WATER QUALITY PREDICTION

A Mini - Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

CHILAMKURTHI C V S VARSHITH	**121710314012**
G RUTWIZ GANGADHAR	**121710314020**
KAVALI SRIKANTH	**121710314027**
THAKSHAK SUNKARA	**121710314055**
VANAPALLI RAHUL	**121710314062**

Under the esteemed guidance of

Mrs. M Raja Mani

Assistant Professor , GIT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM(Deemed to be University)

VISAKHAPATNAM

APRIL 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

VISAKHAPATNAM

DECLARATION

We, hereby declare that the Project entitled "**Water Quality Prediction using ML**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering. The work has not been submitted to any other college or university for the award of any degree or diploma.

Name	**Registration number**
CHILAMKURTHI C V S VARSHITH	121710314012
G RUTWIZ GANGADHAR	121710314020
KAVALI SRIKANTH	121710314027
THAKSHAK SUNKARA	121710314055
VANAPALLI RAHUL	121710314062

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

VISAKHAPATNAM

CERTIFICATE

This is to certify that the project report entitled **"Water Quality Prediction using ML"** is a bonafide record of work carried out by **CHILAMKURTHI C V S VARSHITHI (121710314012), G RUTWIZ GANGADHAR(121710314020), KAVALI SRIKANTH (121710314027), THAKSHAK SUNKARA (121710314055), VANAPALLI RAHUL (121710314062),** submitted in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

PROJECT GUIDE

Mrs. M Raja Mani
Assistant Professor
CSE, GIT
GITAM, Visakhapatnam.

HEAD OF THE DEPARTMENT

Dr.K.THAMMI REDDY
Professor
CSE, GIT
GITAM, Visakhapatnam.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We would like to thank our project guide **Mrs. M Raja Mani** , Assistant Professor, Department of CSE for her stimulating guidance and profuse assistance. We shall always cherish our association with her guidance, encouragement, and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under her guidance and constant support.

We also express our thanks to the project's reviewers **M.Venkata Ramana**, Assistant Professor, and **Dr.Bhramaramba Ravi**, Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance for doing our project.

We consider it a privilege to express our deepest gratitude to **Dr. K. Thammi Reddy**, Head of the Department, Computer Science Engineering for his valuable suggestions and constant motivation that greatly helped us to complete this project.

Our sincere thanks to **Dr. C. Dharma Raj**, Principal, GITAM Institute of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

Name	**Registration number**
Chilamkurthi C V S Varshith	121710314012
Kavali Srikanth	121710314027
Thakshak Sunkara	121710314055
Vanapalli Rahul	121710314062
G Rutwiz Gangadhar	121710314020

ABSTRACT

In this Project, we extracted monitoring data from the water transfer channel of both the water resource and the intake area as training samples and selected some distinct indices as input factors to establish a BP neural network whose connection weight values between network layers and the threshold of each layer had already been optimized by an improved artificial bee colony (IABC) algorithm. Compared with the traditional BP and ABC-BP neural network model, it was shown that the IABC-BP neural network has a greater ability for forecasting and could achieve much better accuracy, nearly 25% more precise than the BP neural network. The new model is particularly practical for the water quality prediction of a water diversion project and could be readily applied in this field. Water shortage and water resource pollution have become major problems in China. The water quality prediction is of great significance to the planning and control of water quality. In order to make the plan for water pollution prevention and control, it is necessary to predict the changes of water quality at different pollution levels in the future so as to formulate a reasonable plan. Although the mechanism model takes into account the physical, chemical, and biological factors that result in the changes of water quality, these models are relatively complex, the required water quality data is very large and these factors limit the further application of the model to the water to some degree [2]. In recent decades, with the development of computer technology, the non-mechanism model has become a hotspot for research on the water quality prediction model, such as the Markov method [3], the grey prediction model method [4]. The characteristic of these methods is to establish a water quality prediction model with a certain algorithm from the perspective of the variation in water quality data and without considering the relationship of the water pollution and the changing mechanism. In other words, the modeling method is a kind of black box type. The water environment system is a system with strong nonlinear and nondeterministic characteristics. The traditional linear prediction model cannot fully reflect its changing regulation and the prediction accuracy also cannot now satisfy requirements. The water pollution process is so complex that it is not only affected by natural factors and pollutant discharge, but also by factors such as social and economic development, resulting in the nonlinear water environment system [6]. Now many of the reaction mechanisms of a chemical, biological process cannot be expressed exactly by a mathematical equation which limits the applicability and accuracy of the traditional water quality mathematical model.

INTRODUCTION

Deep Learning is an artificial intelligence (AI) function that imitates the human brain's workings in processing data and creating patterns for decision-making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from unstructured or unlabeled data. Also known as deep neural learning or deep neural network. Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out machine learning.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural «pylab» interface based on a state machine , designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Overview:

Prediction of water quality which can ensure the water supply and prevent water pollution is essential for a successful water transfer project. In recent years, with the development of artificial intelligence, the backpropagation (BP) neural network has been increasingly applied for the prediction and forecasting field. However, the BP neural network frame cannot satisfy the demand of higher accuracy. In this study, we extracted monitoring data from the water transfer channel of both the water resource and the intake area as training samples and selected some distinct indices as input factors to establish a BP neural network whose connection weight values between network layers and the threshold of each layer had already been optimized by an improved artificial bee colony (IABC) algorithm. Compared with the traditional BP and ABC-BP neural network model, it was shown that the IABC-BP neural network has a greater ability for forecasting and could achieve much better accuracy, nearly 25% more precise than the BP neural network. The new model is particularly practical for the water quality prediction of a water diversion project and could be readily applied in this field.

Purpose:

Water shortage and water resource pollution have become major problems in China. In order to solve the problem of water resource imbalance, water diversion projects in many areas have been constructed. Water quality is the key to the success of a water diversion project. The prediction of water quality is to predict the variation trend of water environment quality at a certain time in the future. The water quality prediction is of great significance to the planning and control of water quality. In order to make the plan for water pollution prevention and control, it is necessary to predict the changes of water quality at different pollution levels in the future so as to formulate a reasonable plan. For a water diversion project it is more important to predict the water quality because quite a significant amount of the water is transferred for solving daily drinking problems. Therefore, it is of great significance to explore the methods of water quality prediction in the present society

LITERATURE SURVEY

Existing Problem:

With the rapid development of the economy and accelerated ization, water pollution has become more and more serious. Water quality is of great importance to our daily lives. Prediction of water quality helps control water pollution and protect human health. To overcome this kind of problem statement, we developed a deep learning model to predict the water quality. Water quality is affected by a wide range of natural and human influences. The most important of the natural influences are geological, hydrological and climatic, since these affect the quantity and the quality of water available. Although the natural ecosystem is in harmony with natural water quality, any significant changes to water quality will usually be disruptive to the ecosystem.

Proposed Solution:

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 & to forecast the predictions. We used the new understanding of the problem to consider different framings of the prediction problem, ways the data may be prepared, and modelling methods that may be used. Finally, we chose the Random Forest Regression model as it offered the highest accuracy. Then we implemented our model to predict the quality of water given by various quantities like pH, D.O , Conductivity, TotalColiform, Wqi etc. using a local host web application.

TASKS TAKEN UP

- Dataset preparation
- Dataset pre-processing
    - Import the libraries
    - Import the Dataset
    - Data visualization
    - Taking care of Missing Data
    - Splitting the Data into Train and Test
- Model Building
    - Training and Testing the Model
    - Evaluation and the final predicting method should be such that the accuracy should probably be at most.
- Application Building
    - Create an HTML Files
    - Structuring a proper Python Code

METHODOLOGY AND LEARNING

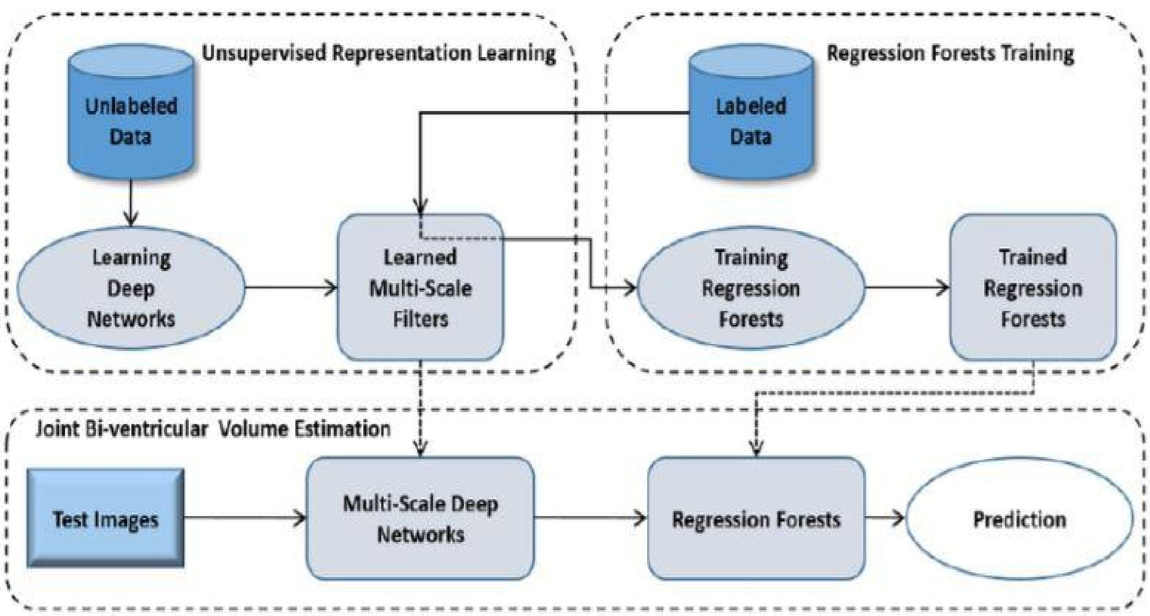
Technologies/Tools Used:

- Jupyter Notebook Environment
- Spyder Ide
- Matplotlib , Seaborn
- Python (pandas, NumPy)
- HTML
- Flask

Source of Data:

Kaggle is an online community for descriptive analysis and predictive modeling. It collects a variety of research fields from analytic data practitioners. Data scientists compete to build the best model for both descriptive and predictive analytics. However, it allows individuals to access their dataset to create models and work with other data scientists to solve various real-world analytics problems. The input dataset used in developing this model has been downloaded from Kaggle.

Block Diagram:



Theoretical Analysis:

Modeling Methods:

The prediction of water quality components can be done by using artificial intelligence (AI) techniques including MLP, SVM, and group method of data handling (GMDH). And also be done using machine learning(ML) techniques including Random forest regressor , Decision tree , Support Vector Machines (SVM), Neural Networks (NN), Deep Neural Networks (Deep NN) and k Nearest Neighbors (kNN), with the highest accuracy of 93% with Deep NN.

The aim of this study is prediction of water quality components is done by using machine learning methods i.e, Random forest Regressor.

Machine learning Methods:

Random Forest Regression: A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. This part is called Bootstrap. Below is a step by step sample implementation of Random Forest Regression.

Step 1 : Import the required libraries.

Step 2 : Import and print the dataset.

Step 3 : Select all rows and column 1 from the dataset to x and all rows and column 2 as y .

Step 4 : Fit Random forest regressor to the dataset

Step 5 : Predicting a new result .

Step 6 : Visualising the result .

Experimental Investigation

The water quality prediction dataset is collected from water samples from various areas in India.There are 12 parameters taken in the dataset some of which are various pollutant measures columns and each column consists of the average values over a period of time.The entire dataset observations are made between 2003-2014.

It is comprised of 12 variables which are:

● STATION CODE: Unique code of area from which water sample is collected.

● Location:City from which water samples are collected.

● State: State of the area.

● Temp : The average values of temperature over time.

● D.O(mg/l) : The average values of D.O of water over time.

● PH: The average values of ph of water over time.

● Conductivity: The average values of conductivity of water over time.

● B.O.D: The average values of B.O.D of water over time.

● Nitrate N N+: The average values of NA over time.

● Fecal coliform: The average values of fecal coliform of water over time.

● Total coliform: The average values of total coliform of water e over time.

In all the columns there are null values.State column is having the highest null values. Different data visualizations are made on the data like box plot,line plot,cat plot etc to interpret each parameter.From scatter plot between station code and total coliform,most of the station codes are having the same range of total coliform in water. In 2003,the ph values are varying from each other and from 2004,ph values seem to remain constant.In year 2012,the distribution intensity is more.With line plot between year and pollutant measure total coliform ,we can conclude that coliform is not depending on the year. To predict the water quality,we need to calculate the water quality index.

Water quality index is determined by using mathematical calculations on ph,na,coliform,conductivity,nitrate,D.O,B.O.D. Station code also needed to predict water quality according to locations. So these columns are considered and calculated which results in a water quality index.

Water quality constraints to decide the purity:

Excellent: (WQI Value 95-100) – Water quality is protected with a virtual absence of impairment; conditions are very close to pristine levels. These index values can only be obtained if all measurements meet recommended guidelines virtually all of the time.

Very Good: (WQI Value 89-94) – Water quality is protected with a slight presence of impairment; conditions are close to pristine levels.

Good: (WQI Value 80-88) – Water quality is protected with only a minor degree of impairment; conditions rarely depart from desirable levels.

Fair: (WQI Value 65-79) – Water quality is usually protected but occasionally impaired; conditions sometimes depart from desirable levels.

Marginal: (WQI Value 45-64) – Water quality is frequently impaired; conditions often depart from desirable levels.

Poor: (WQI Value 0-44) – Water quality is almost always impaired; conditions usually depart from desirable levels.

From the observations of the data visualization using matplot and seaborn and also considering parameters needed for water index calculation,we have finally considered some of the parameters and dropped unnecessary columns. Final parameters taken are:

- Station code
- PH
- Conductivity
- D.O
- B.O.D
- Total coliform
- Nitrate N N+

Saving the model and Predicting the model :

We imported NumPy arrays, Pandas, Matplotliib and also Random Forest Regression algorithms to predict the model.

SCREENSHOTS SOURCE CODE AND RESULTS

Model Development Code:

WATER QUALITY PREDICTION

CATEGORY: MACHINE LEARNING

SKILLS REQUIRED: Python,Python Web Frame Works,Python For Data Analysis,Python For Data Visualization,Exploratory Data Analysis,Data Preprocessing Techniques,Machine Learning,Regression Algorithms,Classification Algorithms.

PROJECT DESCRIPTION: With the rapid development of economy and accelerated urbanization, water pollution has become more and more serious. Urban water quality is of great importance to our daily lives. Prediction of water quality help control water pollution and protect human health. To overcome this kind of problem statement, we developed a deep learning model to predict the water quality.

SOLUTION: 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 & to forecast the predictions.

DATA PREPROCESSING ¶

"IMPORTING THE LIBRARIES"

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

IMPORTING THE DATASET

In [2]:
```
data = pd.read_csv("dataset.csv",encoding ='ISO-8859-1',low_memory =False)
```

In [5]:
```
data
```

Out[5]:

	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (Âµmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393.0	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203.0	NaN	0.1	11	27	2014
1	1399.0	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189.0	2.0	0.2	4953	8391	2014
2	1475.0	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179.0	1.7	0.1	3243	5330	2014
3	3181.0	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64.0	3.8	0.5	5382	8443	2014
4	3182.0	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83.0	1.9	0.4	3428	5500	2014
...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NaN	NaN	7.9	738.0	7.2	2.7	0.518	0.518	202	2003
1987	1450.0	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NaN	29	7.5	585.0	6.3	2.6	0.155	0.155	315	2003
1988	1403.0	GUMTI AT U/S SOUTH TRIPURA,TRIPURA	NaN	28	7.6	98.0	6.2	1.2	NaN	NaN	570	2003
1989	1404.0	GUMTI AT D/S SOUTH TRIPURA, TRIPURA	NaN	28	7.7	91.0	6.5	1.3	NaN	NaN	562	2003
1990	1726.0	CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA	NaN	29	7.6	110.0	5.7	1.1	NaN	NaN	546	2003

1991 rows × 12 columns

In [6]:
```
data.dtypes
```

Out[6]:
```
STATION CODE                          float64
LOCATIONS                              object
STATE                                  object
Temp                                   object
D.O. (mg/l)                           float64
PH                                    float64
CONDUCTIVITY (Âµmhos/cm)              float64
B.O.D. (mg/l)                         float64
NITRATENAN N+ NITRITENANN (mg/l)       object
FECAL COLIFORM (MPN/100ml)             object
TOTAL COLIFORM (MPN/100ml)Mean         object
year                                    int64
dtype: object
```

In [7]:
```
#conversions

data['Temp']=pd.to_numeric(data['Temp'],errors='coerce')
data['NITRATENAN N+ NITRITENANN (mg/l)']=pd.to_numeric(data['NITRATENAN N+ NITRITENANN (mg/l)'],errors='coerce')
data['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(data['TOTAL COLIFORM (MPN/100ml)Mean'],errors='coerce')
data.dtypes
```

Out[7]:
```
STATION CODE                          float64
LOCATIONS                              object
STATE                                  object
Temp                                  float64
D.O. (mg/l)                           float64
PH                                    float64
CONDUCTIVITY (Âµmhos/cm)              float64
B.O.D. (mg/l)                         float64
NITRATENAN N+ NITRITENANN (mg/l)      float64
FECAL COLIFORM (MPN/100ml)             object
TOTAL COLIFORM (MPN/100ml)Mean        float64
year                                    int64
dtype: object
```

TAKING CARE OF MISSING VALUES

In [6]:
```
#to know if there are any missing values

data.isnull().any()
```

Out[6]:
```
STATION CODE                        True
LOCATIONS                           True
STATE                               True
Temp                                True
D.O. (mg/l)                         True
PH                                  True
CONDUCTIVITY (Âµmhos/cm)            True
B.O.D. (mg/l)                       True
NITRATENAN N+ NITRITENANN (mg/l)    True
FECAL COLIFORM (MPN/100ml)          True
TOTAL COLIFORM (MPN/100ml)Mean      True
year                               False
dtype: bool
```

In [8]:
```
data['STATION CODE'].fillna(data['STATION CODE'].median(), inplace = True)
data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].median(), inplace = True)
data['PH'].fillna(data['PH'].median(),inplace = True)
data['CONDUCTIVITY (Âµmhos/cm)'].fillna(data['CONDUCTIVITY (Âµmhos/cm)'].median(), inplace = True)
data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].median(), inplace = True)
data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(data['NITRATENAN N+ NITRITENANN (mg/l)'].median(), inplace = True)
data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM (MPN/100ml)Mean'].median(), inplace = True)
```

In [9]:
```
data
```

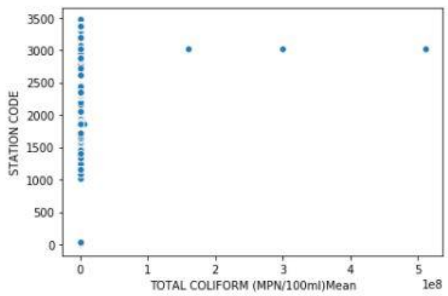
Out[9]:

	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (Âµmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393.0	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203.0	1.8965	0.100	11	27.0	2014
1	1399.0	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189.0	2.0000	0.200	4953	8391.0	2014
2	1475.0	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179.0	1.7000	0.100	3243	5330.0	2014
3	3181.0	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64.0	3.8000	0.500	5382	8443.0	2014
4	3182.0	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83.0	1.9000	0.400	3428	5500.0	2014
...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NaN	NaN	7.9	738.0	7.2	2.7000	0.518	0.518	202.0	2003
		PALAR AT VANIYAMBADI										

data visualization

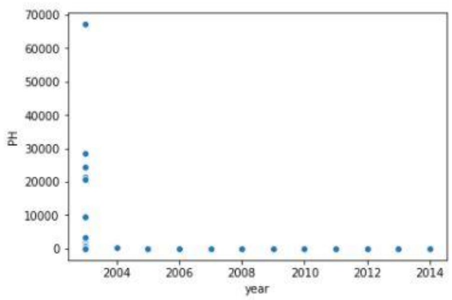
In [9]:
```
sns.scatterplot(x='TOTAL COLIFORM (MPN/100ml)Mean',y='STATION CODE',data=data)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x24abdfec108>



In [10]:
```
sns.scatterplot(x='year',y='PH',data=data)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x24abdcf94c8>

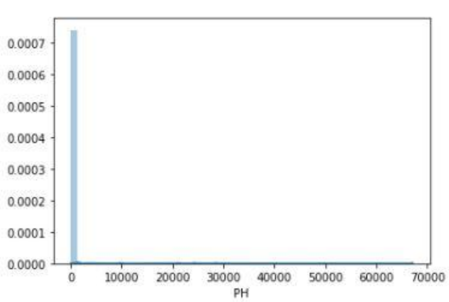


In [11]:
```
sns.scatterplot(x='TOTAL COLIFORM (MPN/100ml)Mean',y='LOCATIONS',data=data)
```

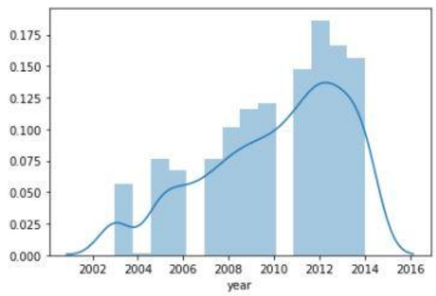
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x24abdd9cb88>

In [12]:
```
sns.distplot(data['PH'])
```

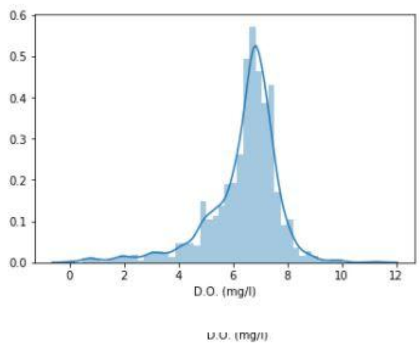
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x24abe60c348>



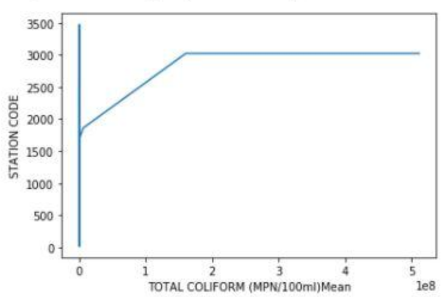
In [13]:
```
sns.distplot(data['year'])
```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x24abe651248>



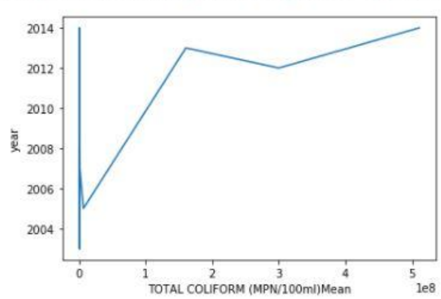
In [14]:
```
sns.distplot(data['D.O. (mg/l)'])
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x24abe843c08>



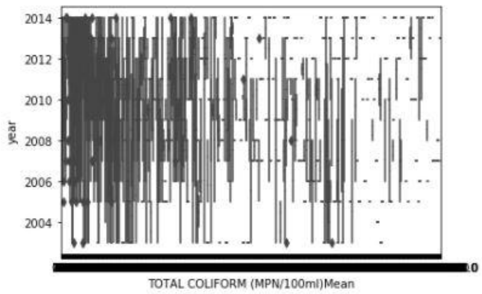
In [15]:
```
sns.lineplot(x="TOTAL COLIFORM (MPN/100ml)Mean",y="STATION CODE",data=data)
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x24abe88dac8>



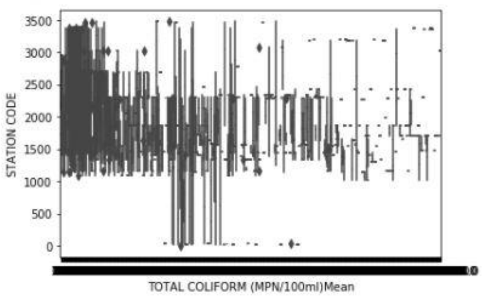
In [16]:
```
sns.lineplot(x="TOTAL COLIFORM (MPN/100ml)Mean",y="year",data=data)
```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x24abe963408>



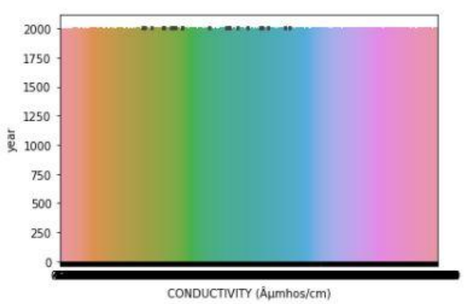
```
In [17]: sns.boxplot(x="TOTAL COLIFORM (MPN/100ml)Mean",y="year",data=data)

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x24abfa6cd88>
```



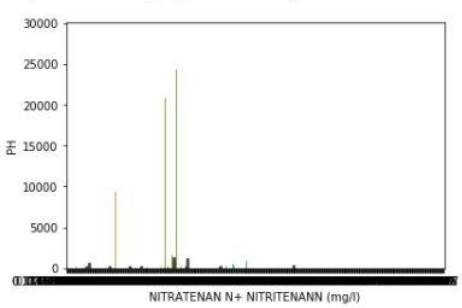

```
In [18]: sns.boxplot(x="TOTAL COLIFORM (MPN/100ml)Mean",y="STATION CODE",data=data)

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x24ac806ea88>
```




```
In [19]: sns.barplot(x="CONDUCTIVITY (Âµmhos/cm)",y="year",data=data)

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x24ab7e93908>
```

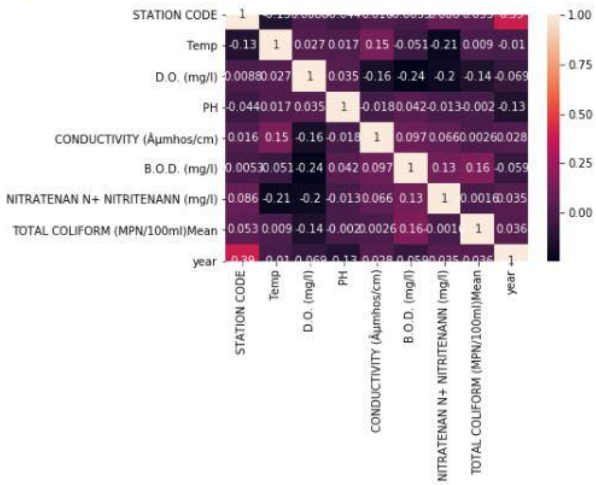



```
In [20]: sns.barplot(x="NITRATENAN N+ NITRITENANN (mg/l)",y="PH",data=data)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x24ad1afaac8>
```



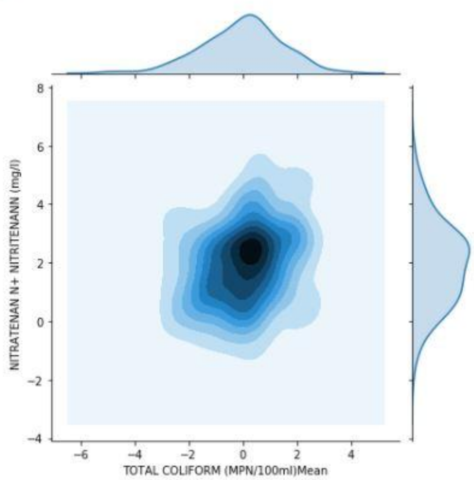
```
In [21]: sns.heatmap(data.corr(),annot=True)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24ad53bbc08>



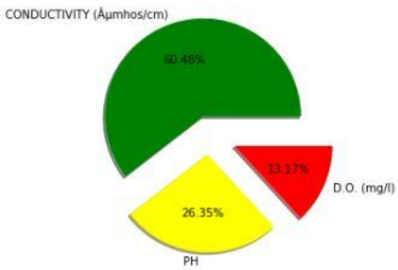

```
In [22]: mean, cov = [0, 2], [(2, .4), (.4, 2)]
         dataset = np.random.multivariate_normal(mean, cov, 200)
         df = pd.DataFrame(dataset, columns=["TOTAL COLIFORM (MPN/100ml)Mean", "NITRATENAN N+ NITRITENANN (mg/l)"])
```

```
In [23]: sns.jointplot(x="TOTAL COLIFORM (MPN/100ml)Mean", y="NITRATENAN N+ NITRITENANN (mg/l)", data=df, kind="kde"
         );
```

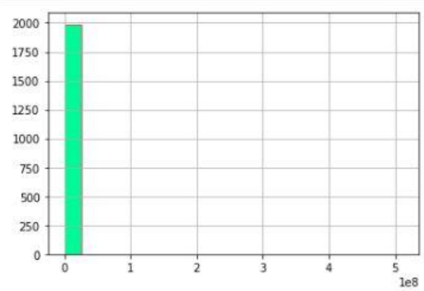


In [24]:
```python
labels=['CONDUCTIVITY (Âµmhos/cm)','PH','D.O. (mg/l)']
size=[9179,4000,1999]
colors = ["green", "yellow", "red"]
explode = (0.2,0.2,0.28)
plt.pie(size,labels=labels,explode=explode,colors=colors,autopct='%1.2f%%',shadow=True)
```

Out[24]:
```
([<matplotlib.patches.Wedge at 0x24ad5669408>,
  <matplotlib.patches.Wedge at 0x24ad566f5c8>,
  <matplotlib.patches.Wedge at 0x24ad5674a08>],
 [Text(-0.42015317992371914, 1.2302322160470303, 'CONDUCTIVITY (Âµmhos/cm)'),
  Text(-0.10992152078653307, -1.2953444558371245, 'PH'),
  Text(1.2635497499077637, -0.5548351372326988, 'D.O. (mg/l)')],
 [Text(-0.258555803029981, 0.7570659791058647, '60.48%'),
  Text(-0.06764401279171264, -0.7971350497459228, '26.35%'),
  Text(0.8057418695064, -0.353807913597663, '13.17%')])
```

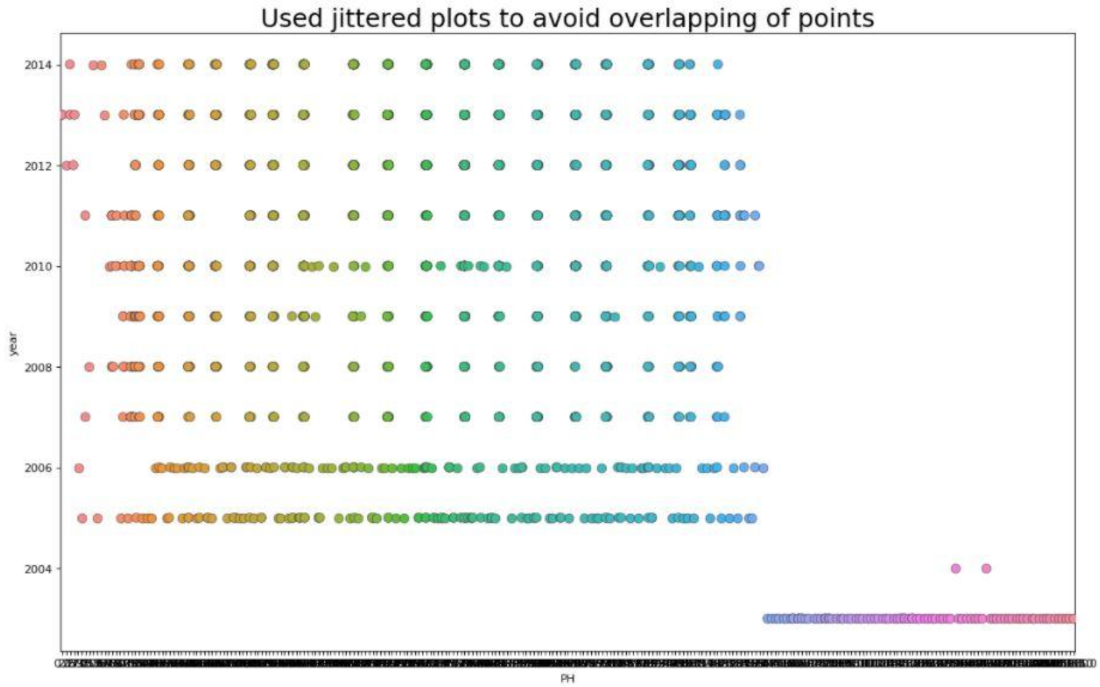


In [25]:
```python
data['TOTAL COLIFORM (MPN/100ml)Mean'].hist(bins=20,color='mediumspringgreen',edgecolor='indianred')
plt.show()
```



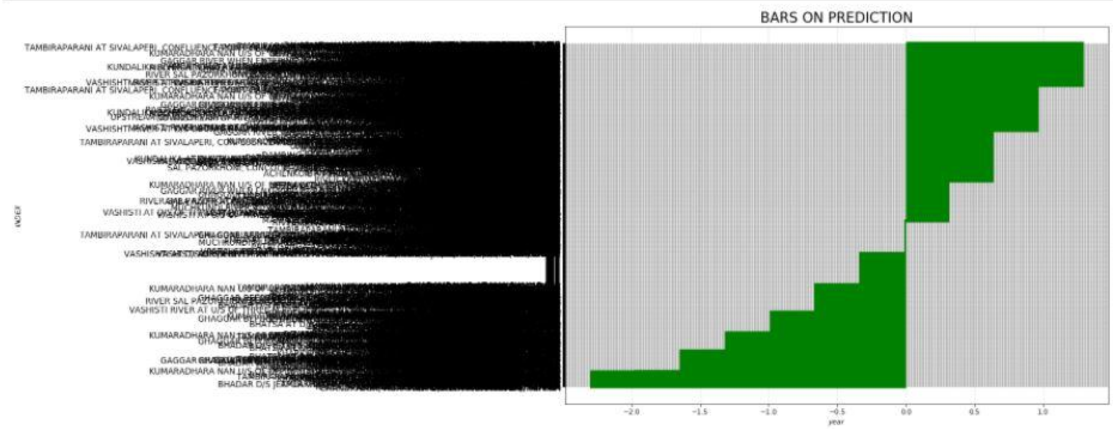
In [26]:
```python
sns.factorplot(x = 'PH',data = data,kind = 'count',hue='LOCATIONS',size=6,aspect=.9)
plt.show()
```

In [27]:
```
# Import Data
df = pd.read_csv("dataset.csv")

# Draw Stripplot
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
sns.stripplot(df.PH, df.year, jitter=0.25, size=8, ax=ax, linewidth=.5)

# Decorations
plt.title('Used jittered plots to avoid overlapping of points', fontsize=22)
plt.show()
```



...

In [28]:
```python
# Prepare Data
df = pd.read_csv("dataset.csv")
x = df.loc[:, ['year']]
df['mpg_z'] = (x - x.mean())/x.std()
df['colors'] = ['red' if x < 0 else 'green' for x in df['year']]
df.sort_values('mpg_z', inplace=True)
df.reset_index(inplace=True)

# Draw plot
plt.figure(figsize=(14,10), dpi= 80)
plt.hlines(y=df.index, xmin=0, xmax=df.mpg_z, color=df.colors, alpha=0.4, linewidth=5)

# Decorations
plt.gca().set(ylabel='$INDEX$', xlabel='$year$')
plt.yticks(df.index, df.LOCATIONS, fontsize=12)
plt.title('BARS ON PREDICTION', fontdict={'size':20})
plt.grid(linestyle='--', alpha=0.5)
plt.show()
```



In [10]:
```python
#INITIALIZATION

start=0
end=1993
station=data.iloc [start:end ,0]
location=data.iloc [start:end ,1]
state=data.iloc [start:end ,2]
do= data.iloc [start:end ,4].astype(np.float64)
value=0
ph = data.iloc[ start:end,5]
co = data.iloc [start:end ,6].astype(np.float64)

year=data.iloc[start:end,11]
tc=data.iloc [2:end ,10].astype(np.float64)


bod = data.iloc [start:end ,7].astype(np.float64)
na= data.iloc [start:end ,8].astype(np.float64)
na.dtype
```

Out[10]: dtype('float64')

In [11]:
```python
data
```

[illegible]	[illegible]	BRIDGE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	3182.0	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83.0	1.9000	0.400	3428	5500.0	2014
...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NaN	NaN	7.9	738.0	7.2	2.7000	0.518	0.518	202.0	2003
1987	1450.0	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NaN	29.0	7.5	585.0	6.3	2.6000	0.155	0.155	315.0	2003
1988	1403.0	GUMTI AT U/S SOUTH TRIPURA,TRIPURA	NaN	28.0	7.6	98.0	6.2	1.2000	0.516	NaN	570.0	2003
1989	1404.0	GUMTI AT D/S SOUTH TRIPURA, TRIPURA	NaN	28.0	7.7	91.0	6.5	1.3000	0.516	NaN	562.0	2003
1990	1726.0	CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA	NaN	29.0	7.6	110.0	5.7	1.1000	0.516	NaN	546.0	2003

1991 rows × 12 columns

```
In [12]: data=pd.concat([station,do,ph,co,bod,na,tc],axis=1)
         data. columns = ['station','do','ph','co','bod','na','tc']
```

```
In [13]: #calulation of Ph
         data['npH']=data.ph.apply(lambda x: (100 if (8.5>=x>=7)
                                          else(80 if  (8.6>=x>=8.5) or (6.9>=x>=6.8)
                                               else(60 if (8.8>=x>=8.6) or (6.8>=x>=6.7)
                                                   else(40 if (9>=x>=8.8) or (6.7>=x>=6.5)
                                                       else 0)))))
```

```
In [14]: #calculation of dissolved oxygen
         data['ndo']=data.do.apply(lambda x:(100 if (x>=6)
                                          else(80 if  (6>=x>=5.1)
                                               else(60 if (5>=x>=4.1)
                                                   else(40 if (4>=x>=3)
                                                       else 0)))))
```

```
In [15]: #calculation of total coliform
         data['nco']=data.tc.apply(lambda x:(100 if (5>=x>=0)
                                          else(80 if  (50>=x>=5)
                                               else(60 if (500>=x>=50)
                                                   else(40 if (10000>=x>=500)
                                                       else 0)))))
```

```
In [16]: #calc of B.D.O
         data['nbdo']=data.bod.apply(lambda x:(100 if (3>=x>=0)
                                          else(80 if  (6>=x>=3)
                                               else(60 if (80>=x>=6)
                                                   else(40 if (125>=x>=80)
                                                       else 0)))))
```

```
In [17]: #calculation of electrical conductivity
         data['nec']=data.co.apply(lambda x:(100 if (75>=x>=0)
                                          else(80 if  (150>=x>=75)
                                               else(60 if (225>=x>=150)
                                                   else(40 if (300>=x>=225)
                                                       else 0)))))
```

```
In [18]: #Calulation of nitrate
         data['nna']=data.na.apply(lambda x:(100 if (20>=x>=0)
                                          else(80 if  (50>=x>=20)
                                               else(60 if (100>=x>=50)
                                                   else(40 if (200>=x>=100)
                                                       else 0)))))
```

```
In [19]: #Calculation of water quality index
         data['wph']=data.npH * 0.165
         data['wdo']=data.ndo * 0.281
         data['wbdo']=data.nbdo * 0.234
         data['wec']=data.nec* 0.009
         data['wna']=data.nna * 0.028
         data['wco']=data.nco * 0.281
         data['wqi']=data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
         data
```

Out[19]:

	station	do	ph	co	bod	na	tc	npH	ndo	nco	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco	wqi
0	1393.0	6.7	7.5	203.0	1.8965	0.100	NaN	100	100	0	100	60	100	16.5	28.10	23.40	0.54	2.8	0.00	71.34
1	1399.0	5.7	7.2	189.0	2.0000	0.200	NaN	100	80	0	100	60	100	16.5	22.48	23.40	0.54	2.8	0.00	65.72
2	1475.0	6.3	6.9	179.0	1.7000	0.100	5330.0	80	100	40	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	3181.0	5.8	6.9	64.0	3.8000	0.500	8443.0	80	80	40	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	3182.0	5.8	7.3	83.0	1.9000	0.400	5500.0	100	80	40	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	7.9	738.0	7.2	2.7000	0.518	202.0	0	100	60	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1987	1450.0	7.5	585.0	6.3	2.6000	0.155	315.0	0	100	60	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1988	1403.0	7.6	98.0	6.2	1.2000	0.516	570.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1989	1404.0	7.7	91.0	6.5	1.3000	0.516	562.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1990	1726.0	7.6	110.0	5.7	1.1000	0.516	546.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44

1991 rows × 20 columns

In [21]:
```
data['tc'].fillna(data['tc'].median(), inplace = True)
```

In [22]:
```
data
```

Out[22]:

	station	do	ph	co	bod	na	tc	npH	ndo	nco	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco	wqi
0	1393.0	6.7	7.5	203.0	1.8965	0.100	468.0	100	100	0	100	60	100	16.5	28.10	23.40	0.54	2.8	0.00	71.34
1	1399.0	5.7	7.2	189.0	2.0000	0.200	468.0	100	80	0	100	60	100	16.5	22.48	23.40	0.54	2.8	0.00	65.72
2	1475.0	6.3	6.9	179.0	1.7000	0.100	5330.0	80	100	40	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	3181.0	5.8	6.9	64.0	3.8000	0.500	8443.0	80	80	40	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	3182.0	5.8	7.3	83.0	1.9000	0.400	5500.0	100	80	40	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	7.9	738.0	7.2	2.7000	0.518	202.0	0	100	60	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1987	1450.0	7.5	585.0	6.3	2.6000	0.155	315.0	0	100	60	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1988	1403.0	7.6	98.0	6.2	1.2000	0.516	570.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1989	1404.0	7.7	91.0	6.5	1.3000	0.516	562.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1990	1726.0	7.6	110.0	5.7	1.1000	0.516	546.0	0	100	40	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44

1991 rows × 20 columns

In [24]:
```
#dropping of columns

data=data.drop(columns=["npH","ndo","nbdo","nec","nna","wph","wdo","wbdo","wec","wna","wco","nco"],axis=1)
data
```

Out[24]:

	station	do	ph	co	bod	na	tc	wqi
0	1393.0	6.7	7.5	203.0	1.8965	0.100	468.0	71.34
1	1399.0	5.7	7.2	189.0	2.0000	0.200	468.0	65.72
2	1475.0	6.3	6.9	179.0	1.7000	0.100	5330.0	79.28
3	3181.0	5.8	6.9	64.0	3.8000	0.500	8443.0	69.34
4	3182.0	5.8	7.3	83.0	1.9000	0.400	5500.0	77.14
...	...	...	...	...	...	...	...	...
1986	1330.0	7.9	738.0	7.2	2.7000	0.518	202.0	72.06
1987	1450.0	7.5	585.0	6.3	2.6000	0.155	315.0	72.06
1988	1403.0	7.6	98.0	6.2	1.2000	0.516	570.0	66.44
1989	1404.0	7.7	91.0	6.5	1.3000	0.516	562.0	66.44
1990	1726.0	7.6	110.0	5.7	1.1000	0.516	546.0	66.44

1991 rows × 8 columns

In [25]:
```
data.head(10)
```

Out[25]:

	station	do	ph	co	bod	na	tc	wqi
0	1393.0	6.7	7.5	203.0	1.8965	0.1	468.0	71.34
1	1399.0	5.7	7.2	189.0	2.0000	0.2	468.0	65.72
2	1475.0	6.3	6.9	179.0	1.7000	0.1	5330.0	79.28
3	3181.0	5.8	6.9	64.0	3.8000	0.5	8443.0	69.34
4	3182.0	5.8	7.3	83.0	1.9000	0.4	5500.0	77.14
5	1400.0	5.5	7.4	81.0	1.5000	0.1	4049.0	77.14
6	1476.0	6.1	6.7	308.0	1.4000	0.3	5672.0	75.44
7	3185.0	6.4	6.7	414.0	1.0000	0.2	9423.0	75.44
8	3186.0	6.4	7.6	305.0	2.2000	0.1	4990.0	82.04
9	3187.0	6.3	7.6	77.0	2.3000	0.1	4301.0	82.76

In [26]:
```
x=data.iloc[:,0:7].values #x contains inputs
y=data.iloc[:,7].values #y contains output
```

In [27]:
```
x
```

Out[27]:
```
array([[1.3930e+03, 6.7000e+00, 7.5000e+00, ..., 1.8965e+00, 1.0000e-01,
        4.6800e+02],
       [1.3990e+03, 5.7000e+00, 7.2000e+00, ..., 2.0000e+00, 2.0000e-01,
        4.6800e+02],
       [1.4750e+03, 6.3000e+00, 6.9000e+00, ..., 1.7000e+00, 1.0000e-01,
        5.3300e+03],
       ...,
       [1.4030e+03, 7.6000e+00, 9.8000e+01, ..., 1.2000e+00, 5.1600e-01,
        5.7000e+02],
       [1.4040e+03, 7.7000e+00, 9.1000e+01, ..., 1.3000e+00, 5.1600e-01,
        5.6200e+02],
       [1.7260e+03, 7.6000e+00, 1.1000e+02, ..., 1.1000e+00, 5.1600e-01,
        5.4600e+02]])
```

In [28]:
```
y
```

Out[28]:
```
array([71.34, 65.72, 79.28, ..., 66.44, 66.44, 66.44])
```

In [29]:
```
x.shape
```

Out[29]:
```
(1991, 7)
```

In [30]:
```
y.shape
```

Out[30]:
```
(1991,)
```

SPLITTING OF DATA INTO TEST AND TRAIN ¶

In [32]:
```
#splitting the dataset into test and train

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)
```

In [33]:
```
x_test.shape
```

Out[33]:
```
(399, 7)
```

In [34]:
```
y_test.shape
```

Out[34]:
```
(399,)
```

In [35]:
```
x_train.shape
```

Out[35]:
```
(1592, 7)
```

In [36]:
```
y_train.shape
```

Out[36]:
```
(1592,)
```

MODEL BUILDING USING RANDOM FOREST REGRESSOR

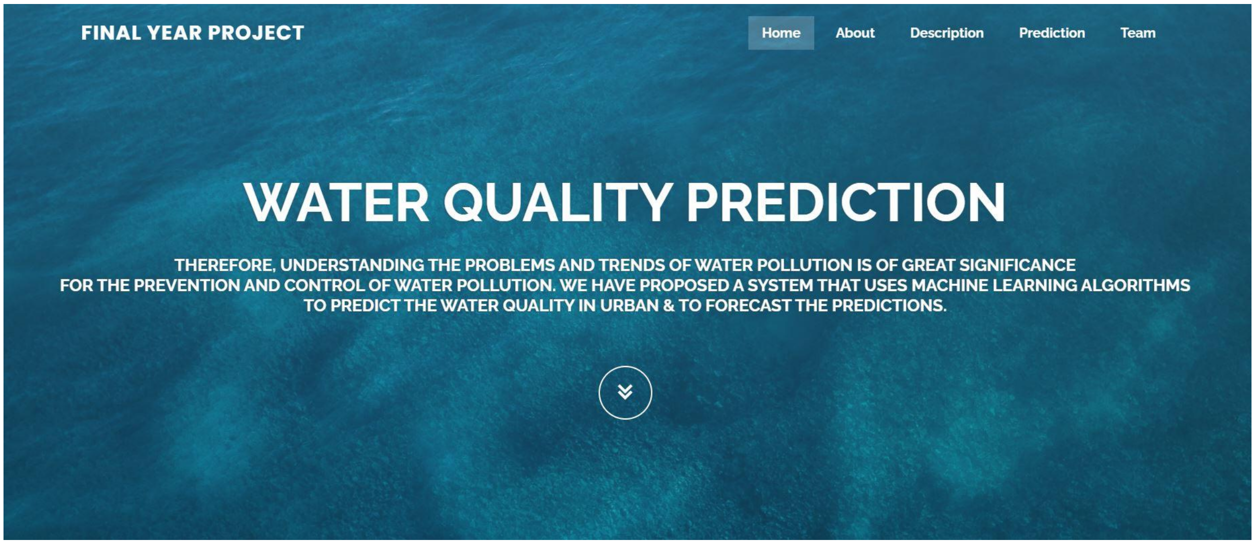
"TRAINING"

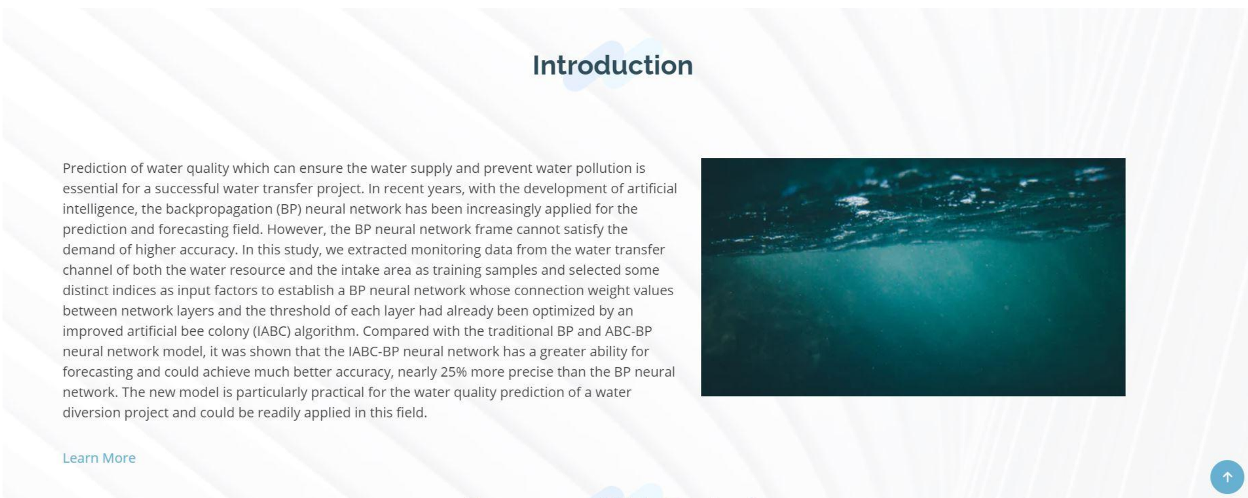
```
In [37]: from sklearn.ensemble import RandomForestRegressor
         rdr=RandomForestRegressor(n_estimators=10,random_state=0)
         rdr.fit(x_train,y_train)

Out[37]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                               max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=10,
                               n_jobs=None, oob_score=False, random_state=0, verbose=0,
                               warm_start=False)

In [38]: yr=rdr.predict(x_test)

In [39]: yr

Out[39]: array([82.94 , 87.66 , 55.82 , 73.04 , 76.168, 50.994, 66.44 , 88.362,
                82.04 , 76.106, 79.64 , 52.622, 87.66 , 73.04 , 77.414, 72.512,
                78.3  , 58.756, 88.362, 70.836, 87.66 , 83.214, 82.04 , 66.192,
                81.95 , 94.18 , 82.04 , 67.646, 82.94 , 87.66 , 83.52 , 88.38 ,
                82.76 , 89.238, 50.2  , 88.38 , 83.7  , 36.324, 87.66 , 88.38 ,
                82.04 , 93.28 , 82.4  , 79.226, 94.18 , 55.82 , 82.706, 72.006,
                74.896, 72.06 , 82.94 , 72.914, 82.04 , 83.7  , 33.376, 83.96 ,
                84.36 , 94.18 , 78.858, 61.44 , 77.988, 67.932, 64.156, 94.18 ,
                82.6  , 83.7  , 85.152, 66.44 , 83.52 , 70.058, 88.38 , 79.424,
                78.776, 76.146, 83.394, 82.476, 61.44 , 82.314, 71.16 , 70.266,
                77.988, 77.32 , 77.868, 82.94 , 66.386, 88.2  , 99.026, 88.924,
                88.56 , 82.94 , 93.694, 99.746, 76.42 , 69.274, 87.66 , 69.944,
                87.66 , 76.42 , 32.706, 79.64 , 76.162, 86.514, 82.94 , 71.332,
                51.262, 81.614, 78.516, 82.94 , 88.38 , 78.3  , 50.2  , 82.98 ,
                98.612, 87.66 , 82.94 , 82.76 , 93.28 , 82.778, 72.112, 75.782,
                71.862, 61.094, 82.94 , 73.394, 65.89 , 71.736, 86.874, 78.282,
                73.484, 93.658, 82.94 , 75.834, 83.7  , 72.914, 82.598, 78.08 ,
                83.7  , 83.52 , 93.28 , 94.18 , 79.46 , 76.34 , 49.596, 88.38 ,
                78.3  , 69.912, 87.66 , 67.21 , 77.64 , 82.94 , 82.598, 82.58 ,
                88.834, 83.7  , 73.238, 45.716, 63.172, 82.94 , 79.64 , 88.38 ,
                65.168, 76.892, 72.06 , 72.06 , 93.604, 88.816, 46.328, 72.06 ,
                83.376, 87.66 , 82.04 , 64.568, 94.18 , 65.48 , 77.302, 78.462,
                88.2  , 72.86 , 55.108, 64.572, 72.342, 62.192, 93.64 , 68.564,

In [40]: from sklearn.metrics import r2_score

In [41]: accu=r2_score(y_test,yr)
         accu

Out[41]: 0.9721948019899869

In [43]: import pickle #saving
         pickle.dump(rdr,open('rdr1.pkl','wb'))
```

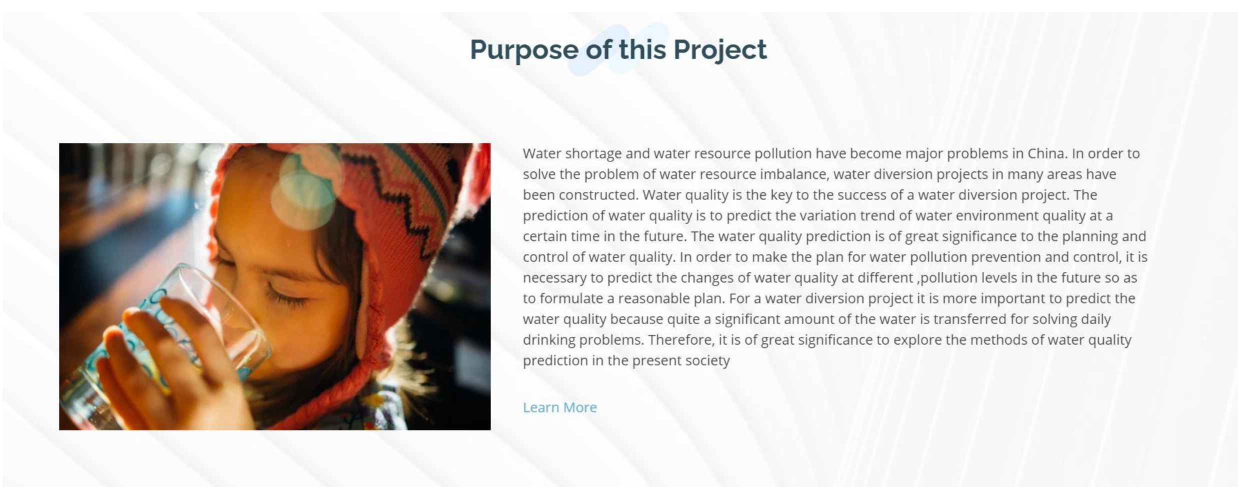
Application Building:

```
import pickle


app = Flask(__name__)

@app.route('/')
def home() :
    return render_template("index.html")

@app.route('/login',methods = ['POST'])
def login() :

    station = request.form["loc"]
    do = request.form["do"]
    ph = request.form["ph"]
    co = request.form["co"]
    bod = request.form["bod"]
    na = request.form["na"]
    tc = request.form["tc"]
    total = [[float(station),float(do),float(ph),float(co),float(bod),float(na),float(tc)]]
    model = pickle.load(open('rdr1.pkl','rb'))
    y_pred = model.predict(total)
    y_pred =y_pred[[0]]
    if(y_pred >= 95 and y_pred <= 100) :
        return render_template("index.html",showcase = 'Excellent,The predicted value is '+ str(y_pred))
    elif(y_pred >= 89 and y_pred <= 94) :
        return render_template("index.html",showcase = 'Very good,The predicted value is '+str(y_pred))
    elif(y_pred >= 80 and y_pred <= 88) :
        return render_template("index.html",showcase = 'Good,The predicted value is'+str(y_pred))
    elif(y_pred >= 65 and y_pred <= 79) :
        return render_template("index.html",showcase = 'Fair,The predicted value is '+str(y_pred))
    elif(y_pred >= 45 and y_pred <= 64) :
        return render_template("index.html",showcase = 'Marginal,The predicted value is '+str(y_pred))
    else :
        return render_template("index.html",showcase = 'Poor,The predicted value is '+str(y_pred))


if __name__ == '__main__' :
    app.run(debug = True,port=5000)
```

Front End Design:







Experimental Investigation

The urban water quality prediction dataset is collected from water samples from various urban areas in India.There are 12 parameters taken in the dataset some of which are various pollutent measures columns and each column consists of the average values over a period of time.The entire dataset observations are made between 2003-2014.

Required Measurements

- Station code
- PH
- Conductivity
- D.O
- B.O.D
- Total coliform
- Nitratenan N+

Water quality constraints

Excellent: (WQI Value 95-100)
Very Good: (WQI Value 89-94)
Good: (WQI Value 80-88)
Fair: (WQI Value 65-79)
Marginal: (WQI Value 45-64)
Poor: (WQI Value 0-44)

Advantages

It's very simple to understand.
Good interpretability.
High accuracy and cheap computational cost.
Ground for more complex machine learning algorithms
Hence it's a high latency algorithm.

Disadvantages

The main disadvantage of Random forests is their complexity.
They are much harder and time-consuming to construct than decision trees.

Prediction

Enter Station Code

Enter Station code

Enter D.O | Enter PH

D.O | PH

Enter Conductivity

Conductivity

Enter B.O.D

B.O.D

Enter Nitratenen

Nitratenen

Enter Total Coliform

Total Coliform

Predict

Very good,The predicted value is [91.878]

Team

We would like to thank our project guide **Mrs M Raja Mani** , Assistant Professor, Department of CSE for her stimulating guidance and profuse assistance. We shall always cherish our association with her guidance, encouragement, and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under her guidance and constant support.

Chilamkurthi C V S Varshith
Team Lead

Kavali Srikanth
Team Member

Thakshak Sunkara
Team Member

Vanapalli Rahul
Team Member

G Rutwiz Gangadhar
Team Member

GITAM (Deemed to be) University

GIT, Visakhapatnam

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- It's very simple to understand.
- Good interpretability.
- High accuracy and cheap computational cost.
- Space complexity is very low, it just needs to save the weights at the end of training.Hence it's a high latency algorithm.
- Feature importance is generated at the time model building. With the help of hyperparameter lamba, you can handle features selection hence we can achieve dimensionality reduction.
- Ground for more complex machine learning algorithms

DISADVANTAGES:

- The main disadvantage of Random forests is their complexity. They are much harder and time-consuming to construct than decision trees.
- They also require more computational resources and are also less intuitive
- In addition, the prediction process using random forests is time-consuming than other algorithms.

APPLICATIONS:

The proposed solution to predict the quality of water using machine learning can be extended and can be used everywhere to check the quality of water. The web application product can then be used to learn from the data and predict the quality of water for any sort of values of the input variables. This proposed solution is more applicable and beneficial in India. This can extend to develop the ANFIS models, and are measured values at outfall measuring stations.

CONCLUSION:

We successfully forecasted the Water Quality Prediction using the Random Forest Regression model. Forecasting consumption in turn, at scale, could aid in the utility company forecasting demand, which is widely studied and an important problem. In this project, we discovered a dataset containing required information for predicting the quality of water at various locations with its states for Water Quality Prediction and better understood the raw data using exploratory analysis. The dataset described the quantity of various chemicals present ins water over different years. We explored and understood the dataset using a suite of line plots for series data and histogram for the data distributions. We used the new understanding of the problem to consider different framings of the prediction problem, ways the data may be prepared, and modelling methods that may be used. Finally, we chose Random Forest Regression model as it offered highest accuracy. Then we implemented our model to predict the quality of water given by various quantities like pH, D.O., Conductivity, TotalColiform, Wqi etc. using a local host web application. We have provided an end-to-end demonstration of how you can use machine learning to determine the water quality at an area based on the data of different quantities.

FUTURE SCOPE:

Among various sources of water supply, due to easy access, rivers have been used more frequently for the development of human societies. Using other water resources such as groundwater and seawater sometimes assisted with problems. So, this system can predict the quality of water whether it is fair, good, or excellent so that many people can use the water for various purposes .

BIBLIOGRAPHY / References :

https://www.kaggle.com/anbarivan/indian-water-quality-data
https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a4517