In [1]:

```
# importing the necessary libraries
import pandas as pd  # for the model training which only requires the dataframe
# for the plotting of regression plots,residual plots and distribution plot
import matplotlib.pyplot as plt
import seaborn as sns  # for the plotting of regression plots
from sklearn.linear_model import LinearRegression  # for the linear model
```

In [2]:

```
# importing the dataset
dataset = open("csv/Concrete_Data.csv")
```

In [3]:

```
# counting the number of features (column) in the dataset
col_names = dataset.readline()  # reading the row that contain column name
num_of_col = col_names.count(",")+1
```

In [4]:

```
# the raw data from which we will extract the required data
data_raw = dataset.readlines()

# now we will get data for each column
required_data = dict()  # first we create dictionary to seperate values of each feature
for col in range(num_of_col):
    required_data["col"+str(col)] = []

# removing unnecessary data (like '\n' or ',' which are seperating the values in csv )
for row in data_raw:
    row = row.rstrip("\n")
    row_val = row.split(",")
    for i in range(len(row_val)):
        # now seprating values of each feature
        required_data["col"+str(i)].append(float(row_val[i]))
```

In [5]:

```
# now we will make the Daraframe of our to train our model
```

```
required_data=pd.DataFrame(required_data)
required_data
```

Out[5]:

	col0	col1	col2	col3	col4	col5	col6	col7	col8
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28.0	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28.0	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270.0	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365.0	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360.0	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28.0	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28.0	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28.0	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28.0	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28.0	32.40

1030 rows × 9 columns

In [6]:

```
# now we create,train and then predict our dataset
lm = LinearRegression()
lm.fit(required_data[["col0","col1","col2","col3","col4","col5",'
 required_data["col8"])
pred =
lm.predict(required_data[["col0","col1","col2","col3","col4","co

pred
```

Out[6]:

```
array([53.46346329, 53.73475651, 56.81258504, ..., 26.46841169,
       29.12237014, 31.89770807])
```

In [7]:

```
# finding the coefficient(slope) for the linear model
slope = lm.coef_
# finding the intercept for the linear model
intercept = lm.intercept_
```

In [8]:

```
slope
```

Out[8]:

```
array([ 0.11980433,  0.10386581,  0.08793432, -0.14991842,  0.2922246 ,
        0.01808621,  0.02019035,  0.11422207])
```

In [9]:

```
# hence the linear equation Y=AX+B1+B2+B3+....
print("the equation is: Y=",intercept,"X +
",slope[0],"+",slope[1],"+",slope[2],"+",slope[3],"+",slope[4],"+
 \
        slope[5],"+",slope[6],"+",slope[7], sep='')
```

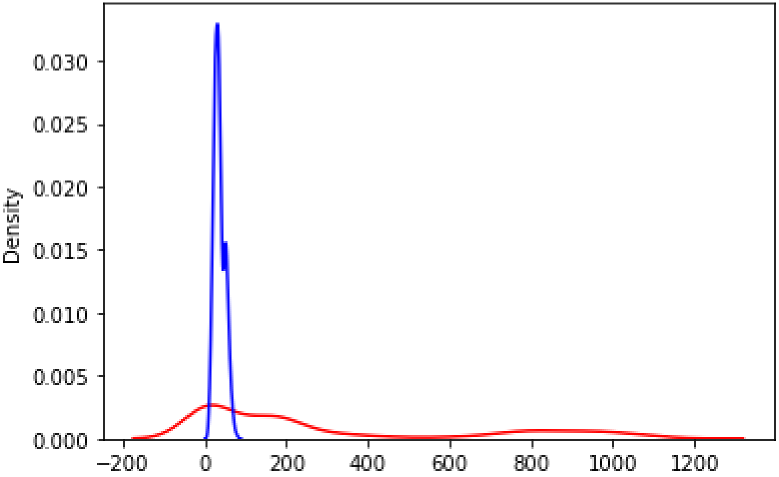
```
the equation is: Y=-23.33121358490314X + 0.1198043344971631+0.1038658088991041 7+0.08793
43215420122+-0.1499184190674037 2+0.2922245951055592 6+0.0180862148274432 42+0.02019035105
301438+0.114222068289382
```

In [10]:

```
# now we plot the distribution
ax1 =
sns.distplot(required_data[["col0","col1","col2","col3","col4","c
 hist=False, color="r",
                   label="Actual Values of The Data Frame")
sns.distplot(pred, hist=False, color="b",
             label="Values used for tejh fitting of the model",
ax=ax1)
```

```
/home/dhanola/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: Futu
reWarning: `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 fl
exibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/home/dhanola/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: Futu
reWarning: `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 fl
exibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

```
Out[10]: <AxesSubplot:ylabel='Density'>
```


Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
−200
0
200
400
600
800
1000
1200