RIVER WATER QUALITY FORECASTING

A UG PROJECT PHASE-2 REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Submitted By

DEEPIKA MATETI	**20UK1A6618**
GAJULA NAGARAJU	**20UK1A6619**
KAKKERLA NATRAJ	**20UK1A6657**

Under the guidance of

Mr.'s. B.SHARADHA (Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD
BOLLIKUNTA, WARANGAL (T.S)-506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project phase-1 entitled “**RIVER WATER QUALITY FORECASTING”** Submitted by **DEEPIKA MATETI (20UK1A6618), GAJULA NAGARAJU (20UK1A6619), KAKKERLA NATRAJ (20UK1A6657)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science and Engineering (AI&ML) to Jawaharlal Nehru Technological University Hyderabad** during the academic year **2023-2024**.

Project Guide	**HOD**
Mr.’s. B.SHARADHA	**Dr. K. SHARMILA REDDY**
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA REDDY,** Head of the Department of CSE (AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG project Phase-2.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the Under Graduate Project Phase-2.

We express heartfelt thanks to the guide **Mr.'s.B.SHARADHA**, Assistant Professor, Department of CSE (AI&ML) for his constant support and giving necessary guidance for completion of this Under Graduate Project Phase-2.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

DEEPIKA MATETI	**20UK1A6618**
GAJULA NAGARAJU	**20UK1A6619**
KAKKERLA NATRAJ	**20UK1A6657**

ABSTRACT

River water quality forecasting is an essential tool for ensuring sustainable water management, environmental conservation, and safeguarding public health. It involves employing various techniques such as empirical models, machine learning, and integrated modelling systems to predict parameters influencing water quality, including pollutants, nutrients, and microbial contaminants. These methodologies utilize historical data, statistical relationships, and complex datasets to forecast water quality variations in river systems.

However, challenges persist in accurately forecasting river water quality due to data scarcity, the intricate nature of water systems, and the dynamic sources of pollutants. Limited real-time data availability on pollutants, nutrients, and microbial contaminants hinders precise forecasting. The complex interplay between natural factors and human activities further complicates modelling efforts, presenting hurdles in accurately predicting water quality variations.

Accurate river water quality forecasting holds significant implications, enabling proactive water resource management, pollution mitigation, and protection of aquatic ecosystems and public health. Timely forecasting assists in implementing interventions, adjusting wastewater treatment processes, and issuing advisories to mitigate potential risks. To enhance forecasting accuracy, addressing challenges related to data availability, model complexity, and comprehensive understanding of water quality dynamics is essential, requiring advancements in technology, improved monitoring systems, and interdisciplinary collaborations. These efforts are crucial for better environmental sustainability, protection of public health, and effective water resource management strategies

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Overview

River water quality forecasting is a multifaceted endeavour crucial for managing freshwater resources sustainably. By integrating data collection, modeling, and risk assessment, it provides insights essential for decision-making in various sectors. This forecasting involves gathering extensive data on physical and chemical parameters through monitoring stations, satellite imagery, and sensor networks. Mathematical models and algorithms are then employed to analyse this data, predicting future water quality based on factors like weather patterns, land use practices, and pollutant sources. Techniques range from statistical methods for short-term forecasts to complex simulations for long-term predictions. However, challenges such as the complexity of river ecosystems, data availability, model uncertainty, and effective communication of forecast uncertainties persist. Despite these challenges, river water quality forecasting remains indispensable for ensuring safe drinking water, protecting aquatic ecosystems, and guiding sustainable water management practices.

1.2. Purpose

The purpose of river water quality forecasting is multifaceted, serving several crucial objectives:

Resource Management: It assists in the efficient management of freshwater resources by providing insights into water quality trends, enabling stakeholders to allocate resources effectively and plan for future needs.

Environmental Protection: By predicting changes in water quality, forecasting helps identify potential threats to aquatic ecosystems, allowing for proactive measures to mitigate pollution and preserve biodiversity.

Public Health: Forecasting enables the early detection of water quality issues that may pose risks to public health, such as contamination from pollutants or harmful algal blooms, thus supporting efforts to ensure safe drinking water supplies.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

Data Availability and Quality: Adequate and timely data collection is crucial for accurate forecasting. However, challenges such as insufficient monitoring stations, gaps in data coverage, and variations in data quality can limit the effectiveness of forecasting models.

Model Complexity and Uncertainty: River ecosystems are complex systems influenced by numerous factors, including weather patterns, land use practices, and pollutant sources. Modelling this complexity introduces uncertainties that can affect the reliability of forecasts, requiring robust validation and calibration processes.

Emerging Contaminants and Unknown Risks: The proliferation of emerging contaminants, such as pharmaceuticals, microplastics, and novel chemicals, poses challenges for traditional forecasting methods that may not account for these evolving threats adequately.

Climate Change Impacts: Climate change exacerbates existing challenges by altering precipitation patterns, increasing the frequency and intensity of extreme weather events, and influencing water temperature and nutrient cycles. Forecasting models must adapt to these changing conditions to provide accurate predictions.

Resource Constraints: Limited financial resources, technical expertise, and infrastructure can hinder the implementation and maintenance of comprehensive monitoring networks and sophisticated modelling techniques, particularly in resource-constrained regions.

2.2. EXISTING SOLUTION:

1.Integrated Monitoring Systems: Implementing comprehensive monitoring systems that collect data on physical, chemical, and biological parameters at multiple points along rivers. This includes traditional monitoring stations, remote sensing technologies, and real-time sensor networks to provide continuous data coverage.

2.Hydrological and Water Quality Models: Developing and refining hydrological models, such as HEC-RAS or SWAT, and water quality models like QUAL2K or CE-QUAL-W2, to simulate river flow dynamics and pollutant transport. These models integrate data on land use, precipitation, temperature, and pollutant sources to forecast water quality.

3.Data Assimilation Techniques: Employing data assimilation techniques to integrate observed data with model simulations, improving the accuracy of forecasts by updating model parameters and initial conditions in real-time.

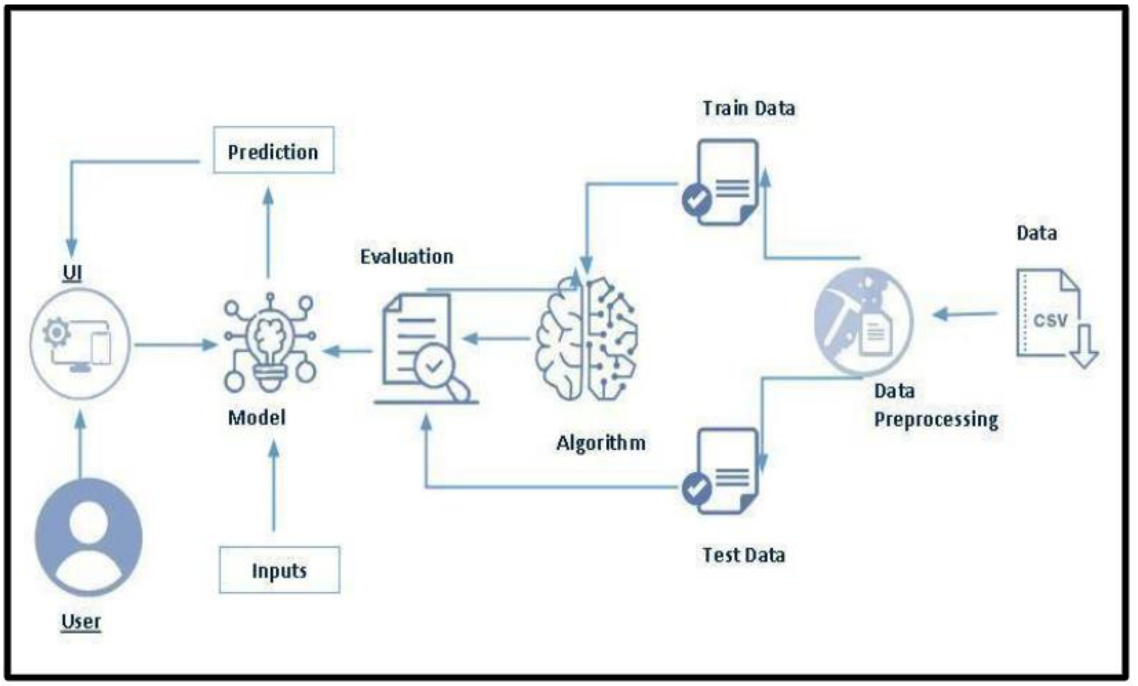
4.Predictive Analytics and Machine Learning: Utilizing predictive analytics and machine learning algorithms to analyse historical data, identify patterns, and predict future water quality trends. These techniques can enhance forecasting accuracy and provide insights into complex relationships within river ecosystems.

5. **Early Warning Systems:** Developing early warning systems that utilize forecasting models to predict and alert stakeholders to potential water quality issues, such as algal blooms, contaminant spills, or excessive nutrient concentrations, allowing for proactive management and response.

6. **Community Engagement and Citizen Science:** Engaging local communities and citizen scientists in water quality monitoring efforts through outreach programs, volunteer initiatives, and citizen science projects. This fosters collaboration, data sharing, and community empowerment in addressing water quality challenges.

3. THEORETICAL ANALYSIS

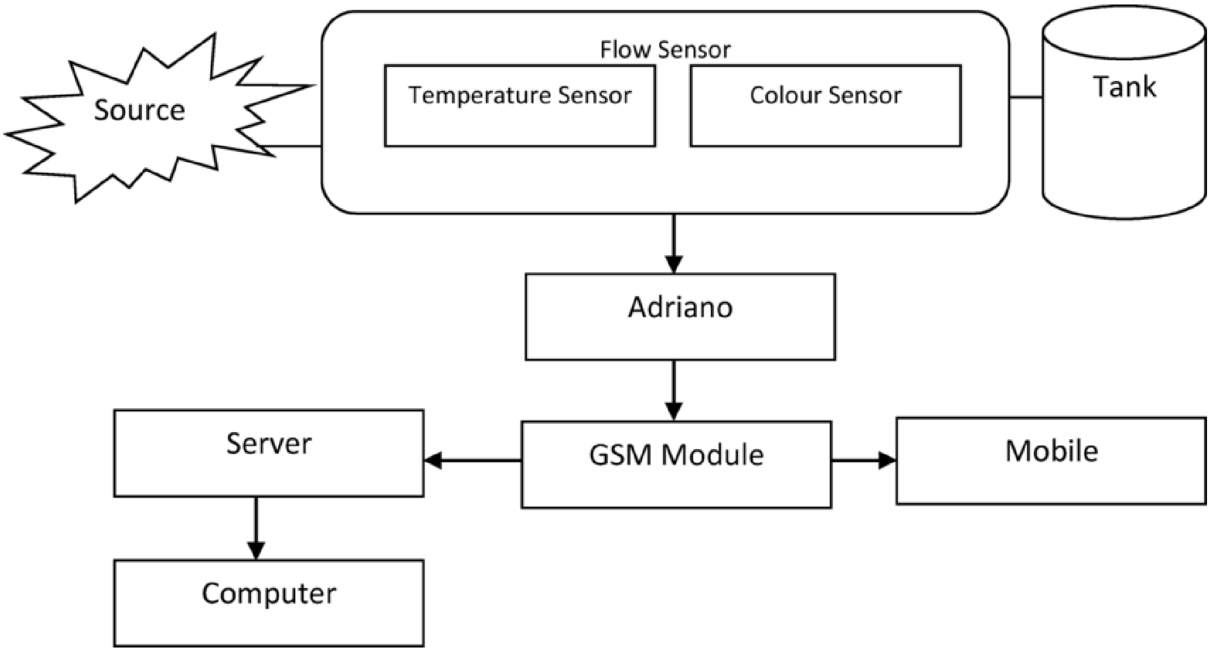
3.1. Technical Architecture



3.2. SOFTWARE REQUIREMENT

- Spyder IDE
- Python (pandas, NumPy,)
- HTML
- Flask
- OpenCV
- Imutils
- Geopy
- Requests
- Anaconda Distribution
- Windows OS

3.3. HARDWARE REQUIREMENTS



4. PROJECT STRUCTURE

Create the Project folder which contains files as shown below

.ipynb_checkpoints	9/11/2023 3:29 AM	File folder	
templates	9/15/2023 2:45 AM	File folder	
PB_All_2000_2021	9/7/2023 1:37 AM	OpenOffice.org X...	158 KB
PB_stations	9/7/2023 1:37 AM	OpenOffice.org X...	2 KB
Project3_app	9/15/2023 2:47 AM	PY File	2 KB
standard_scaler	9/15/2023 2:31 AM	PKL File	2 KB
Water_Quality_Forecasting	9/15/2023 2:34 AM	IPYNB File	4,979 KB
xgb_regressor_model	9/15/2023 1:19 AM	PKL File	79 KB

We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

- xgb_regressor_model is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Data-set used
- The Notebook file contains procedure for building the model.

5. PROPOSED METHODOLOGY

1. Data Collection and Preprocessing:

Gather historical data on physical, chemical, and biological parameters from monitoring stations, satellite imagery, and sensor networks.

Preprocess the data to remove outliers, correct errors, and ensure consistency across datasets.

2.Feature Selection and Engineering:

Identify relevant features that influence water quality, such as weather variables, land use patterns, pollutant sources, and river flow characteristics.

Engineer new features or transform existing ones to capture relationships and trends within the data.

3. Model Development:

Select appropriate modelling techniques, such as hydrological models, water quality models, statistical methods, or machine learning algorithms.

Develop models that integrate data on river flow, meteorological conditions, land use, and pollutant loads to simulate water quality dynamics.

Validate the models using historical data and calibration techniques to ensure accuracy and reliability.

4. Data Assimilation and Model Integration:

Implement data assimilation techniques to integrate observed data with model simulations, updating model parameters and initial conditions in real-time.

Integrate multiple models, such as hydrological and water quality models, to capture the complex interactions between different components of the river ecosystem.

5. Forecast Generation:

Use the developed models to generate forecasts of water quality parameters, such as nutrient concentrations, dissolved oxygen levels, and pollutant loads.

Produce short-term forecasts for immediate decision-making and long-term projections for planning and policy development.

6. Uncertainty Quantification and Risk Assessment:

Quantify uncertainties associated with the forecasts using probabilistic methods, sensitivity analyses, and ensemble modelling techniques.

Assess the risks associated with different scenarios, such as extreme weather events, pollution incidents, or changes in land use practices.

7. Validation and Performance Evaluation:

Validate the forecasts against independent datasets and observational data to assess their accuracy and reliability.

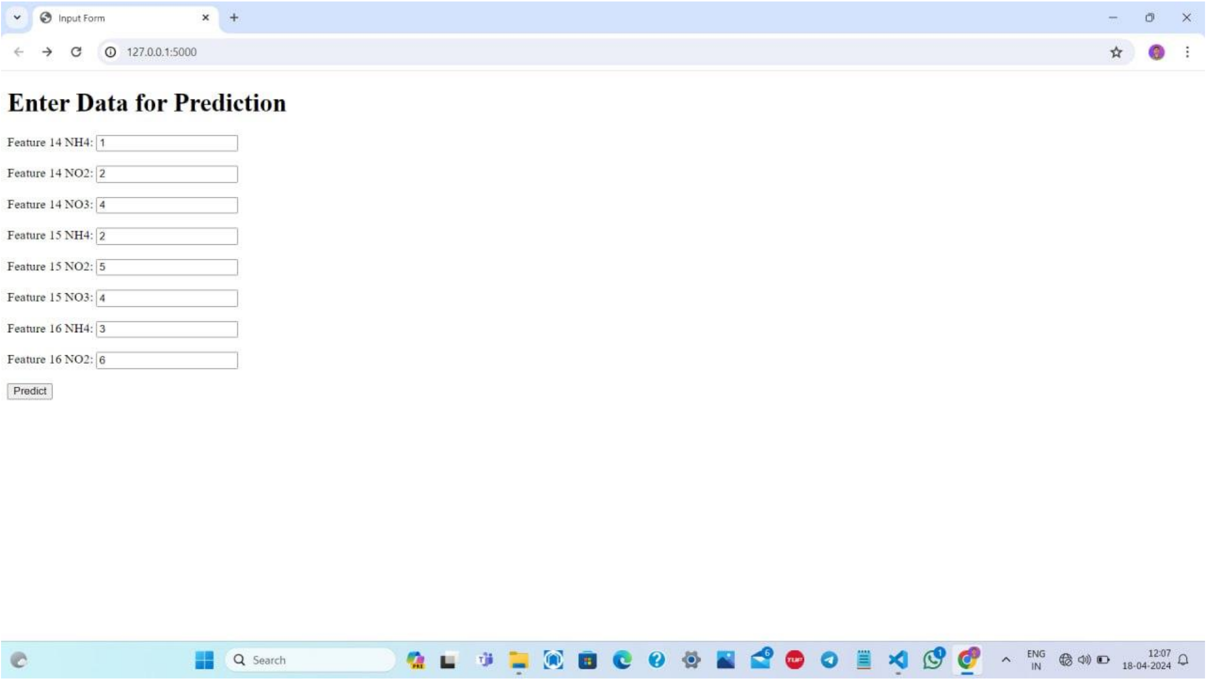
Evaluate the performance of the forecasting system using metrics such as mean squared error, correlation coefficient, and skill scores.

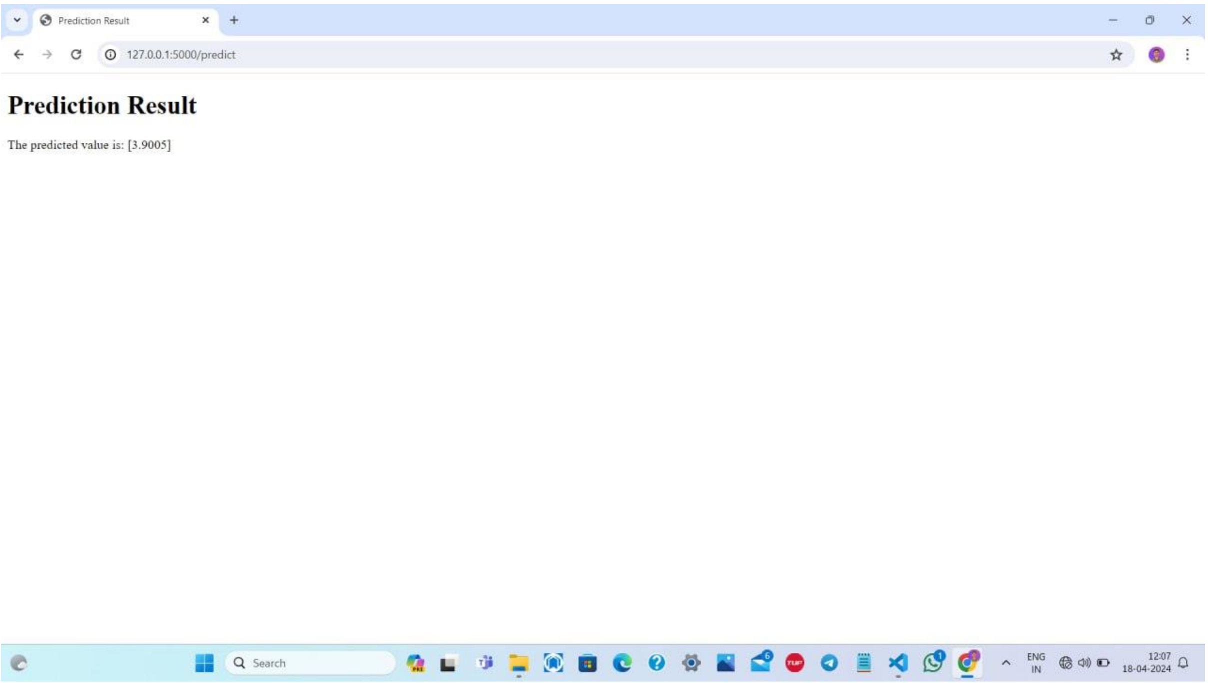
8.Communication and Stakeholder Engagement:

Communicate forecast results, uncertainties, and risks to decision-makers, stakeholders, and the public using clear and accessible formats.

Engage stakeholders in the forecasting process through workshops, meetings, and outreach activities to solicit feedback and ensure relevance to end-users.

RESULTS


Input Form
127.0.0.1:5000
Enter Data for Prediction
Feature 14 NH4: 1
Feature 14 NO2: 2
Feature 14 NO3: 4
Feature 15 NH4: 2
Feature 15 NO2: 5
Feature 15 NO3: 4
Feature 16 NH4: 3
Feature 16 NO2: 6
Predict
Search
ENG
IN
12:07
18-04-2024


Prediction Result
127.0.0.1:5000/predict
Prediction Result
The predicted value is: [3.9005]
Search
ENG
IN
12:07
18-04-2024

6. FUTURE SCOPE

In the future, the scope of river water quality forecasting is poised for significant advancements across various fronts. The integration of emerging technologies, such as remote sensing and IoT devices, promises to revolutionize data collection and monitoring, providing real-time insights with unprecedented spatial and temporal resolution. Moreover, predictive modelling techniques will undergo refinement, leveraging advancements in machine learning and data-driven approaches to enhance accuracy and reliability, especially in capturing complex ecological dynamics. Climate change adaptation will become paramount, with forecasting systems evolving to incorporate the anticipated impacts of climate variability on water quality, enabling proactive mitigation measures. Additionally, there's a growing recognition of the need to integrate social, economic, and institutional factors into forecasting models, ensuring that decisions align with broader environmental and socioeconomic objectives. Predictive analytics will also extend its reach to anticipate the behaviour of emerging contaminants, addressing evolving threats to water quality and human health. Improved data visualization and communication strategies will facilitate the dissemination of forecast results and uncertainties to diverse stakeholders, fostering informed decision-making and community engagement. Furthermore, citizen science initiatives and participatory monitoring programs will play an increasingly pivotal role in expanding monitoring networks and enhancing public awareness of freshwater conservation. Finally, interdisciplinary research and collaboration will drive innovation, enabling stakeholders to address complex challenges and advance the effectiveness of river water quality forecasting efforts collectively. Embracing these opportunities holds the promise of a future where river water quality forecasting serves as a cornerstone for sustainable water management in a rapidly changing world.

7. CONCLUSION

In conclusion, the future of river water quality forecasting holds immense promise for advancing our understanding of freshwater ecosystems and enhancing their management and conservation. By integrating cutting-edge technologies, refining predictive modelling techniques, and embracing interdisciplinary collaboration, we can address emerging challenges and improve the accuracy and reliability of forecasts. Climate change adaptation, social integration, and citizen engagement will be central themes, ensuring that forecasting efforts are holistic, inclusive, and aligned with broader environmental and socioeconomic objectives. As we navigate the complexities of a changing world, river water quality forecasting will remain a vital tool for safeguarding water resources, protecting ecosystems, and promoting public health. By seizing these opportunities and working together, we can build a future where clean and sustainable rivers thrive, supporting the well-being of both current and future generations.

Appendix

Model building:

1)Dataset

2)Google Collab and VS code Application Building

1.HTML file (index file, predict file)

2.CSS file

3.models in pickle format

SOURCE CODE :-

Index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Input Form</title>
</head>

<body>
    <h1>Enter Data for Prediction</h1>
    <form action="/predict" method="POST">
        <label for="14_NH4">Feature 14 NH4:</label>
        <input type="number" id="14_NH4" name="14_NH4" required><br><br>

        <label for="14_NO2">Feature 14 NO2:</label>
        <input type="number" id="14_NO2" name="14_NO2" required><br><br>

        <label for="14_NO3">Feature 14 NO3:</label>
        <input type="number" id="14_NO3" name="14_NO3" required><br><br>

        <label for="15_NH4">Feature 15 NH4:</label>
        <input type="number" id="15_NH4" name="15_NH4" required><br><br>

        <label for="15_NO2">Feature 15 NO2:</label>
        <input type="number" id="15_NO2" name="15_NO2" required><br><br>

        <label for="15_NO3">Feature 15 NO3:</label>
        <input type="number" id="15_NO3" name="15_NO3" required><br><br>
```

```
        <label for="16_NH4">Feature 16 NH4:</label>
        <input type="number" id="16_NH4" name="16_NH4" required><br><br>

        <label for="16_NO2">Feature 16 NO2:</label>
        <input type="number" id="16_NO2" name="16_NO2" required><br><br>

        <input type="submit" value="Predict">
    </form>
</body>
</html>
```

Result.html:

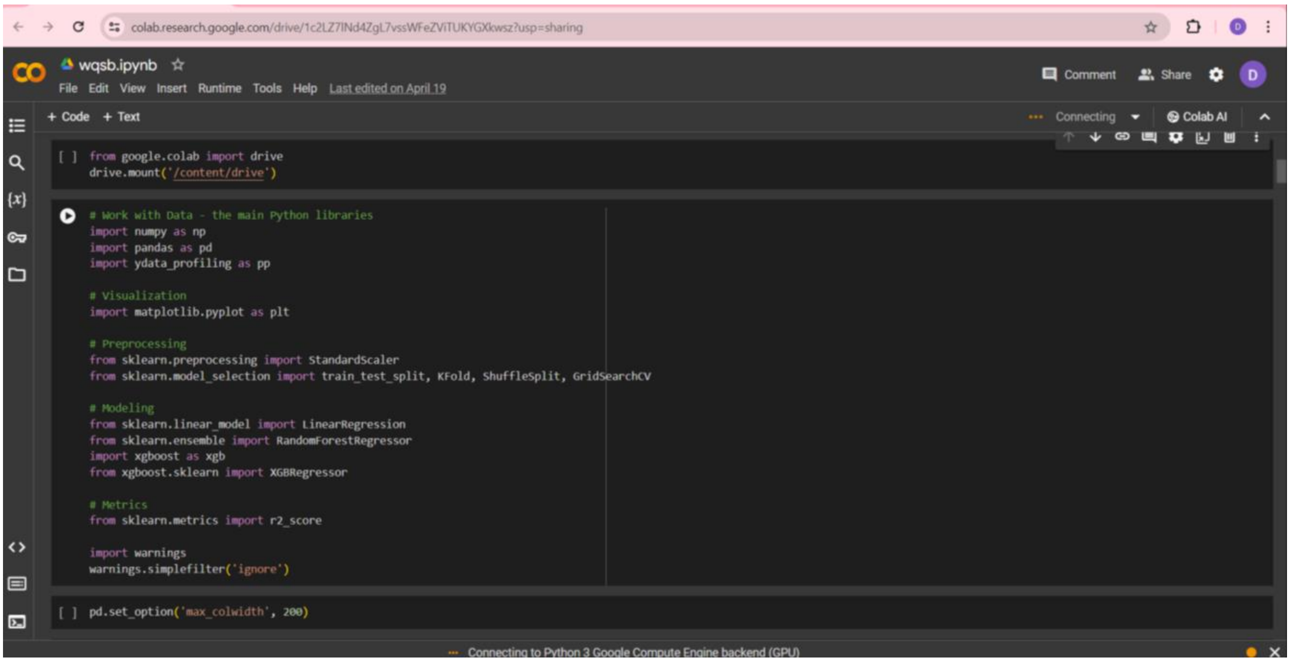
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Prediction Result</title>
</head>

<body>
    <h1>Prediction Result</h1>

    <div>
        <p>The predicted value is: {{ predictions }}</p>
    </div>
</body>

</html>
```

App.py :-

```
from flask import Flask, request, render_template
import numpy as np
import joblib

app = Flask(_name_)
# Load the trained model and scaler
try:
    model=joblib.load('xgb_regressor_model.pkl')
    scaler=joblib.load('standard_scaler.pkl')
except Exception as e:
    print("Error loading model or scaler:", e)



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



@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get Input data from the form
        feature_14_NH4 = float(request.form['14_NH4'])
        feature_14_NO2 = float(request.form['14_NO2'])
        feature_14_NO3 = float(request.form['14_NO3'])
        feature_15_NH4 = float(request.form['15_NH4'])
        feature_15_NO2 = float(request.form['15_NO2'])
        feature_15_NO3 = float(request.form['15_NO3'])
        feature_16_NH4 = float(request.form['16_NH4'])
        feature_16_NO2 = float(request.form['16_NO2'])
```

```
        # Create an Input data array for prediction
        input_data = np.array([
            [feature_14_NH4, feature_14_NO2, feature_14_NO3, feature_15_NH4,
             feature_15_NO2, feature_15_NO3, feature_16_NH4, feature_16_NO2]
        ])

        # Scale the input data using the loaded StandardScaler
        scaled_input_data = scaler.transform(input_data)

        # Perform predictions using your model
        predictions = model.predict(scaled_input_data)

        # Render results on a new page
        return render_template('result.html', predictions=predictions)
    except Exception as e:
        error_message = f"An error occurred: {str(e)}"
        return render_template('error.html', error_message=error_message)


if _name_ == '_main_':
    app.run()
```

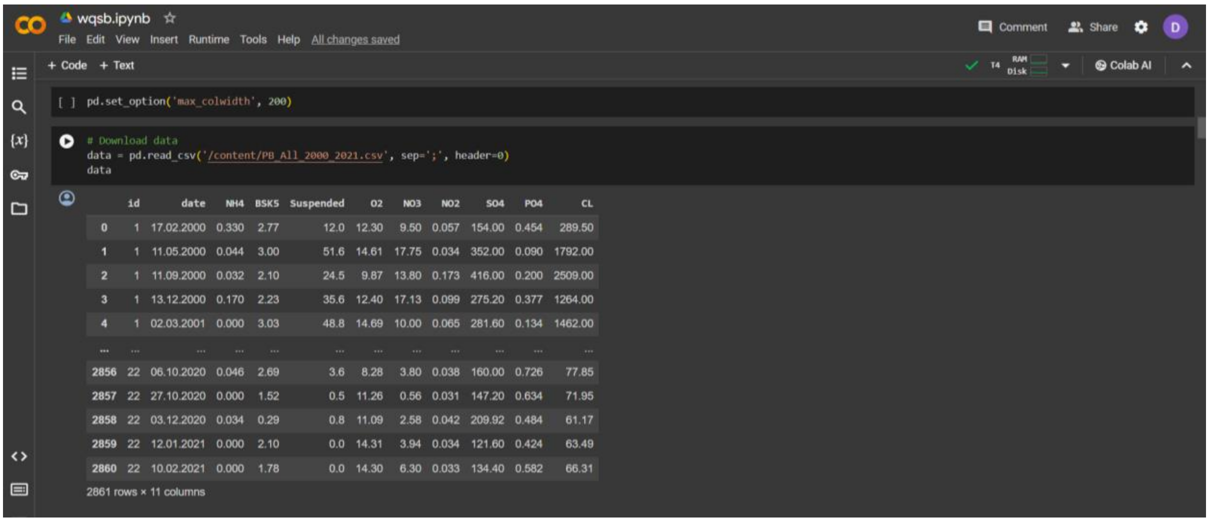
CODE SNIPPETS



```python
from google.colab import drive
drive.mount('/content/drive')
```

```python
# Work with Data - the main Python libraries
import numpy as np
import pandas as pd
import ydata_profiling as pp

# Visualization
import matplotlib.pyplot as plt

# Preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, ShuffleSplit, GridSearchCV

# Modeling
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from xgboost.sklearn import XGBRegressor

# Metrics
from sklearn.metrics import r2_score

import warnings
warnings.simplefilter('ignore')
```

```python
pd.set_option('max_colwidth', 200)
```



```python
pd.set_option('max_colwidth', 200)
```

```python
# Download data
data = pd.read_csv('/content/PB_All_2000_2021.csv', sep=';', header=0)
data
```

	id	date	NH4	BSK5	Suspended	O2	NO3	NO2	SO4	PO4	CL
0	1	17.02.2000	0.330	2.77	12.0	12.30	9.50	0.057	154.00	0.454	289.50
1	1	11.05.2000	0.044	3.00	51.6	14.61	17.75	0.034	352.00	0.090	1792.00
2	1	11.09.2000	0.032	2.10	24.5	9.87	13.80	0.173	416.00	0.200	2509.00
3	1	13.12.2000	0.170	2.23	35.6	12.40	17.13	0.099	275.20	0.377	1264.00
4	1	02.03.2001	0.000	3.03	48.8	14.69	10.00	0.065	281.60	0.134	1462.00
...	...	...	...	...	...	...	...	...	...	...	...
2856	22	06.10.2020	0.046	2.69	3.6	8.28	3.80	0.038	160.00	0.726	77.85
2857	22	27.10.2020	0.000	1.52	0.5	11.26	0.56	0.031	147.20	0.634	71.95
2858	22	03.12.2020	0.034	0.29	0.8	11.09	2.58	0.042	209.92	0.484	61.17
2859	22	12.01.2021	0.000	2.10	0.0	14.31	3.94	0.034	121.60	0.424	63.49
2860	22	10.02.2021	0.000	1.78	0.0	14.30	6.30	0.033	134.40	0.582	66.31

2861 rows × 11 columns

wqsb.ipynb

2861 rows × 11 columns

```
# Information for training data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2861 entries, 0 to 2860
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         2861 non-null   int64
 1   date       2861 non-null   object
 2   NH4        2858 non-null   float64
 3   BSK5       2860 non-null   float64
 4   Suspended  2845 non-null   float64
 5   O2         2858 non-null   float64
 6   NO3        2860 non-null   float64
 7   NO2        2858 non-null   float64
 8   SO4        2812 non-null   float64
 9   PO4        2833 non-null   float64
 10  CL         2812 non-null   float64
dtypes: float64(9), int64(1), object(1)
memory usage: 246.0+ KB
```

```
# Download data about monitoring stations
data_about = pd.read_csv('/content/PB_stations.csv', sep=';', header=0, encoding='cp1251')
data_about.sort_values(by=['length'], ascending=False)
```

wqsb.ipynb

```
# Download data about monitoring stations
data_about = pd.read_csv('/content/PB_stations.csv', sep=';', header=0, encoding='cp1251')
data_about.sort_values(by=['length'], ascending=False)
```

	id	length	name_station
20	21	773.0	р. Південний Буг, 773 км, смт. Чорний Острів, Мар'янівське вдсх.
19	20	755.0	р. Південний Буг, 755 км, м. Хмельницький , Хмельницьке вдсх.
18	19	744.0	р. Південний Буг, 744 км, с. Копистин, нижче м.Хмельницький
17	18	711.0	р. Південний Буг, 711 км, смт. Меджибіж, Меджибіжське вдсх.
16	17	692.0	р. Південний Буг, 692 км, с. Щедрове, Щедрівське вдсх.
15	16	652.0	р. Південний Буг, 652 км, м. Хмільник, питний в/з, вище міста
14	15	607.0	р. Південний Буг, 607 км, с. Гущинці, нижче села , питний водозабір м.Калинівка
13	14	582.0	р. Південний Буг, 582 км, м. Вінниця, Сабарівське вдсх, питний в/з міста, вище міста
12	13	569.5	р. Південний Буг, 569,5 км, 500 м нижче скиду ВОКВП ВКГ "Вінницяводоканал" (1,5 км нижче греблі Сабарівського вдсх.)
11	12	537.0	р. Південний Буг, 537 км, смт. Сутиски, Сутиське вдсх., н/б'єф
9	10	413.0	р. Південний Буг, 413 км, с. Маньківка, вище села, питний в/з м.Ладижин
8	9	400.0	р. Південний Буг, 400 км, м. Ладижин, Ладижинське вдсх.
7	8	372.0	р. Південний Буг, 372 км, с. Глибочок, Глибочекське вдсх.
6	7	327.0	р. Південний Буг, 327 км, с. Ставки, кордон Вінницької та Кіровоградської обл.
5	6	316.0	р. Південний Буг, 316 км, м.Гайворон, Гайворонське вдсх.

10s completed at 11:01 AM

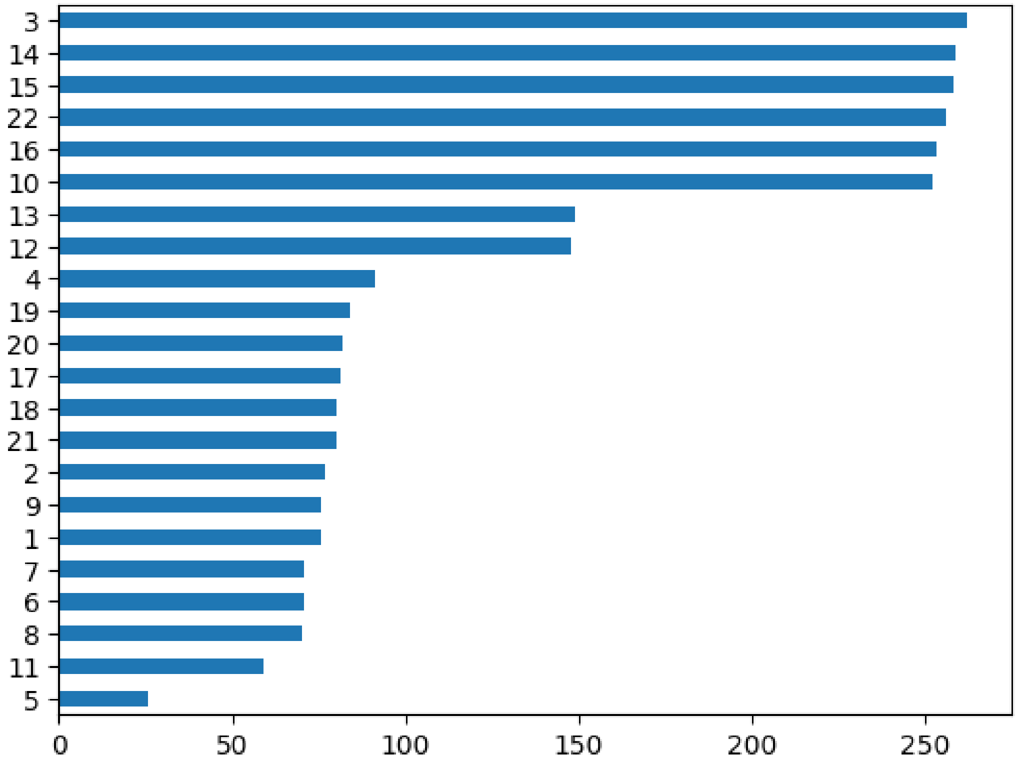
colab.research.google.com/drive/1c2LZ7INd4ZgL7vssWFeZViTUKYGXkwsz?usp=sharing#scrollTo=9MRLNC8o8UAL

wqsb.ipynb

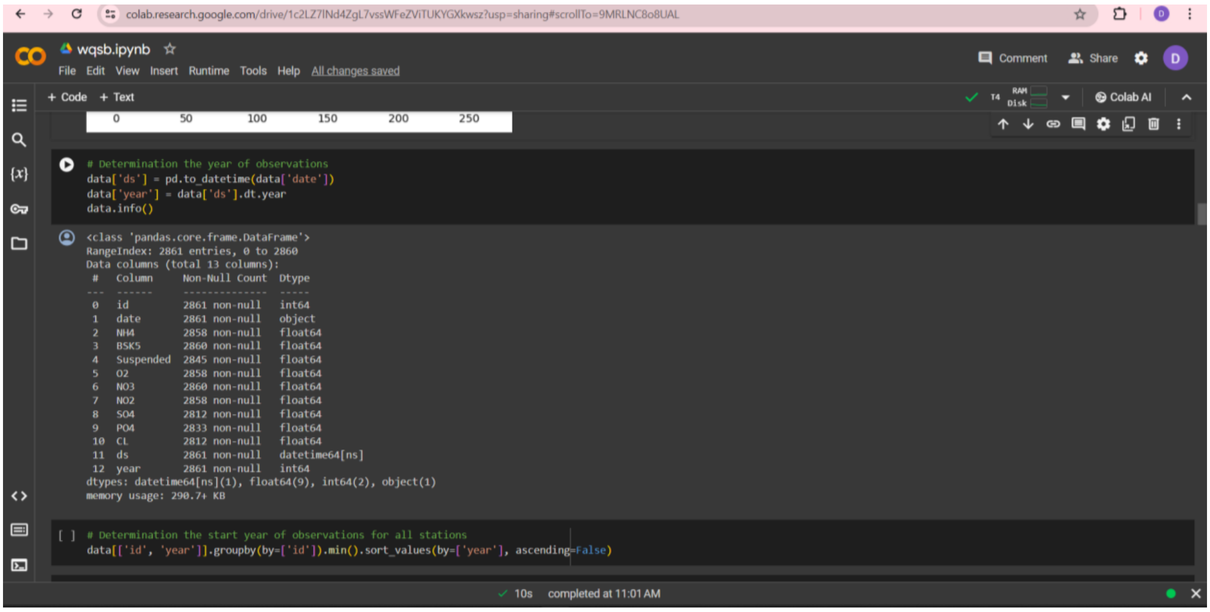
2	3	153.0	р. Південний Буг, 153 км, с. Олексіївка, питний в/з м. Південно-Українськ
1	2	136.0	р. Південний Буг, 136 км, с. Олександрівка, Олександрівське вдсх.
21	22	97.0	р. Південний Буг, 97 км, м. Вознесенськ, пит.в/з м. Вознесенськ, 2 км до в'їзду у м. Вознесенськ по трасі з м. Миколаїв
10	11	50.0	р. Південний Буг, 50 км, с. Ковалівка, Південно-Бузька ЗС
0	1	0.5	р. Південний Буг, 0,5 км, м. Миколаїв, Бузький лиман, тех. в/з Миколаївської ТЕЦ (ліва частина морського порту)

```
# Amount data observations of stations
data['id'].value_counts().sort_values().plot(kind='barh')
```

```
<Axes: >
```



3
14
15
22
16
10
13
12
4
19
20
17
18
21
2
9
1
7
6
8
11
5
0
50
100
150
200
250



colab.research.google.com/drive/1c2LZ7lNd4ZgL7vssWFeZViTUKYGXkwsz?usp=sharing#scrollTo=9MRLNC8o8UAL
wqsb.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Colab AI
0 50 100 150 200 250
Determination the year of observations
data['ds'] = pd.to_datetime(data['date'])
data['year'] = data['ds'].dt.year
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2861 entries, 0 to 2860
Data columns (total 13 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 id 2861 non-null int64
1 date 2861 non-null object
2 NH4 2858 non-null float64
3 BSK5 2860 non-null float64
4 Suspended 2845 non-null float64
5 O2 2858 non-null float64
6 NO3 2860 non-null float64
7 NO2 2858 non-null float64
8 SO4 2812 non-null float64
9 PO4 2833 non-null float64
10 CL 2812 non-null float64
11 ds 2861 non-null datetime64[ns]
12 year 2861 non-null int64
dtypes: datetime64[ns](1), float64(9), int64(2), object(1)
memory usage: 290.7+ KB
[ ] # Determination the start year of observations for all stations
data[['id', 'year']].groupby(by=['id']).min().sort_values(by=['year'], ascending=False)
10s completed at 11:01 AM

```python
# Determination the start year of observations for all stations
data[['id', 'year']].groupby(by=['id']).min().sort_values(by=['year'], ascending=False)
```

```python
# Determination the final year of observations for all stations
data[['id', 'year']].groupby(by=['id']).max().sort_values(by=['year'], ascending=False)
```

	year
id	
22	2021
14	2021
3	2021
5	2021
10	2021
16	2021
15	2021
21	2018
20	2018
19	2018
18	2018
17	2018

Runtime disconnected

Allocating runtime

```python
# Information about stations 14, 15, 16
stations_good = [14, 15, 16]
data_about[data_about['id'].isin(stations_good)]
```

	id	length	name_station
13	14	582.0	р. Південний Буг, 582 км, м. Вінниця, Сабарівське вдсх, питний в/з міста, вище міста
14	15	607.0	р. Південний Буг, 607 км, с. Гущинці, нижче села , питний водозабір м.Калинівка
15	16	652.0	р. Південний Буг, 652 км, м. Хмільник, питний в/з, вище міста

```python
# Set target indicator
target_data_name = 'BSK5'
#feature_target_all = ['NH4', 'BSK5', 'NO3', 'NO2', 'SO4', 'PO4', 'CL']
feature_target_all = ['NH4', 'NO3']
feature_data_all = feature_target_all + [target_data_name]
feature_data_all
```

```
['NH4', 'NO3', 'BSK5']
```

```python
# Data sampling only for good stations
df_indicator = data[['id', 'ds'] + feature_data_all]
df_indicator = df_indicator[df_indicator['id'].isin(stations_good)].dropna().reset_index(drop=True)
df_indicator
```

	id	ds	NH4	NO3	BSK5
0	14	2000-10-01	1.30	6.30	1.9
1	14	2000-01-02	2.20	8.80	2.5
2	14	2000-01-03	0.68	8.80	4.1
3	14	2000-04-04	0.81	4.60	3.4
4	14	2000-05-16	0.27	3.00	3.0
...	...	...	...	...	...
763	16	2020-08-12	0.17	3.38	5.6
764	16	2021-01-28	0.26	2.01	5.4
765	16	2021-02-16	1.64	7.71	6.8
766	16	2021-03-16	2.29	7.70	4.4
767	16	2021-06-04	0.16	6.38	6.1

768 rows × 5 columns

wqsb.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Comment Share Reconnect T4 Colab AI

```
cols = []
for station in stations_good:
    for feature in feature_data_all:
        cols.append(str(station) + "_" + feature)
cols
```

```
['14_NH4',
 '14_NO3',
 '14_BSK5',
 '15_NH4',
 '15_NO3',
 '15_BSK5',
 '16_NH4',
 '16_NO3',
 '16_BSK5']
```

```
df = pd.pivot_table(df_indicator, index=["ds"], columns=["id"], values=feature_data_all).dropna()
df.columns = cols
df
```

ds	14_NH4	14_NO3	14_BSK5	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5
2000-01-02	2.5	2.7	3.1	2.20	2.20	2.40	8.80	8.40	7.70
2000-01-03	4.1	4.4	6.0	0.68	0.87	0.54	8.80	9.10	8.80
2000-01-08	2.9	3.4	9.3	0.37	0.25	0.14	1.50	7.00	0.90
2000-04-04	3.4	3.1	3.4	0.81	1.22	0.51	4.60	4.90	3.50

✓ 10s completed at 11:01 AM

wqsb.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

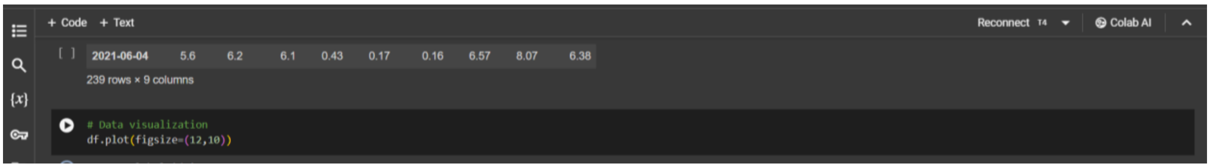
Comment Share Reconnect T4 Colab AI

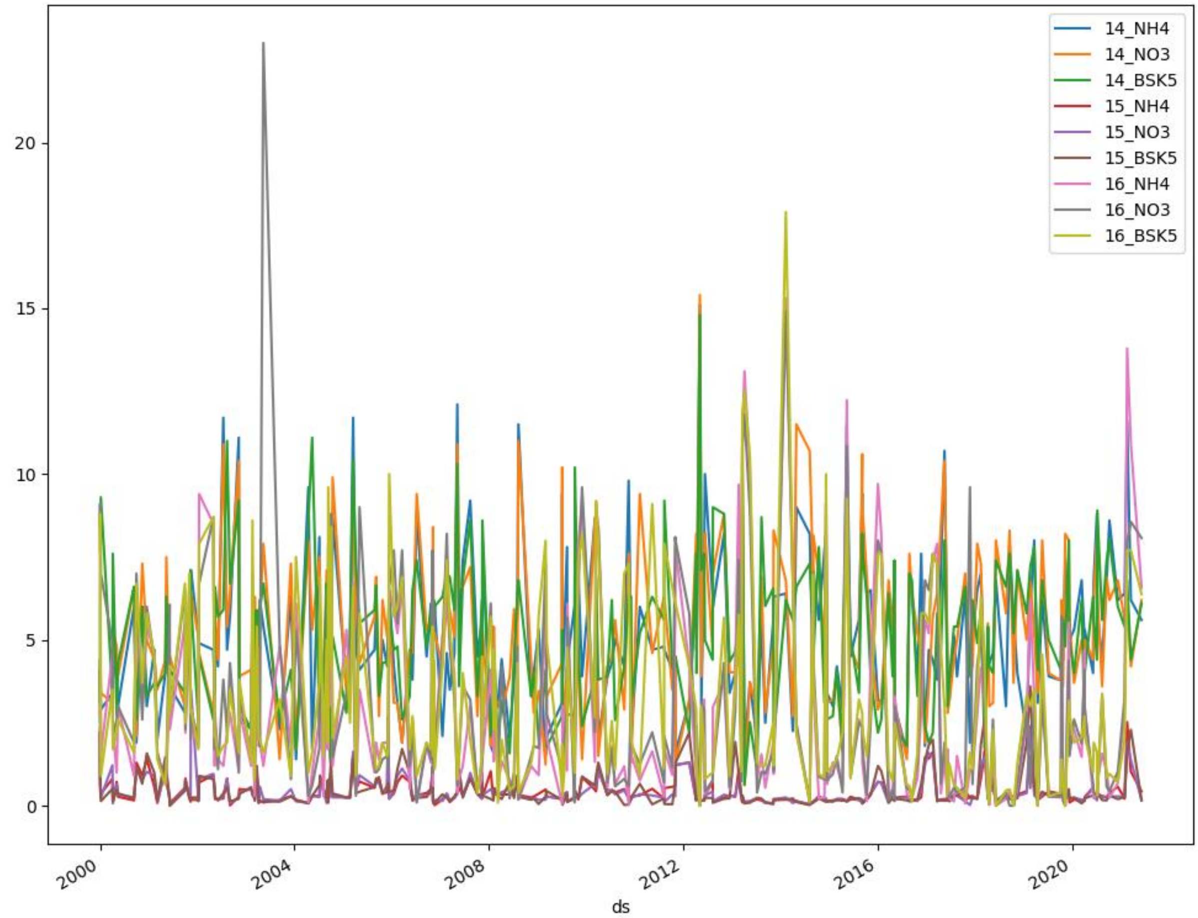
```
df = pd.pivot_table(df_indicator, index=["ds"], columns=["id"], values=feature_data_all).dropna()
df.columns = cols
df
```


ds	14_NH4	14_NO3	14_BSK5	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5
2000-01-02	2.5	2.7	3.1	2.20	2.20	2.40	8.80	8.40	7.70
2000-01-03	4.1	4.4	6.0	0.68	0.87	0.54	8.80	9.10	8.80
2000-01-08	2.9	3.4	9.3	0.37	0.25	0.14	1.50	7.00	0.90
2000-04-04	3.4	3.1	3.4	0.81	1.22	0.51	4.60	4.90	3.50
2000-04-07	4.4	5.3	7.6	0.10	0.07	0.14	2.30	2.10	1.70
...	...	...	...	...	...	...	...	...	...
2020-12-08	6.2	6.8	6.0	0.57	0.28	0.20	0.68	1.03	0.71
2021-01-28	6.4	5.6	5.4	0.21	0.27	0.26	2.95	3.27	2.01
2021-02-16	11.6	6.2	6.8	2.52	1.92	1.64	13.79	7.95	7.71
2021-03-16	6.2	4.2	4.4	1.03	1.38	2.29	10.90	8.56	7.70
2021-06-04	5.6	6.2	6.1	0.43	0.17	0.16	6.57	8.07	6.38

239 rows × 9 columns

```
# Data visualization
```


+ Code + Text
Reconnect T4
Colab AI
[ ]
2021-06-04 5.6 6.2 6.1 0.43 0.17 0.16 6.57 8.07 6.38
239 rows × 9 columns
# Data visualization
df.plot(figsize=(12,10))


14_NH4
14_NO3
14_BSK5
15_NH4
15_NO3
15_BSK5
16_NH4
16_NO3
16_BSK5
0
5
10
15
20
2000
2004
2008
2012
2016
2020
ds

colab.research.google.com/drive/1c2LZ7lNd4ZgL7vssWFeZViTUKYGXkwsz?usp=sharing#scrollTo=jhlTr-Zt8SNk

wqsb.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
# Set target data
target_name = '14_' + target_data_name
target_data = df.pop(target_name)
target_data
```

```
ds
2000-01-02    3.1
2000-01-03    6.0
2000-01-08    9.3
2000-04-04    3.4
2000-04-07    7.6
             ...
2020-12-08    6.0
2021-01-28    5.4
2021-02-16    6.8
2021-03-16    4.4
2021-06-04    6.1
Name: 14_BSK5, Length: 239, dtype: float64
```

```
# Dividing data into training and test
train, test, target, target_test = train_test_split(df, target_data, test_size=0.25, random_state=0)
print(train.shape, test.shape)
```

```
(179, 8) (60, 8)
```

```
# Display the statistics for training data
train.describe()
```

	14_NH4	14_NO3	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5
max	11.700000	11.300000	2.680000	3.500000	3.600000	13.790000	23.000000	12.500000

wqsb.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
# Display the statistics for test data
test.describe()
```

	14_NH4	14_NO3	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5
count	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000
mean	5.121500	5.459667	0.373267	0.357567	0.383000	2.203067	2.452833	2.496333
std	2.656615	2.639714	0.302669	0.277844	0.364025	2.739223	2.846391	2.919597
min	1.600000	2.000000	0.040000	0.022000	0.020000	0.000000	0.000000	0.000000
25%	3.145000	3.650000	0.190000	0.190000	0.180000	0.517500	0.600000	0.940000
50%	4.550000	4.925000	0.270000	0.260000	0.230000	1.250000	1.500000	1.600000
75%	6.225000	6.650000	0.422500	0.440000	0.435000	2.775000	3.027500	2.550000
max	15.100000	15.400000	1.620000	1.480000	1.990000	15.300000	14.900000	17.900000

```
# Standartization data
scaler = StandardScaler()
train = pd.DataFrame(scaler.fit_transform(train), columns = train.columns)

# Display training data
train
```

	14_NH4	14_NO3	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5

✓ 10s completed at 11:01 AM

wqsb.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
# Standartization data
scaler = StandardScaler()
train = pd.DataFrame(scaler.fit_transform(train), columns = train.columns)

# Display training data
train
```

	14_NH4	14_NO3	15_NH4	15_NO3	15_BSK5	16_NH4	16_NO3	16_BSK5
0	-0.305682	-0.694641	-0.159143	-0.104064	-0.294777	-0.966210	-1.026232	-0.973465
1	-0.173876	-0.247629	0.146785	-0.308851	-0.800708	-0.493179	-0.381357	1.989500
2	0.616959	1.227509	-0.559203	-0.491111	-0.498956	-0.114083	0.257257	-0.047762
3	1.891083	2.255635	-0.418006	-0.452202	-0.439328	-0.912533	-0.810230	-0.598180
4	-0.657164	-0.292331	-0.888664	-0.943691	-0.872984	-0.271760	-0.196659	-0.262210
...	...	...	...	...	...	...	...	...
174	-0.393552	-0.918147	-0.182676	-0.022149	-0.023742	0.935979	1.337267	0.774291
175	-0.393552	-0.337032	-0.653335	-0.493159	-0.583880	-0.946081	-0.850926	-1.141450
176	-1.360129	-1.633365	1.511696	1.145137	1.475983	1.774688	1.556399	1.574900
177	-0.657164	-0.560537	1.088103	0.633170	0.699017	0.634045	0.523347	1.203189
178	1.363859	1.719221	-0.888664	-0.820819	-0.728432	-0.607243	-0.509706	-0.440918

179 rows × 8 columns

```
# Standartization data
```

```
# Standartization data
test = pd.DataFrame(scaler.transform(test), columns = test.columns)
```

```
# Training data splitting to new training (part of the all training) and validation data
train_all = train.copy()
target_all = target.copy()
train, valid, target_train, target_valid = train_test_split(train_all, target_all, test_size=0.2, random_state=0)
```

```
# Display information about new training data
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 143 entries, 55 to 172
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   14_NH4   143 non-null    float64
 1   14_NO3   143 non-null    float64
 2   15_NH4   143 non-null    float64
 3   15_NO3   143 non-null    float64
 4   15_BSK5  143 non-null    float64
 5   16_NH4   143 non-null    float64
 6   16_NO3   143 non-null    float64
 7   16_BSK5  143 non-null    float64
dtypes: float64(8)
memory usage: 10.1 KB
```

```
# Display information about validation data
valid.info()
```

```
# Creation the dataframe with the resulting score of all models
result = pd.DataFrame({'model' : ['Linear Regression', 'Random Forest Regressor', 'XGBoost Regressor'],
                       'train_score': 0, 'valid_score': 0})
result
```

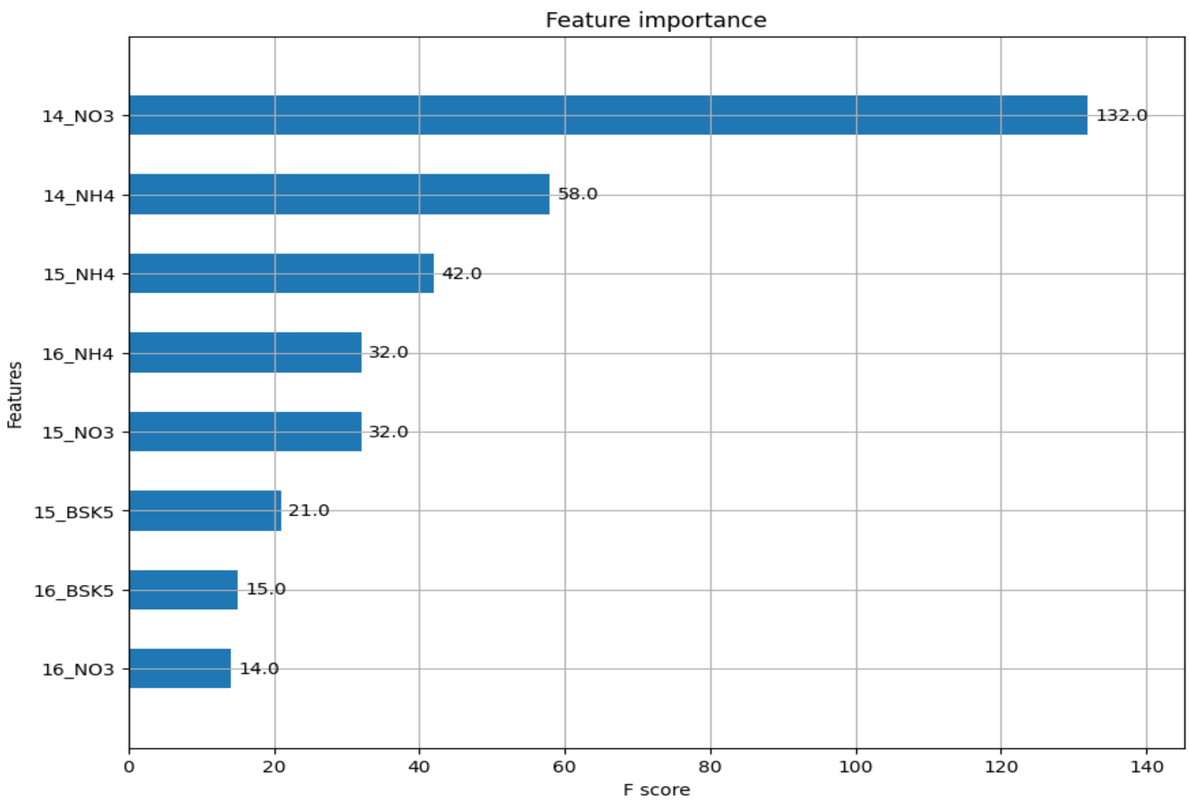
	model	train_score	valid_score
0	Linear Regression	0	0
1	Random Forest Regressor	0	0
2	XGBoost Regressor	0	0

```
# Linear Regression
lr = LinearRegression()
lr.fit(train, target_train)

# Prediction for training data
y_train_lr = lr.predict(train)

# Accuracy of model
r2_score_acc = round(r2_score(target_train, y_train_lr), 2)
print(f'Accuracy of Linear Regression model training is {r2_score_acc}')

# Save to result dataframe
result.loc[result['model'] == 'Linear Regression', 'train_score'] = r2_score_acc
```

```
Accuracy of Linear Regression model training is 0.48
```

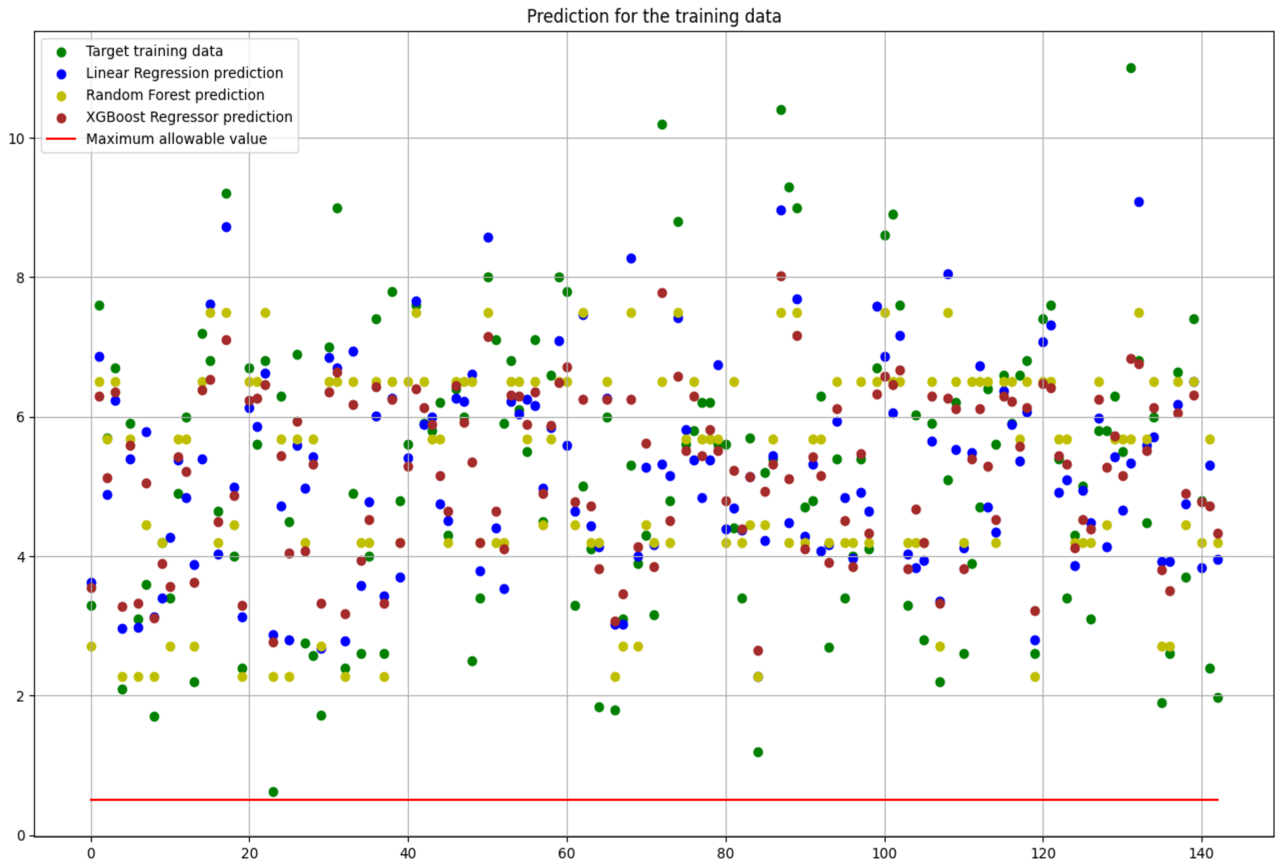
```
%%time
# XGBoost Regressor
xgbr = xgb.XGBRegressor()
# parameters = {'n_estimators': [60, 70, 80, 90, 95, 100, 105, 110, 120, 130, 140],
#                'learning_rate': [0.005, 0.01, 0.05, 0.075, 0.1],
#                'max_depth': [3, 5, 7, 9],
#                'reg_lambda': [0.1, 0.3, 0.5]}

parameters = {'n_estimators': [50, 65, 100],
              'learning_rate': [0.03, 0.04],
              'max_depth': [3, 4],
              'reg_lambda': [0.1, 0.4]}

# Training model
xgb_CV = GridSearchCV(estimator=xgbr, param_grid=parameters, cv=cv_train, n_jobs=-1)
xgb_CV.fit(train, target_train)
print("Best score: %0.3f" % xgb_CV.best_score_)
print("Best parameters set:", xgb_CV.best_params_)

# Prediction for training data
y_train_xgb = xgb_CV.predict(train)

# Accuracy of model
r2_score_acc = round(r2_score(target_train, y_train_xgb),2)
print(f'Accuracy of XGBoost Regressor model training is {r2_score_acc}')

# Save to result dataframe
result.loc[result['model'] == 'XGBoost Regressor', 'train_score'] = r2_score_acc
```


wqsb.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI
result.loc[result['model'] == 'XGBoost Regressor', 'train_score'] = r2_score_acc
Best score: 0.213
Best parameters set: {'learning_rate': 0.03, 'max_depth': 3, 'n_estimators': 50, 'reg_lambda': 0.1}
Accuracy of XGBoost Regressor model training is 0.65
CPU times: user 199 ms, sys: 74 ms, total: 273 ms
Wall time: 3.11 s
# Print rounded r2_score_acc to 2 decimal values after the text
y_val_xgb = xgb_CV.predict(valid)
r2_score_acc_valid = round(r2_score(target_valid, y_val_xgb),2)
result.loc[result['model'] == 'XGBoost Regressor', 'valid_score'] = r2_score_acc_valid
print(f'Accuracy of XGBoost Regressor model prediction for valid dataset is {r2_score_acc_valid}')
Accuracy of XGBoost Regressor model prediction for valid dataset is 0.29
# Feature importance diagram
xgbr = xgb.XGBRegressor(**xgb_CV.best_params_)
xgbr.fit(train, target_train)
fig = plt.figure(figsize = (10,8))
axes = fig.add_subplot(111)
xgb.plot_importance(xgbr,ax = axes,height = 0.5)
plt.show();
plt.close()

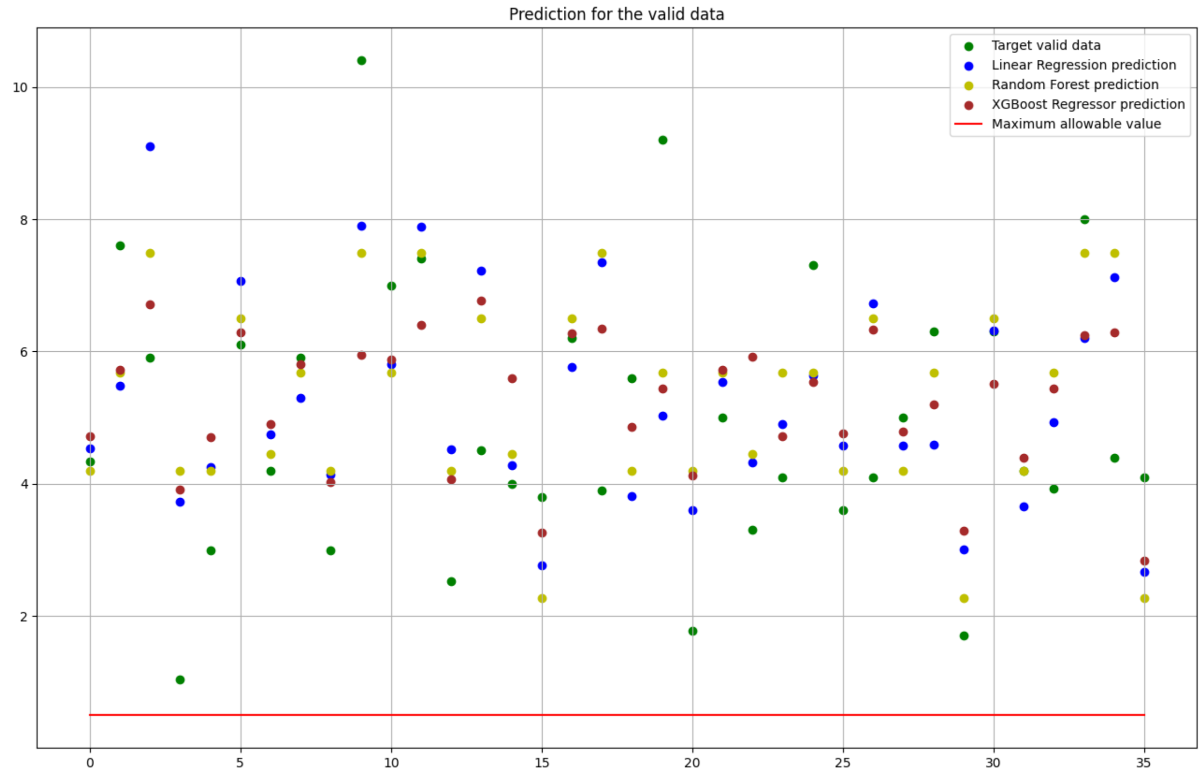

Feature importance
14_NO3
14_NH4
15_NH4
16_NH4
15_NO3
15_BSK5
16_BSK5
16_NO3
132.0
58.0
42.0
32.0
32.0
21.0
15.0
14.0
Features
0
20
40
60
80
100
120
140
F score

```python
# Prediction of target for test data for all models
y_test_lr = lr.predict(test)
y_test_rf = rf_CV.predict(test)
y_test_xgb = xgb_CV.predict(test)
```

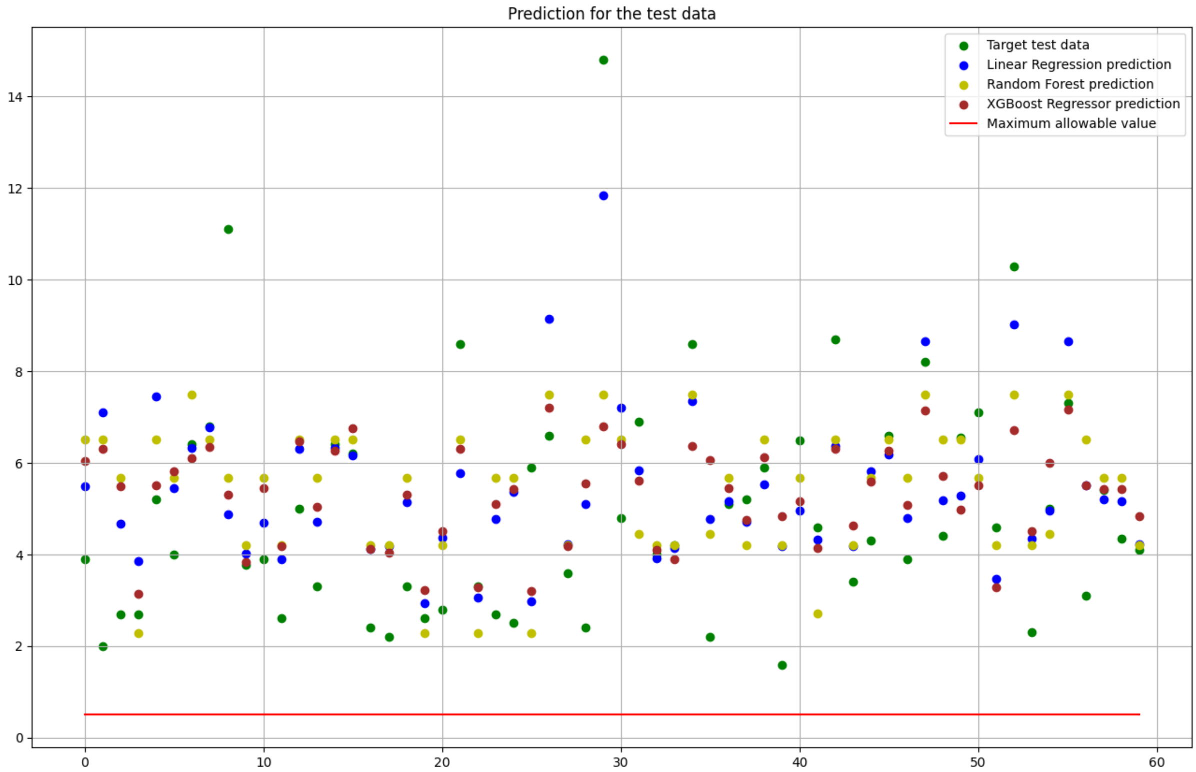
```python
# Building plot for prediction for the training data
x = np.arange(len(train))
plt.figure(figsize=(16,10))
plt.scatter(x, target_train, label = "Target training data", color = 'g')
plt.scatter(x, y_train_lr, label = "Linear Regression prediction", color = 'b')
plt.scatter(x, y_train_rf, label = "Random Forest prediction", color = 'y')
plt.scatter(x, y_train_xgb, label = "XGBoost Regressor prediction", color = 'brown')
plt.plot(x, np.full(len(train), 0.5), label = "Maximum allowable value", color = 'r')
plt.title('Prediction for the training data')
plt.legend(loc='best')
plt.grid(True)
```



```python
# Building plot for prediction for the valid data
x = np.arange(len(valid))
plt.figure(figsize=(16,10))
plt.scatter(x, target_valid, label = "Target valid data", color = 'g')
plt.scatter(x, y_val_lr, label = "Linear Regression prediction", color = 'b')
plt.scatter(x, y_val_rf, label = "Random Forest prediction", color = 'y')
plt.scatter(x, y_val_xgb, label = "XGBoost Regressor prediction", color = 'brown')
plt.plot(x, np.full(len(valid), 0.5), label = "Maximum allowable value", color = 'r')
plt.title('Prediction for the valid data')
plt.legend(loc='best')
plt.grid(True)
```




```python
# Building plot for prediction for the test data
x = np.arange(len(test))
plt.figure(figsize=(16,10))
plt.scatter(x, target_test, label = "Target test data", color = 'g')
plt.scatter(x, y_test_lr, label = "Linear Regression prediction", color = 'b')
plt.scatter(x, y_test_rf, label = "Random Forest prediction", color = 'y')
plt.scatter(x, y_test_xgb, label = "XGBoost Regressor prediction", color = 'brown')
plt.plot(x, np.full(len(test), 0.5), label = "Maximum allowable value", color = 'r')
plt.title('Prediction for the test data')
plt.legend(loc='best')
plt.grid(True)
```




```
# Display results of modeling
result.sort_values(by=['valid_score', 'train_score'], ascending=False)
```

	model	train_score	valid_score
1	Random Forest Regressor	0.55	0.32
2	XGBoost Regressor	0.65	0.29
0	Linear Regression	0.48	0.24

```
# Select models
#result_best = result[(result['train_score'] - result['valid_score']).abs() < 0.15]
result_best = result
result_best.sort_values(by=['valid_score', 'train_score'], ascending=False)
```

	model	train_score	valid_score
1	Random Forest Regressor	0.55	0.32
2	XGBoost Regressor	0.65	0.29
0	Linear Regression	0.48	0.24

```
# Select the best model
```

+ Code + Text Reconnect T4 Colab AI

```
# Select the best model
result_best.nlargest(1, 'valid_score')
```

	model	train_score	valid_score
1	Random Forest Regressor	0.55	0.32

```
# Find a name of the best model (with maximal valid score)
best_model_name = result_best.loc[result_best['valid_score'].idxmax(result_best['valid_score'].max()), 'model']
```

```
print(f'The best model is "{best_model_name}"')
```

```
The best model is "Random Forest Regressor"
```

```
import joblib
# Save the best trained model
best_xgb_model = xgb_CV
joblib.dump(best_xgb_model,'xgb_regressor_model.pkl')
joblib.dump(scaler, 'standard_scaler.pkl')
```

```
['standard_scaler.pkl']
```

```
import pickle
with open('xgb.pkl','wb') as f:
    pickle.dump(xgb_CV, f)
with open('scaler.pkl','wb') as f:
    pickle.dump(scaler,f)
```