Urban Water Quality Prediction

INDEX

1.INTRODUCTION

1.1 Overview

With the rapid development of economy and accelerated urbanization, water pollution has become more serious. Urban water quality is of great importance in our daily lives. Prediction of urban water quality helps control water pollution and protects the human health. To overcome this kind of problem statement, we developed a machine learning model to predict the Urban water quality.

1.2 Purpose

Water quality is an important topic of discussion when it comes to your overall health. Water is essential to life. Water pollution can cause major health problems that can lead to serious illness. The chance of getting sick from polluted water is extremely high.

And hence keeping in mind the importance of water quality on human health and environment we developed a model that can predict the water quality which can enhance our understanding towards the problem. And thereby helps in taking counter measures to the decreasing water quality.

2.LITERATURE SURVEY

2.1 Existing Problem

Declining water quality has become a global issue of concern as human populations grow. As the human population increases, industrial and agricultural activities expand, and climate change threatens to cause major alternations to the hydrological cycle.

The most prevalent water quality problem is 'Eutrophication', a result of high nutrient loads like nitrogen, phosphorus which substantially impairs beneficial uses of water. Poor water quality has a direct impact on aquatic life in numerous ways.

2.2 Proposed Solution

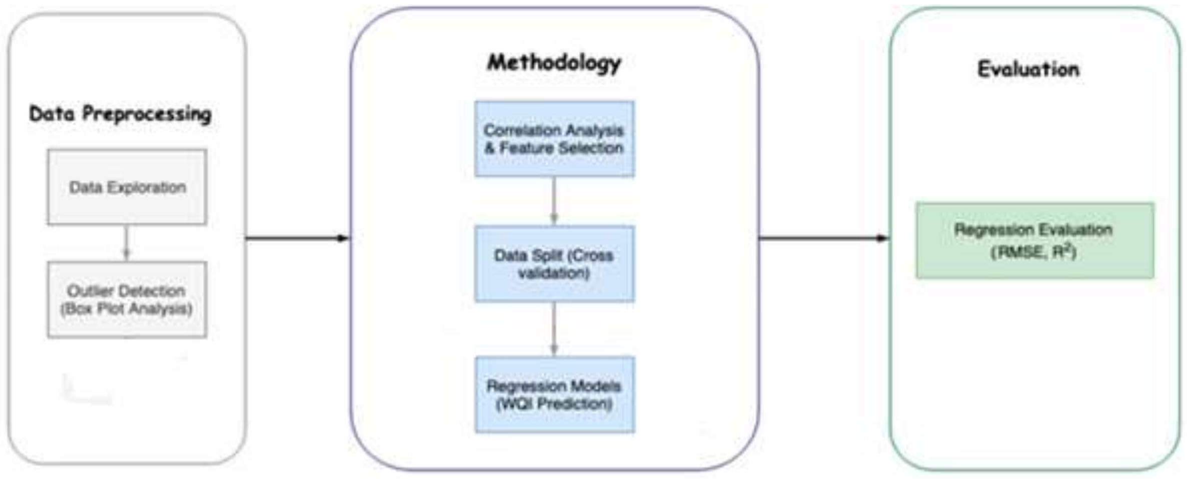
Prediction of water quality is to predict the variation in the trends of water environment quality at a certain time in the future. The water quality prediction is of great significance in planning and control of water quality.

Therefore, understanding the problems and trends of water pollution is of great significance for the prevention and control of water pollution.

We have proposed a system that uses Machine learning algorithms to predict the water quality in Urban areas and forecasts the predictions.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



Data Preprocessing
Data Exploration
Outlier Detection (Box Plot Analysis)
Methodology
Correlation Analysis & Feature Selection
Data Split (Cross validation)
Regression Models (WQI Prediction)
Evaluation
Regression Evaluation (RMSE, R²)

Figure 3.1.1 Block Diagram

3.2 Hardware/ Software Designing

Table 3.2.1 Weightage of Different parameters to calculate WQI

Parameter	Range	Weightage
pH	6.5-8.5	0.14
Temperature	25-27	0.1
D.O. (mg/l)	6-8	0.2
Conductivity (μmhos/cm)	50-2000	0.06
NITRATENAN N+ NITRITENANN (mg/l)	0-1	0.18
B.O.D. (mg/l)	0-3	0.12
Faecal Coliform (MPN/100ml)	0-5000	0.2

4.ENVIRONMENT INVESTIGATION

A lot of parameters are responsible for the degradation of water quality among them the following parameters have more prominent effect:

pH:

pH is a measure of how acidic/basic water is. The range goes from 0 to 14, with 7 being neutral. pH’s of less than 7 indicate acidity, whereas a pH of greater than 7 indicates a base.

The pH of water determines the solubility (amount that can be dissolved in the water) and biological availability (amount that can be utilized by aquatic life) of chemical constituents such as nutrients (phosphorus, nitrogen, and carbon) and heavy metals (lead, copper, cadmium, etc.).

Excessively high and low pH’s can be detrimental for the use of water. High pH causes a bitter taste, water pipes and water-using appliances become encrusted with deposits, and it depresses the effectiveness of the disinfection of chlorine, thereby causing the need for additional chlorine when pH is high. Low-pH water will corrode or dissolve metals and other substances.

Pollution can change a water's pH, which in turn can harm animals and plants living in the water. For instance, water coming out of an abandoned coal mine can have a pH of 2, which is very acidic and would definitely affect any fish crazy enough to try to live in it! By using the logarithm scale, this mine-drainage water would be 100,000 times more acidic than neutral water -- so stay out of abandoned mines.

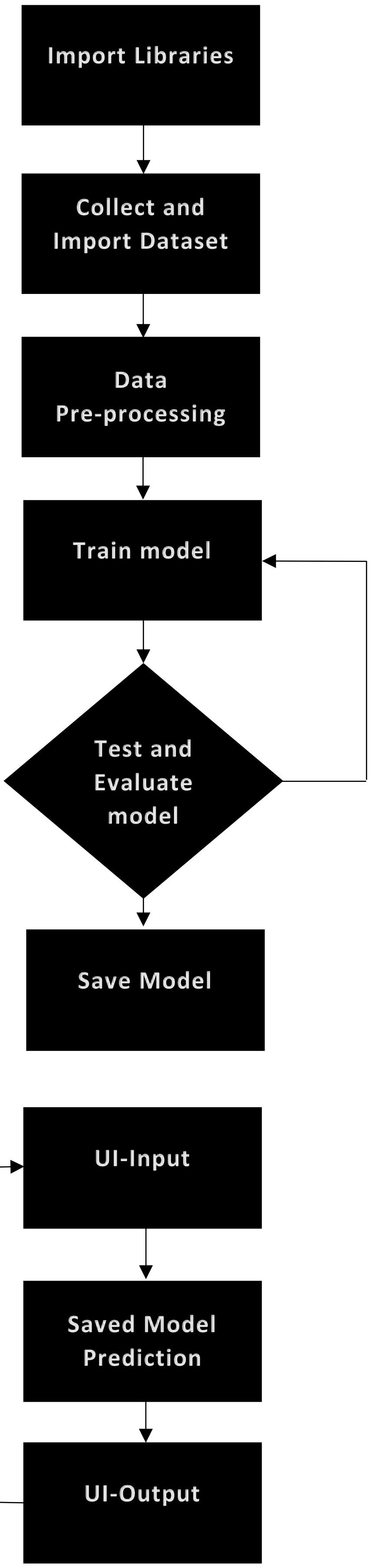
D.O (mg/L):

Unlike air, which is normally about 21 percent oxygen, water contains only a tiny fraction of a percentage of dissolved oxygen. In water it usually is expressed in milligrams per litre (mg/L), parts per million (ppm), or percent of saturation. At sea level, typical DO concentrations in 100-percent saturated fresh water will range from 7.56 mg/L (or 7.56 parts oxygen in 1,000,000 parts water) at 30 degrees Celsius to 14.62 mg/L at zero degrees Celsius.

As environmental protection is becoming an important global attention and focuses on getting a value added by product from waste which does not cause environmental pollution or eco-friendly. Chitosan is a natural biopolymer obtained as by-product from sea food industry waste, has versatile application in various fields.

Chitosan is used as a coagulant to remove turbidity and compared with other commonly used coagulant, alum (Aluminium sulphate). In present work, we also focused on parameters such as pH, B.O.D, D.O and nitrate. also studied the various characteristics of chitosan.

5. FLOWCHART


Import Libraries
Collect and Import Dataset
Data Pre-processing
Train model
Test and Evaluate model
Save Model
UI-Input
Saved Model Prediction
UI-Output

6.RESULT

The proposed machine learning model has a r2 score of 0.8721 and mean square error of 22.967. The following graph represents the water quality index from the year 2007 to 2025.

```
lr.predict([[2025]])
```

```
array([87.54050881])
```

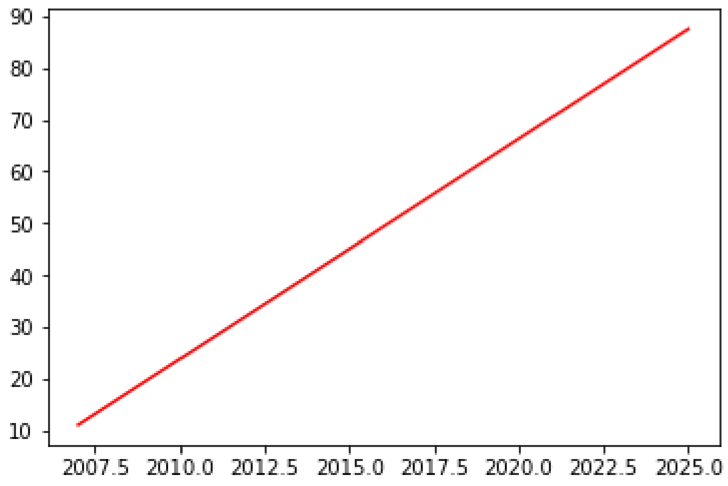


Figure 6.1 Water Quality Prediction From 2007 to 2025(WQI)

7.ADVANTAGES AND DISADVANTAGES

Advantages:

- Create public and private places that harvest, clean, and recycle water, resulting in water resource, environmental and social livability benefits.
- Since it predicts the future water quality measures can be taken accordingly.
- No need of human interference in the prediction of water quality.

Disadvantages:

- This is a supervised algorithm and hence needs to be monitored.
- The data provided to the model should be accurate.
- Lack of large dataset.

8.APPLICATIONS

The application of the model demonstrates how it can be a useful tool for water management and planning in multiple settings, especially in data-limited scenarios.

The model can also be useful for urban water planners in developed countries, emphasizing how sustainability is central to assimilative capacity-based water quality modelling.

It lets the environmentalist to know about the current scenario of water quality depletion in urban areas.

9.CONCLUSION

Few water bodies are not advisable for drinking as well as for washing or any other purposes. One of the reasons affecting the quality of water in the various states of India is that the wastewater and industrial effluents are discharged untreated directly into the river. Less than 50 per cent of the population in India has access to safely managed drinking water, Groundwater from over 30 million access points supplies 85 per cent of drinking water in rural areas and 48 per cent of water requirements in urban areas.

The Water quality prediction model can give an overview of how the upcoming future water qualities will be, so that we take precautions to overcome the situations,

If the current trend continues and then by 2025 or soon, we can expect the water to be polluted to an extent from which we might not be able to recover.

10.FUTURE SCOPE

For this water predication machine learning model, we can add few add ones to make it into advanced model. In future, we plan to deal with the water quality inference problems in the urban water distribution systems through a limited number of water quality monitor stations.

In future works, we propose integrating the findings of this project in a large-scale IoT-based online predicting system using only the sensors of the required parameters. The tested algorithms would predict the water quality immediately based on the real-time data fed from the IoT system.

The proposed IoT system would employ the parameter sensors of pH, turbidity, temperature, and TDS for parameter readings and communicate those readings using an Arduino microcontroller and ZigBee transceiver. It would identify poor quality water before it is released for consumption and alert concerned authorities. It will hopefully result in curtailment of people consuming poor quality water and consequently de-escalate harrowing diseases like typhoid and diarrhoea.

11. BIBILOGRAPHY

[1] AliNajah Ahmed ,FaridahBinti Othman, "Machine learning methods for better water quality prediction",2019.
[2] "Clean water for a healthy world", *Development*, 2010.

[3] David Rosenblum, "Predicting Urban Water Quality with Ubiquitous Data", 10 February 2020.

[4] K. Farrell-Poe, "Water Quality & Monitoring", 2000.

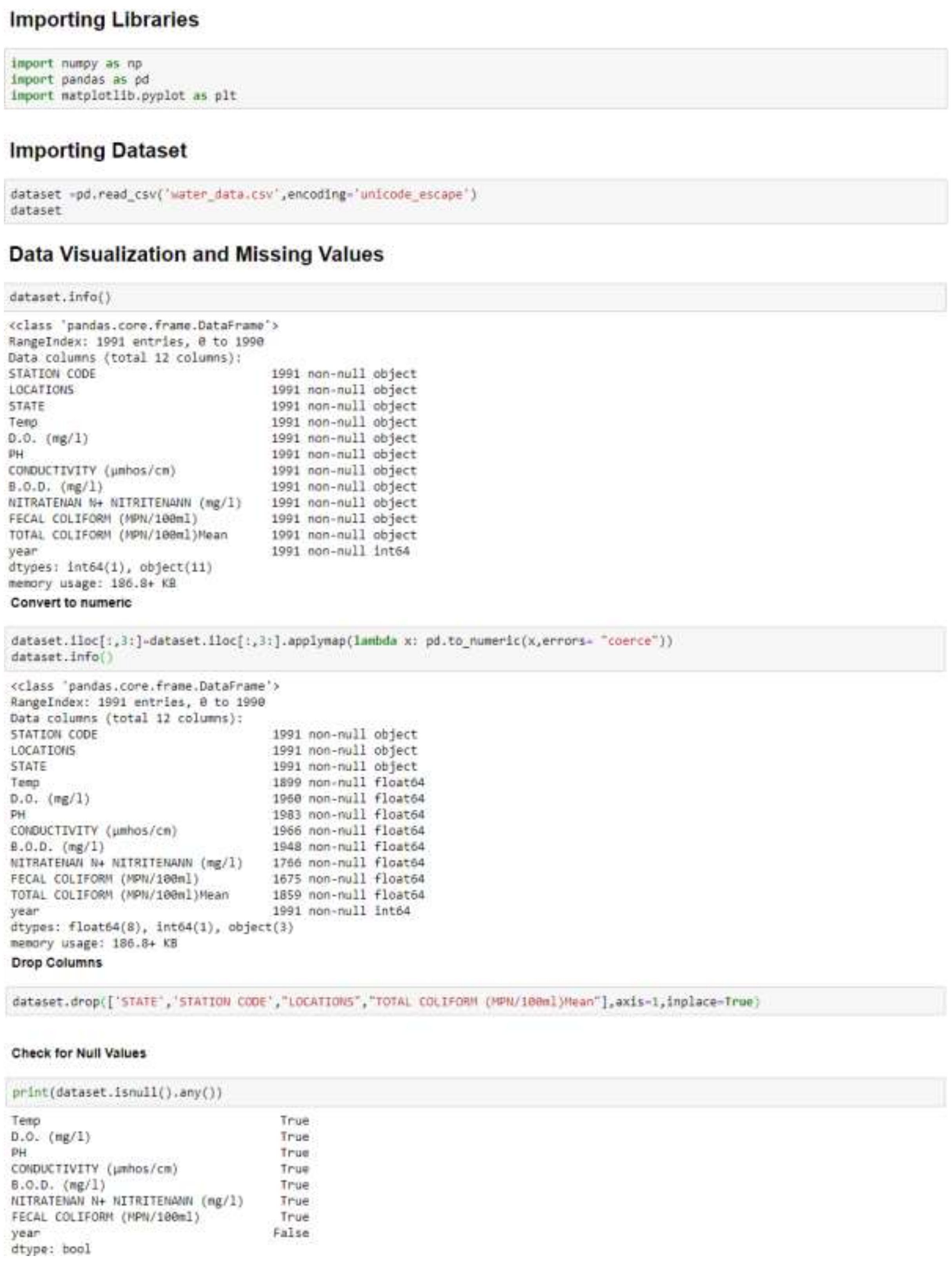
[5] Umair Ahmed, Rafia Mumtaz, "Efficient Water Quality Prediction Using Supervised Machine Learning",2019.

[6] Yafra Khan ," Predicting and analysing water quality using Machine Learning",2016.

12.APPENDIX

a. Source Code

(i) Data Pre-processing:



Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing Dataset

```
dataset =pd.read_csv('water_data.csv',encoding='unicode_escape')
dataset
```

Data Visualization and Missing Values

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
STATION CODE                       1991 non-null object
LOCATIONS                          1991 non-null object
STATE                              1991 non-null object
Temp                               1991 non-null object
D.O. (mg/l)                        1991 non-null object
PH                                 1991 non-null object
CONDUCTIVITY (µmhos/cm)            1991 non-null object
B.O.D. (mg/l)                      1991 non-null object
NITRATENAN N+ NITRITENANN (mg/l)   1991 non-null object
FECAL COLIFORM (MPN/100ml)         1991 non-null object
TOTAL COLIFORM (MPN/100ml)Mean     1991 non-null object
year                               1991 non-null int64
dtypes: int64(1), object(11)
memory usage: 186.8+ KB
```

Convert to numeric

```
dataset.iloc[:,3:]=dataset.iloc[:,3:].applymap(lambda x: pd.to_numeric(x,errors= "coerce"))
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
STATION CODE                       1991 non-null object
LOCATIONS                          1991 non-null object
STATE                              1991 non-null object
Temp                               1899 non-null float64
D.O. (mg/l)                        1960 non-null float64
PH                                 1983 non-null float64
CONDUCTIVITY (µmhos/cm)            1966 non-null float64
B.O.D. (mg/l)                      1948 non-null float64
NITRATENAN N+ NITRITENANN (mg/l)   1766 non-null float64
FECAL COLIFORM (MPN/100ml)         1675 non-null float64
TOTAL COLIFORM (MPN/100ml)Mean     1859 non-null float64
year                               1991 non-null int64
dtypes: float64(8), int64(1), object(3)
memory usage: 186.8+ KB
```

Drop Columns

```
dataset.drop(['STATE','STATION CODE',"LOCATIONS","TOTAL COLIFORM (MPN/100ml)Mean"],axis=1,inplace=True)
```

Check for Null Values

```
print(dataset.isnull().any())
```

```
Temp                                True
D.O. (mg/l)                         True
PH                                  True
CONDUCTIVITY (µmhos/cm)             True
B.O.D. (mg/l)                       True
NITRATENAN N+ NITRITENANN (mg/l)    True
FECAL COLIFORM (MPN/100ml)          True
year                               False
dtype: bool
```

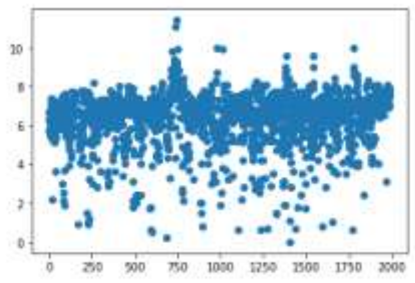
Removing Outliners

```
dataset.describe()
```

	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	year
count	1899.000000	1960.000000	1983.000000	1966.000000	1948.000000	1766.000000	1.675000e+03	1991.000000
mean	26.209814	6.392637	112.090674	1786.466394	6.940049	1.623079	3.625294e+05	2010.038172
std	3.366388	1.332938	1878.930716	5552.276223	29.400026	4.090481	8.764767e+06	3.057333
min	10.000000	0.000000	0.000000	0.400000	0.100000	0.000000	0.000000e+00	2003.000000
25%	24.750000	5.900000	6.900000	78.000000	1.200000	0.240000	2.600000e+01	2008.000000
50%	27.000000	6.700000	7.300000	183.000000	1.896500	0.516000	2.210000e+02	2011.000000
75%	28.400000	7.200000	7.700000	592.750000	3.600000	1.500000	9.965000e+02	2013.000000
max	35.000000	11.400000	67115.000000	65700.000000	534.500000	108.700000	2.725216e+08	2014.000000

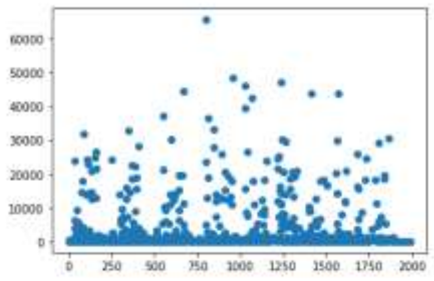
```
plt.scatter(range(1991),dataset["D.O. (mg/l)"])
```

```
<matplotlib.collections.PathCollection at 0x1a7538eb708>
```



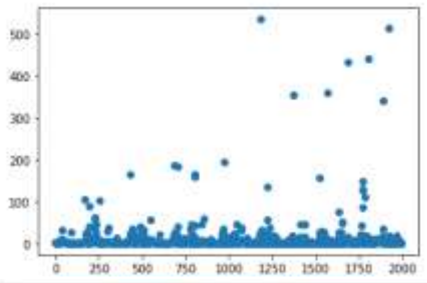
```
plt.scatter(range(1991),dataset["CONDUCTIVITY (µmhos/cm)"])
```

```
<matplotlib.collections.PathCollection at 0x1a753966148>
```



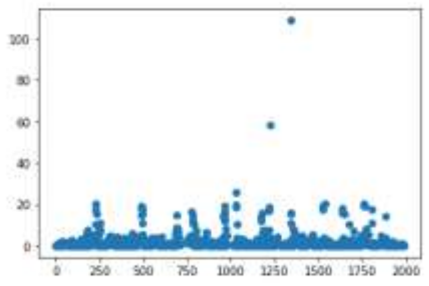
```
plt.scatter(range(1991),dataset["B.O.D. (mg/l)"])
```

```
<matplotlib.collections.PathCollection at 0x1a7539d39c8>
```



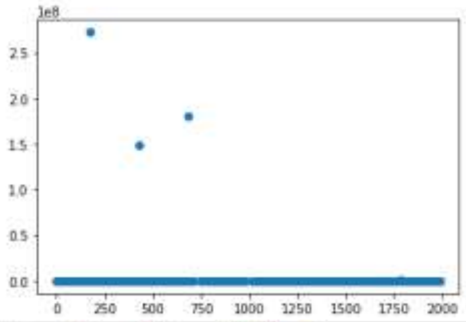
```
plt.scatter(range(1991),dataset["NITRATENAN N+ NITRITENANN (mg/l)"])
```

```
<matplotlib.collections.PathCollection at 0x1a753a40808>
```



```
plt.scatter(range(1991),dataset["FECAL COLIFORM (MPN/100ml)"])
```

```
<matplotlib.collections.PathCollection at 0x1a753aa4f88>
```



```
dataset=dataset[dataset["PH"]<14]
dataset=dataset[dataset["PH"]>4]
dataset=dataset[dataset["B.O.D. (mg/l)"]<190]
dataset=dataset[dataset["FECAL COLIFORM (MPN/100ml)"]<1000000000]
```

```
print(dataset.info())
dataset.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1596 entries, 1 to 1900
Data columns (total 8 columns):
Temp                                 1550 non-null float64
D.O. (mg/l)                          1594 non-null float64
PH                                   1596 non-null float64
CONDUCTIVITY (µmhos/cm)              1594 non-null float64
B.O.D. (mg/l)                        1596 non-null float64
NITRATENAN N+ NITRITENANN (mg/l)     1563 non-null float64
FECAL COLIFORM (MPN/100ml)           1596 non-null float64
year                                 1596 non-null int64
dtypes: float64(7), int64(1)
memory usage: 112.2 KB
None
```

	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	year
count	1550.000000	1594.000000	1596.000000	1594.000000	1596.000000	1563.000000	1.596000e+03	1596.000000
mean	26.321120	6.321489	7.210962	1840.548701	4.570059	1.391031	3.797053e+05	2010.441103
std	3.263076	1.285105	0.503566	5339.636857	10.977493	2.809978	8.978805e+06	2.716658
min	10.000000	0.000000	5.600000	11.000000	0.100000	0.000000	0.000000e+00	2005.000000
25%	25.000000	5.900000	6.900000	79.000000	1.100000	0.250000	3.400000e+01	2008.000000
50%	27.000000	6.700000	7.200000	180.500000	1.700000	0.500000	2.360000e+02	2011.000000
75%	28.400000	7.100000	7.600000	611.250000	3.400000	1.406000	1.082750e+03	2013.000000
max	35.000000	10.000000	9.010000	47156.000000	185.800000	58.100000	2.725216e+08	2014.000000

Fill NaN

```
dataset['Temp']=dataset['Temp'].replace(np.NaN,26.318446)
dataset['D.O. (mg/l)']=dataset['D.O. (mg/l)'].replace(np.NaN,6.330581)
dataset['CONDUCTIVITY (µmhos/cm)']=dataset['CONDUCTIVITY (µmhos/cm)'].replace(np.NaN,1839.723212)
dataset['NITRATENAN N+ NITRITENANN (mg/l)']=dataset['NITRATENAN N+ NITRITENANN (mg/l)'].replace(np.NaN,1.391046)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1596 entries, 1 to 1900
Data columns (total 8 columns):
Temp                                 1596 non-null float64
D.O. (mg/l)                          1596 non-null float64
PH                                   1596 non-null float64
CONDUCTIVITY (µmhos/cm)              1596 non-null float64
B.O.D. (mg/l)                        1596 non-null float64
NITRATENAN N+ NITRITENANN (mg/l)     1596 non-null float64
FECAL COLIFORM (MPN/100ml)           1596 non-null float64
year                                 1596 non-null int64
dtypes: float64(7), int64(1)
memory usage: 112.2 KB
```

Organize Dataset

```
df=dataset.groupby(by=["year"],sort=True,as_index=True).mean()
df
```

```
          Temp  D.O. (mg/l)        PH  CONDUCTIVITY (µmhos/cm)  B.O.D. (mg/l)  NITRATENAN N+ NITRITENANN (mg/l)  FECAL COLIFORM (MPN/100ml)
year
2005 26.542677     6.458511  7.244787              1792.402128       3.940489                          0.733938                1.918894e+03
2006 25.047078     6.443632  7.227909              1544.766173       4.118500                          0.994501                3.184227e+03
2007 26.204301     6.466667  7.149462              2237.419355       3.040860                          0.821810                6.136817e+03
2008 26.788321     6.088321  7.070073              1504.569343       4.265766                          1.104491                5.323146e+03
2009 27.264151     6.223742  7.150440              3535.508511       4.636478                          1.471996                1.344756e+03
2010 26.683836     6.426419  7.119560              1664.389937       4.538805                          1.173485                1.558840e+03
2011 26.830233     6.389535  7.275000              2172.546512       4.900000                          1.536808                1.333326e+03
2012 25.953586     6.352321  7.279325              1396.464135       5.073840                          1.594190                7.620333e+05
2013 26.037288     6.312288  7.258898              1529.766949       4.661441                          1.774788                6.340517e+05
2014 25.854751     6.216742  7.238009              1477.076923       4.929864                          1.602715                1.236742e+06
```

```python
df.describe()
```

```
            Temp  D.O. (mg/l)         PH  CONDUCTIVITY (µmhos/cm)  B.O.D. (mg/l)  NITRATENAN N+ NITRITENANN (mg/l)  FECAL COLIFORM (MPN/100ml)
count  10.000000    10.000000  10.000000                10.000000      10.000000                         10.000000                1.000000e+01
mean   26.320622     6.337818   7.201346              1885.490997       4.410604                          1.280872                2.653627e+05
std     0.633982     0.126348   0.072999               648.078106       0.603959                          0.362439                4.482261e+05
min    25.047078     6.088321   7.070073              1396.464135       3.040860                          0.733938                1.333326e+03
25%    25.974512     6.245879   7.149707              1510.868745       4.155317                          1.021998                1.648853e+03
50%    26.373489     6.370928   7.232959              1604.578055       4.587642                          1.322741                4.253687e+03
75%    26.762200     6.439328   7.255371              2077.510416       4.840360                          1.579845                4.770730e+05
max    27.264151     6.466667   7.279325              3535.508511       5.073840                          1.774788                1.236742e+06
```

```python
y=pd.Series()
yy=pd.DataFrame()
```

```python
y=df["PH"].apply(lambda x: (0 if (8>=x>=7)
                   else (0.028 if (8.5>=x>=8) or (7>=x>=6.5)
                   else (0.084 if (9>=x>=8.8) or (6.5>=x>=6.3)
                   else (0.112 if (10>=x>=9)  or (6.3>=x>=6)
                   else  0.14)))))
yy["PH"]=df["PH"].apply(lambda x: (0 if (8>=x>=7)
                   else (0.028 if (8.5>=x>=8) or (7>=x>=6.5)
                   else (0.084 if (9>=x>=8.8) or (6.5>=x>=6.3)
                   else (0.112 if (10>=x>=9)  or (6.3>=x>=6)
                   else  0.14)))))
```

```python
yy["D.O. (mg/l)"]=df["D.O. (mg/l)"].apply(lambda x: (0 if (8>=x>=6.5)
                              else (0.04 if (6.5>=x>=6)
                              else  0.2)))
y=y+yy["D.O. (mg/l)"]
```

```python
yy["CONDUCTIVITY (µmhos/cm)"]=df["CONDUCTIVITY (µmhos/cm)"].apply(lambda x: (0 if (1500>=x>=50)
                                          else (0.012 if (2000>=x>=1500)
                                          else (0.048 if (2500>=x>=2000)
                                          else  0.06))))
y=y+yy["CONDUCTIVITY (µmhos/cm)"]
```

```python
yy["NITRATENAN N+ NITRITENANN (mg/l)"]=df["NITRATENAN N+ NITRITENANN (mg/l)"].apply(lambda x: (0 if (1>=x)
                                          else (0.036 if (1.5>=x>=1)
                                          else (0.144 if (2>=x>=1.5)
                                          else  0.18))))
y=y+yy["NITRATENAN N+ NITRITENANN (mg/l)"]
```

```python
yy["B.O.D. (mg/l)"]=df["B.O.D. (mg/l)"].apply(lambda x: (0 if (3>=x>=0)
                                else (0.024 if (5>=x>=3)
                                else (0.072 if (10>=x>=5)
                                else  0.12))))
y=y+yy["B.O.D. (mg/l)"]
```

```python
yy["FECAL COLIFORM (MPN/100ml)"]=df["FECAL COLIFORM (MPN/100ml)"].apply(lambda x: (0 if (5000>=x>=0)
                                       else (0.04 if (10000>=x>=5000)
                                       else (0.12 if (100000>=x>=10000)
                                       else  0.2))))
y=y+yy["FECAL COLIFORM (MPN/100ml)"]
```

```python
y=y*100
yy["y"]=y
```

	PH	D.O. (mg/l)	CONDUCTIVITY (µmhos/cm)	NITRATENAN N+ NITRITENANN (mg/l)	B.O.D. (mg/l)	FECAL COLIFORM (MPN/100ml)	y
year							
2005	0	0.04	0.012	0.000	0.024	0.00	7.6
2006	0	0.04	0.012	0.000	0.024	0.00	7.6
2007	0	0.04	0.048	0.000	0.024	0.04	15.2
2008	0	0.04	0.012	0.036	0.024	0.04	15.2
2009	0	0.04	0.060	0.036	0.024	0.00	16.0
2010	0	0.04	0.012	0.036	0.024	0.00	11.2
2011	0	0.04	0.048	0.144	0.024	0.00	25.6
2012	0	0.04	0.000	0.144	0.072	0.20	45.6
2013	0	0.04	0.012	0.144	0.024	0.20	42.0
2014	0	0.04	0.000	0.144	0.024	0.20	40.8

```
x=df.index.tolist()
```

```
x=list(map(lambda z:[z,],x))
```

```
y=list(y)
```

Splitting Data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Model Training

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

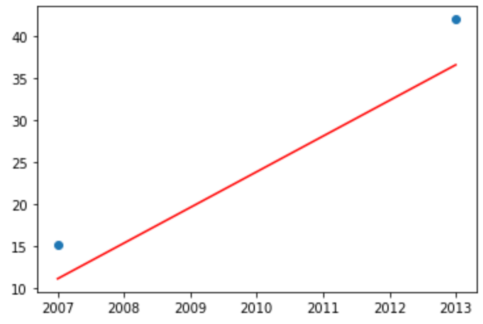
Predicting for Test Data

```
y_pre=lr.predict(x_test)
y_pre
```

```
array([11.11624266, 36.59099804])
```

```
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pre,"r")
```

```
[<matplotlib.lines.Line2D at 0x215c8f50548>]
```



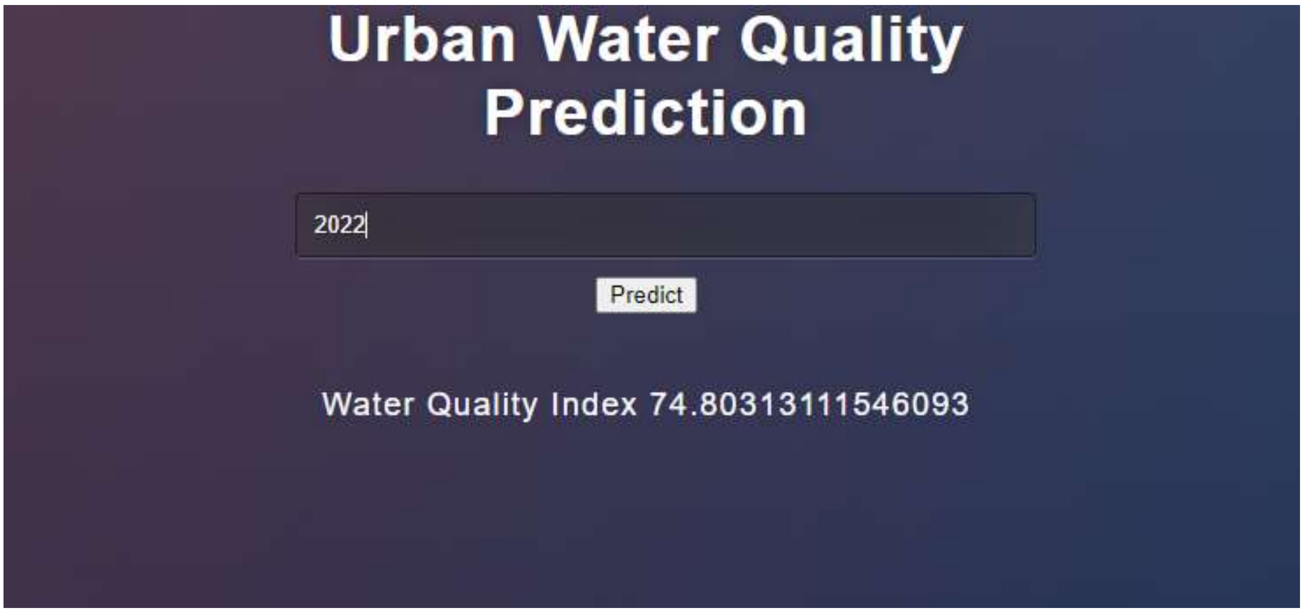
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pre)
```

```
0.8720918462618956
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pre)
```

```
22.967188085214037
```

b. UI Output screenshot


Urban Water Quality Prediction

2022

Water Quality Index 74.80313111546093