

# DATA COLLECTION

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

```
In [2]: # To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\bottle.csv")
sd
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...
<b>0</b>	1	1	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0000A-3	0	10.50	33.440	NaN	25.649	NaN	...
<b>1</b>	1	2	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0008A-3	8	10.46	33.440	NaN	25.656	NaN	...
<b>2</b>	1	3	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0010A-7	10	10.46	33.437	NaN	25.654	NaN	...
<b>3</b>	1	4	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0019A-3	19	10.45	33.420	NaN	25.643	NaN	...
<b>4</b>	1	5	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0020A-7	20	10.45	33.421	NaN	25.643	NaN	...
...	...	...	...	...	...	...	...	...	...	...	...
<b>5240</b>	173	5241	069.0 144.0	19- 4905CR- HY-125- 1712- 06901440- 0279A-3	279	7.86	33.870	3.81	26.411	57.0	...
<b>5241</b>	173	5242	069.0 144.0	19- 4905CR- HY-125- 1712- 06901440- 0300A-7	300	7.52	33.896	3.55	26.481	52.7	...
<b>5242</b>	173	5243	069.0 144.0	19- 4905CR- HY-125- 1712- 06901440- 0371A-3	371	6.54	33.930	2.67	26.642	38.8	...
<b>5243</b>	173	5244	069.0 144.0	19- 4905CR- HY-125- 1712- 06901440- 0400A-7	400	6.25	33.951	2.32	26.697	33.5	...

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...
				19-4905CR-HY-125-1712-06901440-0500A-7							
5244	173	5245	069.0 144.0		500	5.44	34.027	1.30	26.858	18.4	...

5245 rows × 71 columns

In [3]:

# to display top 5 rows  
sd.head()

Out[3]:

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

5 rows × 71 columns



# DATA CLEANING AND PRE\_PROCESSING

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

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5245 entries, 0 to 5244
Data columns (total 71 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Cst_Cnt                5245 non-null   int64
1   Btl_Cnt                5245 non-null   int64
2   Sta_ID                 5245 non-null   object
3   Depth_ID               5245 non-null   object
4   Depthm                 5245 non-null   int64
5   T_degC                 5225 non-null   float64
6   Salnty                 5092 non-null   float64
7   O2ml_L                 3051 non-null   float64
8   STheta                 5077 non-null   float64
9   O2Sat                  2954 non-null   float64
10  Oxy_μmol/Kg            2954 non-null   float64
11  BtlNum                  0 non-null      float64
12  RecInd                  5245 non-null   int64
13  T_prec                 5225 non-null   float64
14  T_qual                  52 non-null     float64
15  S_prec                 5092 non-null   float64
16  S_qual                  239 non-null    float64
17  P_qual                  5245 non-null   int64
18  O_qual                  2198 non-null   float64
19  SThtaq                  283 non-null    float64
20  O2Satq                  2362 non-null   float64
21  ChlorA                  0 non-null      float64
22  Chlqua                  5245 non-null   int64
23  Phaeop                  0 non-null      float64
24  Phaqua                  5245 non-null   int64
25  PO4uM                   1236 non-null   float64
26  PO4q                     4009 non-null   float64
27  SiO3uM                   0 non-null      float64
28  SiO3qu                   5245 non-null   int64
29  NO2uM                     0 non-null      float64
30  NO2q                     5245 non-null   int64
31  NO3uM                     0 non-null      float64
32  NO3q                     5245 non-null   int64
33  NH3uM                     0 non-null      float64
34  NH3q                     5245 non-null   int64
35  C14As1                   0 non-null      float64
36  C14A1p                   0 non-null      float64
37  C14A1q                   5245 non-null   int64
38  C14As2                   0 non-null      float64
39  C14A2p                   0 non-null      float64
40  C14A2q                   5245 non-null   int64
41  DarkAs                   0 non-null      float64
42  DarkAp                   0 non-null      float64
43  DarkAq                   5245 non-null   int64
44  MeanAs                   0 non-null      float64
45  MeanAp                   0 non-null      float64
46  MeanAq                   5245 non-null   int64
47  IncTim                   0 non-null      float64
48  LightP                   0 non-null      float64
49  R_Depth                  5245 non-null   int64
50  R_TEMP                   5225 non-null   float64
51  R_POTEMP                 5019 non-null   float64

```

```

52  R_SALINITY    5092 non-null    float64
53  R_SIGMA      4962 non-null    float64
54  R_SVA        4962 non-null    float64
55  R_DYNHT      5031 non-null    float64
56  R_O2         3051 non-null    float64
57  R_O2Sat      2930 non-null    float64
58  R_SIO3       0 non-null       float64
59  R_PO4        1236 non-null    float64
60  R_NO3        0 non-null       float64
61  R_NO2        0 non-null       float64
62  R_NH4        0 non-null       float64
63  R_CHLA       0 non-null       float64
64  R_PHAEO      0 non-null       float64
65  R_PRES       5245 non-null    int64
66  R_SAMP       0 non-null       float64
67  DIC1         0 non-null       float64
68  DIC2         0 non-null       float64
69  TA1          0 non-null       float64
70  TA2          0 non-null       float64
dtypes: float64(52), int64(17), object(2)
memory usage: 2.8+ MB

```

```

In [5]: # to display summary of statistics
sd.describe()

```

```

Out[5]:

```

	Cst_Cnt	Btl_Cnt	Depthm	T_degC	Salnty	O2ml_L	SThet
<b>count</b>	5245.000000	5245.000000	5245.000000	5225.000000	5092.000000	3051.000000	5077.000000
<b>mean</b>	86.191992	2623.000000	345.400572	8.993144	33.827093	3.129839	26.11996
<b>std</b>	49.742962	1514.245412	356.298143	3.902631	0.492518	2.259226	0.92301
<b>min</b>	1.000000	1.000000	0.000000	2.700000	32.520000	0.210000	23.56800
<b>25%</b>	43.000000	1312.000000	54.000000	5.430000	33.490000	0.670000	25.11700
<b>50%</b>	86.000000	2623.000000	200.000000	8.620000	33.890000	2.990000	26.27900
<b>75%</b>	129.000000	3934.000000	600.000000	12.230000	34.255000	5.490000	26.99800
<b>max</b>	173.000000	5245.000000	1547.000000	19.760000	34.700000	6.630000	27.62300

8 rows × 69 columns



```
In [6]: #to display colums heading
sd.columns
```

```
Out[6]: Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC',
              'Salnty', 'O2m1_L', 'STheta', 'O2Sat', 'Oxy_μmol/Kg', 'BtlNum',
              'RecInd', 'T_prec', 'T_qual', 'S_prec', 'S_qual', 'P_qual', 'O_qual',
              'SThetaq', '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'],
              dtype='object')
```

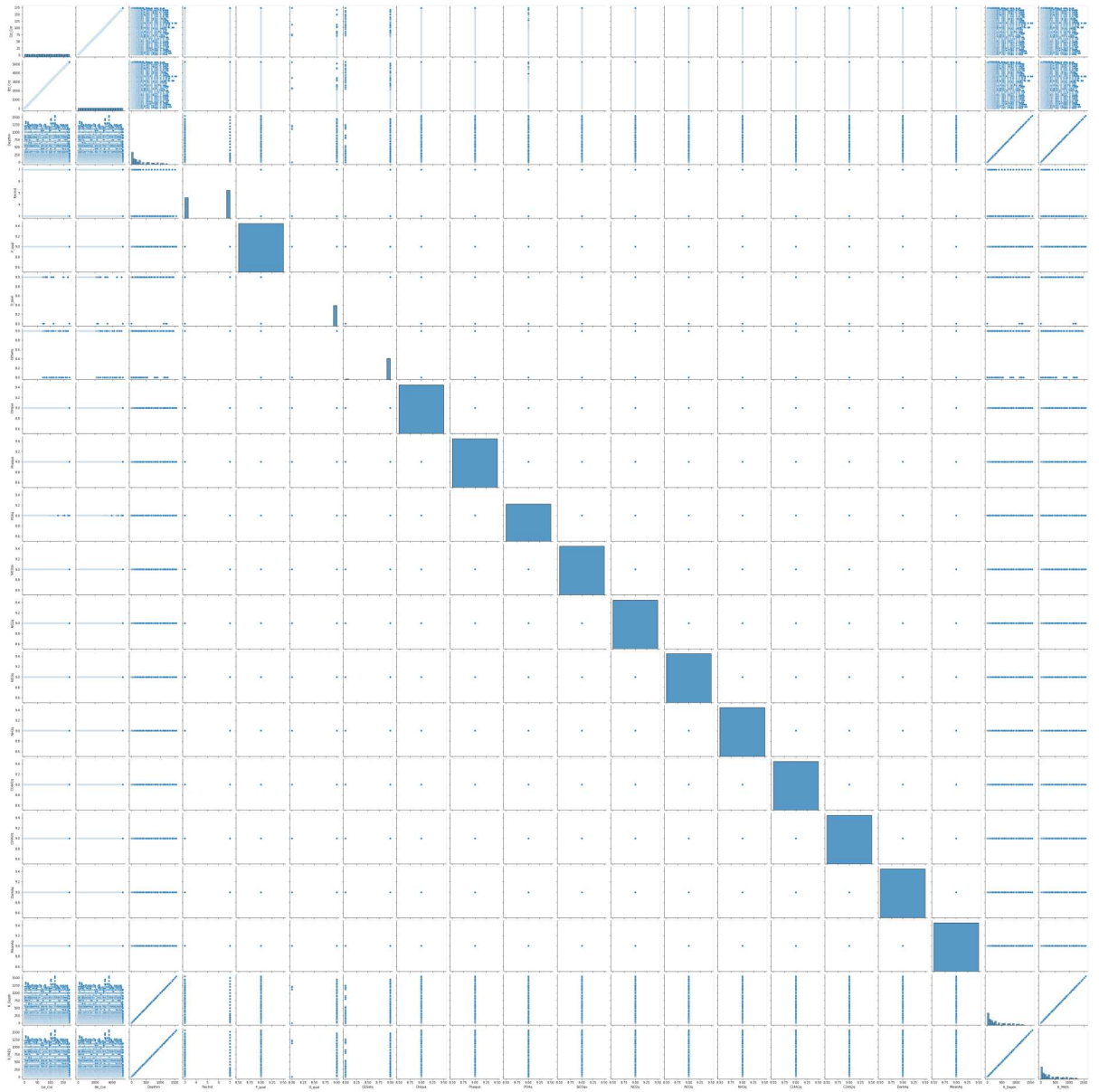
## EDA and visualization

```
In [96]: sd1=sd[['Cst_Cnt', 'Btl_Cnt', 'Depthm', 'RecInd',
                 'P_qual', 'O_qual', 'O2Satq', 'Chlqua', 'Phaqua', 'PO4q', 'SiO3qu',
                 'NO2q', 'NO3q', 'NH3q', 'C14A1q', 'C14A2q', 'DarkAq', 'MeanAq',
                 'R_Depth', 'R_PRES']]
```



```
In [97]: sns.pairplot(sd1)
```

```
Out[97]: <seaborn.axisgrid.PairGrid at 0x2ed22dca9a0>
```

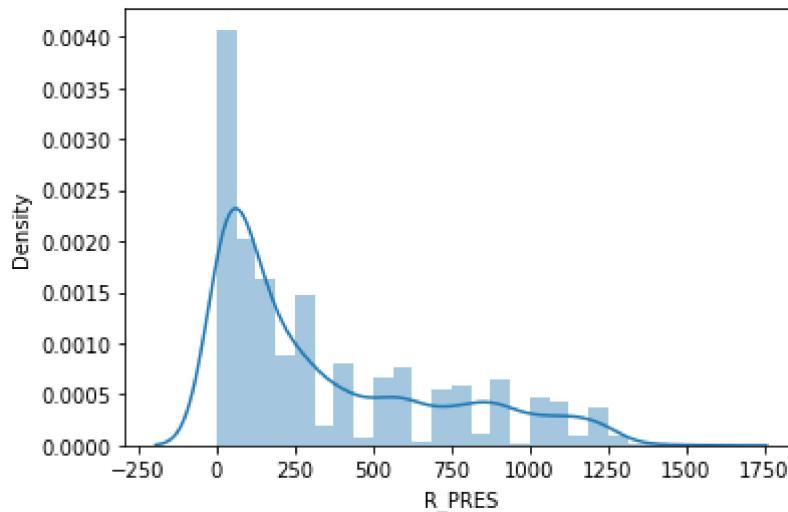


```
In [98]: sns.distplot(sd['R_PRES'])
```

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).

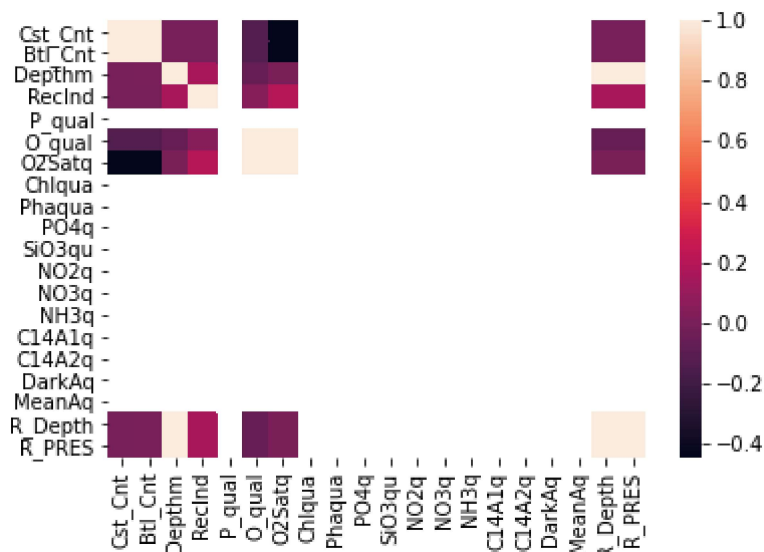
```
warnings.warn(msg, FutureWarning)
```

```
Out[98]: <AxesSubplot:xlabel='R_PRES', ylabel='Density'>
```



```
In [99]: sns.heatmap(sd1.corr())
```

```
Out[99]: <AxesSubplot:>
```



## TO TRAIN THE MODEL \_MODEL BUILDING

we are going to train a Linear Regression model; we need to split out the data into two variables x and y where x is independent (input) and y is dependent (output) address column as it is not required for our model

```
In [100]: x= sd1[['Cst_Cnt', 'Btl_Cnt', 'RecInd',  
                'P_qual', 'O_qual', 'O2Satq', 'Chlqua', 'Phaqua', 'P04q', 'SiO3qu',  
                'NO2q', 'NO3q', 'NH3q', 'C14A1q', 'C14A2q', 'DarkAq', 'MeanAq',  
                'R_Depth', 'R_PRES']]  
y=sd1['Depthm']
```

```
In [101]: # To split my dataset into training data 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=0.5)
```

In [105]: `from sklearn.linear_model import LinearRegression`

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

```

-----
ValueError                                Traceback (most recent call last)
<ipython-input-105-b0fd2c20cba9> in <module>
      2
      3 lr=LinearRegression()
----> 4 lr.fit(x_train,y_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in fit(self, X, y, sample_weight)
    516         accept_sparse = False if self.positive else ['csr', 'csc', 'coo']
    517
--> 518         X, y = self._validate_data(X, y, accept_sparse=accept_sparse,
    519                                     y_numeric=True, multi_output=True)
    520

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_params)
    431         y = check_array(y, **check_y_params)
    432     else:
--> 433         X, y = check_X_y(X, y, **check_params)
    434         out = X, y
    435

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_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\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    812         raise ValueError("y cannot be None")
    813
--> 814         X = check_array(X, accept_sparse=accept_sparse,
    815                         accept_large_sparse=accept_large_sparse,
    816                         dtype=dtype, order=order, copy=copy,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_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\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    661
    662         if force_all_finite:
--> 663             _assert_all_finite(array,

```

```

664                                     allow_nan=force_all_finite == 'allow-n
an')
665

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in _as
sert_all_finite(X, allow_nan, msg_dtype)
101         not allow_nan and not np.isfinite(X).all()):
102         type_err = 'infinity' if allow_nan else 'NaN, infinity'
--> 103         raise ValueError(
104             msg_err.format
105             (type_err,

```

**ValueError:** Input contains NaN, infinity or a value too large for dtype('float64').

In [103]: `print(lr.intercept_)`

```

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-103-182bb45ab960> in <module>
----> 1 print(lr.intercept_)

AttributeError: 'LinearRegression' object has no attribute 'intercept_'

```

In [104]: `coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])`  
`coeff`

```

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-104-31ae3bd46ea9> in <module>
----> 1 coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
      2 coeff

AttributeError: 'LinearRegression' object has no attribute 'coef_'

```

```
In [23]: prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

```
-----
NotFittedError                                Traceback (most recent call last)
<ipython-input-23-10d398fd7dc3> in <module>
----> 1 prediction = lr.predict(x_test)
      2 plt.scatter(y_test, prediction)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in p
redict(self, X)
    236         Returns predicted values.
    237         """
--> 238         return self._decision_function(X)
    239
    240     _preprocess_data = staticmethod(_preprocess_data)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in _
decision_function(self, X)
    216
    217     def _decision_function(self, X):
--> 218         check_is_fitted(self)
    219
    220         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])

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\utils\validation.py in che
ck_is_fitted(estimator, attributes, msg, all_or_any)
    1039
    1040     if not attrs:
-> 1041         raise NotFittedError(msg % {'name': type(estimator).__name_
_})
    1042
    1043
```

**NotFittedError:** This LinearRegression instance is not fitted yet. Call 'fit' with appropriate arguments before using this estimator.

```
In [24]: print(lr.score(x_test,y_test))
```

```
-----  
NotFittedError                                Traceback (most recent call last)  
<ipython-input-24-6bc23016a4ce> in <module>  
----> 1 print(lr.score(x_test,y_test))  
  
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in score(self, X,  
y, sample_weight)  
    551  
    552         from .metrics import r2_score  
--> 553         y_pred = self.predict(X)  
    554         return r2_score(y, y_pred, sample_weight=sample_weight)  
    555  
  
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in p  
redict(self, X)  
    236         Returns predicted values.  
    237         """  
--> 238         return self._decision_function(X)  
    239  
    240     _preprocess_data = staticmethod(_preprocess_data)  
  
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in _  
decision_function(self, X)  
    216  
    217     def _decision_function(self, X):  
--> 218         check_is_fitted(self)  
    219  
    220         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])  
  
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\utils\validation.py in che  
ck_is_fitted(estimator, attributes, msg, all_or_any)  
    1039  
    1040     if not attrs:  
-> 1041         raise NotFittedError(msg % {'name': type(estimator).__name_  
_})  
    1042  
    1043  
  
NotFittedError: This LinearRegression instance is not fitted yet. Call 'fit'  
with appropriate arguments before using this estimator.
```

```
In [ ]:
```



