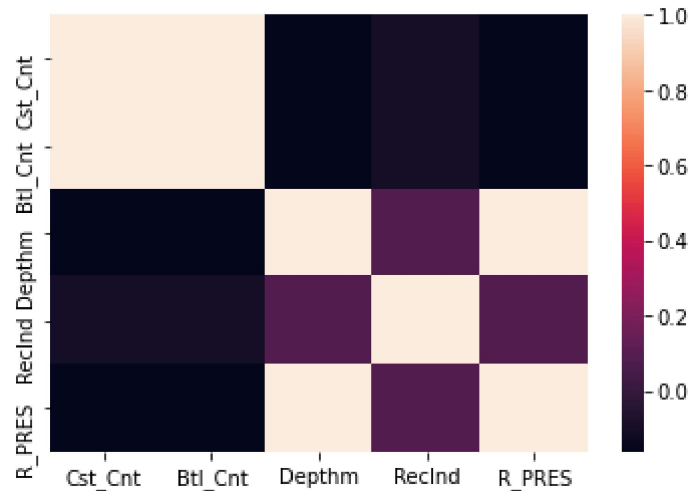
Out[23]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depthm	RecInd	R_PRES
0	1	1	054.0 056.0	0	3	0
1	1	2	054.0 056.0	8	3	8
2	1	3	054.0 056.0	10	7	10
3	1	4	054.0 056.0	19	3	19
4	1	5	054.0 056.0	20	7	20
...
864858	34404	864859	093.4 026.4	0	7	0
864859	34404	864860	093.4 026.4	2	3	2
864860	34404	864861	093.4 026.4	5	3	5
864861	34404	864862	093.4 026.4	10	3	10
864862	34404	864863	093.4 026.4	15	3	15

864863 rows × 6 columns

```
In [24]: sns.heatmap(df1.corr())
```

```
Out[24]: <AxesSubplot:>
```



To train the model - MODEL BUILD

Going to train linear regression model; We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

```
In [32]: x=df1[['Cst_Cnt', 'RecInd', 'Depthm',  
              'R_PRES']]  
y=df1[['Depthm']]
```

To split the dataset into test data

```
In [33]: # importing lib for splitting test data  
from sklearn.model_selection import train_test_split
```

```
In [34]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```



```
In [35]: from sklearn.linear_model import LinearRegression
```

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

```
Out[35]: LinearRegression()
```

```
In [38]: print(lr.intercept_)
```

```
[-5.68434189e-13]
```

```
In [39]: print(lr.score(x_test,y_test))
```

```
1.0
```

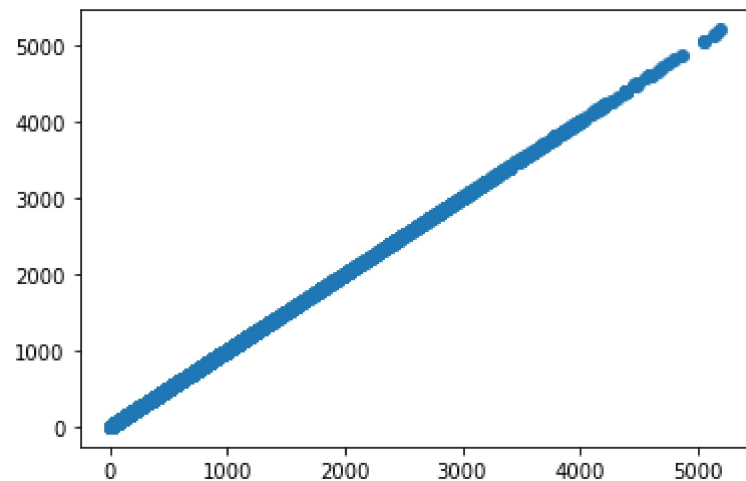
```
In [48]: coeff=pd.DataFrame(lr.coef_)  
coeff
```

```
Out[48]:
```

	0	1	2	3
0	1.181664e-17	-8.144321e-14	1.0	-7.741626e-16

```
In [49]: pred = lr.predict(x_test)  
plt.scatter(y_test,pred)
```

```
Out[49]: <matplotlib.collections.PathCollection at 0x220d45a2d90>
```



In []: