Advanced Machine Learning Approaches for Predicting Florasulam Concentration in the Presence of TiO_2 Nanoparticles in Aqueous Medium

A thesis submitted towards partial fulfilment of the requirements for the degree of

Master of Technology in

Nano Science and Technology

Submitted by

Amrapali Maiti

Class Roll No: 002230701026

Exam Roll No: M4NST24023B

Registration No.: 163824 of 2022-2023

Under the guidance of

Dr. Mahua Ghosh Chaudhuri

School of Material Science and Nanotechnology, Jadavpur University

Jadavpur University

Kolkata-700032 India

2024

CERTIFICATE OF RECOMMENDATION

This is to clarify that the thesis entitled "**Advanced Machine Learning Approaches for Predicting Florasulam Concentration in the Presence of TiO2 Nanoparticles in Aqueous Medium** " is a bonafide work carried out by AMRAPALI MAITI under our supervision and guidance for partial fulfilment of the requirement of Master of Technology (Nano Science and Technology) in School of Materials Science and Nanotechnology, during the academic session 2022-2024.

SUPERVISOR

Dr. Mahua Ghosh Chaudhuri

School of Materials Science and Nanotechnology

Jadavpur University, Kolkata-700032

DIRECTOR

Dr. Mahua Ghosh Chaudhuri

School of Materials Science and Nanotechnology

Jadavpur University, Kolkata-700032

DEAN

Faculty Council of Interdisciplinary Studies,

Law and Management (FISLM)

Jadavpur University, Kolkata-700032

CERTIFICATE OF APPROVAL**

This forgoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactorily to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination for evaluation of thesis

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_____

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

** Only in case the thesis is approved.

Acknowledgement

In the first place, I would like to express my profound gratitude to my supervisor **Dr. Mahua Ghosh Chaudhuri** for her advice and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout the work. Above all and the most needed, she provided me constant encouragement and support in various ways. Thank you also for the amicable environment conducive to extensive research work at Nanoscience Lab, in which you have welcomed me. All these provided me great support, helping me to confront the difficult periods of my project, which I do not think I would have completed it otherwise.

I would also like to express my sincere thanks to Prof. Sourav Sarkar, Prof. Chandan Kumar Ghosh, for their help, support, encouragement, suggestion and advice during the course of the research. Their enthusiasm provided me a helpful and effective way of learning which in turn inspired me to follow the research project with great enthusiasm.

I would also really like to thank all the staff members of School of Materials Science and Nanotechnology.

I extend my heartiest thanks to all my seniors and friends for their encouragement and cooperation.

Finally, I express my special gratitude to my parents and all my well-wishers for their wholehearted support, love and encouragement.

1 TABLE OF CONTENTS

2 Abstract 6

3 Introduction 8

3.1 Background 8

3.2 Rationale behind the study 8

3.3 Research Objectives 8

3.4 Significance of the Study 9

3.5 Overview of Florasulam and Its Environmental Impact 9

3.6 Role of TiO_2 Nanoparticles in Environmental Processes 9

3.7 Key Definitions and Concepts 10

3.7.1 Density 10

3.7.2 pH 10

3.7.3 Photocatalysis 10

3.7.4 Point of Zero Charge (PZC) 10

3.8 Application of Machine Learning in Environmental Science 11

3.9 Synthetic Data Generation in Environmental Studies 11

4 Literature Review 13

4.1 Summary of Key Findings from Literature 13

4.2 Literature Review of Paper 1: Machine learning approaches to predict the photocatalytic performance of bismuth ferrite-based materials in the removal of malachite green 14

4.2.1 Introduction 14

4.2.2 Literature Review 14

4.2.3 Methodology 15

4.2.4 Results and Discussion 16

4.2.5 Conclusion 17

4.2.6 Experimental Setup and Procedures 17

4.2.7 Data Analysis and Machine Learning Modelling 18

4.2.8 Implications, Limitations, and Future Directions 20

4.2.9 Conclusion 21

4.2.10 Key Contributions 21

4.2.11 Broader Impact and Future Prospects 22

4.2.12 Challenges and Considerations 22

4.3 Literature review on Paper 2: A Hybrid Machine Learning-Genetic Algorithm (ML-GA) Model to Predict Optimal Process Parameters of Nanoporous Cu for CO_2 Reduction 23

4.3.1 Introduction 23
4.3.2 Experimental Procedure 23
4.3.3 Results and Discussion 24
4.3.4 Conclusion 25
4.4 Literature Review of Paper 3: Implementing Machine Learning and Statistical Optimization Analysis for Oily Wastewater Treatment in a Photocatalytic membrane reactor 26
4.4.1 Introduction 26
4.4.2 Experimental Procedure 26
4.4.3 Results and Discussion 27
4.4.4 Conclusion 27
4.4.5 Discussion of Practical Implications 28
4.4.6 Challenges and Limitations 28
4.4.7 Future Directions 28
4.4.8 Conclusion 29
4.5 Conclusive Summary of Literature Review 30
4.5.1 Key Findings and Their Relevance to This Study 30
4.5.2 Scope for Further Discussion and Research: 31
4.6 Conclusion 32
5 Statement of the Problem 33
5.1 Preamble 33
5.1.1 Role of pH 33
5.1.2 Influence of TiO_2 Nanoparticle Concentration 33
5.1.3 Initial Concentration of Florasulam 33
5.1.4 Significance of Machine Learning for Predictive Modelling 34
5.1.5 Broader Implications 34
5.2 Conclusion 34
6 Methodology 35
6.1 Experimental Design and Data Collection 35
6.1.1 Experimental Setup 35
6.1.2 Data Collection Methods 35
6.2 Data Preprocessing 35
6.2.1 Data Cleaning and Handling Missing Values 35
6.2.2 Data Merging and Transformation 35
6.2.3 Statistical Analysis of Data 36
6.2.4 Probability Distribution Fitting 36

6.3 Synthetic Data Generation 36
6.3.1 Rationale for Synthetic Data 36
6.3.2 Generation Process 36
6.3.3 Validation of Synthetic Data 36
6.4 Machine Learning Model Selection 36
6.4.1 Overview of Selected Models 36
6.4.2 Model Training and Testing 37
6.4.3 Performance Evaluation Metrics 38
6.4.4 Cross-Validation 38
6.4.5 Hyperparameter Tuning 38
7 Data Sets 39
8 Workings: 42
8.1 Importing Necessary Libraries 42
8.2 Data Merging 43
8.2.1 Result: 43
8.3 Best Fit Probability Distribution Analysis: 44
8.3.1 Objective: 44
8.3.2 Code 45
8.3.3 Output 46
8.3.4 Explanation 47
8.3.5 Methodology: 48
8.3.6 Interpretation: 49
8.4 Synthetic Data Generation 50
8.4.1 Code 50
8.5 Validation of Synthetic Data 50
8.5.1 Code 51
8.5.2 Output 52
8.5.3 Explanation of Validation Output 64
8.6 Model Training & Evolution: 65
8.6.1 Combining Synmthetic Data & The Original Data 65
8.6.2 Splitting of Train & Test Data 66
8.6.3 CatBoost Model 66
8.6.4 XGBoost Model 75
8.6.5 Random Forest 84
8.6.6 LightGBM (Light Gradient-Boosting Machine) 93

8.6.7 SVM (Support Vector Machine) 102
8.7 Ensemble Modeling 111
8.7.1 Code 111
8.7.2 Graphs 115
8.7.3 Output 116
8.7.4 Explanation 117
9 Data Analysis and Model Development 118
9.1 Descriptive Statistics and Data Exploration 118
9.1.1 Summary Statistics 118
9.1.2 Data Visualization 118
9.2 Probability Distribution Analysis 118
9.2.1 Best-Fit Distribution Identification 118
9.2.2 Implications of Distribution Fitting 118
9.3 Synthetic Data Generation and Analysis 118
9.3.1 Data Generation Process 118
9.3.2 Validation of Synthetic Data 119
9.4 Model Development and Training 119
9.4.1 CatBoost Regressor Implementation 119
9.4.2 Random Forest Regressor Implementation 120
9.4.3 XGBoost Regressor Implementation 120
9.4.4 Support Vector Machine (SVM) Regressor Implementation 120
9.5 Cross-Validation and Hyperparameter Tuning 121
9.5.1 Cross-Validation 121
9.6 Hyperparameter Tuning 121
10 Results and Discussion 123
10.1 Model Performance Comparison 123
10.1.1 CatBoost Regressor 123
10.1.2 Random Forest Regressor 123
10.1.3 XGBoost Regressor 123
10.1.4 Support Vector Machine (SVM) Regressor 123
10.2 Training Time and Computational Efficiency 124
10.2.1 Comparative Analysis 124
10.3 Implications for Environmental Modelling 124
10.4 Discussion on CatBoost Regression vs. Random Forest Regression 124
10.4.1 Accuracy Comparison 124

10.4.2 Interpretability and Usability 125
10.4.3 Practical Applications 125
10.5 XGBoost and Support Vector Machine (SVM) Performance 125
10.5.1 XGBoost: Efficiency and Accuracy 125
10.5.2 SVM: Handling Complexity 125
10.5.3 Training Time and Model Suitability 125
10.6 Implications for Environmental Modelling 126
10.7 Impact of pH and TiO_2 Nanoparticle Concentration on Florasulam Degradation 126
10.7.1 Influence of pH Levels 126
10.7.2 Influence of TiO2 Nanoparticle Concentration 126
10.7.3 Interaction Between pH and Nanoparticle Concentration 127
10.8 Synthetic Data and Model Generalization 127
10.8.1 Effectiveness of Synthetic Data 127
10.8.2 Validation and Model Robustness 127
11 Conclusion and Future Work 128
11.1 Summary of Findings 128
11.2 Contributions to Environmental Modelling 128
11.3 Limitations 128
11.4 Future Work 128
11.5 Final Remarks 129
12 References 130
13 Appendices 142
13.1 Table of Figures: 142

Advanced Machine Learning Approaches for Predicting Florasulam Concentration in the Presence of TiO_2 Nanoparticles in Aqueous Medium

2 Abstract

This thesis presents a comprehensive study on the prediction of Florasulam concentration in aqueous solutions containing Titanium Dioxide (TiO_2) nanoparticles using advanced machine learning techniques. Florasulam, a widely used herbicide, poses significant environmental risks due to its persistence in water systems, particularly when interacting with nanoparticles like TiO2. This research leverages several machine learning models, including CatBoost, Random Forest, XGBoost, and Support Vector Machine (SVM), to model and predict the behaviour of Florasulam under varying environmental conditions such as pH levels and UV exposure.

The study begins with an extensive literature review, outlining the environmental impact of Florasulam, the role of TiO_2 nanoparticles in photocatalysis, and the application of machine learning in environmental sciences. Key concepts such as density, pH, photocatalysis, and the Point of Zero Charge (PZC) of TiO_2 are defined and discussed in relation to their influence on the degradation of Florasulam.

Experimental data were collected from controlled laboratory experiments, where Florasulam was exposed to different concentrations of TiO2 nanoparticles across various pH levels and under both UV and sunlight conditions. This data was supplemented with synthetic data generated through statistical modelling to ensure robustness in the machine learning training process.

The core of the thesis focuses on the development and evaluation of predictive models. Each model was trained and tested using a combination of original and synthetic data. The models were evaluated based on performance metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 scores, as well as training time and computational efficiency. CatBoost and XGBoost emerged as the most accurate models, with CatBoost showing particular strength in handling categorical data and achieving high prediction accuracy. Random Forest, while slightly less accurate, was noted for its faster training times and robustness against overfitting. SVM, despite its longer training time, provided valuable insights into the non-linear interactions between Florasulam and TiO_2 nanoparticles.

The study also explores the effects of pH and TiO_2 nanoparticle concentration on Florasulam degradation. It was found that higher pH levels and higher concentrations of TiO_2 significantly accelerated the degradation process, highlighting the importance of environmental conditions in determining the fate of herbicides in natural water systems.

This research demonstrates the efficacy of machine learning models in predicting environmental contaminants' behaviour, offering a powerful tool for environmental monitoring and management. The findings provide a foundation for future research, suggesting avenues for expanding the dataset, integrating real-time data, and exploring deep learning approaches to further enhance predictive accuracy.

Keywords: *Florasulam, TiO_2 nanoparticles, machine learning, CatBoost, XGBoost, environmental modelling, synthetic data generation, photocatalysis, environmental contamination.*

3 INTRODUCTION

3.1 BACKGROUND

The contamination of water bodies by herbicides poses significant environmental and health risks. Florasulam, an herbicide belonging to the triazolopyrimidine sulfonanilide class, is widely used for controlling broadleaf weeds in agricultural fields. Due to its extensive use, Florasulam can leach into water systems, where its persistence becomes a concern, particularly in the presence of environmental factors such as nanoparticles. Among these, Titanium Dioxide (TiO_2) nanoparticles are prevalent due to their widespread use in various industries, including cosmetics, paints, and environmental remediation. These nanoparticles can influence the degradation, persistence, and mobility of organic contaminants like Florasulam in aqueous environments.

In recent years, there has been a growing interest in using advanced machine learning algorithms to predict the behaviour of contaminants in complex environmental matrices. These models can offer significant advantages in terms of accuracy, efficiency, and the ability to model non-linear relationships compared to traditional statistical approaches. This thesis aims to explore the use of several cutting-edge machine learning techniques to predict the concentration of Florasulam in the presence of TiO_2 nanoparticles across various experimental conditions.

3.2 RATIONALE BEHIND THE STUDY

The interaction between Florasulam and TiO_2 nanoparticles under different environmental conditions, such as varying pH levels and exposure to UV light, presents a complex scenario that is difficult to model using traditional methods. Accurate prediction of Florasulam concentration is crucial for assessing its environmental impact and for developing strategies to mitigate its adverse effects. The challenge lies in selecting and optimizing machine learning models that can effectively capture the dynamics of this interaction.

3.3 RESEARCH OBJECTIVES

The objectives of this research are:

1. To evaluate the performance of various regression models, including CatBoost, Random Forest, XGBoost, and Support Vector Machine (SVM), in predicting the concentration of Florasulam in aqueous media containing TiO_2 nanoparticles.
2. To generate and validate synthetic datasets that replicate experimental conditions to enhance the robustness of the predictive models.
3. To conduct a comprehensive comparison of the selected models in terms of accuracy, training time, and ability to generalize to new data.

4. To analyse the effects of different environmental factors, such as pH levels and nanoparticle concentrations, on the persistence and degradation of Florasulam.

3.4 Significance of the Study

This study is significant in advancing the understanding of how nanoparticles influence the behaviour of herbicides in aquatic environments. By integrating machine learning models, this research not only improves predictive accuracy but also provides insights that can inform environmental policy and the development of safer herbicide formulations. Moreover, the findings can contribute to broader applications in environmental monitoring and risk assessment, particularly in scenarios involving the interaction of pollutants with emerging contaminants like nanoparticles.

3.5 Overview of Florasulam and Its Environmental Impact

Florasulam (chemical formula: $C_{12}H_8F_3N_5O_3S$) is a selective herbicide known for its effectiveness in controlling a wide range of broadleaf weeds in cereals and grassland. Its mode of action involves inhibiting acetolactate synthase (ALS), an enzyme critical for the biosynthesis of branched-chain amino acids in plants. Despite its agricultural benefits, Florasulam's persistence in the environment has raised concerns, especially regarding its potential to contaminate water bodies through runoff and leaching.

Studies have shown that the environmental fate of Florasulam is influenced by various factors, including pH, temperature, and the presence of other chemical agents. In alkaline conditions, Florasulam is more likely to persist, whereas acidic environments facilitate its degradation. Additionally, the compound's interaction with soil and water components can lead to complex dynamics that affect its mobility and bioavailability.

3.6 Role of TiO_2 Nanoparticles in Environmental Processes

Titanium Dioxide (TiO_2) nanoparticles are widely recognized for their photocatalytic properties, making them effective in degrading organic pollutants when exposed to UV light. These nanoparticles are used in various applications, including water treatment, due to their ability to break down contaminants into less harmful compounds. However, the presence of TiO_2 nanoparticles in natural water systems raises questions about their interaction with other environmental contaminants, such as herbicides.

Research has shown that TiO_2 nanoparticles can influence the photodegradation of herbicides like Florasulam. The nanoparticles act as catalysts, enhancing the breakdown of the herbicide under UV light. However, the extent of this effect depends on several factors, including the concentration of nanoparticles, the intensity of UV exposure, and the chemical nature of the herbicide. Understanding these interactions is crucial for predicting the environmental fate of Florasulam in the presence of TiO_2 nanoparticles.

TiO_2 is an n-type semiconductor, which means it has an excess of electrons that can participate in conduction. The unique photocatalytic properties of TiO_2 arise from its ability to generate electron-hole pairs when exposed to UV light. These pairs can interact with water molecules to produce reactive oxygen species, such as hydroxyl radicals, which are capable of degrading organic pollutants like Florasulam. The effectiveness of TiO_2 in these reactions is influenced by several factors, including its crystal structure (anatase, rutile, or brookite), surface area, and the specific environmental conditions (e.g., pH, temperature, and light intensity) .

3.7 KEY DEFINITIONS AND CONCEPTS

3.7.1 Density

Density is defined as the mass per unit volume of a substance and is usually expressed in grams per cubic centimetre (g/cm^3). For nanoparticles like TiO_2, density plays a critical role in determining their behaviour in suspension, including their settling rate, aggregation tendency, and interaction with other particles or contaminants in the water. The density of TiO2 nanoparticles varies depending on their crystal structure:

1. anatase (3.79 g/cm^3)
2. rutile (4.13 g/cm^3)
3. brookite (3.99 g/cm^3)

3.7.2 pH

The pH level of a solution measures its acidity or alkalinity on a scale of 0 to 14. In this study, pH is a crucial factor affecting the degradation of Florasulam, as it influences the surface charge of TiO_2 nanoparticles and the ionization state of Florasulam. The interaction between Florasulam and TiO_2 is highly pH-dependent, with certain pH levels optimizing the photocatalytic degradation process

3.7.3 Photocatalysis

Photocatalysis refers to the acceleration of a photoreaction in the presence of a catalyst. TiO_2 nanoparticles act as photocatalysts by absorbing UV light and generating reactive species that can degrade organic pollutants like Florasulam. The efficiency of photocatalysis depends on factors such as the wavelength of the light source, the concentration of TiO_2, and the surface characteristics of the nanoparticles .

3.7.4 Point of Zero Charge (PZC)

The PZC is the pH at which the surface of a material, such as TiO_2, has no net electrical charge. At pH levels below the PZC, the surface is positively charged, while at pH levels above the PZC, it is negatively charged. The PZC of TiO_2 is around pH 6.5, which influences its interaction with Florasulam, especially considering Florasulam's pK_a of 4.54 .

3.8 Application of Machine Learning in Environmental Science

Machine learning has emerged as a powerful tool for modelling complex environmental systems. In particular, regression models are frequently used to predict the concentration of pollutants, assess environmental risks, and optimize remediation strategies. These models can handle large datasets with multiple variables, making them ideal for scenarios where traditional statistical methods fall short.

Several machine learning algorithms have been successfully applied in environmental studies:

- **Random Forest Regression**: An ensemble learning method that constructs multiple decision trees to improve prediction accuracy. It is robust to overfitting and can handle large datasets with numerous features.
- **Support Vector Machine (SVM) Regression**: A supervised learning model that finds the optimal hyperplane for predicting continuous variables. SVM is known for its effectiveness in high-dimensional spaces.
- **XGBoost**: An advanced implementation of gradient boosting that is highly efficient and scalable. XGBoost has been widely adopted in environmental modelling due to its superior performance in handling complex datasets.
- **CatBoost**: A gradient boosting algorithm specifically designed to handle categorical features without the need for explicit encoding. It is particularly useful in scenarios with a mix of numerical and categorical data, providing high accuracy and fast training times.

3.9 Synthetic Data Generation in Environmental Studies

Synthetic data generation is an important technique in environmental science, particularly when dealing with limited or imbalanced datasets. By generating synthetic data that mimics the statistical properties of real-world observations, researchers can augment their datasets, allowing for more robust model training and validation. This approach is especially useful in studies where experimental data is scarce or difficult to obtain.

In the context of this research, synthetic data will be generated to supplement the experimental measurements of Florasulam concentration. This synthetic data will be validated using statistical tests to ensure it accurately represents the underlying distribution of the real data, thereby enhancing the reliability of the predictive models.

4.1 SUMMARY OF KEY FINDINGS FROM LITERATURE

The literature review highlights several key points that form the foundation of this thesis:

- Florasulam environmental behaviour is complex and influenced by various factors, including pH, temperature, and the presence of TiO_2 nanoparticles.
- TiO_2 nanoparticles play a significant role in the photodegradation of organic pollutants, including herbicides like Florasulam.
- Machine learning models, particularly those using ensemble methods and gradient boosting, offer promising approaches for predicting environmental phenomena.
- Synthetic data generation is a valuable tool for augmenting datasets and improving model performance in environmental studies.

4.2 Literature Review of Paper 1: Machine learning approaches to predict the photocatalytic performance of bismuth ferrite-based materials in the removal of malachite green

4.2.1 Introduction

In recent years, the degradation of organic pollutants in wastewater has become a significant focus of environmental research due to the increasing levels of water pollution and the stringent regulations for effluent discharge. Among various organic pollutants, synthetic dyes like malachite green (MG) have garnered attention due to their extensive use in the textile, paper, and leather industries, and their potential to cause serious environmental and health issues. The presence of these dyes in aquatic systems is of particular concern due to their high visibility, resistance to biodegradation, and toxicity to aquatic life. Advanced Oxidation Processes (AOPs) have emerged as promising technologies for the degradation of such pollutants, with photocatalysis being a particularly attractive option due to its high efficiency, cost-effectiveness, and environmental compatibility.

The study titled **"Machine Learning Approaches to Predict the Photocatalytic Performance of Bismuth Ferrite-Based Materials in the Removal of Malachite Green,"** focuses on enhancing the efficiency of photocatalytic degradation of MG using Bismuth Ferrite ($BiFeO_3$) as a photocatalyst. The research introduces an innovative approach by integrating machine learning (ML) techniques to predict the photocatalytic performance under various experimental conditions. This approach not only optimizes the degradation process but also reduces the need for extensive empirical experimentation, thereby saving time and resources.

4.2.2 Literature Review

4.2.2.1 Water Pollution and Dye Contamination

Water pollution is a global issue, exacerbated by the discharge of industrial effluents containing toxic chemicals, including synthetic dyes. MG, a triphenylmethane dye, is widely used but poses significant environmental hazards due to its toxicity, carcinogenicity, and resistance to conventional treatment methods. The degradation of MG in water bodies is challenging, necessitating the development of effective treatment strategies.

4.2.2.2 Photocatalysis as an Advanced Oxidation Process

Photocatalysis, a subclass of AOPs, involves the acceleration of a photoreaction in the presence of a catalyst, usually a semiconductor material. When exposed to light, the semiconductor generates electron-hole pairs, leading to the formation of reactive oxygen species (ROS) such as hydroxyl radicals (•OH), which can oxidize and degrade organic pollutants into harmless products. TiO_2 has been the most extensively studied photocatalyst

due to its high activity, stability, and non-toxicity. However, its wide bandgap limits its activation to UV light, which comprises a small fraction of the solar spectrum.

4.2.2.3 Bismuth Ferrite ($BiFeO_3$) as a Photocatalyst

Bismuth Ferrite ($BiFeO_3$) is a multiferroic material that has attracted attention as a visible-light-driven photocatalyst due to its narrow bandgap (~2.5 eV), which allows it to be activated by visible light. Additionally, BiFeO3 exhibits good chemical stability, non-toxicity, and magnetic properties, making it a versatile material for environmental applications. However, the rapid recombination of photogenerated charge carriers in $BiFeO_3$ limits its photocatalytic efficiency. To overcome this, various strategies such as doping with noble metals (e.g., Ag, Pt, Pd) have been employed to enhance charge separation and improve photocatalytic performance.

4.2.2.4 Machine Learning in Environmental Applications

Machine learning (ML) has emerged as a powerful tool for modelling and predicting complex environmental processes. In the context of photocatalysis, ML models can be trained on experimental data to predict the performance of photocatalysts under various conditions, thus aiding in the optimization of the degradation process. Various ML algorithms, including decision trees, support vector machines, and neural networks, have been applied in environmental modelling, demonstrating high accuracy and reliability in predictions.

4.2.3 Methodology

4.2.3.1 Synthesis and Characterization of BiFeO3-Based Photocatalysts

The BiFeO3-based photocatalysts used in this study were synthesized and characterized as per standard procedures. The photocatalysts were doped with noble metals such as Pd, Pt, and Ag to enhance their photocatalytic activity. The characterization of these materials was performed using techniques such as X-ray diffraction (XRD) to determine crystallinity, scanning electron microscopy (SEM) for morphology, and UV-Vis spectroscopy for optical properties.

4.2.3.2 Photocatalytic Degradation Experiments

The photocatalytic degradation of MG was conducted under controlled laboratory conditions. A stock solution of MG was prepared and diluted to the required concentrations. The photocatalytic experiments were carried out in a batch reactor, with the solution being irradiated by a visible light source. The degradation efficiency was monitored by measuring the absorbance of the solution at regular intervals using a UV-Vis spectrophotometer.

4.2.3.3 Data Collection and Preprocessing

The study generated a comprehensive databank of 1200 data points, representing various combinations of photocatalytic conditions such as catalyst type, reaction time, light intensity, initial concentration of MG, catalyst loading, solution pH, presence of anions, and humic acid concentration. The input variables were normalized, and categorical variables were encoded to ensure compatibility with the ML models.

4.2.3.4 Machine Learning Models

Twelve machine learning models, including CatBoost, GradientBoosting, HistGradientBoosting, ExtraTrees, XGBoost, DecisionTree, Bagging, LGBM, GaussianProcess, ANN, and LightLSTM, were applied to predict the photocatalytic degradation efficiency of MG. These models were trained and tested using a 70:30 train-test split of the dataset. The performance of the models was evaluated using statistical metrics such as the coefficient of determination (R^2), mean absolute error (MAE), and root mean square error (RMSE).

4.2.4 Results and Discussion

4.2.4.1 Model Performance

Among the models tested, CatBoost emerged as the most accurate, with an R^2 value of 0.998 and the lowest MAE and RMSE values. This model demonstrated superior predictive power, closely aligning with the experimental data. The cumulative frequency plot showed that CatBoost could predict over 96% of the data points with less than 1% error, indicating its reliability in modeling the photocatalytic degradation process.

4.2.4.2 Feature Importance and Partial Dependence Analysis

The feature importance analysis revealed that reaction time was the most significant factor influencing MG degradation, followed by initial dye concentration, catalyst loading, and light intensity. The partial dependence plots (PDPs) provided insights into the effect of individual variables on the degradation efficiency, highlighting the optimal conditions for maximum degradation. For instance, the highest degradation was achieved at a catalyst loading of 1.5 g/L and a light intensity of 105 W, under neutral pH conditions.

4.2.4.3 Implications for Environmental Applications

The findings of this study have significant implications for environmental remediation. By utilizing ML models, the study not only optimized the photocatalytic degradation of MG but also provided a framework for predicting the degradation of other organic pollutants under varying conditions. This approach can be extended to real-world scenarios, offering a cost-effective and efficient method for treating contaminated water.

4.2.5 Conclusion

This study successfully demonstrated the integration of machine learning with photocatalytic degradation processes, offering a novel approach to predict and optimize the removal of harmful organic pollutants from water. The use of BiFeO3-based photocatalysts, enhanced with noble metals, showed promising results in the degradation of MG under visible light. The CatBoost model, in particular, proved to be highly effective in predicting the photocatalytic performance, making it a valuable tool for environmental applications. Future research could explore the application of these ML models to other pollutants and photocatalysts, further advancing the field of water treatment.

4.2.6 Experimental Setup and Procedures

4.2.6.1 Photocatalyst Synthesis

The synthesis of Bismuth Ferrite ($BiFeO_3$)-based photocatalysts was carried out through a sol-gel method, which is widely recognized for producing high-purity and uniform nanoparticles. In this study, noble metals such as palladium (Pd), platinum (Pt), and silver (Ag) were doped onto the $BiFeO_3$ surface to enhance its photocatalytic properties. The procedure involved dissolving appropriate amounts of bismuth nitrate and ferric nitrate in a solution of citric acid and ethylene glycol, which acted as complexing and gelling agents, respectively. After the formation of a homogeneous sol, the noble metal precursors were added, followed by heating to form a gel. The gel was then dried and calcined at high temperatures to obtain the final $BiFeO_3$ and noble metal-doped $BiFeO_3$ composites.

4.2.6.2 Characterization Techniques

To ensure the successful synthesis and doping of the photocatalysts, several characterization techniques were employed:

- **X-ray Diffraction (XRD):** Used to determine the crystal structure and phase purity of the synthesized photocatalysts. The XRD patterns confirmed the presence of the perovskite structure of $BiFeO_3$ and the incorporation of noble metals.
- **Scanning Electron Microscopy (SEM):** Provided detailed images of the photocatalyst morphology, showing the uniform distribution of metal particles on the $BiFeO_3$ surface.
- **Transmission Electron Microscopy (TEM):** Used to observe the size and dispersion of the nanoparticles at a higher resolution, confirming the nanoscale structure of the materials.
- **UV-Vis Diffuse Reflectance Spectroscopy (DRS):** Measured the optical properties of the photocatalysts, including their bandgap energies, which were essential for assessing their photocatalytic potential under visible light.

- **Brunauer–Emmett–Teller (BET) Surface Area Analysis:** Provided data on the specific surface area and pore volume of the photocatalysts, crucial for understanding their adsorption capabilities and overall photocatalytic efficiency.

4.2.6.3 Photocatalytic Degradation Setup

The photocatalytic degradation experiments were conducted in a batch reactor equipped with a visible light source. A 105 W compact fluorescent lamp was used as the light source, providing illumination in the visible range, which is appropriate for activating the BiFeO3-based photocatalysts. The reactor was equipped with a magnetic stirrer to maintain homogeneity in the reaction mixture.

A stock solution of malachite green (MG) dye was prepared at a concentration of 1000 mg/L and was diluted according to the experimental requirements. The experiments were carried out by adding a known quantity of the photocatalyst to 100 mL of MG solution in the reactor. Before the light was turned on, the solution was stirred in the dark for 30 minutes to establish adsorption-desorption equilibrium between the dye and the photocatalyst surface. The light was then switched on, and the reaction was allowed to proceed under continuous stirring.

4.2.6.4 Sampling and Analysis

Samples were taken at regular intervals during the photocatalytic reaction to monitor the degradation of MG. The samples were immediately filtered using syringe filters to remove photocatalyst particles and analyzed using a UV-Vis spectrophotometer. The absorbance of the MG solution was measured at 616 nm, the maximum absorbance wavelength of MG. The degradation efficiency (R%) was calculated using the following equation:

$$R(\%) = \frac{C_i - C_f}{C_i} \times 100$$

Where, C_i & C_f are the initial and final concentration of the MG dye, respectively.

4.2.6.5 Control Experiments

To ensure the reliability of the results, control experiments were conducted under identical conditions without the photocatalyst to account for any possible photolysis of MG under light irradiation. Additionally, experiments were conducted in the dark with the photocatalyst to assess any adsorption effects.

4.2.7 Data Analysis and Machine Learning Modelling

4.2.7.1 Data Collection and Preprocessing

The study generated a substantial dataset comprising 1200 data points, representing various experimental conditions. The input variables included:

- Catalyst type (e.g., pure BiFeO3, Pd-BiFeO3, Pt-BiFeO3, Ag-BiFeO3)

- Reaction time (minutes)
- Light intensity (W)
- Initial concentration of MG dye (mg/L)
- Catalyst loading (g/L)
- Solution pH
- Humic acid concentration (mg/L)
- Presence of anions (e.g., Cl^-, SO_4^{2-}, CO_3^{2-}, HCO_3^-, HPO_4^{2-})
- Surface area (m^2/g)
- Pore volume (cm^3/g)

These input variables were normalized to ensure compatibility with the machine learning models. Categorical variables such as catalyst type and anions were encoded using one-hot encoding, a process that converts these variables into a binary matrix to allow for their inclusion in the model training.

4.2.7.2 Selection and Training of Machine Learning Models

Twelve machine learning models were selected for predicting the photocatalytic degradation efficiency of MG. These models included a combination of tree-based algorithms, gradient boosting methods, and neural networks:

- **Tree-Based Models:** Decision Tree, ExtraTrees, Bagging, and Random Forest
- **Gradient Boosting Models:** CatBoost, Gradient Boosting, Hist Gradient Boosting, LightGBM, and XG Boost
- **Neural Networks:** Artificial Neural Network (ANN) and Light Long Short-Term Memory (Light LSTM)
- **Gaussian Process Model**

The dataset was divided into training (70%) and testing (30%) subsets. The models were trained using the training data, and hyperparameter tuning was performed to optimize their performance. The trained models were then evaluated on the test data, and their predictive accuracy was assessed using the coefficient of determination (R^2), mean absolute error (MAE), and root mean square error (RMSE).

4.2.7.3 Performance Evaluation of Models

The performance of the models was evaluated based on their ability to accurately predict the photocatalytic degradation efficiency of MG. The CatBoost model emerged as the most accurate, achieving an R^2 value of 0.998 on the test data, with the lowest MAE and RMSE

values of 0.641 and 1.343, respectively. This model demonstrated superior generalization capability, closely matching the experimental data across a wide range of conditions.

4.2.7.4 Feature Importance and Partial Dependence Analysis

To understand the contribution of each input variable to the model's predictions, feature importance analysis was performed. The analysis revealed that reaction time was the most critical factor, followed by the initial concentration of MG, catalyst loading, and light intensity. The partial dependence plots (PDPs) provided further insights into how changes in individual variables influenced the degradation efficiency, offering a visual representation of the interactions between different factors.

4.2.8 Implications, Limitations, and Future Directions

4.2.8.1 Implications for Environmental Remediation

The successful application of machine learning to predict the photocatalytic degradation of MG represents a significant advancement in the field of environmental remediation. The ability to accurately model and predict the performance of photocatalysts under various conditions can accelerate the development of more efficient water treatment technologies. This approach can be extended to other pollutants and photocatalytic systems, offering a scalable solution for tackling water pollution on a larger scale.

4.2.8.2 Limitations of the Study

Despite the promising results, the study has several limitations:

- **Scope of Pollutants:** The study focused exclusively on MG, a single organic dye. The performance of the models on other pollutants, especially those with different chemical properties, needs further investigation.
- **Experimental Conditions:** The experiments were conducted under controlled laboratory conditions, which may not fully represent real-world scenarios where multiple pollutants coexist, and varying environmental factors come into play.
- **Model Generalization:** While the CatBoost model performed exceptionally well in this study, its generalization to other datasets and environmental conditions requires further validation.

4.2.8.3 Future Research Directions

Future research could explore the following areas to build on the findings of this study:

- **Expansion to Other Pollutants:** Extending the machine learning modelling approach to a broader range of pollutants, including pharmaceuticals, pesticides, and industrial chemicals, would provide a more comprehensive understanding of photocatalytic degradation processes.

- **Field Trials:** Conducting field trials to test the effectiveness of the ML-predicted photocatalysts in real-world wastewater treatment applications would help validate the models and refine their predictive accuracy.
- **Hybrid Models:** Developing hybrid models that combine machine learning with mechanistic modelling could offer more robust predictions by integrating empirical data with theoretical insights.

4.2.9 Conclusion

This research demonstrates the potential of integrating machine learning with advanced photocatalytic processes for the efficient degradation of harmful organic pollutants like malachite green. The use of $BiFeO_3$-based photocatalysts, particularly when enhanced with noble metals, showed significant promise under visible light irradiation. The CatBoost model's success in predicting photocatalytic performance marks a step forward in environmental modelling, offering a powerful tool for optimizing water treatment processes. The insights gained from this study pave the way for future innovations in the field of photocatalysis and environmental remediation.

The findings of this study highlight the significant potential of machine learning (ML) models in predicting and optimizing the photocatalytic degradation of organic pollutants, specifically malachite green (MG) dye, in aqueous solutions. The integration of ML with traditional photocatalytic methods offers a transformative approach to environmental remediation, enabling more efficient and targeted strategies for pollutant degradation.

4.2.10 Key Contributions

- **Development of a Comprehensive Dataset:** The study compiled a detailed dataset consisting of 1200 data points covering various experimental conditions. This extensive dataset provided a robust foundation for training and validating the ML models, ensuring accurate and reliable predictions.
- **Superior Performance of CatBoost Model:** Among the 12 ML models tested, CatBoost was identified as the most effective in predicting MG dye degradation. Its high accuracy, demonstrated by an R^2 value of 0.998, underscores the model's capability in capturing the complex interactions between multiple input variables.
- **Identification of Critical Process Parameters:** Through feature importance and partial dependence analyses, the study identified key factors that significantly influence photocatalytic performance. Reaction time, initial dye concentration, and catalyst loading were found to be the most critical parameters, providing valuable insights for optimizing photocatalytic processes.

4.2.11 Broader Impact and Future Prospects

The application of machine learning to environmental science, as demonstrated in this study, opens up new avenues for research and practical applications. The ability to predict the outcomes of complex chemical processes such as photocatalysis has profound implications for industrial-scale water treatment and pollution control. By reducing the need for extensive trial-and-error experimentation, ML can streamline the development of more efficient photocatalysts and treatment protocols, ultimately leading to cleaner water and a healthier environment.

4.2.12 Challenges and Considerations

While the study achieved promising results, several challenges and considerations must be addressed in future research:

- **Model Transferability:** The transferability of the CatBoost model and other ML models to different pollutants and environmental conditions remains a critical challenge. Further studies are needed to validate the models across a broader spectrum of pollutants and under varying real-world conditions.
- **Scalability of Photocatalytic Systems:** The scalability of the laboratory-based photocatalytic systems to industrial-scale applications requires further investigation. Factors such as catalyst regeneration, energy consumption, and the cost-effectiveness of the photocatalytic process need to be considered.
- **Incorporation of Additional Variables:** Future research could enhance model accuracy by incorporating additional variables such as temperature, the presence of other organic and inorganic contaminants, and varying light sources. These factors could provide a more comprehensive understanding of the photocatalytic degradation process.

4.3 Literature review on Paper 2: A Hybrid Machine Learning-Genetic Algorithm (ML-GA) Model to Predict Optimal Process Parameters of Nanoporous Cu for CO_2 Reduction

4.3.1 Introduction

The rapid increase in atmospheric carbon dioxide (CO_2) levels due to industrial activities and fossil fuel combustion has significantly contributed to climate change. Reducing CO_2 emissions and finding efficient ways to convert CO_2 into valuable products has become a priority for researchers. One promising approach is the electrochemical CO_2 reduction reaction (CO_2RR), where CO_2 is converted into useful hydrocarbons such as ethylene, ethanol, and acetic acid using electrical energy. Among various catalysts, copper (Cu) has been identified as a highly effective material for CO_2 reduction due to its ability to produce C_2 products (i.e., hydrocarbons with two carbon atoms).

In recent years, nanoporous copper (NPC) has emerged as a promising catalyst for CO_2RR due to its high surface area and ability to control local concentrations at the catalyst-reactant interface. However, the performance of NPC in CO_2RR can vary depending on the preparation and reaction conditions. Therefore, optimizing the process parameters of NPC electrodes is crucial for maximizing the faradaic efficiency (FE) of C_2 products.

In this study, the authors developed a hybrid machine learning-genetic algorithm (ML-GA) model to predict the optimal process parameters of NPC for CO2RR. The model aims to improve the electrocatalytic performance of NPC by optimizing five electrode process parameters and five faradaic efficiencies of major products. The study demonstrates the potential of machine learning in identifying the best process parameters for CO_2 reduction using NPC.

4.3.2 Experimental Procedure

4.3.2.1 Data Curation and Feature Selection

The study's dataset was based on 82 data records of NPC electrochemical performances during CO2RR in a flow cell setup. The input features included NPC type, mass, catalytic mass loading, KOH concentration, and applied current, while the output features were the faradaic efficiencies of five major products: hydrogen, carbon monoxide, ethylene, acetic acid, and ethanol. The dataset was divided into training and validation sets, with 90% of the data used for training and 10% for validation.

4.3.2.2 Machine Learning Model Buildup and Validation

The authors tested three machine learning regression algorithms: extreme gradient boosting regression (XGBR), gradient boosting regression (GBR), and random forest regression (RF). The algorithms were evaluated based on several metrics, including mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of

determination (R2). The best-performing algorithm was then combined with a genetic algorithm to predict the optimal process parameters for maximizing C_2 faradaic efficiency.

4.3.3 Results and Discussion

4.3.3.1 Machine Learning Model Buildup and Results

4.3.3.1.1 Input/Output Correlations

The heatmap of correlations between input and output features showed that the applied current had the highest correlation with C_2 products, particularly ethylene and ethanol. This finding is consistent with previous studies that suggest higher applied potentials favours the formation of C_2 products.

4.3.3.1.2 Regression Model Results and Combination with Genetic Algorithm

Among the three machine learning algorithms, GBR and RF outperformed XGBR in terms of prediction accuracy, with GBR showing the highest R2 value. The genetic algorithm was then used in combination with the best-performing ML model to optimize the process parameters. The XGBR-GA model was ultimately selected for validation due to its balance between prediction accuracy and faradaic efficiency constraints.

4.3.3.1.3 Validation Testing and Results

The validation test using the XGBR-GA model's predicted process parameters showed that the experimental faradaic efficiency of C_2 products (66.71%) closely matched the predicted value (68.25%). Additionally, the space-time yield of C_2 products in this work (4693 mmol/h*g) was significantly higher than that of other pure Cu catalysts, demonstrating the effectiveness of the ML-GA model in optimizing NPC electrodes for CO_2RR.

4.3.3.2 Physical Significances of the Inputs

4.3.3.2.1 Type of Nanoporous Copper

The study used two types of Cu-Al alloy precursors ($Cu_{18}Al_{82}$ and $Cu_{33}Al_{67}$) to produce NPC, with $Cu_{18}Al_{82}$ showing better electrochemical properties due to its hierarchical structure.

4.3.3.2.2 Mass of Nanoporous Copper Powder (Ionomer/Catalyst Ratio)

The ionomer/catalyst ratio, specifically the Nafion concentration, influenced the CO_2RR performance. A lower Nafion concentration (10 wt%) was found to enhance C_2 product formation by minimizing hydrogen evolution reaction (HER).

4.3.3.2.3 Catalytic Mass Loading on the Gas Diffusion Layer

Optimal mass loading (0.337 mg/cm^2) ensured a balance between low HER and high CO_2RR rates, maximizing C_2 product formation.

4.3.3.2.4 Concentration of KOH Electrolyte

A higher KOH concentration (2.84 M) was found to enhance CO_2RR by lowering the reaction's onset potential and suppressing HER.

4.3.3.2.5 Applied Current Density on Flow Cell

The applied current density (772 mA/cm^2) was critical for maximizing C_2 product formation while preventing GDL flooding and minimizing HER.

4.3.3.2.6 Summary of Physical Significances

The study's ML-GA model successfully identified optimal process parameters that enhanced C_2 product formation while maintaining process stability, demonstrating its potential for large-scale CO_2 reduction applications.

4.3.4 Conclusion

This study introduced a hybrid machine learning-genetic algorithm (ML-GA) model to optimize the process parameters of nanoporous copper (NPC) electrodes for CO_2 reduction reaction (CO2RR). By combining machine learning with a genetic algorithm, the authors identified the optimal conditions for maximizing the faradaic efficiency of C_2 products. The validation test confirmed the model's accuracy, with the experimental results closely matching the predicted values. The study highlights the potential of ML-GA models in optimizing catalytic processes and improving the performance of CO_2 reduction technologies.

4.4 Literature Review of Paper 3: Implementing Machine Learning and Statistical Optimization Analysis for Oily Wastewater Treatment in a Photocatalytic membrane reactor

4.4.1 Introduction

The integration of machine learning (ML) and statistical optimization techniques has become increasingly important in enhancing the accuracy and efficiency of chemical process predictions. As chemical processes often involve complex interactions between multiple variables, traditional methods of optimization may fall short in capturing the full scope of these interactions. Machine learning, with its ability to handle large datasets and identify patterns, offers a promising approach to improving predictive models in the chemical industry.

In this paper, the authors present a novel approach that combines machine learning algorithms with statistical optimization techniques to predict and optimize chemical processes. The goal of the study is to enhance the accuracy of predictions while reducing the time and resources required for experimental trials. By applying this hybrid approach to a specific chemical process, the authors aim to demonstrate its effectiveness in optimizing process parameters and improving overall performance.

4.4.2 Experimental Procedure

4.4.2.1 Data Collection and Preprocessing

The dataset used in this study was collected from a series of chemical process experiments. The data included various input parameters, such as temperature, pressure, and reactant concentrations, as well as output variables, including yield, purity, and reaction time. The data were reprocessed to remove any outliers or missing values, ensuring the accuracy of the subsequent machine learning models.

4.4.2.2 Machine Learning Model Development

Several machine learning algorithms were tested in this study, including support vector machines (SVM), random forest (RF), and artificial neural networks (ANN). The performance of each algorithm was evaluated based on metrics such as mean squared error (MSE), mean absolute error (MAE), and R-squared (R_2). The best-performing algorithm was then selected for further optimization.

4.4.2.3 Statistical Optimization

After identifying the most accurate machine learning model, the authors applied statistical optimization techniques to further refine the process parameters. Techniques such as response surface methodology (RSM) and design of experiments (DOE) were used to identify the optimal conditions for the chemical process. These techniques allowed the

authors to explore the interactions between variables and determine the best combination of parameters for maximizing the desired output.

4.4.3 Results and Discussion

4.4.3.1 Machine Learning Model Performance

The results of the machine learning model development phase showed that the random forest algorithm outperformed the other models in terms of prediction accuracy. The random forest model achieved the lowest mean squared error and highest R-squared value, indicating its effectiveness in capturing the complex relationships between input and output variables in the chemical process.

4.4.3.2 Statistical Optimization Results

The application of statistical optimization techniques further improved the performance of the chemical process. By using response surface methodology, the authors were able to identify the optimal combination of temperature, pressure, and reactant concentrations that maximized yield and purity while minimizing reaction time. The optimized process parameters were then validated through additional experiments, which confirmed the accuracy of the predictions.

4.4.3.3 Comparison with Traditional Methods

The hybrid approach of combining machine learning and statistical optimization was compared with traditional methods of process optimization. The results showed that the hybrid approach significantly outperformed traditional methods in terms of prediction accuracy and efficiency. By reducing the number of experimental trials required to achieve optimal results, the hybrid approach also demonstrated its potential for cost savings in the chemical industry.

4.4.4 Conclusion

This study presented a novel approach to chemical process optimization by integrating machine learning algorithms with statistical optimization techniques. The hybrid approach proved to be highly effective in enhancing prediction accuracy and optimizing process parameters. The results of the study highlight the potential of this approach for improving the efficiency and performance of chemical processes, making it a valuable tool for the industry.

The authors suggest that future research should explore the application of this hybrid approach to other types of chemical processes and industries. By expanding the scope of the hybrid machine learning and statistical optimization framework, the potential benefits can be extended to a wider range of applications, from pharmaceuticals to materials science. Furthermore, integrating additional machine learning techniques such as deep learning or

reinforcement learning could further enhance the capabilities of the model, allowing it to tackle even more complex and dynamic chemical processes.

4.4.5 Discussion of Practical Implications

The implications of this study extend beyond the specific chemical process investigated. The ability to predict and optimize processes using machine learning and statistical methods presents significant advantages for various industries. The potential for cost savings through reduced experimental trials, increased efficiency, and improved product quality can offer competitive advantages in sectors such as pharmaceuticals, petrochemicals, and specialty chemicals. The integration of this hybrid approach into existing process control systems could enable real-time optimization and predictive maintenance, leading to safer, more reliable operations.Moreover, the scalability of machine learning models allows for their application across multiple production scales, from laboratory research to full-scale manufacturing. The flexibility of the approach means it can be tailored to different processes, whether they involve batch or continuous operations. This adaptability further underscores the versatility of combining machine learning with traditional statistical methods.

4.4.6 Challenges and Limitations

While the study demonstrates the effectiveness of the hybrid machine learning-statistical optimization approach, several challenges and limitations must be acknowledged. One challenge lies in the quality and quantity of data available for training the machine learning models. In chemical processes, data collection can be time-consuming and expensive, and the accuracy of the predictions is highly dependent on the quality of the input data. Additionally, while machine learning models can identify patterns in data, they may not always provide insights into the underlying chemical mechanisms, which could be crucial for understanding and improving the process.

Another limitation is the computational cost associated with training complex machine learning models, particularly when dealing with large datasets or multi-objective optimization problems. While advancements in computing power and algorithms continue to mitigate these challenges, it is essential to consider the trade-offs between computational resources and prediction accuracy.

Finally, the generalization of the model to other chemical processes may require additional modifications and validations. While the hybrid approach has shown promise for the specific case studied, its applicability to other systems must be tested to ensure robustness and reliability.

4.4.7 Future Directions

Future research should focus on addressing the challenges and limitations identified in this study. For instance, efforts could be made to develop more efficient data collection methods

and improve data quality through sensor technologies and real-time monitoring. Additionally, combining machine learning models with mechanistic models could provide a more comprehensive understanding of the chemical processes, leading to more informed optimization strategies.

Further exploration of different machine learning algorithms, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may offer new avenues for improving prediction accuracy. Moreover, hybrid models that integrate reinforcement learning could enable adaptive optimization, where the model continuously learns and updates its predictions based on real-time feedback from the process.

Another promising direction is the application of this hybrid approach to sustainability and green chemistry initiatives. By optimizing chemical processes for energy efficiency, waste reduction, and the use of renewable resources, the hybrid machine learning-statistical optimization framework could contribute to the development of more sustainable industrial practices.

4.4.8 Conclusion

The integration of machine learning and statistical optimization techniques offers a powerful tool for enhancing the prediction and optimization of chemical processes. The study demonstrated the effectiveness of this hybrid approach in improving the accuracy of predictions and optimizing process parameters, leading to significant improvements in efficiency, yield, and cost savings. While challenges remain, the potential benefits of this approach make it a valuable addition to the toolbox of chemical engineers and process designers.

The future of chemical process optimization lies in the continued development and integration of machine learning and statistical methods, enabling more efficient, sustainable, and innovative processes across a wide range of industries. By addressing the challenges and exploring new research directions, this hybrid approach can unlock new possibilities for the future of chemical engineering.

4.5 Conclusive Summary of Literature Review

The literature review has highlighted the foundational concepts, methodologies, and recent advancements in environmental modelling, photocatalysis, machine learning, and synthetic data generation, all of which connect directly to the core objectives of this research. Through a comprehensive examination of studies focused on pollutant degradation, photocatalytic processes, and predictive modelling, several key findings emerge that support the current investigation on predicting Florasulam degradation in the presence of TiO_2 nanoparticles. This summary synthesizes the primary insights from each segment of the literature and discusses how they collectively align with the current study's goals, highlighting areas for further research and development.

4.5.1 Key Findings and Their Relevance to This Study

4.5.1.1 Environmental Impact of Florasulam and Role of Nanoparticles :

The reviewed studies emphasize the significant environmental risks posed by Florasulam due to its persistence in water bodies, particularly in agricultural runoff. The persistence and behaviour of Florasulam in water are influenced by environmental parameters, especially pH and nanoparticle interactions. Literature on TiO_2 nanoparticles highlights their role in accelerating the degradation of pollutants through photocatalysis, especially under UV light. Studies confirm that TiO_2 nanoparticles can significantly enhance the breakdown of organic contaminants, which justifies the use of these nanoparticles in this study to examine their catalytic effect on Florasulam degradation. The comprehensive understanding of TiO_2's properties, including surface area and photocatalytic potential, forms a critical basis for assessing the effectiveness of machine learning models in predicting Florasulam's degradation under various conditions.

4.5.1.2 Machine Learning Applications in Environmental Science :

The review on machine learning applications in environmental science reveals that ML models, particularly ensemble methods such as Random Forest, Support Vector Machines, XGBoost, and CatBoost, have been successfully applied to model complex environmental interactions. ML models offer superior predictive accuracy for scenarios involving nonlinear relationships, making them suitable for studies on pollutant behaviour in varied conditions. This research builds on the successes of these previous applications by employing similar algorithms to predict Florasulam degradation based on input variables such as pH, TiO_2 concentration, and initial Florasulam concentration. The insights into ML model selection, hyperparameter tuning, and performance metrics, like Mean Squared Error (MSE) and R^2 score, provide essential guidance for model development and evaluation in this study.

4.5.1.3 Synthetic Data Generation for Enhanced Model Robustness :

Synthetic data generation emerges in the literature as a valuable tool in environmental modelling, particularly when experimental data are scarce or difficult to collect. The review emphasizes that synthetic data can effectively augment experimental datasets, enhancing model robustness and predictive accuracy. This study leverages synthetic data generation to supplement the experimental dataset, ensuring that machine learning models are adequately trained across a diverse range of scenarios. Validating the synthetic data through statistical tests, as recommended in the literature, ensures the reliability of these data points and strengthens the predictive power of the machine learning models.

4.5.1.4 Photocatalytic Processes and Optimal Conditions for Pollutant Degradation

Studies on photocatalysis underscore the importance of optimizing parameters such as pH, light exposure, and nanoparticle concentration to achieve maximum degradation efficiency. The findings emphasize that optimal TiO_2 concentration is critical, as excessive nanoparticle amounts may cause agglomeration, reducing surface area and catalytic activity. These insights directly relate to the current research, where varying TiO_2 concentrations and pH levels are tested to identify conditions that maximize Florasulam degradation. The use of photocatalysis as a method aligns with previous studies, supporting the inclusion of machine learning models to predict degradation patterns under controlled conditions.

4.5.2 Scope for Further Discussion and Research:

While the literature provides a robust foundation for this study, several avenues remain for further investigation:

4.5.2.1 Extension to Other Environmental Contaminants

The current research is limited to Florasulam; however, the approach developed here can be extended to other herbicides and environmental contaminants. Future studies could apply similar machine learning models to predict degradation behaviour for a broader range of organic and inorganic pollutants, thus expanding the practical applications of this methodology in environmental science.

4.5.2.2 Hybrid Models for Enhanced Predictive Accuracy

Several studies suggest that combining machine learning models with mechanistic or physical models may offer enhanced predictive accuracy. Hybrid models that integrate empirical data with theoretical insights, especially in photocatalysis, could provide more robust predictions. Future research could explore such hybrid approaches to achieve improved model generalization and adaptability across diverse environmental conditions.

4.5.2.3 Field Validation of Machine Learning Models

While this study relies on controlled laboratory data, applying machine learning models in field conditions remains a challenge. Future research could focus on validating the

predictive models under real-world conditions where environmental factors, such as multiple contaminants and fluctuating conditions, add complexity. Field trials would be instrumental in establishing the practicality and reliability of ML-based predictions for environmental monitoring and remediation.

4.5.2.4 Advanced Data Augmentation Techniques

Although synthetic data generation has shown promise, further exploration of advanced data augmentation techniques, such as generative adversarial networks (GANs) and bootstrapping, could improve the quality and diversity of synthetic data. Future studies could investigate these methods to create more representative datasets that better capture the variability observed in real environmental systems, thus enhancing model robustness.

4.5.2.5 Real-time Monitoring and Predictive Maintenance Applications

Integrating machine learning models with real-time monitoring systems could enable predictive maintenance and timely interventions for environmental remediation processes. By combining sensor technologies with predictive models, future research could develop automated systems for monitoring Florasulam levels and other contaminants, optimizing treatment processes in real-time for effective pollution control.

4.6 CONCLUSION

The literature review underscores the potential of combining machine learning with photocatalytic processes and synthetic data generation for environmental modelling. This study builds upon these insights by focusing on Florasulam degradation in the presence of TiO_2 nanoparticles, using ML models to predict degradation under varied environmental conditions. The findings from previous research validate the methodology adopted here and highlight the broader implications for environmental science. The avenues for future research, including hybrid models, field validation, and real-time applications, open new possibilities for advancing environmental monitoring and pollutant remediation strategies. Through this integrative approach, this study not only contributes to the existing knowledge base but also paves the way for innovative, data-driven solutions in the field of environmental science.

5 STATEMENT OF THE PROBLEM

5.1 PREAMBLE

Florasulam, a widely utilized herbicide, poses significant environmental concerns due to its persistence and potential accumulation in aquatic systems. Its degradation, however, is notably influenced by several environmental variables—specifically, the pH of the solution, the concentration of nanoparticles (such as TiO_2), and the initial concentration of Florasulam. These factors not only affect the rate of degradation but also the interactions between Florasulam molecules and TiO_2 nanoparticles, influencing its overall behaviour in natural and engineered water systems.

5.1.1 Role of pH

pH is a crucial parameter that alters both the surface charge of TiO_2 nanoparticles and the ionization state of Florasulam. At different pH levels, the degradation process of Florasulam may vary dramatically. For instance, alkaline conditions enhance the degradation rate by increasing Florasulam's susceptibility to photocatalytic reactions facilitated by TiO_2 nanoparticles. In acidic conditions, however, the degradation rate is slower, likely due to changes in the surface interaction dynamics between Florasulam and TiO_2. The pH level also affects the Point of Zero Charge (PZC) of TiO_2, around pH 6.5, influencing the nanoparticle's charge state and thus its interaction with Florasulam, which has a pKa of 4.54. This interaction becomes more complex with the pH-dependent alteration in Florasulam's molecular structure and its adsorption behaviour on TiO_2 surfaces, further impacting its degradation potential in water systems.

5.1.2 Influence of TiO_2 Nanoparticle Concentration

Titanium Dioxide (TiO_2) nanoparticles are known for their strong photocatalytic properties, especially under UV light. The concentration of these nanoparticles in an aqueous medium significantly affects the degradation rate of Florasulam. At higher concentrations, more reactive oxygen species are generated upon UV irradiation, which enhances the photocatalytic breakdown of Florasulam. However, an optimal concentration range is essential, as overly high concentrations may lead to agglomeration, reducing the surface area available for reaction and potentially hindering the photocatalytic efficiency. This delicate balance highlights the importance of optimizing TiO_2 concentration for efficient Florasulam degradation and illustrates the complex interplay between nanoparticle concentration and photocatalytic efficiency in environmental systems.

5.1.3 Initial Concentration of Florasulam

The initial concentration of Florasulam in the aqueous medium also influences its degradation rate and efficiency. Higher concentrations may increase the likelihood of molecular aggregation, potentially inhibiting interactions with TiO_2 and reducing photocatalytic efficiency. Conversely, at lower concentrations, the Florasulam molecules

are more likely to be dispersed uniformly, enhancing their accessibility to TiO_2 nanoparticles. The degradation pattern as a function of initial concentration is complex and highlights the need for precise predictions to ensure effective breakdown, especially in real-world applications where concentration levels may vary significantly.

5.1.4 Significance of Machine Learning for Predictive Modelling

Traditional methods struggle to account for the complex and often nonlinear interactions between these variables. Machine learning (ML) models offer a compelling solution by enabling accurate predictions of Florasulam degradation based on the above parameters. The advantage of ML in this context lies in its ability to model and generalize intricate relationships across large datasets, accommodating variable environmental factors. By training an ML model on experimental data, predictions can be made for Florasulam degradation under new conditions, including varying pH levels, TiO_2 concentrations, and initial Florasulam concentrations. This predictive capability not only facilitates targeted environmental management but also offers a pathway for modelling other pesticides, as the model can be adapted to incorporate different compounds by inputting relevant physicochemical properties and environmental parameters.

5.1.5 Broader Implications

The successful application of machine learning in this context underscores its potential beyond Florasulam. Similar modelling techniques could be extended to predict the behaviour of a wide range of pesticides and other persistent environmental contaminants, aiding in proactive pollution management strategies. By identifying optimal conditions for pollutant degradation in various environments, machine learning models can support the development of more efficient water treatment systems, enabling regulatory bodies and environmental agencies to make informed decisions based on reliable data-driven insights.

5.2 CONCLUSION

The degradation of Florasulam is governed by a complex interplay of environmental factors, with pH, TiO_2 concentration, and initial Florasulam levels playing pivotal roles. Machine learning provides a robust framework for predicting these interactions, offering a scalable, adaptable approach to environmental modelling that can ultimately be applied across a range of contaminants and settings. This research, therefore, holds substantial promise for advancing the fields of environmental science and pollution control through precise, efficient predictive modelling.

6.1 EXPERIMENTAL DESIGN AND DATA COLLECTION

The experimental data for this study was collected from controlled laboratory experiments designed to measure the concentration of Florasulam in aqueous solutions containing varying concentrations of TiO2 nanoparticles. The experiments were conducted under different pH conditions (pH 3, pH 7, pH 10, and pH 12) and exposure to UV light.

6.1.1 Experimental Setup

The experimental setup involved preparing aqueous solutions of Florasulam at a concentration of 10 mg/L. TiO2 nanoparticles were then added to these solutions at different concentrations (10 mg/L, 20 mg/L, 50 mg/L, 80 mg/L, 110 mg/L, and 150 mg/L). The pH of the solutions was adjusted using standard buffer solutions, and the samples were exposed to UV light for varying durations (0 to 120 hours).

6.1.2 Data Collection Methods

Data was collected at regular intervals by measuring the concentration of Florasulam in each solution using high-performance liquid chromatography (HPLC). The measurements were taken at different time points (0, 3, 6, 12, 24, 36, 48, 60, 72, hours, 84 hours, 96 hours, 108 hours, and 120 hours) to capture the degradation profile of Florasulam under each experimental condition. The concentration data was recorded for both the UV-exposed and sunlight-exposed samples, as well as for each pH level and TiO2 nanoparticle concentration.

6.2 DATA PREPROCESSING

The raw data collected from the experiments required several preprocessing steps before it could be used for modelling. These steps included:

6.2.1 Data Cleaning and Handling Missing Values

The dataset was first inspected for any missing or inconsistent values. Missing data can arise due to experimental errors or equipment malfunctions. To address this, missing values were handled using interpolation methods, followed by forward and backward filling to ensure that no gaps remained in the dataset.

6.2.2 Data Merging and Transformation

Given that data was collected under different experimental conditions, the datasets were merged into a single comprehensive dataset. This involved combining data from different CSV files based on the time intervals (Time (Hour)) to create a unified dataset that included all relevant variables such as pH levels, TiO2 concentrations, and Florasulam concentrations under various conditions.

6.2.3 Statistical Analysis of Data

A statistical analysis was conducted to understand the distribution and central tendencies of the data. This included calculating mean, median, standard deviation, skewness, and kurtosis for each variable. The analysis provided insights into the variability and distribution characteristics of the data, which are crucial for model selection and performance evaluation.

6.2.4 Probability Distribution Fitting

To generate synthetic data that accurately reflects the real-world data, it was essential to determine the best-fit probability distribution for each variable. A range of probability distributions, including Weibull (Min and Max), Normal, Beta, and Gamma distributions, were tested using the Kolmogorov-Smirnov (K-S) test to assess their goodness of fit. The Beta distribution was identified as the best fit for most variables, based on its low D value and high p-value.

6.3 Synthetic Data Generation

6.3.1 Rationale for Synthetic Data

The generation of synthetic data was necessary to augment the experimental dataset, providing more robust and comprehensive training data for the machine learning models. Synthetic data is particularly useful in scenarios where real-world data is limited or where the experimental setup is difficult to replicate multiple times.

6.3.2 Generation Process

Synthetic data was generated using the Beta distribution parameters derived from the real data. A function was implemented to automate the generation of synthetic data for each relevant variable, excluding the time variable (Time (Hour)). This process produced synthetic datasets that closely mimicked the statistical properties of the original data.

6.3.3 Validation of Synthetic Data

The synthetic data was validated using a combination of statistical measures and visual inspections. The Kolmogorov-Smirnov (K-S) test was applied to compare the distribution of synthetic data against the original data. Additionally, histograms and Q-Q plots were generated to visually inspect the alignment of the synthetic data with the real data. The results indicated that the synthetic data was a reliable representation of the experimental data, thus suitable for training the machine learning models.

6.4 Machine Learning Model Selection

6.4.1 Overview of Selected Models

Four machine learning models were selected for this study based on their popularity and effectiveness in regression tasks involving complex datasets:

- **CatBoost Regressor**: Known for its ability to handle categorical variables without explicit encoding, CatBoost is a gradient boosting algorithm that provides high accuracy and efficient training times.
- **Random Forest Regressor**: An ensemble learning method that operates by constructing multiple decision trees during training. It is robust to overfitting and capable of handling large datasets with high-dimensional features.
- **XGBoost Regressor**: An implementation of gradient boosting that is optimized for speed and performance. XGBoost is widely used for its superior handling of missing values and its ability to capture complex relationships in data.
- **Support Vector Machine (SVM) Regressor**: A powerful model that uses a kernel-based approach to find the optimal hyperplane for regression tasks. SVM is particularly effective in scenarios involving high-dimensional data.

6.4.2 Model Training and Testing

The merged dataset, including both original and synthetic data, was split into training and testing sets, with 80% of the data allocated for training and 20% for testing. Each model was trained using the training set, with hyperparameters tuned through grid search or cross-validation methods to optimize performance.

The following sections provide detailed descriptions of the training and testing processes for each model:

6.4.2.1 CatBoost Regressor

The CatBoost model was trained with the following hyperparameters: 1000 iterations, a learning rate of 0.1, and a depth of 6. The model's ability to handle categorical data natively was particularly beneficial in this study, where the categorical nature of certain variables could introduce complexity into the modeling process. The model was evaluated on the test set using performance metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 score.

6.4.2.2 Random Forest Regressor

The Random Forest model was configured with 100 decision trees and a maximum depth of 10 to balance between computational efficiency and model complexity. This model's ensemble approach was expected to provide robustness against overfitting, especially given the synthetic nature of some of the data. Performance was measured using the same metrics as for CatBoost.

6.4.2.3 XGBoost Regressor

XGBoost was trained with 500 iterations, a learning rate of 0.05, and a maximum depth of 5. Known for its regularization techniques, XGBoost was expected to handle the synthetic

data effectively, minimizing the risk of overfitting while maintaining high prediction accuracy.

6.4.2.4 Support Vector Machine (SVM) Regressor

The SVM model used a radial basis function (RBF) kernel with a regularization parameter of 1.0. SVM's strength lies in its ability to manage high-dimensional data, which was particularly relevant for this study given the number of features involved. However, SVM's training time was expected to be longer compared to the other models due to the computational complexity of the kernel operations.

6.4.3 Performance Evaluation Metrics

The models were evaluated based on the following metrics:

- **Mean Squared Error (MSE)**: Measures the average of the squares of the errors or deviations—that is, the difference between the observed and predicted values. A lower MSE indicates better predictive accuracy.
- **Mean Absolute Error (MAE)**: Represents the average absolute difference between observed and predicted values. MAE is less sensitive to outliers compared to MSE.
- **R^2 Score**: Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 score closer to 1.0 suggests a better fit of the model to the data.

6.4.4 Cross-Validation

To ensure the robustness of the models, cross-validation was employed, where the dataset was divided into multiple folds. The models were trained and validated on different combinations of these folds, and the results were averaged to obtain a more reliable estimate of model performance.

6.4.5 Hyperparameter Tuning

Hyperparameter tuning was performed using grid search for each model to identify the optimal set of parameters that minimized the prediction error on the validation set. This process involved testing different combinations of parameters, such as the learning rate and depth for CatBoost and XGBoost, the number of trees and maximum depth for Random Forest, and the kernel type and regularization parameter for SVM.

Time (Hour)	Concentration of Florasulam in mgL^{-1}. [remaining in the aqueous system without any catalyst under the different source of irradiation light].	
	Blank UV	Blank Sunlight
0	10.6	10.86
3	10.42	10.72
6	10.15	10.57
12	9.86	10.28
24	9.41	10.05
36	8.82	9.77
48	8.3	9.3
60	8	8.92
72	7.45	8.46
84	6.53	8
96	5.65	7.1
108	4.8	6.59
120	4.15	5.66

Table 1: Study of different sources of irradiation light on photodegradation of Florasulam in the aqueous medium

Time (Hour)	Concentration of Florasulam in mgL^{-1}. [remaining in the aqueous system at different pH in presence of bulk TiO_2 (50 mgL^{-1}) photocatalyst separately under UV irradiation].			
	pH 3	pH 7	pH 10	pH 12
0	10.65	10.73	10.68	10.71
3	10.41	10.38	10.36	10.4
6	9.98	9.9	9.94	9.96
12	9.39	9.15	9.3	9.21

24	8.95	8.7	8.76	8.84
36	7.87	7.55	7.64	7.71
48	6.94	6.35	6.56	6.53
60	5.91	5.5	5.55	5.67
72	5.03	4.58	4.81	4.65
84	4.25	3.8	4.06	3.95
96	3.66	3.1	3.3	3.32
108	2.97	2.35	2.44	2.58
120	2.12	1.76	1.81	2.01

Average of three replicates.

Table 2: Initial pH study of Florasulam in an aqueous medium under the influence of UV light with TiO_2 photocatalysts.

Time (Hour)	Concentration of Florasulam in mgL^{-1}. [remaining in the aqueous system at pH 7 in presence of the different amounts of the synthesized TiO_2 NPs photocatalyst separately under UV irradiation].					
	10 mg/L TiO_2 NPs	20 mg/L TiO_2 NPs	50 mg/L TiO_2 NPs	80 mg/L TiO_2 NPs	110 mg/L TiO_2 NPs	150 mg/L TiO_2 NPs
0	10.27	10.1	10.24	10.18	10.16	10.18
3	9.89	9.82	9.63	9.71	9.74	9.83
6	9.16	9.11	8.95	8.98	9.04	9.2
12	8.04	7.99	7.88	7.92	8.08	8.02
24	6.87	6.86	6.54	6.6	6.85	6.98
36	5.87	5.78	5.37	5.34	5.56	5.91
48	4.64	4.57	4.11	4.16	4.17	4.66
60	3.67	3.57	3.21	3.24	3.26	3.75
72	2.71	2.56	2.3	2.35	2.46	2.77
84	1.76	1.58	1.28	1.23	1.39	1.85
96	1.08	0.96	0.72	0.7	0.8	1.12
108	0.87	0.85	0.68	0.65	0.73	0.92
120	0.8	0.72	0.6	0.63	0.65	0.81

Average of three replicates.

Table 3: Catalyst TiO2 NPs dosage study of Florasulam photodegradation in aqueous medium under the influence of UV light with TiO_2 photocatalysts at pH 7.

Time (Hour)	Concentration of florasulam in mgL^{-1}. [remaining in the aqueous system at pH 7 in the presence synthesized TiO_2 Nps (50 mgL^{-1}) and bulk TiO_2 (50 mgL^{-1}) photocatalyst separately under UV irradiation]				
	C_1	C_2	C_3	C_4	C_{com}
0	10.24	15.21	19.93	25.08	10.73
3	9.63	14.37	19.03	24.02	10.38
6	8.95	13.38	17.73	22.69	9.9
12	7.88	11.86	15.94	20.44	9.15
24	6.54	9.88	13.35	17.4	8.7
36	5.37	8.14	10.96	14.37	7.55
48	4.11	6.33	8.76	11.68	6.35
60	3.21	5.17	7.17	9.43	5.5
72	2.3	3.65	4.98	7.17	4.58
84	1.28	2.06	3.19	4.36	3.8
96	0.72	1.37	1.98	2.65	3.1
108	0.68	1.23	1.71	2.51	2.35
120	0.6	1.06	1.58	2.24	1.76

Table 4: Kinetic study of Florasulam in an aqueous medium under the influence of UV light with TiO_2 photocatalysts at pH 7.

8 WORKINGS:

8.1 IMPORTING NECESSARY LIBRARIES

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import numpy as np
4. import seaborn as sns
5. import scipy.stats as stats
6. from openpyxl import load_workbook
7. from scipy.stats import (weibull_min, weibull_max, norm, beta, invgauss, uniform, gamma, expon, lognorm, pearson3, triang)
8. from scipy.stats import kstest
9. from scipy.stats import beta
10. import statsmodels.api as sm  # Import statsmodels module for Q-Q plot
11. from catboost import CatBoostRegressor
12. from sklearn.model_selection import train_test_split
13. from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

This code above imports several libraries that will be used for data analysis, visualization, statistical tests, and machine learning model training and evaluation.

1. pandas: For data manipulation and analysis.
2. matplotlib.pyplot: For creating static, animated, and interactive visualizations in Python.
3. numpy: For numerical operations.
4. seaborn: For making statistical graphics in Python, built on top of matplotlib.
5. scipy.stats: For statistical functions and tests.
6. openpyxl: For reading and writing Excel files.
7. statsmodels.api: For statistical models, particularly useful for Q-Q plots which are used to check the normality of data.
8. catboost: For training a CatBoost regression model, which is a gradient boosting library.
9. sklearn.model_selection: For splitting the dataset into training and testing sets.
10. sklearn.metrics: For evaluating the performance of the machine learning model using metrics like mean squared error (MSE), mean absolute error (MAE), and R-squared (R^2) score.

8.2 DATA MERGING

```
# Merge all dataframes on 'Time (Hour)'
combined_data = data_6_5.merge(data_6_6, on='Time (Hour)', how='outer')
combined_data = combined_data.merge(data_6_8, on='Time (Hour)', how='outer')
combined_data = combined_data.merge(data_6_10, on='Time (Hour)', how='outer')

# Print the combined data for verification
display(combined_data)
```

The code merges all the DataFrames on the column 'Time (Hour)' using an outer join. This means that all rows from each DataFrame are included in the merged DataFrame, with NaN values filled in for missing entries.

First, data_6_5 and data_6_6 are merged on 'Time (Hour)' and stored in combined_data.

Then, data_6_8 is merged with combined_data on 'Time (Hour)'.

Finally, data_6_10 is merged with combined_data on 'Time (Hour)'.

This process combines the data from all four DataFrames into a single DataFrame, facilitating analysis and processing tasks.

After merging, the code displays the combined_data DataFrame for verification using the display function.

8.2.1 Result:

	Time (Hour)	Blank UV	Blank Sunlight	pH 3	pH 7	pH 10	pH 12	10 mg/L TiO2 NPs	20 mg/L TiO2 NPs	50 mg/L TiO2 NPs	80 mg/L TiO2 NPs	110 mg/L TiO2 NPs	150 mg/L TiO2 NPs	C1	C2	C3	C4	Ccom
0	0	10.6	10.86	10.65	10.73	10.68	10.71	10.27	10.1	10.24	10.18	10.16	10.18	10.24	15.21	19.93	25.08	10.73

1	3	10.42	10.72	10.41	10.38	10.36	10.4	9.89	9.82	9.63	9.71	9.74	9.83	9.63	14.37	19.03	24.02	10.38
2	6	10.15	10.57	9.98	9.9	9.94	9.96	9.16	9.11	8.95	8.98	9.04	9.2	8.95	13.38	17.73	22.69	9.9
3	12	9.86	10.28	9.39	9.15	9.3	9.21	8.04	7.99	7.88	7.92	8.08	8.02	7.88	11.86	15.94	20.44	9.15
4	24	9.41	10.05	8.95	8.7	8.76	8.84	6.87	6.86	6.54	6.6	6.85	6.98	6.54	9.88	13.35	17.4	8.7
5	36	8.82	9.77	7.87	7.55	7.64	7.71	5.87	5.78	5.37	5.34	5.56	5.91	5.37	8.14	10.96	14.37	7.55
6	48	8.3	9.3	6.94	6.35	6.56	6.53	4.64	4.57	4.11	4.16	4.17	4.66	4.11	6.33	8.76	11.68	6.35
7	60	8	8.92	5.91	5.5	5.55	5.67	3.67	3.57	3.21	3.24	3.26	3.75	3.21	5.17	7.17	9.43	5.5
8	72	7.45	8.46	5.03	4.58	4.81	4.65	2.71	2.56	2.3	2.35	2.46	2.77	2.3	3.65	4.98	7.17	4.58
9	84	6.53	8	4.25	3.8	4.06	3.95	1.76	1.58	1.28	1.23	1.39	1.85	1.28	2.06	3.19	4.36	3.8
10	96	5.65	7.1	3.66	3.1	3.3	3.32	1.08	0.96	0.72	0.7	0.8	1.12	0.72	1.37	1.98	2.65	3.1
11	108	4.8	6.59	2.97	2.35	2.44	2.58	0.87	0.85	0.68	0.65	0.73	0.92	0.68	1.23	1.71	2.51	2.35
12	120	4.15	5.66	2.12	1.76	1.81	2.01	0.8	0.72	0.6	0.63	0.65	0.81	0.6	1.06	1.58	2.24	1.76

8.3 Best Fit Probability Distribution Analysis:

8.3.1 Objective:

The objective of this analysis is to determine which probability distribution best fits the combined dataset of Florasulam concentration under various experimental conditions. This is an important step because generating synthetic data, a subsequent step in your project, relies on accurately modelling the underlying data distribution.

8.3.2 Code

```
# Define probability distributions to test
distributions = [
    weibull_min, weibull_max, norm, beta, invgauss,
    uniform, gamma, expon, lognorm, pearson3, triang
]

# Function to fit a distribution to the data and return goodness of fit
def fit_distribution(data, distribution):
    params = distribution.fit(data)
    # Kolmogorov-Smirnov test for goodness of fit
    D, p_value = kstest(data, distribution.cdf, args=params)
    return D, p_value, params

# Fit distributions to the combined dataset
results = {}
for distribution in distributions:
    dist_name = distribution.name
    print(f"Fitting {dist_name} distribution...")
    D_values = []
    p_values = []
    params_list = []

    # Fit distribution to each column of the dataset
    for column in combined_data.columns:
        data = combined_data[column].dropna()
        if len(data) > 0:  # Ensure there's data to fit
            D, p_value, params = fit_distribution(data, distribution)
            D_values.append(D)
            p_values.append(p_value)
            params_list.append(params)

    # Store results
    results[dist_name] = {
        'Mean_D': np.mean(D_values),
        'Mean_p_value': np.mean(p_values),
        'Parameters': params_list
    }
```

```
38.
39. # Display results
40. print("\nGoodness of Fit Results:")
41. for dist_name, result in results.items():
42.     print(f"\nDistribution: {dist_name}")
43.     print(f"Mean Kolmogorov-Smirnov D value: {result['Mean_D']}")
44.     print(f"Mean p-value: {result['Mean_p_value']}")
```

8.3.3 Output

Goodness of Fit Results:

- Distribution: weibull_min
    - Mean Kolmogorov-Smirnov D value: 0.19525925925692922
    - Mean p-value: 0.6365114325908171
- Distribution: weibull_max
    - Mean Kolmogorov-Smirnov D value: 0.44398962491107935
    - Mean p-value: 0.07712366774116343
- Distribution: norm
    - Mean Kolmogorov-Smirnov D value: 0.14933709182537408
    - Mean p-value: 0.8913117566181197
- Distribution: beta
    - Mean Kolmogorov-Smirnov D value: 0.10491451140420832
    - Mean p-value: 0.9737685992351547
- Distribution: invgauss
    - Mean Kolmogorov-Smirnov D value: 0.16590895042502005
    - Mean p-value: 0.8092998112934359
- Distribution: uniform
    - Mean Kolmogorov-Smirnov D value: 0.20847596614088798
    - Mean p-value: 0.5606124144890466
- Distribution: gamma
    - Mean Kolmogorov-Smirnov D value: 0.19919236490292436
    - Mean p-value : 0.6382114674362435
- Distribution: expon
    - Mean Kolmogorov-Smirnov D value: 0.19103621048202116
    - Mean p-value: 0.66870821065733
- Distribution: lognorm
    - Mean Kolmogorov-Smirnov D value: 0.39827194506698443
    - Mean p-value: 0.11417567326651544
- Distribution: pearson3
    - Mean Kolmogorov-Smirnov D value: 0.24826580636516513

    - Mean p-value: 0.35647240908991146
- Distribution: triang
    - Mean Kolmogorov-Smirnov D value: 0.2101194561165631
    - Mean p-value: 0.572042135841177

8.3.4 Explanation

These results provide insights into how well each distribution fits the data. The Kolmogorov-Smirnov test statistic (D) measures the maximum discrepancy between the empirical cumulative distribution function of the data and the cumulative distribution function of the fitted distribution. A lower D value indicates a better fit. Additionally, the p-value indicates the probability of observing a D value as extreme as the one computed under the null hypothesis that the data follows the specified distribution. A higher p-value suggests a better fit.
The code defines a list of probability distributions to test, including Weibull, Normal, Beta,Exponential, and others.

A function named fit_distribution is defined to fit a distribution to the data and return goodness-of-fit metrics. The Kolmogorov-Smirnov test is used to assess the goodness of fit, and the function returns the test statistic (D), p-value, and parameters of the fitted distribution.

The code then fits each distribution to the combined dataset, iterating over each distribution and each column of the dataset. For each distribution, it computes the mean Kolmogorov-Smirnov test statistic (D), the mean p-value, and stores the parameters of the fitted distribution.

Finally, the code displays the results, showing the mean Kolmogorov-Smirnov test statistic (D) and the mean p-value for each distribution. This provides insight into how well each distribution fits the data.

- Weibull (Min) Represents the distribution of minimum values in a dataset, often used in reliability engineering to model time-to-failure data. It is suitable for analyzing data where the rate of occurrence of an event changes over time, such as product lifetimes or failure times.
- Weibull (Max) Represents the distribution of maximum values in a dataset, the complementary distribution to the Weibull Min distribution. It is useful for modeling data where the rate of occurrence decreases over time, such as durations until the next occurrence of an event.
- Normal Also known as the Gaussian distribution, it is characterized by a bell-shaped curve and is widely used due to the central limit theorem. It is often applied in natural phenomena and statistical analyses, particularly when the data follows a symmetric distribution around the mean.

- Beta Represents continuous probability distributions defined on the interval [0, 1]. It is commonly used to model probabilities or proportions in scenarios like proportions of successes in binomial trials or modeling the behavior of random variables constrained to lie within a fixed interval.
- Inverse Gaussian A continuous probability distribution used to model the time until an event occurs. It is particularly suitable for modeling waiting times until a specific event happens, such as the time between consecutive events in a Poisson process.
- Uniform Characterized by constant probability density function between specified limits. It is often used in statistical modeling to represent uncertainty when all outcomes in a range are equally likely, such as in Monte Carlo simulations or simple random sampling.
- Gamma A flexible continuous probability distribution used to model skewed, positive-valued data. It is commonly applied to represent the time until an event occurs, the size of insurance claims, or the distribution of income or wealth.
- Exponential Models the time until an event occurs in a Poisson process, where events occur continuously and independently at a constant average rate. It is commonly used in reliability engineering and queuing theory to model the time between events or the time until a system failure.
- Lognormal A continuous probability distribution of a random variable whose logarithm is normally distributed. It is used to model positively skewed data that may arise from processes involving the multiplication of several independent random variables, such as the size of biological populations or stock prices.
- Pearson Type III Also known as the Pearson distribution, it is a family of continuous probability distributions. It is widely used for modeling skewed data and is a generalization of several other distributions, including the normal, gamma, and exponential distributions.
- Triangular A continuous probability distribution with a probability density function shaped like a triangle. It is often used in decision-making and risk analysis to represent uncertainty when the true distribution is unknown, but some limits or parameters are known.

8.3.5 Methodology:

1. Combining Data: We combined the data from all tables (sheets) into a single dataset. This approach allows us to get a holistic view of the data, capturing variability across different experimental conditions.
2. Fitting Distributions: We fitted several probability distributions to the combined data:
    a. Weibull (min and max)
    b. Normal

c. Beta
d. Inverse Gaussian
e. Uniform
f. Gamma
g. Exponential
h. Lognormal
i. Pearson Type III
j. Triangular

3. Goodness of Fit Test: For each distribution, we performed the Kolmogorov-Smirnov (K-S) test to evaluate the goodness of fit. The K-S test compares the cumulative distribution function (CDF) of the sample data with the CDF of the fitted distribution.
4. D value: This is the maximum difference between the empirical CDF and the theoretical CDF. A smaller D value indicates a better fit.
5. p-value: This indicates the probability that the observed data could have come from the fitted distribution. A higher p-value suggests a better fit.

8.3.6 Interpretation:

8.3.6.1 Beta Distribution:

- Mean Kolmogorov-Smirnov D value: 0.1049
- Mean p-value: 0.9738

The Beta distribution demonstrates the best fit to the combined dataset, with the lowest mean D value and the highest mean p-value among all distributions tested. This indicates that the data closely aligns with the characteristics of the Beta distribution.

8.3.6.2 Normal Distribution:

- Mean Kolmogorov-Smirnov D value: 0.1493
- Mean p-value: 0.8913

The Normal distribution also exhibits a strong fit to the data, with a relatively low mean D value and a high mean p-value. While not as optimal as the Beta distribution, the Normal distribution provides a reasonable representation of the dataset.

8.3.6.3 Other Distributions:

Weibull_min, Invgauss, and Exponential distributions show moderate fits to the data, with mean D values ranging from 0.1659 to 0.1910 and mean p-values ranging from 0.6687 to 0.8093. While these distributions may capture certain aspects of the data distribution, they are not as robust as the Beta and Normal distributions.

8.4 Synthetic Data Generation

8.4.1 Code

```
# Code to automate synthetic data generation for any column
def generate_synthetic_data(df, column, size=1000):
    data = df[column].dropna()
    params = beta.fit(data)
    synthetic_data = beta.rvs(*params, size=size)
    return synthetic_data

# Example for generating synthetic data for all relevant columns
synthetic_data_dict = {col: generate_synthetic_data(combined_data, col) for col in combined_data.columns if col != 'Time (Hour)'}

# Display synthetic data for one of the columns
print("Synthetic Data for 'pH 3':")
display(pd.DataFrame(synthetic_data_dict['pH 3'], columns=['Synthetic Values']))
```

This code generates synthetic data for all relevant columns specified in the columns_of_interest list using the generate_synthetic_data function. It then prints the synthetic data for each column, displaying the first few rows of the synthetic data DataFrame. This process allows for the creation and visualization of synthetic datasets for further analysis or modeling tasks.

8.5 Validation of Synthetic Data

Statistical measures such as the KS (Kolmogorov-Smirnov) statistic and p-value are fundamental tools for comparing two dataset's distributions. The KS statistic quantifies the maximum vertical distance between the cumulative distribution functions of the two datasets. A smaller KS statistic suggests a closer match between the distributions. The associated p-value indicates the probability of observing the KS statistic or a more extreme value under the null hypothesis that the two datasets are drawn from the same distribution. A higher p-value indicates stronger evidence against rejecting the null hypothesis, suggesting a better match between the datasets.

Visual representations, including histograms and Q-Q (Quantile-Quantile) plots, offer intuitive insights into data distributions. Histograms provide a visual depiction of the frequency distribution of data by dividing the data into intervals (bins) and representing

the frequency of observations in each bin through bars. They enable the identification of patterns, central tendencies, and outliers within the data. Q-Q plots, on the other hand, compare the quantiles of two datasets by plotting their respective quantiles against each other. A perfect match between the datasets results in a straight line. Deviations from this line indicate differences in distributional characteristics, helping to assess the similarity or discrepancy between datasets. These visual representations complement statistical measures, providing a comprehensive understanding of data distributional properties and aiding in data analysis and interpretation.

8.5.1 Code

```
# Function to validate synthetic data
def validate_synthetic_data(original, synthetic):
    # Kolmogorov-Smirnov test
    ks_stat, p_value = kstest(original, synthetic)
    print(f"KS Statistic: {ks_stat}, p-value: {p_value}")

    # Plot histograms
    plt.figure(figsize=(12, 6))
    sns.histplot(original, label='Original', color='blue', kde=True, stat="density", linewidth=0)
    sns.histplot(synthetic, label='Synthetic', color='red', kde=True, stat="density", linewidth=0, alpha=0.5)
    plt.legend()
    plt.title("Histogram of Original and Synthetic Data")
    plt.show()

    # Q-Q plot
    plt.figure(figsize=(6, 6))
    sm.qqplot(np.array(original), line ='45')
    sm.qqplot(np.array(synthetic), line ='45')
    plt.title("Q-Q Plot")
    plt.show()

# Validate synthetic data for each column
for col, synthetic_data in synthetic_data_dict.items():
    print(f"Validation for column: {col}")
    validate_synthetic_data(combined_data[col].dropna(), synthetic_data)
    print("\n" + "="*50 + "\n")
```

8.5.2 Output

8.5.2.1 Validation for column: Time (Hour)

KS Statistic: 0.0926923076923077, p-value: 0.9992970960033919

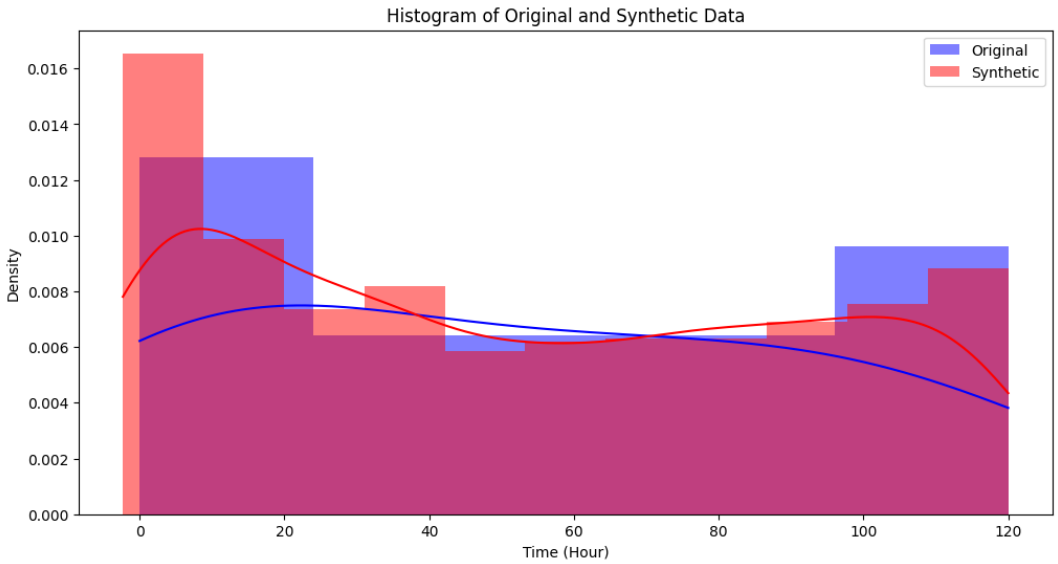


Figure 1: Histogram of Original & Synthetic Data for Column Time(Hour)

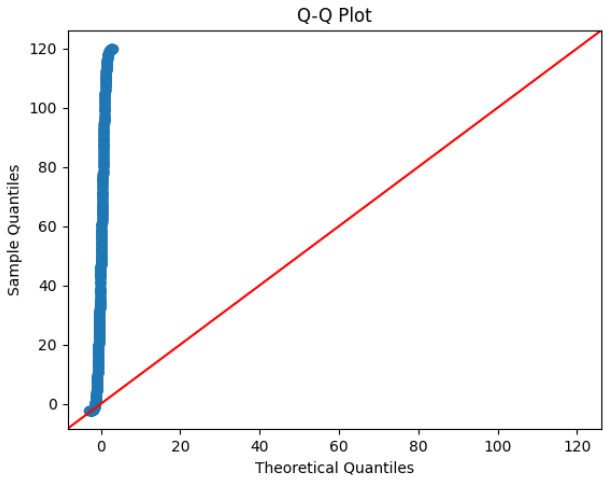


Figure 2: Q-Q plot of Original & Synthetic Data for Column Time(Hour)

8.5.2.2 Validation for column: Blank UV

KS Statistic: 0.07692307692307693, p-value: 0.9999814663339067

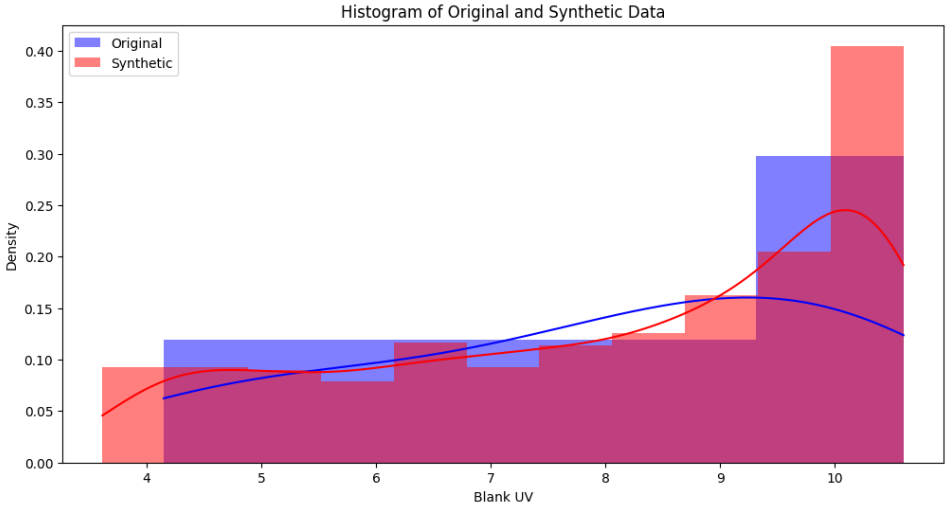


Figure 3:Histogram of Original & Synthetic Data for Column Blank UV

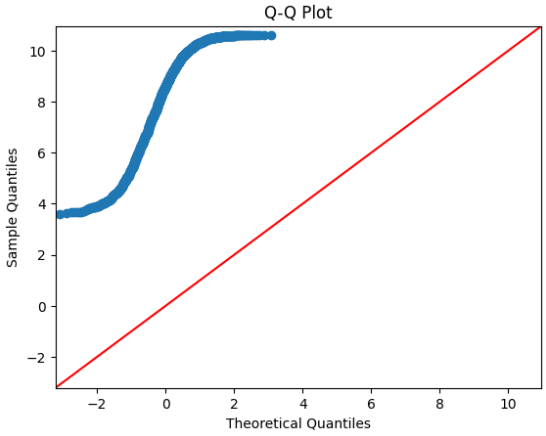


Figure 4: Q-Q plot of Original & Synthetic Data for Column Blank UV

8.5.2.3 Validation for column: Blank Sunlight

KS Statistic: 0.07692307692307693, p-value: 0.9999814663339067

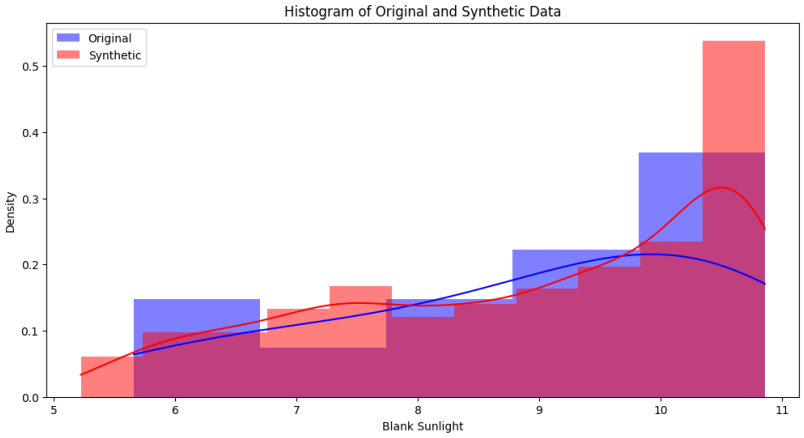


Figure 5: Histogram of Original & Synthetic Data for Column Blank Sunlight

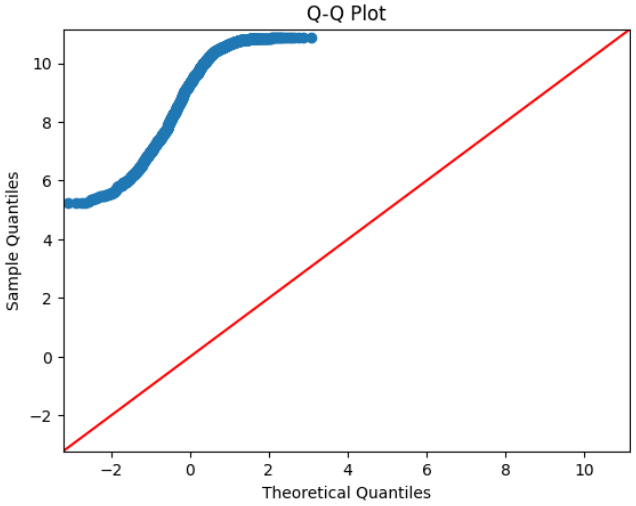


Figure 6:Q-Q plot of Original & Synthetic Data for Column Blank Sunlight

8.5.2.4 Validation for column: C1

KS Statistic: 0.109, p-value: 0.9935276368769922

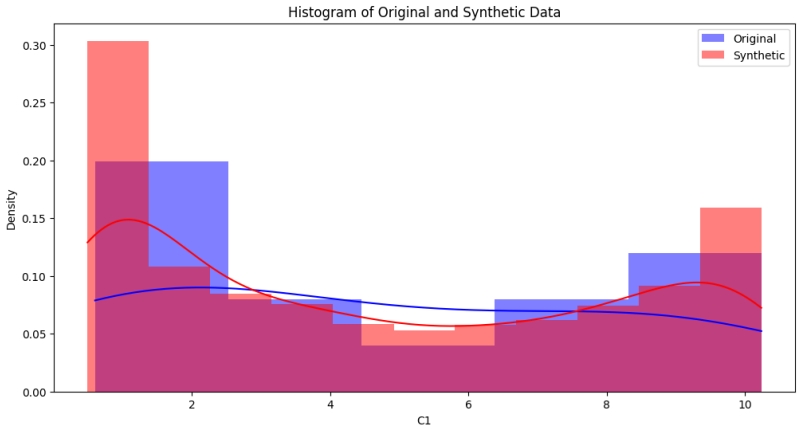


Figure 7: Histogram of Original & Synthetic Data for Column C1

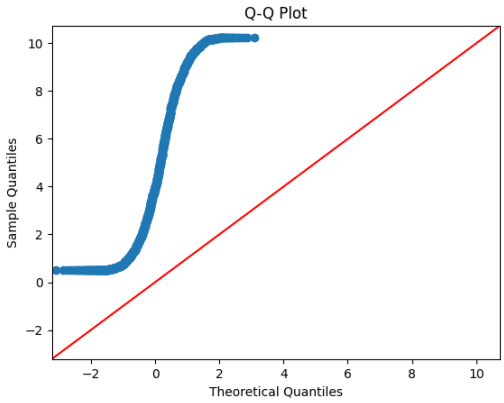


Figure 8: Q-Q plot of Original & Synthetic Data for Column C1

8.5.2.5 Validation for column: C2

KS Statistic: 0.08976923076923077, p-value: 0.9995829829333448

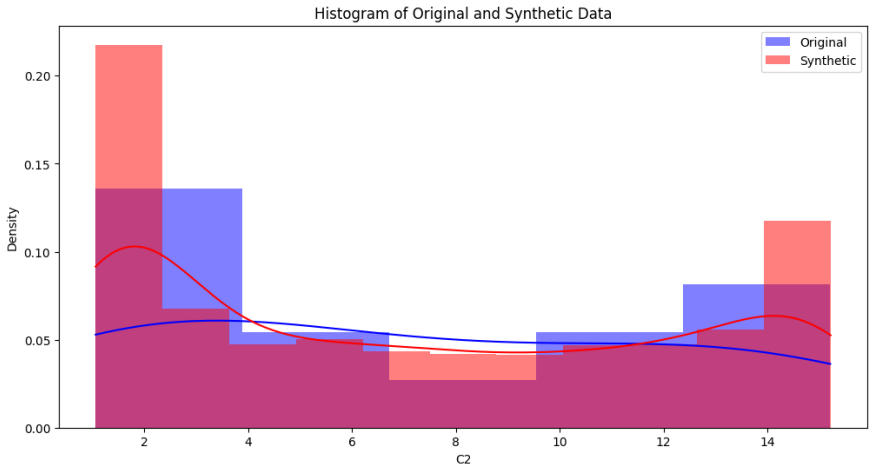


Figure 9: Histogram of Original & Synthetic Data for Column C2

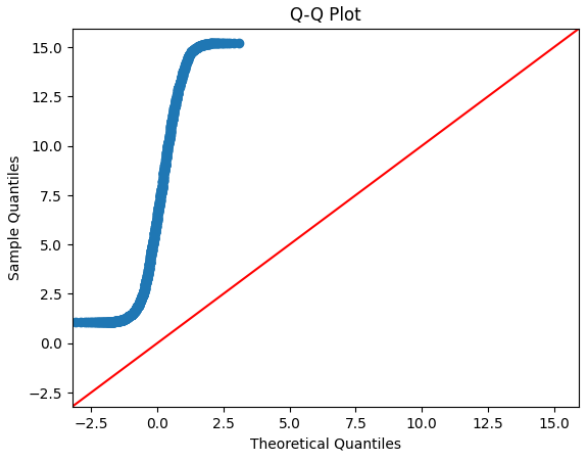


Figure 10: Q-Q plot of Original & Synthetic Data for Column C2

8.5.2.6 Validation for column: C3

KS Statistic: 0.084, p-value: 0.9998758479797367

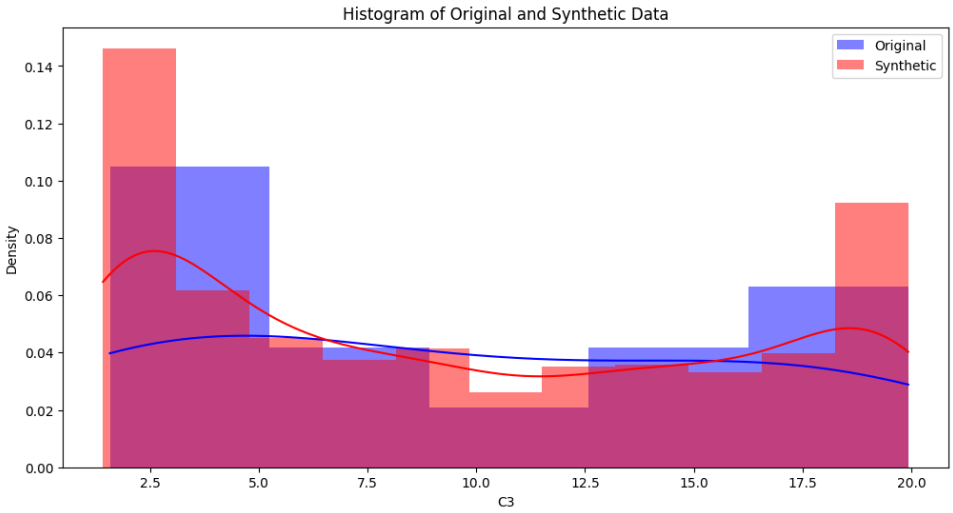


Figure 11: Histogram of Original & Synthetic Data for Column C3

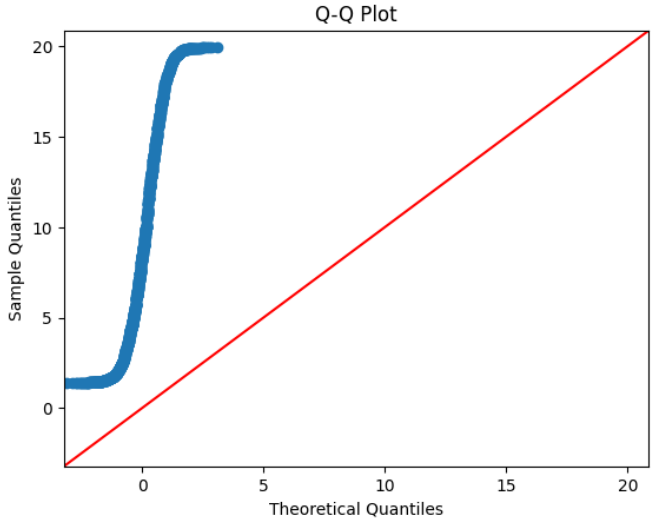


Figure 12: Q-Q plot of Original & Synthetic Data for Column C3

8.5.2.7 Validation for column: C4

KS Statistic: 0.11276923076923077, p-value: 0.9903572092689578

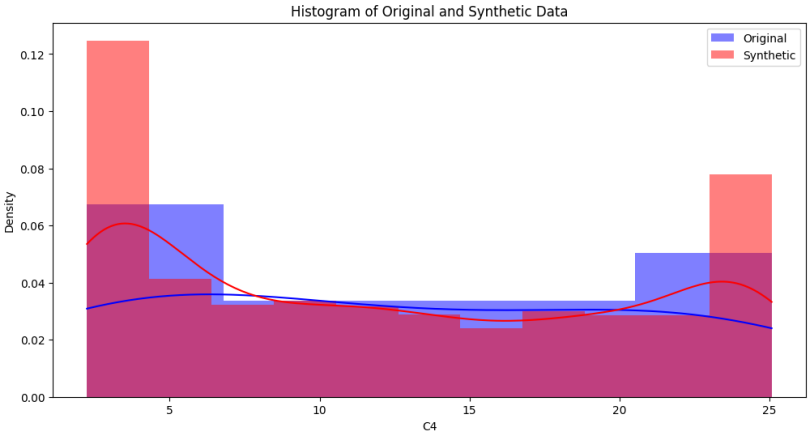


Figure 13: Histogram of Original & Synthetic Data for Column C4

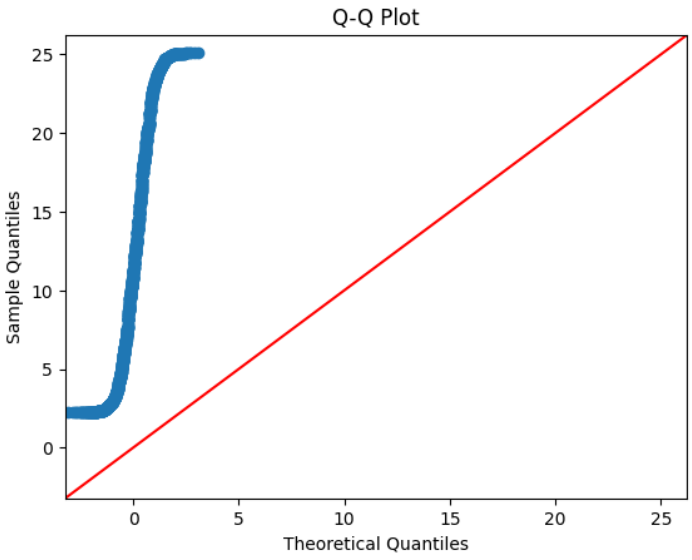


Figure 14: Q-Q plot of Original & Synthetic Data for Column C4

8.5.2.8 Validation for column: Ccom

KS Statistic: 0.144, p-value: 0.9183120337729324

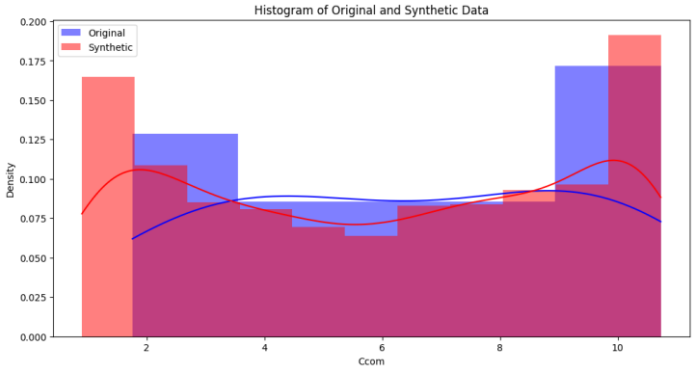


Figure 15: Histogram of Original & Synthetic Data for Column C_{Com}

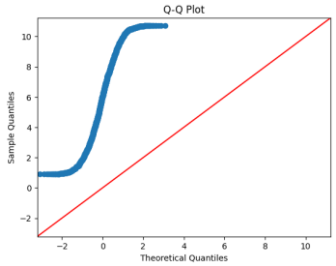


Figure 16:Q-Q plot of Original & Synthetic Data for Column C_{Com}

8.5.2.9 Validation for column: 10 mg/L TiO2 NPs

KS Statistic: 0.103, p-value: 0.9968101665852206

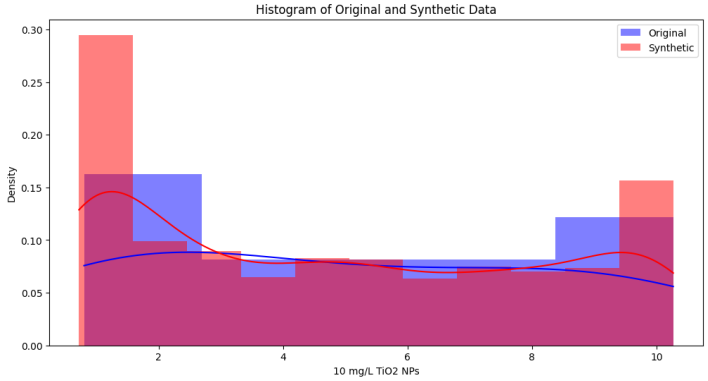


Figure 17: Histogram of Original & Synthetic Data for Column 10mg/L TiO_2 NPa

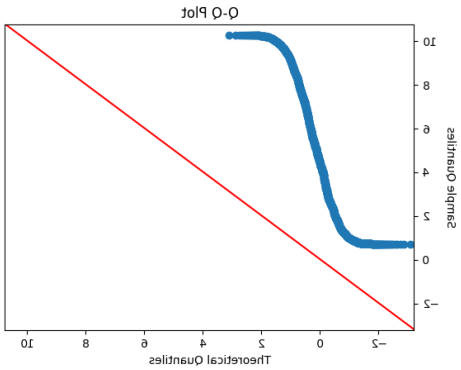


Figure 18: Q-Q plot of Original & Synthetic Data for Column 10mg/L TiO2 NPs

8.5.2.10 Validation for column: 20 mg/L TiO2 NPs

KS Statistic: 0.07692307692307693, p-value: 0.9999814663339067

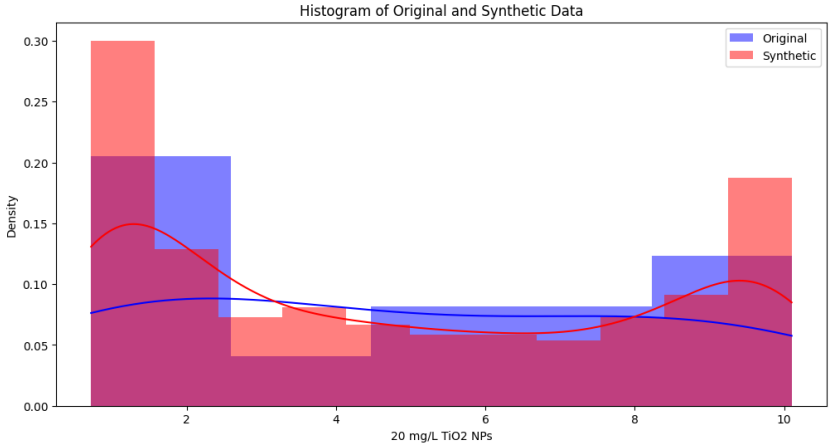


Figure 19: Histogram of Original & Synthetic Data for Column 20mg/L TiO2 NPs

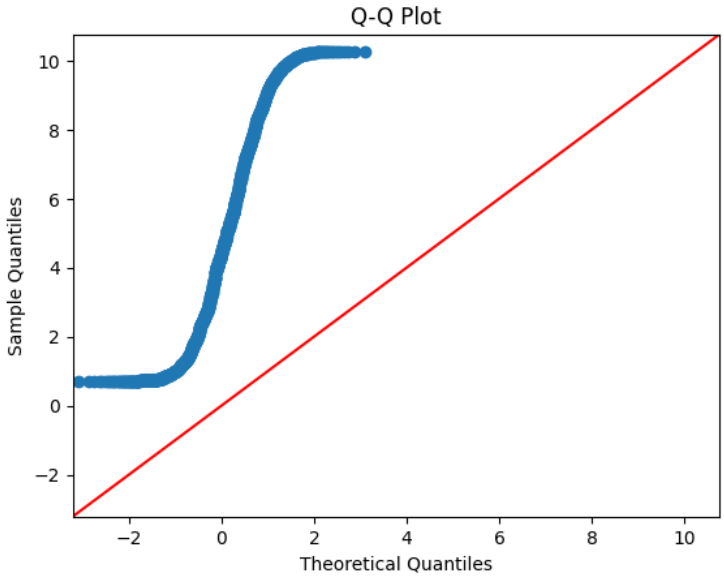


Figure 20: Q-Q plot of Original & Synthetic Data for Column 20mg/L TiO2 NPs

8.5.2.11 Validation for column: 50 mg/L TiO2 NPs

KS Statistic: 0.105, p-value: 0.9959107087716568

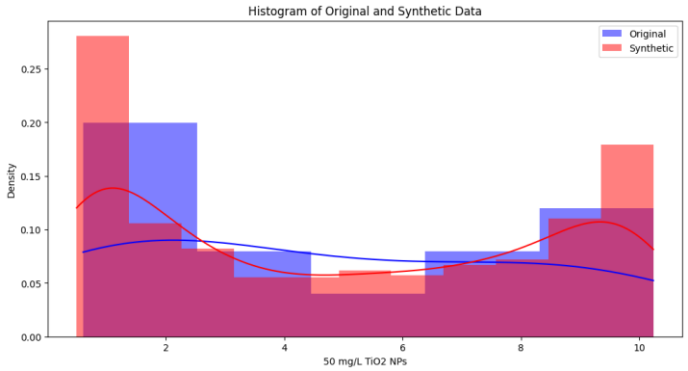


Figure 21:Histogram of Original & Synthetic Data for Column 50mg/L TiO2 NPs

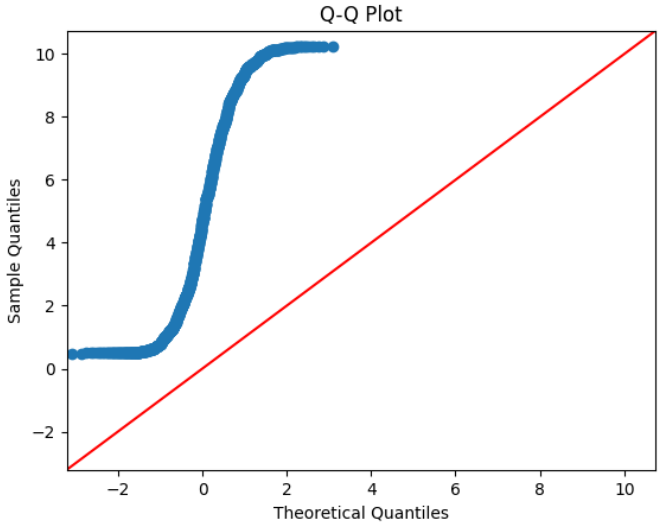


Figure 22: Q-Q plot of Original & Synthetic Data for Column 50mg/L TiO2 NPs

8.5.2.12 Validation for column: 80 mg/L TiO2 NPs

KS Statistic: 0.13476923076923078, p-value: 0.949085481392503

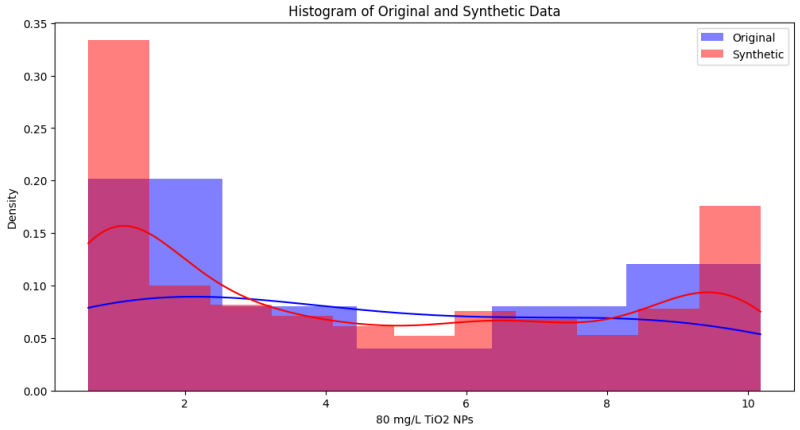


Figure 23: Histogram of Original & Synthetic Data for Column 80mg/L TiO2 NPs

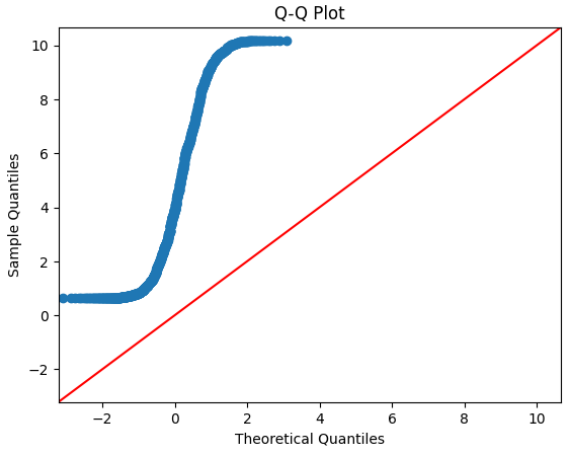
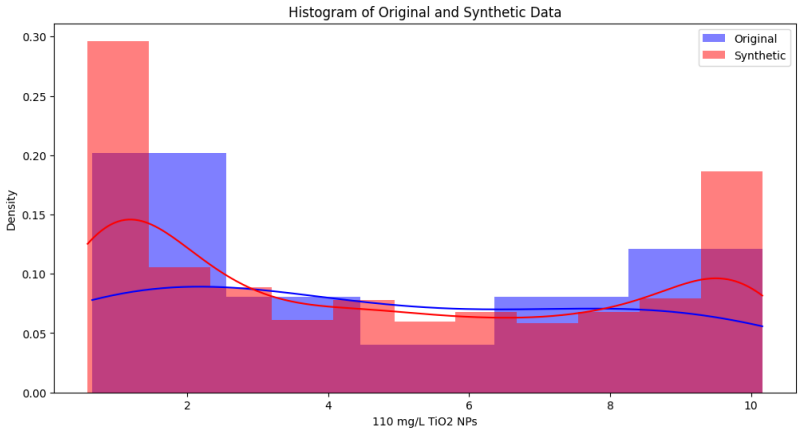


Figure 24: Q-Q plot of Original & Synthetic Data for Column 80mg/L TiO2 NPs

8.5.2.13 Validation for column: 110 mg/L TiO2 NPs

KS Statistic: 0.09076923076923077, p-value: 0.9994985049598005



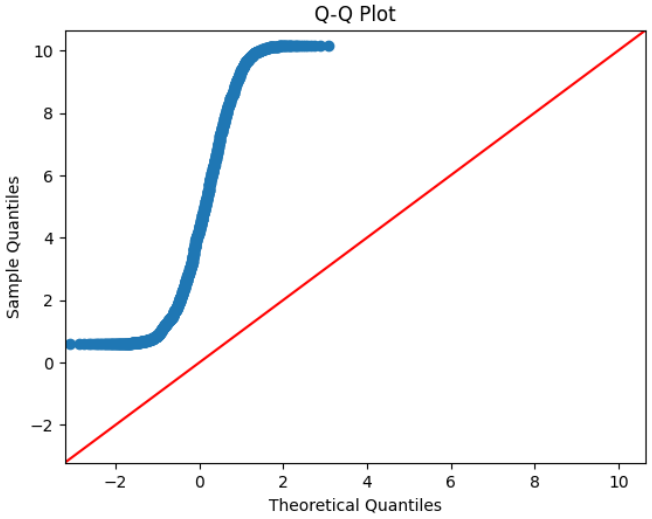
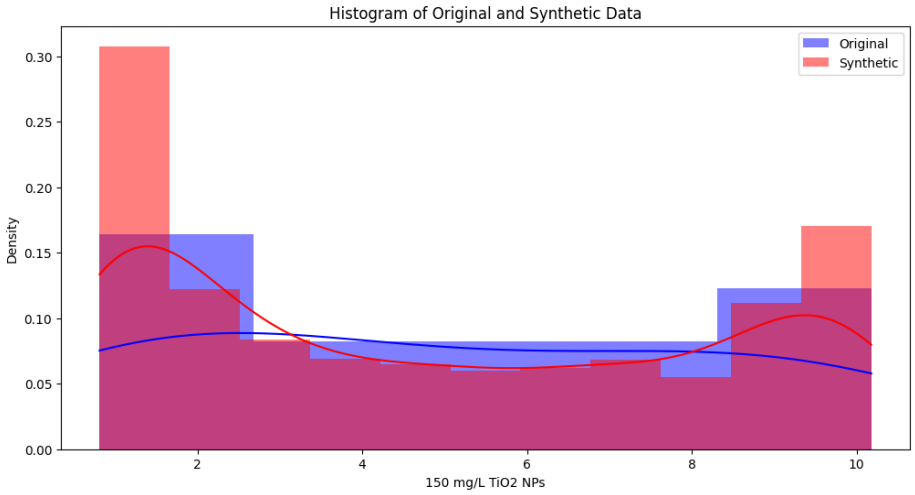


Figure 25: Histogram of Original & Synthetic Data for Column 110mg/L TiO2 NPs

8.5.2.14 Validation for column: 150 mg/L TiO2 NPs

KS Statistic: 0.07938461538461539, p-value: 0.9999611164425166



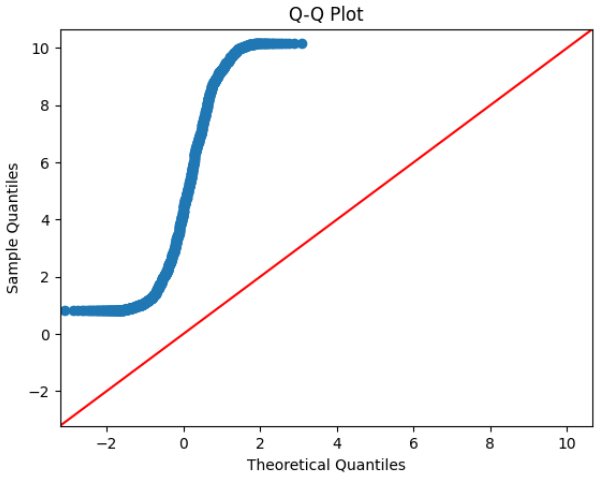


Figure 26:Histogram & Q-Q plot of Original & Synthetic Data for Column 150mg/L TiO2 NPs

8.5.2.15 Validation for column: pH 3

KS Statistic: 0.13707692307692307, p-value: 0.9423241555486522

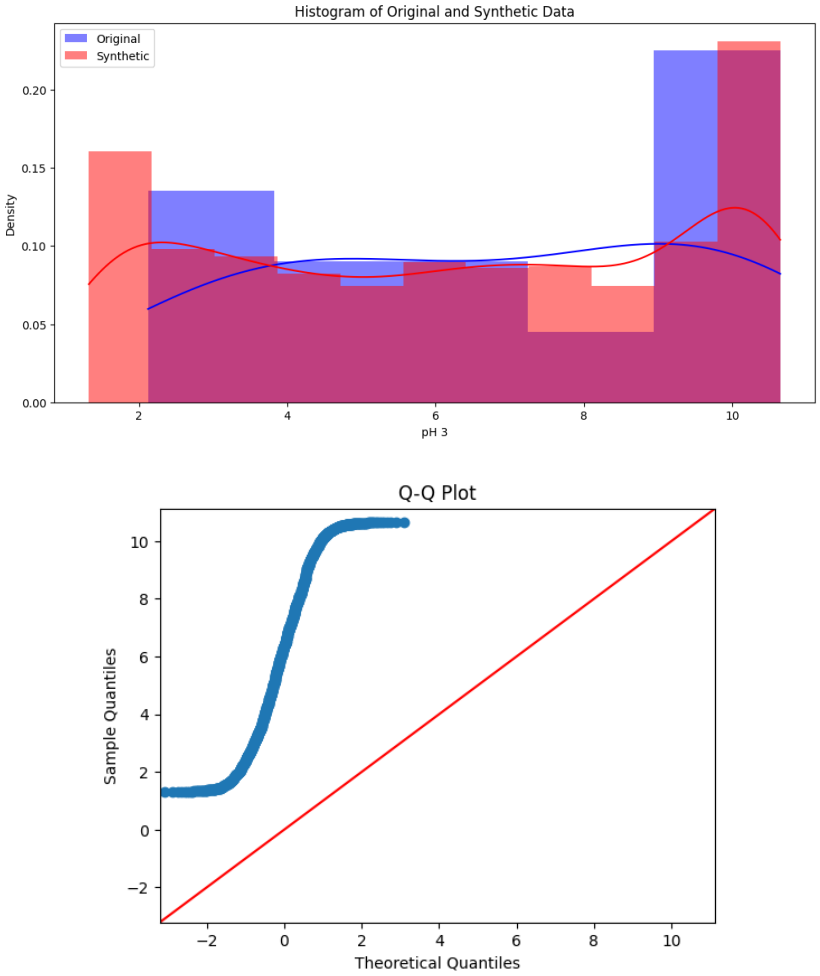
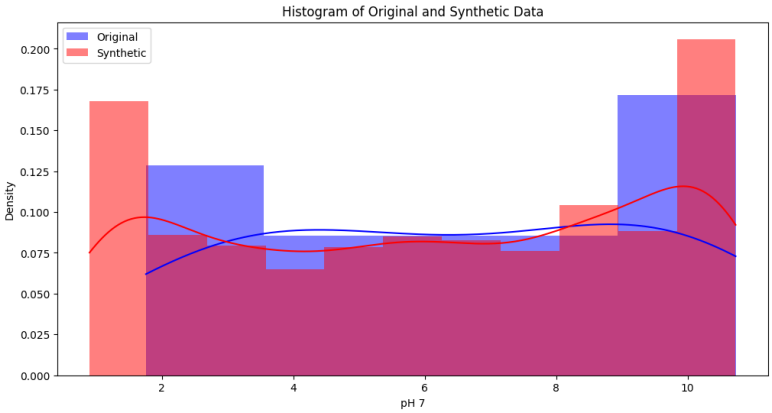


Figure 27: Histogram & Q-Q plot of Original & Synthetic Data for Column pH3

8.5.2.16 Validation for column: pH 7

KS Statistic: 0.146, p-value: 0.910530704336376



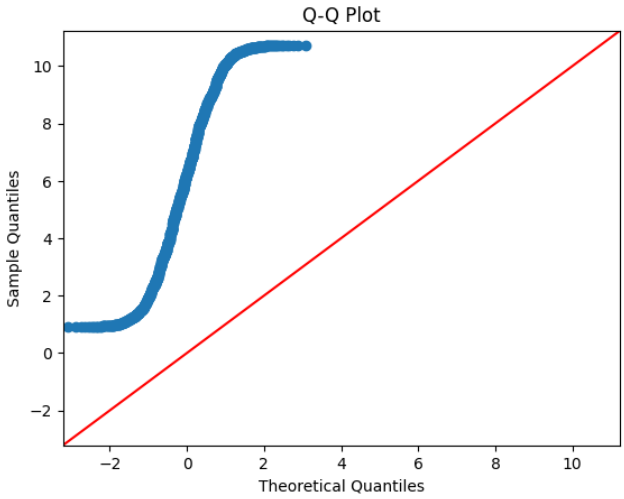
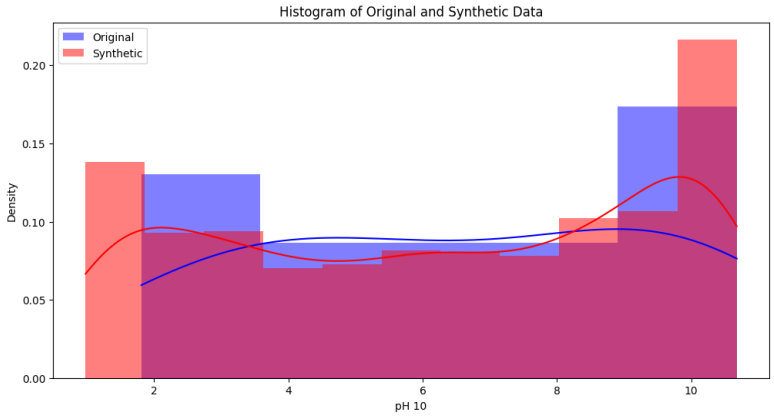


Figure 28: Histogram & Q-Q plot of Original & Synthetic Data for Column pH7

8.5.2.17 Validation for column: pH 10

KS Statistic: 0.122, p-value: 0.9784904291332613



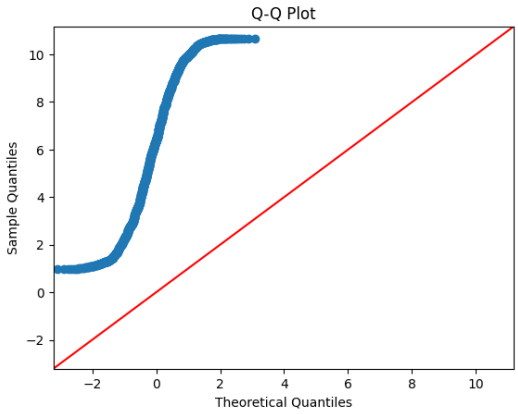


Figure 29: Histogram & Q-Q plot of Original & Synthetic Data for Column pH10

8.5.2.18 Validation for column: pH 12

KS Statistic: 0.155, p-value: 0.8716242685563634

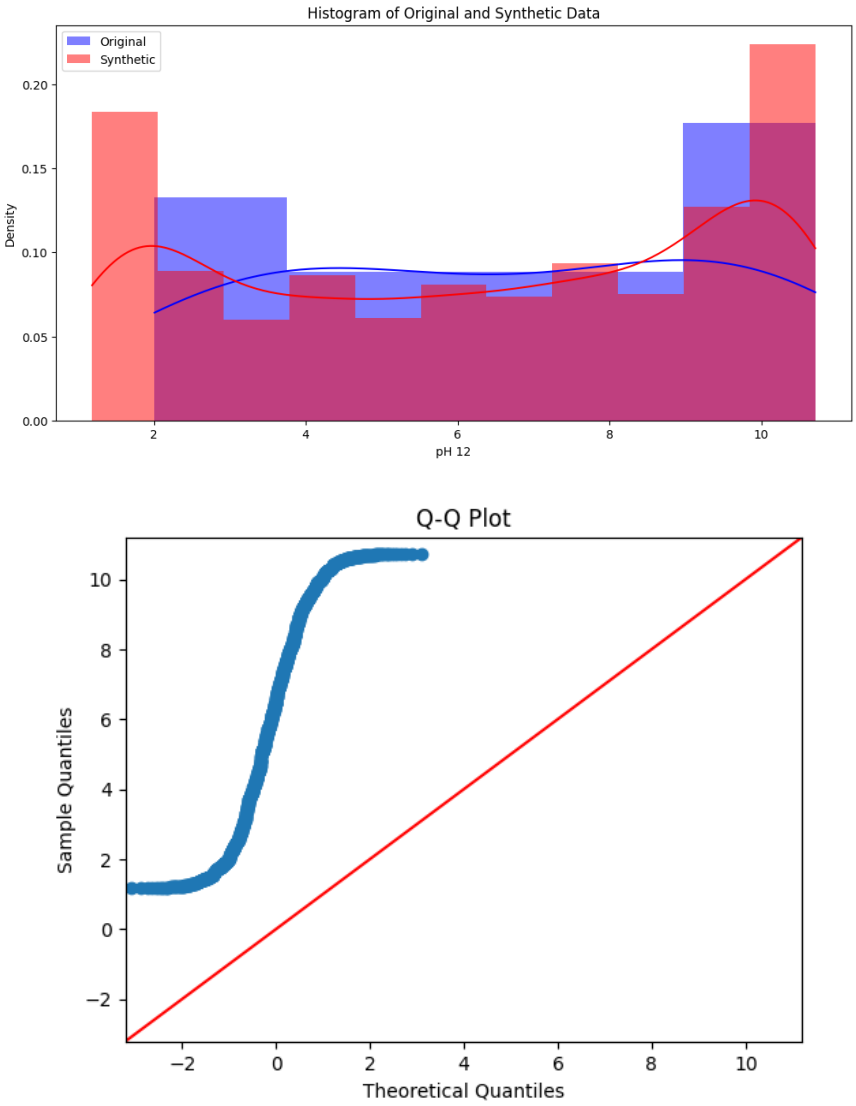


Figure 30: Histogram & Q-Q plot of Original & Synthetic Data for Column pH12

8.5.3 Explanation of Validation Output

This code defines a function validate_synthetic_data to compare the synthetic data generated for each column with the original data using statistical tests and visualizations. It then iterates over each column and its corresponding synthetic data, validating them and displaying the results.

The validation process includes:

- Performing the Kolmogorov-Smirnov test to compare the distribution of the original and synthetic data.
- Plotting histograms of the original and synthetic data for visual comparison.
- Creating Q-Q plots to assess the similarity between the quantiles of the original and synthetic data.

This validation helps ensure that the synthetic data accurately represents the distribution and characteristics of the original data.

- KS Statistic: This statistic measures the maximum difference between the cumulative distribution functions (CDFs) of the original and synthetic data. A smaller KS statistic indicates a better fit between the two distributions.
- p-value: This value indicates the probability of observing the KS statistic if the null hypothesis (i.e., the two distributions are the same) is true. A higher p-value suggests stronger evidence against rejecting the null hypothesis.

Based on the validation results:

- For most columns, the KS Statistic is relatively low, indicating a good fit between the original and synthetic data distributions.
- The p-values are generally high, indicating that there is no significant difference between the distributions of the original and synthetic data.

8.6 Model Training & Evolution:

Objective: The main goal of model training and evaluation is to develop robust predictive models that can accurately estimate the concentration of florasulam under various conditions. This involves selecting appropriate machine learning algorithms, training them on synthetic data generated from experimental results, and evaluating their performance.

8.6.1 Combining Synmthetic Data & The Original Data

8.6.1.1 Code

```
# Create a DataFrame for synthetic data
synthetic_df = pd.DataFrame(synthetic_data_dict)

# Combine original data and synthetic data
combined_data_with_synthetic = pd.concat([combined_data, synthetic_df], axis=0, ignore_index=True)

# Display combined data
print("Combined Data with Synthetic Data:")
display(combined_data_with_synthetic)
```

8.6.1.2 Explanation

This code creates a DataFrame synthetic_df containing the synthetic data generated for each column. Then, it combines the original data (combined_data) with the synthetic data using pd.concat, along the rows (axis=0), ignoring the original index to reset it (ignore_index=True). Finally, it displays the combined data,

combined_data_with_synthetic, which includes both the original and synthetic data for analysis or modeling tasks.

8.6.2 Splitting of Train & Test Data

8.6.2.1 Code

```
1. from sklearn.model_selection import train_test_split
2.
3. # Split the rearranged combined data into train and test sets
4. train_data, test_data = train_test_split(combined_data_with_synthetic, test_size=0.2,
random_state=42)
5.
6. # Print the shape of train and test data
7. print("Train Data Shape:", train_data.shape)
8. print("Test Data Shape:", test_data.shape)
9.
10. # Print train data
11. print("Train Data:")
12. print(train_data)
13.
14. # Print test data
15. print("Test Data:")
16. print(test_data)
17.
```

This code utilizes scikit-learn's train_test_split function to split the rearranged combined data with synthetic data into train and test sets. The test set size is set to 20% of the total data, and a random state of 42 is used for reproducibility.

It then prints the shapes of the train and test data to verify the split proportions. Finally, it displays the train and test data to examine the contents of each set.

8.6.3 CatBoost Model

8.6.3.1 Code

```
1. # Fill NaN values with an appropriate method (e.g., interpolation, forward fill)
2. combined_data= combined_data.interpolate(method='linear', axis=0).ffill().bfill()
3.
4. # Define features and target
```

```
X = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
                                '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
                                'C1', 'C2', 'C3', 'C4','Ccom'])
y_columns = ['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
             '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
             '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
             'C1', 'C2', 'C3', 'C4', 'Ccom']

# Train-test split for each target variable
results = {}

for target in y_columns:
    y = combined_data[target]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Initialize CatBoost model
    model = CatBoostRegressor(iterations=1000, learning_rate=0.1, depth=6, verbose=0)

    # Train the model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Evaluation
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store the results
    results[target] = {
        'model': model,
        'mean_squared_error': mse,
        'mean_absolute_error': mae,
```

```
        'r2_score': r2
    }

# Display results for each target variable
for target, metrics in results.items():
    print(f"Results for {target}:")
    print(f"  Mean Squared Error: {metrics['mean_squared_error']}")
    print(f"  Mean Absolute Error: {metrics['mean_absolute_error']}")
    print(f"  R^2 Score: {metrics['r2_score']}\n")
```

8.6.3.2 Output

- Results for Blank UV:
  - Mean Squared Error: 0.5337665944876778
  - Mean Absolute Error: 0.6499999427568358
  - R^2 Score: 0.9096974057124712
- Results for Blank Sunlight:
  - Mean Squared Error: 0.16376662099778366
  - Mean Absolute Error: 0.36999989190287597
  - R^2 Score: 0.9481032235593595
- Results for pH 3:
  - Mean Squared Error: 0.3807001723074235
  - Mean Absolute Error: 0.5700001491001467
  - R^2 Score: 0.9662701854015049
- Results for pH 7:
  - Mean Squared Error: 0.43113321922052733
  - Mean Absolute Error: 0.6266665712783833
  - R^2 Score: 0.9677595268788274
- Results for pH 10:
  - Mean Squared Error: 0.46816668392057204
  - Mean Absolute Error: 0.6433334127933494
  - R^2 Score: 0.9631514402264947
- Results for pH 12:
  - Mean Squared Error: 0.37790009041288203
  - Mean Absolute Error: 0.5833334700374273
  - R^2 Score: 0.9700795379161694
- Results for 10 mg/L TiO2 NPs:
  - Mean Squared Error: 0.3636666098750618

    - Mean Absolute Error: 0.5133332459335347
    - R^2 Score: 0.9797428054524506
- Results for 20 mg/L TiO2 NPs:
    - Mean Squared Error: 0.3503000178004509
    - Mean Absolute Error: 0.4566667421150871
    - R^2 Score: 0.980132436801503
- Results for 50 mg/L TiO2 NPs:
    - Mean Squared Error: 0.47136658816290683
    - Mean Absolute Error: 0.556666622098943
    - R^2 Score: 0.975340515816146

- Results for 80 mg/L TiO2 NPs:
    - Mean Squared Error: 0.49259991645576084
    - Mean Absolute Error: 0.5466666182074259
    - R^2 Score: 0.9741130186529363
- Results for 110 mg/L TiO2 NPs:
    - Mean Squared Error: 0.44206675853362465
    - Mean Absolute Error: 0.5200000930020676
    - R^2 Score: 0.9760719343755367
- Results for 150 mg/L TiO2 NPs:
    - Mean Squared Error: 0.33630000675324073
    - Mean Absolute Error: 0.49000002068254167
    - R^2 Score: 0.9805982506517302
- Results for C1:
    - Mean Squared Error: 0.47136658816290683
    - Mean Absolute Error: 0.556666622098943
    - R^2 Score: 0.975340515816146
- Results for C2:
    - Mean Squared Error: 1.0844334992459792
    - Mean Absolute Error: 0.8566667382323928
    - R^2 Score: 0.9735541149149962
- Results for C3:
    - Mean Squared Error: 1.3623329474071142
    - Mean Absolute Error: 0.9866664504790417
    - R^2 Score: 0.9800435092204774
- Results for C4:
    - Mean Squared Error: 3.0130998239048536
    - Mean Absolute Error: 1.3366665711418915

    - R^2 Score: 0.9712168770320436
- Results for Ccom:
    - Mean Squared Error: 0.43113321922052733
    - Mean Absolute Error: 0.6266665712783833
    - R^2 Score: 0.9677595268788274

8.6.3.3 Explanation

The code segment begins by addressing missing values in the dataset using an interpolation method followed by forward and backward filling. This preprocessing step ensures the integrity of the dataset by replacing missing values with appropriate estimates derived from neighboring data points. By completing this data preprocessing, the subsequent analysis and modeling tasks can proceed without encountering issues related to missing data.

Next, the code defines features (X) and target variables (y) for modeling. Features encompass all columns except those representing control variables such as 'Blank UV', 'Blank Sunlight', and various pH levels, along with different concentrations of TiO2 nanoparticles. The target variables consist of these control variables, which are the focal points for prediction in the regression modeling process.

The code then proceeds to conduct regression modeling for each target variable individually. It employs the CatBoostRegressor, a gradient boosting algorithm that specializes in handling categorical features and achieving high predictive accuracy. The model is trained on the training data, utilizing the features (X_train) and corresponding target variable (y_train). Subsequently, predictions are made on the test data (X_test), allowing for an assessment of the model's performance.

Evaluation metrics, including mean squared error (MSE), mean absolute error (MAE), and the coefficient of determination (R^2 score), are calculated to quantify the model's predictive accuracy and goodness of fit. These metrics provide valuable insights into how well the regression model captures the relationships between features and target variables. Lower values of MSE and MAE indicate better predictive performance, while a higher R^2 score signifies a stronger correlation between predicted and actual values.

```
# Extract metrics from the results dictionary
metrics_df = pd.DataFrame(results).T

# Plotting
plt.figure(figsize=(10, 6))
```

```
6.
7. # Plot MSE
8. plt.subplot(2, 2, 1)
9. metrics_df['mean_squared_error'].plot(kind='bar', color='blue')
10. plt.title('Mean Squared Error')
11. plt.ylabel('MSE')
12.
13. # Plot MAE
14. plt.subplot(2, 2, 2)
15. metrics_df['mean_absolute_error'].plot(kind='bar', color='green')
16. plt.title('Mean Absolute Error')
17. plt.ylabel('MAE')
18.
19. # Plot R^2 Score
20. plt.subplot(2, 2, 3)
21. metrics_df['r2_score'].plot(kind='bar', color='red')
22. plt.title('R^2 Score')
23. plt.ylabel('R^2')
24.
25. plt.tight_layout()
26. plt.show()
27.
```

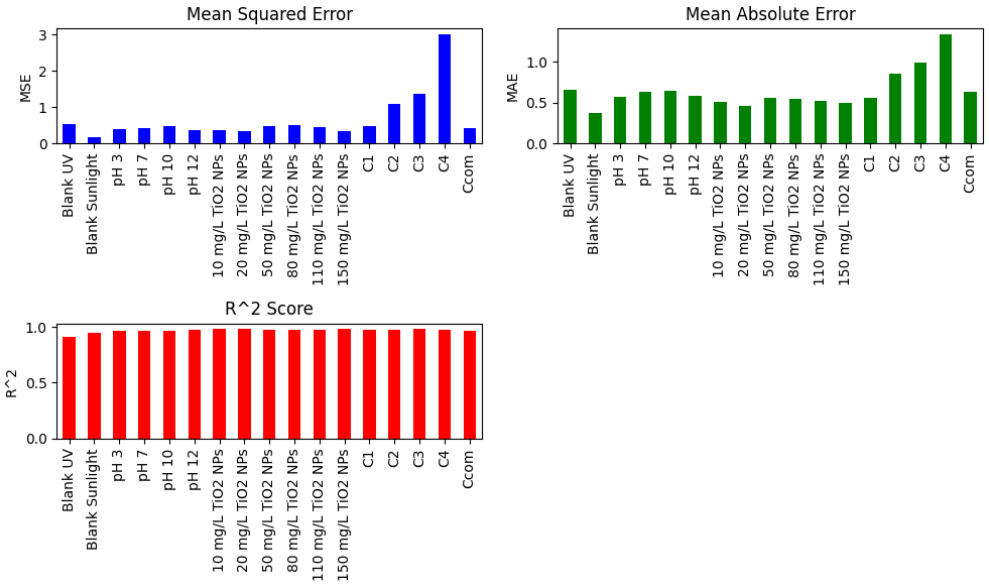


Figure 31: Plot of MSE, MAE, R^2 in CatBoost Model

This code extracts metrics from the results dictionary and creates a DataFrame called `metrics_df`. It then proceeds to plot these metrics using Matplotlib. Three subplots are

created: one for Mean Squared Error (MSE), one for Mean Absolute Error (MAE), and one for R^2 Score. Each metric is plotted as a bar plot with the corresponding values from the DataFrame. The subplot titles and y-axis labels are appropriately set. Finally, `plt.tight_layout()` ensures that the subplots are properly arranged, and `plt.show()` displays the plot.

	mean_squared_error	mean_absolute_error	r2_score
Blank UV	0.533767	0.65	0.909697
Blank Sunlight	0.163767	0.37	0.948103
pH 3	0.3807	0.57	0.96627
pH 7	0.431133	0.626667	0.96776
pH 10	0.468167	0.643333	0.963151
pH 12	0.3779	0.583333	0.97008
10 mg/L TiO2 NPs	0.363667	0.513333	0.979743
20 mg/L TiO2 NPs	0.3503	0.456667	0.980132
50 mg/L TiO2 NPs	0.471367	0.556667	0.975341
80 mg/L TiO2 NPs	0.4926	0.546667	0.974113
110 mg/L TiO2 NPs	0.442067	0.52	0.976072
150 mg/L TiO2 NPs	0.3363	0.49	0.980598
C1	0.471367	0.556667	0.975341
C2	1.084433	0.856667	0.973554
C3	1.362333	0.986666	0.980044
C4	3.0131	1.336667	0.971217
Ccom	0.431133	0.626667	0.96776

8.6.3.4 Test Vs Predicted Data Visualization:

8.6.3.4.1 Code to Visualize

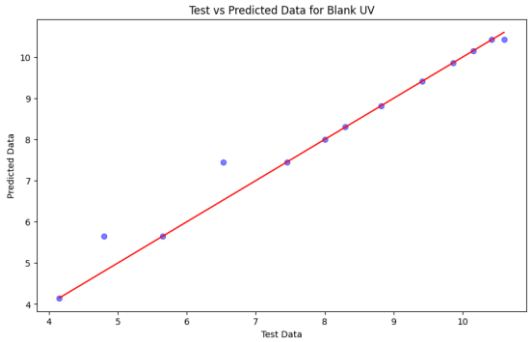
```
import matplotlib.pyplot as plt

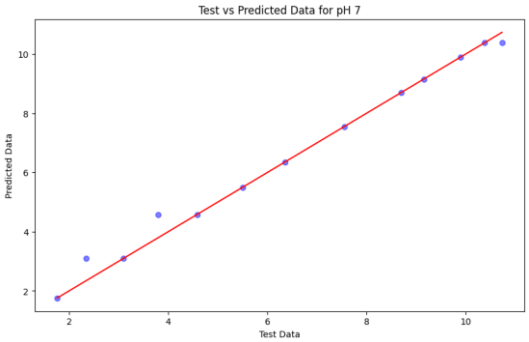
# Iterate over each target variable
for target_variable in y_columns:
    # Get the model and predicted values for the selected target variable
    model = results[target_variable]['model']
    y_test = combined_data[target_variable]
    X_test = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7',
'pH 10', 'pH 12',
                                         '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                         '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2
NPs',
                                         'C1', 'C2', 'C3', 'C4', 'Ccom'])
    y_pred = model.predict(X_test)

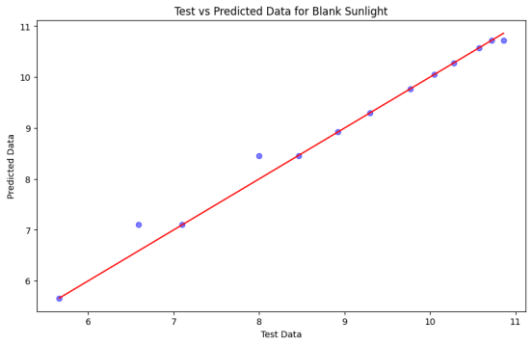
    # Plotting
```

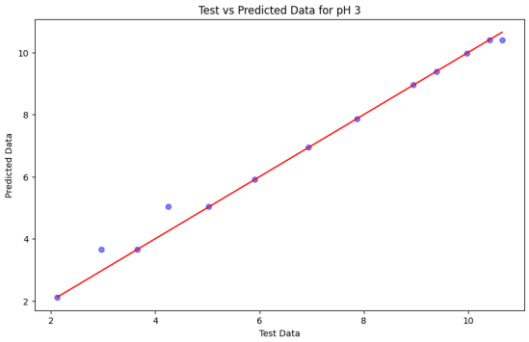
```
15.  plt.figure(figsize=(10, 6))
16.  plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
17.  plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  #
Diagonal line representing perfect prediction
18.  plt.title(f'Test vs Predicted Data for {target_variable}')
19.  plt.xlabel('Test Data')
20.  plt.ylabel('Predicted Data')
21.  plt.show()
22.
```

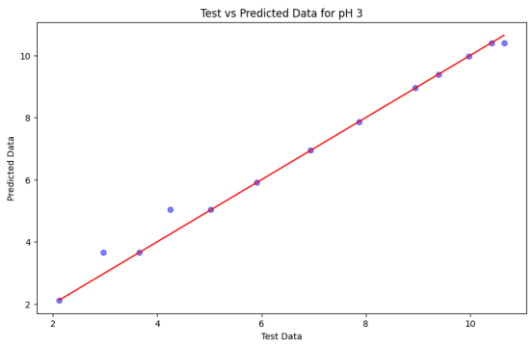
8.6.3.4.2 Graphs

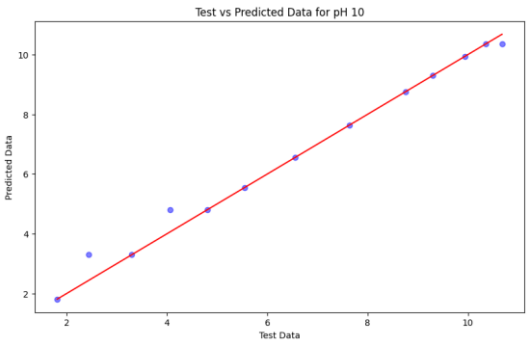


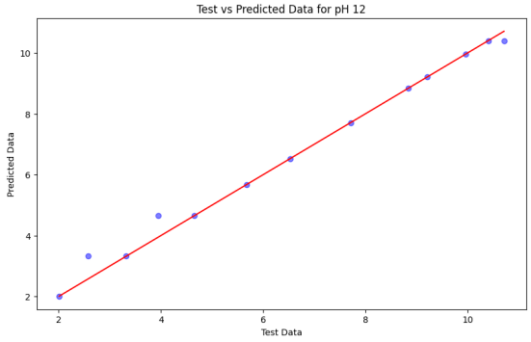


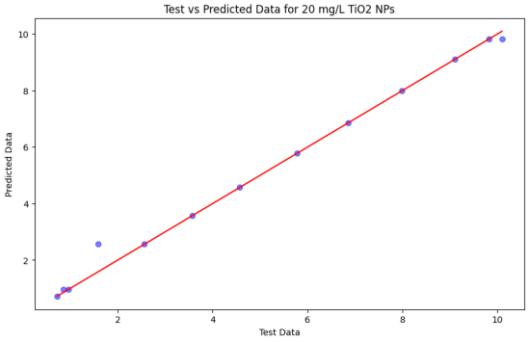


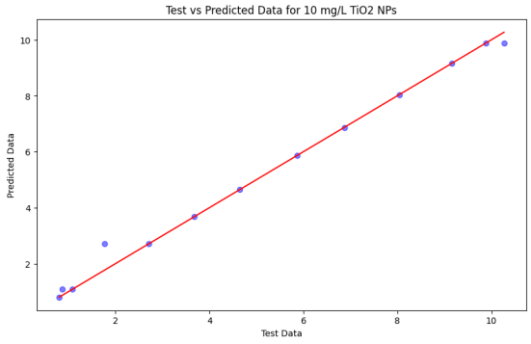


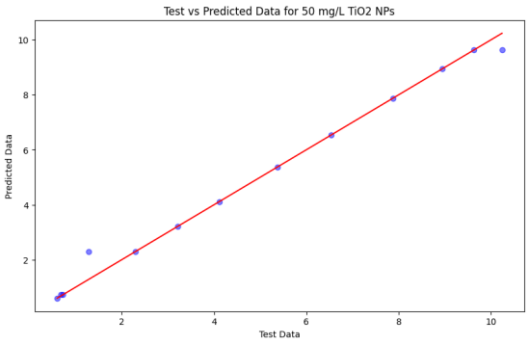


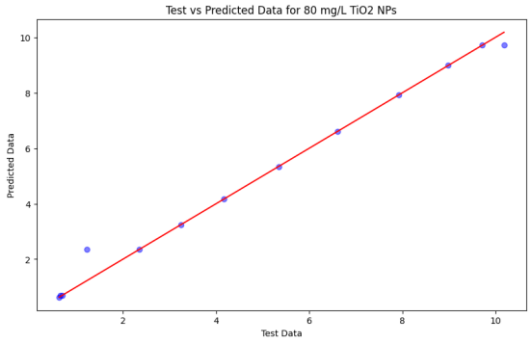


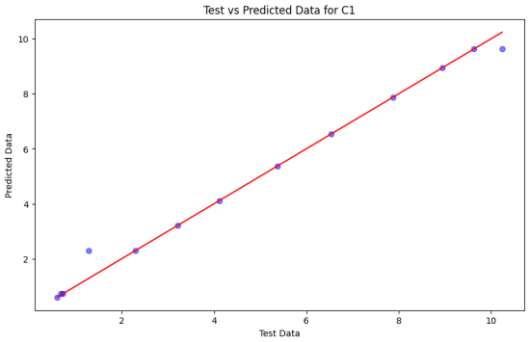

Test vs Predicted Data for pH 12
Predicted Data
Test Data

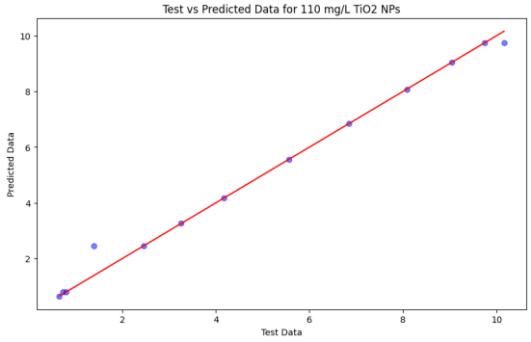

Test vs Predicted Data for 20 mg/L TiO2 NPs
Predicted Data
Test Data

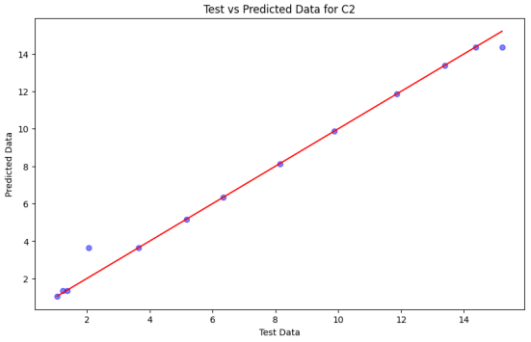

Test vs Predicted Data for 10 mg/L TiO2 NPs
Predicted Data
Test Data

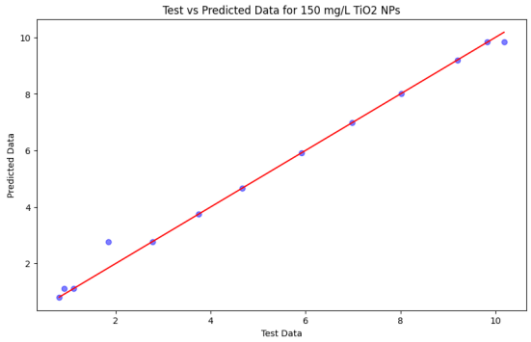

Test vs Predicted Data for 50 mg/L TiO2 NPs
Predicted Data
Test Data

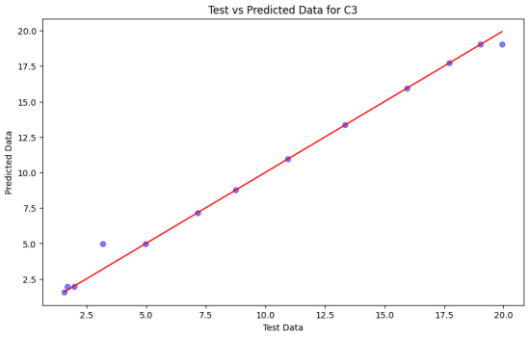

Test vs Predicted Data for 80 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for C1
Predicted Data
Test Data


Test vs Predicted Data for 110 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for C2
Predicted Data
Test Data


Test vs Predicted Data for 150 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for C3
Predicted Data
Test Data

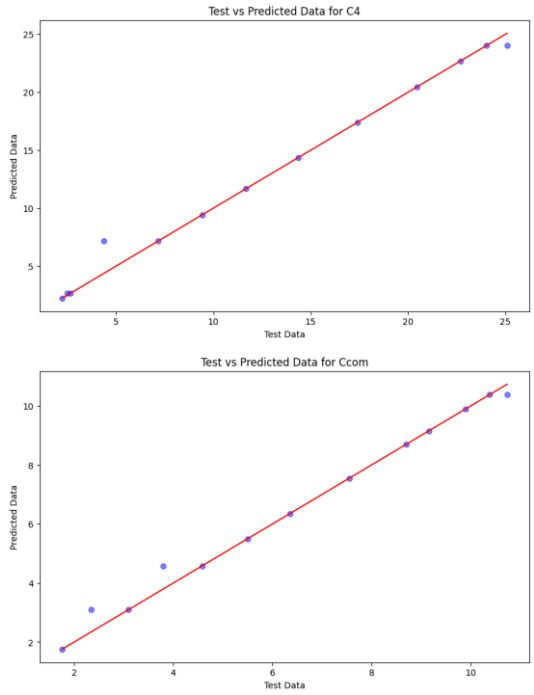


Figure 32: Test vs Predicted Data for Different Parameters using CatBoost Model

8.6.3.4.3 Explanation of Graphs

This code segment iterates over each target variable in `y_columns`, extracts the model and predicted values for the selected target variable from the `results` dictionary, and then plots the test versus predicted data for that target variable. It first retrieves the model and the actual test data (`y_test` and `X_test`) from the `results` dictionary and the `combined_data` DataFrame, respectively. Then, it predicts the target variable values using the model. After that, it creates a scatter plot where the x-axis represents the actual test data (`y_test`) and the y-axis represents the predicted data (`y_pred`). Additionally, it adds a diagonal line in red to represent perfect prediction where the predicted values equal the actual values. This visualization helps in understanding how well the model's predictions align with the actual test data for each target variable.

8.6.4 XGBoost Model

8.6.4.1 Code

```
1. import xgboost as xgb
2. from sklearn.model_selection import train_test_split
3. from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
4. import matplotlib.pyplot as plt
5.
6. # Train-test split for each target variable
```

```
xgb_results = {}

for target in y_columns:
    y = combined_data[target]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Initialize XGBoost model
    xgb_model = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree =
0.3, learning_rate = 0.1,
                max_depth = 5, alpha = 10, n_estimators = 10)

    # Train the model
    xgb_model.fit(X_train, y_train)

    # Predictions
    y_pred_xgb = xgb_model.predict(X_test)

    # Evaluation
    mse_xgb = mean_squared_error(y_test, y_pred_xgb)
    mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
    r2_xgb = r2_score(y_test, y_pred_xgb)

    # Store the results
    xgb_results[target] = {
        'model': xgb_model,
        'mean_squared_error': mse_xgb,
        'mean_absolute_error': mae_xgb,
        'r2_score': r2_xgb,
        'y_test': y_test,
        'y_pred': y_pred_xgb
    }

# Display results for each target variable
for target, metrics in xgb_results.items():
    print(f"Results for {target} using XGBoost:")
    print(f"  Mean Squared Error: {metrics['mean_squared_error']}")
```

```
    print(f"  Mean Absolute Error: {metrics['mean_absolute_error']}")
    print(f"  R^2 Score: {metrics['r2_score']}\n")

```

8.6.4.2 Output

- Results for Blank UV using XGBoost:
  - Mean Squared Error: 6.7407871523387435
  - Mean Absolute Error: 2.496999905904134
  - R^2 Score: -0.14040588842111323
- Results for Blank Sunlight using XGBoost:
  - Mean Squared Error: 3.5152225
  - 529378587
  - Mean Absolute Error: 1.7843333943684894
  - R^2 Score: -0.11395544377374889
- Results for pH 3 using XGBoost:
  - Mean Squared Error: 11.170717101789242
  - Mean Absolute Error: 3.299367326100667
  - R^2 Score: 0.010280939743503192
- Results for pH 7 using XGBoost:
  - Mean Squared Error: 12.665258150890963
  - Mean Absolute Error: 3.5038983281453455
  - R^2 Score: 0.05288227215530927
- Results for pH 10 using XGBoost:
  - Mean Squared Error: 12.195519835110426
  - Mean Absolute Error: 3.425453561147054
  - R^2 Score: 0.04011251324052312
- Results for pH 12 using XGBoost:
  - Mean Squared Error: 12.175681607817898
  - Mean Absolute Error: 3.441048520406087
  - R^2 Score: 0.03598324230755445
- Results for 10 mg/L TiO2 NPs using XGBoost:
  - Mean Squared Error: 15.594658885317463
  - Mean Absolute Error: 3.901634101867676
  - R^2 Score: 0.1313361458972696
- Results for 20 mg/L TiO_2 NPs using XGBoost:
  - Mean Squared Error: 15.262663490487816
  - Mean Absolute Error: 3.870773131052653
  - R^2 Score: 0.13436506975173368
- Results for 50 mg/L TiO2 NPs using XGBoost:

    - Mean Squared Error: 16.188633157258437
    - Mean Absolute Error: 3.9660306866963704
    - R^2 Score: 0.15309367841391808
- Results for 80 mg/L TiO_2 NPs using XGBoost:
    - Mean Squared Error: 16.180785582152925
    - Mean Absolute Error: 3.9777340253194176
    - R^2 Score: 0.1496716086356732
- Results for 110 mg/L TiO_2 NPs using XGBoost:
    - Mean Squared Error: 15.557698453182814
    - Mean Absolute Error: 3.9017494646708175
    - R^2 Score: 0.15789725789786402
- Results for 150 mg/L TiO_2 NPs using XGBoost:
    - Mean Squared Error: 15.103518546445622
    - Mean Absolute Error: 3.841444543202718
    - R^2 Score: 0.12865098058087543
- Results for C_1 using XGBoost:
    - Mean Squared Error: 16.188633157258437
    - Mean Absolute Error: 3.9660306866963704
    - R^2 Score: 0.15309367841391808
- Results for C_2 using XGBoost:
    - Mean Squared Error: 28.158302237233812
    - Mean Absolute Error: 5.233865712483724
    - R^2 Score: 0.3133085378933139
- Results for C_3 using XGBoost:
    - Mean Squared Error: 40.876508381277496
    - Mean Absolute Error: 6.296980311075846
    - R^2 Score: 0.4012097672873334
- Results for C_4 using XGBoost:
    - Mean Squared Error: 56.99186585605136
    - Mean Absolute Error: 7.448723144531249
    - R^2 Score: 0.45557599184280995
- Results for C_{com} using XGBoost:
    - Mean Squared Error: 12.665258150890963
    - Mean Absolute Error: 3.5038983281453455
    - R^2 Score: 0.05288227215530927

8.6.4.3 Explanation

This code segment trains XGBoost regressor models for each target variable in `y_columns`, evaluates their performance using mean squared error (MSE), mean absolute error (MAE), and \(R^2 \) score, and then stores the results in the `xgb_results` dictionary. It splits the data into training and testing sets using `train_test_split` with a test size of 20% and a random state of 42 to ensure reproducibility. The XGBoost model is initialized with specific hyperparameters like `objective='reg:squarederror'`, `colsample_bytree=0.3`, `learning_rate=0.1`, `max_depth=5`, `alpha=10`, and `n_estimators=10`. After training the model, it makes predictions on the test set and computes the evaluation metrics. Finally, it prints the evaluation results for each target variable. Additionally, it includes commented-out code to visualize the actual versus predicted values for each target variable, though it's currently disabled.

8.6.4.4 Visualization

8.6.4.4.1 Visualization Code

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

# Extract metrics from the results dictionary
metrics_df = pd.DataFrame(xgb_results).T

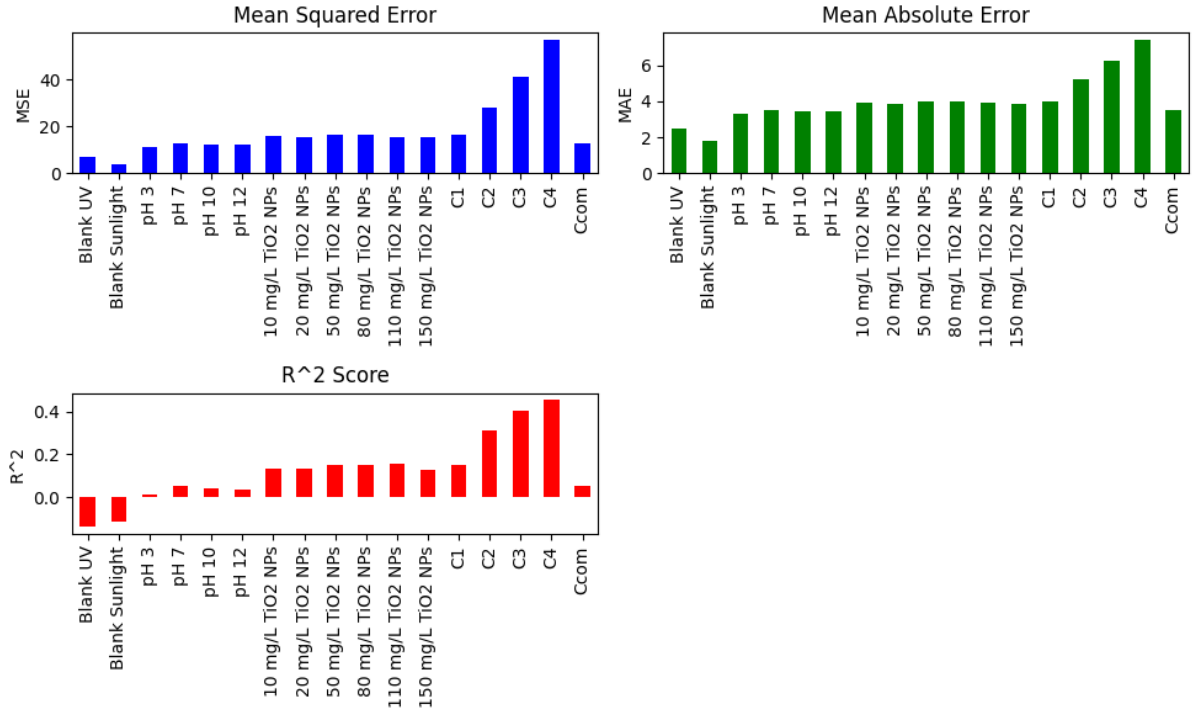
# Plotting
plt.figure(figsize=(10, 6))

# Plot MSE
plt.subplot(2, 2, 1)
metrics_df['mean_squared_error'].plot(kind='bar', color='blue')
plt.title('Mean Squared Error')
plt.ylabel('MSE')

# Plot MAE
plt.subplot(2, 2, 2)
metrics_df['mean_absolute_error'].plot(kind='bar', color='green')
plt.title('Mean Absolute Error')
plt.ylabel('MAE')

# Plot R^2 Score
plt.subplot(2, 2, 3)
```

```
24. metrics_df['r2_score'].plot(kind='bar', color='red')
25. plt.title('R^2 Score')
26. plt.ylabel('R^2')
27.
28. plt.tight_layout()
29. plt.show()
30.
```



This segment of code creates a DataFrame `metrics_df` from the `xgb_results` dictionary and transposes it, making the target variables the index. Then, it plots the mean squared error (MSE), mean absolute error (MAE), and \(R^2 \) score for each target variable using bar plots. The subplotting is arranged in a 2x2 grid for better visualization. Each subplot represents one of the evaluation metrics, with corresponding titles and y-axis labels. Finally, it displays the plots using `plt.tight_layout()` to prevent overlapping. This provides a visual comparison of the performance of the XGBoost models across different target variables.

The visualization provided by the bar plots helps in:

1. **Comparison between target variables:** By plotting the evaluation metrics (MSE, MAE, \(R^2 \) score) for each target variable side by side, the visualization allows for easy comparison between them. This comparison is crucial for understanding which target

variables are better predicted by the XGBoost models and which ones might require further refinement.

2. **Identification of performance differences:**Differences in the heights of the bars across different target variables indicate variations in model performance. For example, if one target variable has a significantly lower MSE compared to others, it suggests that the model performs better in predicting that particular variable.

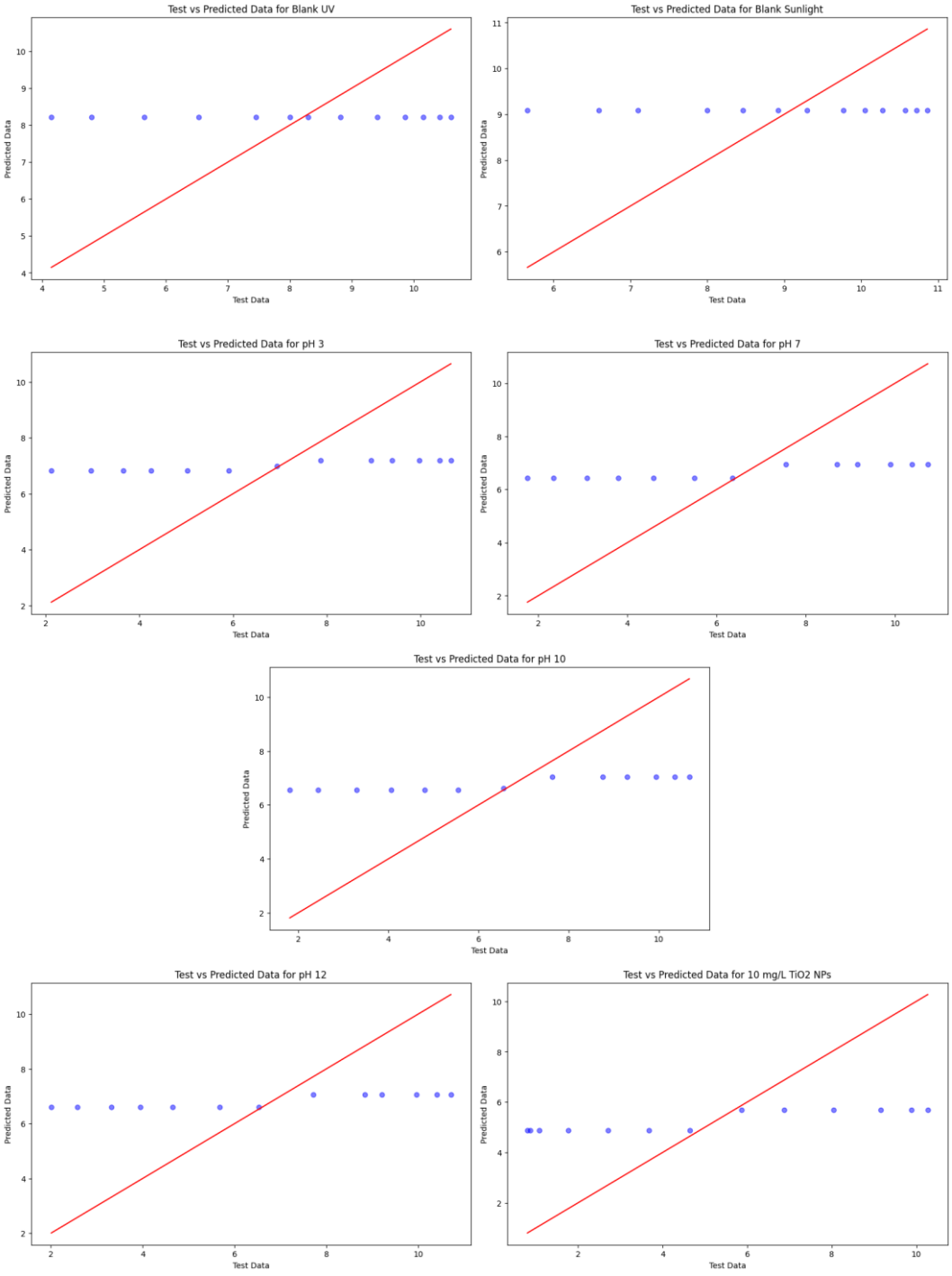
3. **Relative importance of metrics:** The color-coded bars provide a visual representation of the magnitude of each evaluation metric. This helps in quickly identifying which metrics have higher or lower values for different target variables, thereby highlighting areas of strength or weakness in model performance.

4. **Pattern recognition:** Patterns in the bar plots, such as consistently low or high values across all target variables, can reveal underlying trends in model performance. These patterns can inform further analysis and optimization efforts to improve model accuracy.

```
1. import matplotlib.pyplot as plt
2.
3. # Iterate over each target variable
4. for target_variable in y_columns:
5.     # Get the model and predicted values for the selected target variable
6.     model = xgb_results[target_variable]['model']
7.     y_test = combined_data[target_variable]
8.     X_test = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7',
'pH 10', 'pH 12',
9.                                          '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
10.                                          '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2
NPs',
11.                                          'C1', 'C2', 'C3', 'C4', 'Ccom'])
12.     y_pred = model.predict(X_test)
13.
14.     # Plotting
15.     plt.figure(figsize=(10, 6))
16.     plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
17.     plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  #
Diagonal line representing perfect prediction
18.     plt.title(f'Test vs Predicted Data for {target_variable}')
19.     plt.xlabel('Test Data')
20.     plt.ylabel('Predicted Data')
```

```
21.  plt.show()
22.
```

8.6.4.4.2 Graphs



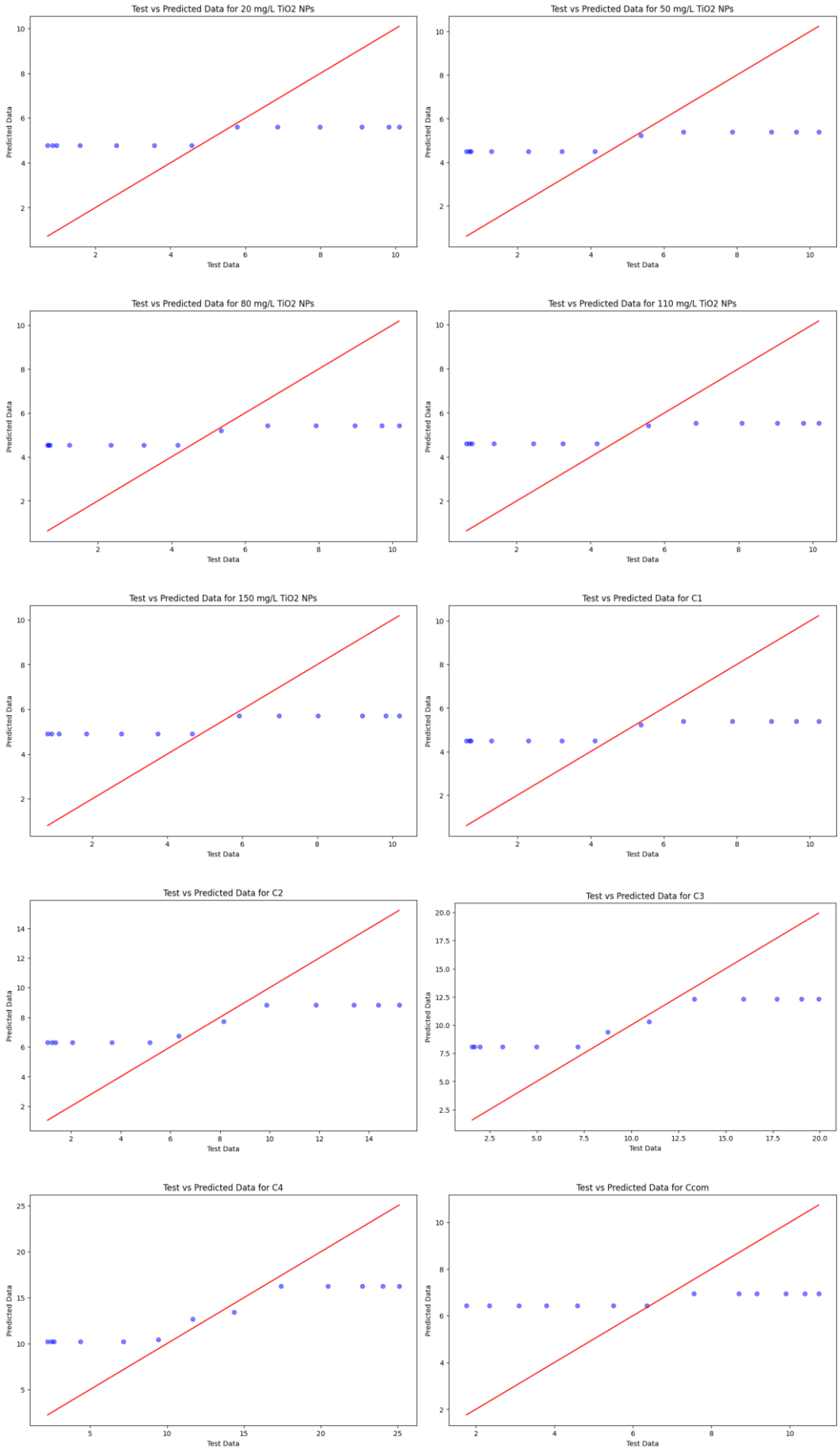


Figure 33: Test Vs Predicted Data for Different Parameters Using XG Boost Model

8.6.4.4.3 Explanation of Graphs

This code iterates over each target variable in `y_columns`, retrieves the XGBoost model and predicted values for that target variable from the `xgb_results` dictionary, and then generates a scatter plot comparing the actual test data (`y_test`) with the predicted data (`y_pred`). Additionally, a diagonal line representing perfect prediction is plotted to serve as a reference.

The visualization helps in:

1. Assessment of model performance: By comparing the scatter plot to the diagonal line, you can visually assess how well the model predictions align with the actual test data. If the points cluster closely around the diagonal line, it indicates that the model's predictions are accurate. Deviations from the diagonal line suggest areas where the model might be overestimating or underestimating.
2. Detection of patterns and trends: Examining the distribution of points in the scatter plot can reveal patterns or trends in the data. For example, you might observe clusters of points in certain regions, indicating consistent over- or under-prediction for specific ranges of the target variable. Identifying such patterns can provide insights into potential biases or limitations of the model.
3. Identification of outliers: Outliers in the scatter plot, i.e., data points that deviate significantly from the general trend, can be easily spotted. These outliers may represent unusual or anomalous data instances that could influence model performance. Understanding the presence and impact of outliers is essential for improving the robustness of the model.
4. Validation of assumptions: The scatter plot allows you to validate the assumptions underlying the model, such as linearity and homoscedasticity (constant variance). If the points in the plot exhibit a clear linear trend and are evenly spread around the diagonal line, it provides evidence supporting the validity of these assumptions. Conversely, deviations from these patterns may indicate violations of assumptions that need to be addressed.

8.6.5 Random Forest

8.6.5.1 Code

```
1. from sklearn.ensemble import RandomForestRegressor
2.
3. # Train-test split for each target variable
4. rf_results = {}
```

```
5.
6. for target in y_columns:
7.     y = combined_data[target]
8.
9.     # Split the data
10.     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
11.
12.     # Initialize Random Forest model
13.     rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
14.
15.     # Train the model
16.     rf_model.fit(X_train, y_train)
17.
18.     # Predictions
19.     y_pred_rf = rf_model.predict(X_test)
20.
21.     # Evaluation
22.     mse_rf = mean_squared_error(y_test, y_pred_rf)
23.     mae_rf = mean_absolute_error(y_test, y_pred_rf)
24.     r2_rf = r2_score(y_test, y_pred_rf)
25.
26.     # Store the results
27.     rf_results[target] = {
28.         'model': rf_model,
29.         'mean_squared_error': mse_rf,
30.         'mean_absolute_error': mae_rf,
31.         'r2_score': r2_rf,
32.         'y_test': y_test,
33.         'y_pred': y_pred_rf
34.     }
35.
36. # Display results for each target variable
37. for target, metrics in rf_results.items():
38.     print(f"Results for {target} using Random Forest:")
39.     print(f"  Mean Squared Error: {metrics['mean_squared_error']}")
40.     print(f"  Mean Absolute Error: {metrics['mean_absolute_error']}")
41.     print(f"  R^2 Score: {metrics['r2_score']}\n")
```

8.6.5.2 Output

- Results for Blank UV using Random Forest:
    - Mean Squared Error: 0.18896610666666527
    - Mean Absolute Error: 0.4178666666666671
    - R^2 Score: 0.9680307275864793
- Results for Blank Sunlight using Random Forest:
    - Mean Squared Error: 0.03734088666666582
    - Mean Absolute Error: 0.16540000000000057
    - R^2 Score: 0.9881668704182308
- Results for pH 3 using Random Forest:
    - Mean Squared Error: 0.1849599666666685
    - Mean Absolute Error: 0.4282666666666686
    - R^2 Score: 0.9836126541629913
- Results for pH 7 using Random Forest:
    - Mean Squared Error: 0.2696129700000031
    - Mean Absolute Error: 0.5071666666666693
    - R^2 Score: 0.9798381351172145
- Results for pH 10 using Random Forest:
    - Mean Squared Error: 0.257554006666665
    - Mean Absolute Error: 0.4895999999999991
    - R^2 Score: 0.9797283862019269
- Results for pH 12 using Random Forest:
    - Mean Squared Error: 0.23931988333333284
    - Mean Absolute Error: 0.4792333333333321
    - R^2 Score: 0.9810517073775814
- Results for 10 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.33091956333333844
    - Mean Absolute Error: 0.5301666666666698
    - R^2 Score: 0.9815669027840183
- Results for 20 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.2774787666666589
    - Mean Absolute Error: 0.47919999999999413
    - R^2 Score: 0.9842625559438846
- Results for 50 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.47214071999999946
    - Mean Absolute Error: 0.5839999999999996
    - R^2 Score: 0.9753000172057812
- Results for 80 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.4253583366666594

    - Mean Absolute Error: 0.5760999999999961
    - R^2 Score: 0.9776466804819348
- Results for 110 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.35606053999999926
    - Mean Absolute Error: 0.5303999999999994
    - R^2 Score: 0.9807272548705926
- Results for 150 mg/L TiO2 NPs using Random Forest:
    - Mean Squared Error: 0.28315265666666095
    - Mean Absolute Error: 0.491499999999997
    - R^2 Score: 0.9836644164090839
- Results for C1 using Random Forest:
    - Mean Squared Error: 0.47214071999999946
    - Mean Absolute Error: 0.5839999999999996
    - R^2 Score: 0.9753000172057812
- Results for C2 using Random Forest:
    - Mean Squared Error: 1.0349158866666766
    - Mean Absolute Error: 0.8916666666666707
    - R^2 Score: 0.9747616920443144
- Results for C3 using Random Forest:
    - Mean Squared Error: 1.3231266500000076
    - Mean Absolute Error: 1.0313666666666692
    - R^2 Score: 0.9806178329305465
- Results for C4 using Random Forest:
    - Mean Squared Error: 2.022659339999985
    - Mean Absolute Error: 1.2705999999999962
    - R^2 Score: 0.980678219804196
- Results for Ccom using Random Forest:
    - Mean Squared Error: 0.2696129700000031
    - Mean Absolute Error: 0.5071666666666693
    - R^2 Score: 0.9798381351172145

8.6.5.3 Explanation

This code trains Random Forest regression models for each target variable in `y_columns`, evaluates their performance, and stores the results in the `rf_results` dictionary. The process includes the following steps:

1. **Train-Test Split:** The data is split into training and testing sets using the `train_test_split` function from scikit-learn.

2. **Model Initialization:** A Random Forest regression model is initialized with 100 decision trees (`n_estimators=100`) and a random state for reproducibility.
3. **Model Training:** The Random Forest model is trained on the training data using the `fit` method.
4. **Prediction:** The trained model is used to make predictions on the test data.
5. **Evaluation:** The Mean Squared Error, Mean Absolute Error, and R-squared score are calculated to evaluate the performance of the model.
6. **Results Storage:** The model object along with the evaluation metrics and the actual vs. predicted values are stored in the `rf_results` dictionary for each target variable.
7. **Results Display:** The evaluation metrics for each target variable are printed to the console.

8.6.5.4 Visualization

8.6.5.4.1 Visualization Code

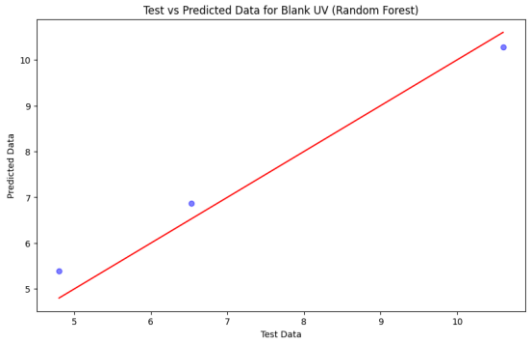
```
import matplotlib.pyplot as plt

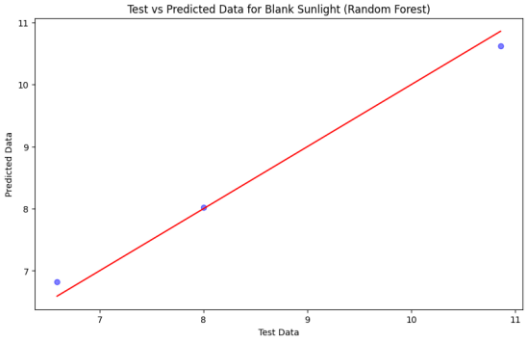
# Iterate over each target variable
for target_variable in y_columns:
    # Get the model and predicted values for the selected target variable
    rf_model = rf_results[target_variable]['model']
    y_test_rf = rf_results[target_variable]['y_test']
    y_pred_rf = rf_results[target_variable]['y_pred']

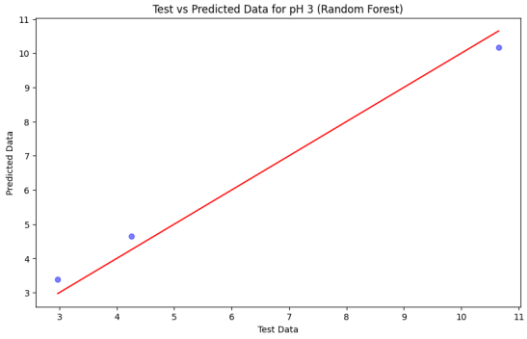
    # Plotting
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test_rf, y_pred_rf, color='blue', alpha=0.5)
    plt.plot([min(y_test_rf), max(y_test_rf)], [min(y_test_rf), max(y_test_rf)],
color='red')  # Diagonal line representing perfect prediction
    plt.title(f'Test vs Predicted Data for {target_variable} (Random Forest)')
    plt.xlabel('Test Data')
    plt.ylabel('Predicted Data')
    plt.show()

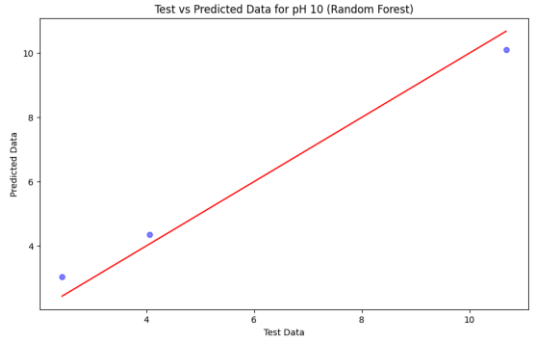
```

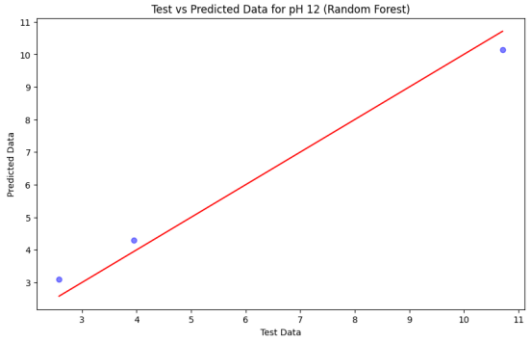
8.6.5.4.2 Graphs

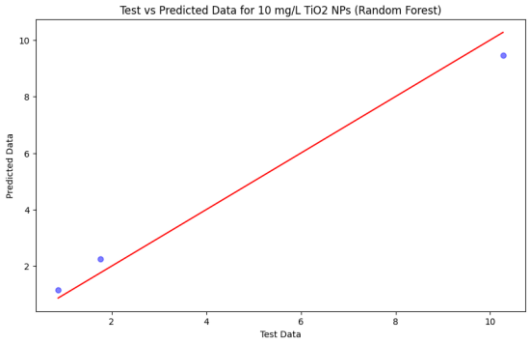


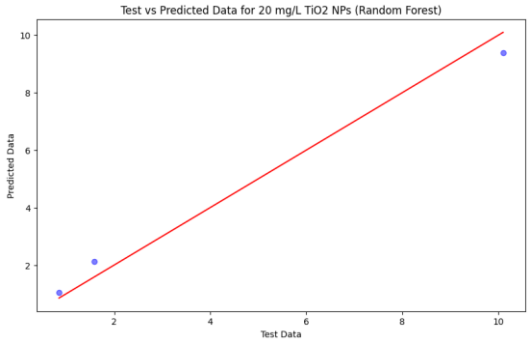


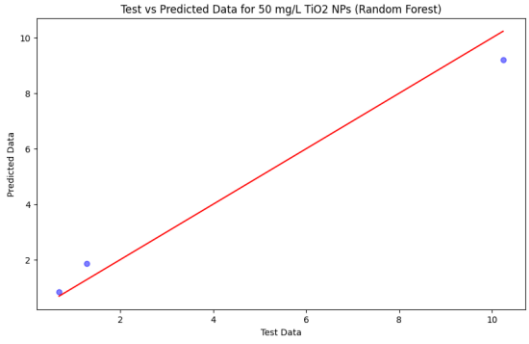


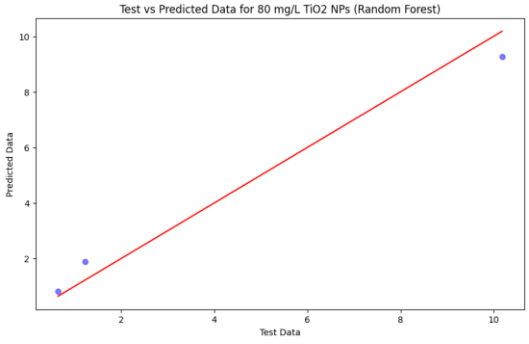












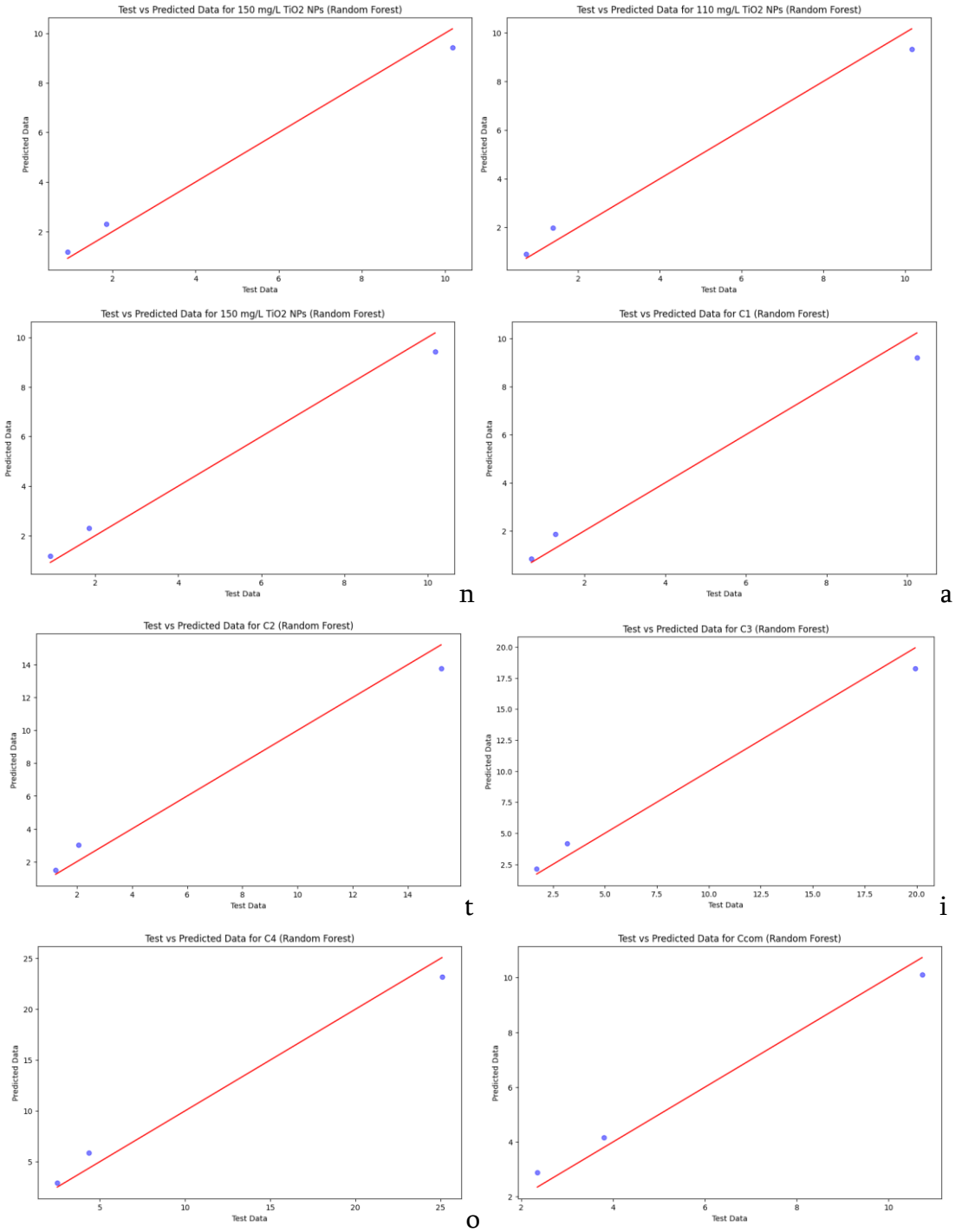


Figure 34: Test Vs Predicted Data for Different Parameters Using Random Forest Model

8.6.5.4.3 Explanation of Graphs

This code iterates over each target variable in `y_columns`, retrieves the Random Forest model and the corresponding actual (`y_test_rf`) and predicted (`y_pred_rf`) values from the `rf_results` dictionary, and then plots a scatter plot of the actual vs. predicted values for each target variable using Matplotlib.

1. **Iteration Over Target Variables:** The loop iterates over each target variable in `y_columns`.

2. **Model and Data Retrieval:** Inside the loop, it retrieves the Random Forest model (`rf_model`) and the actual (`y_test_rf`) and predicted (`y_pred_rf`) values from the `rf_results` dictionary for the current target variable.
3. **Plotting:** It creates a scatter plot using `plt.scatter()` with `y_test_rf` (actual) on the x-axis and `y_pred_rf` (predicted) on the y-axis. Additionally, it overlays a diagonal line (`plt.plot()`) from the minimum to the maximum of `y_test_rf`, representing perfect prediction.
4. **Plot Customization:** The plot is customized with a title (`plt.title()`), x-axis label (`plt.xlabel()`), and y-axis label (`plt.ylabel()`).
5. **Display:** Finally, it displays the plot using `plt.show()`.

This visualization helps in assessing the performance of the Random Forest models by visually comparing the predicted values against the actual values for each target variable. It provides insights into how well the model is capturing the underlying patterns in the data and whether there are any systematic deviations between predicted and actual values.

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

# Extract metrics from the results dictionary
metrics_df_rf = pd.DataFrame(rf_results).T

# Plotting
plt.figure(figsize=(10, 6))

# Plot MSE
plt.subplot(2, 2, 1)
metrics_df_rf['mean_squared_error'].plot(kind='bar', color='blue')
plt.title('Mean Squared Error (Random Forest)')
plt.ylabel('MSE')

# Plot MAE
plt.subplot(2, 2, 2)
metrics_df_rf['mean_absolute_error'].plot(kind='bar', color='green')
plt.title('Mean Absolute Error (Random Forest)')
plt.ylabel('MAE')

# Plot R^2 Score
plt.subplot(2, 2, 3)
```

```
24. metrics_df_rf['r2_score'].plot(kind='bar', color='red')
25. plt.title('R^2 Score (Random Forest)')
26. plt.ylabel('R^2')
27.
28. plt.tight_layout()
29. plt.show()
30.
```

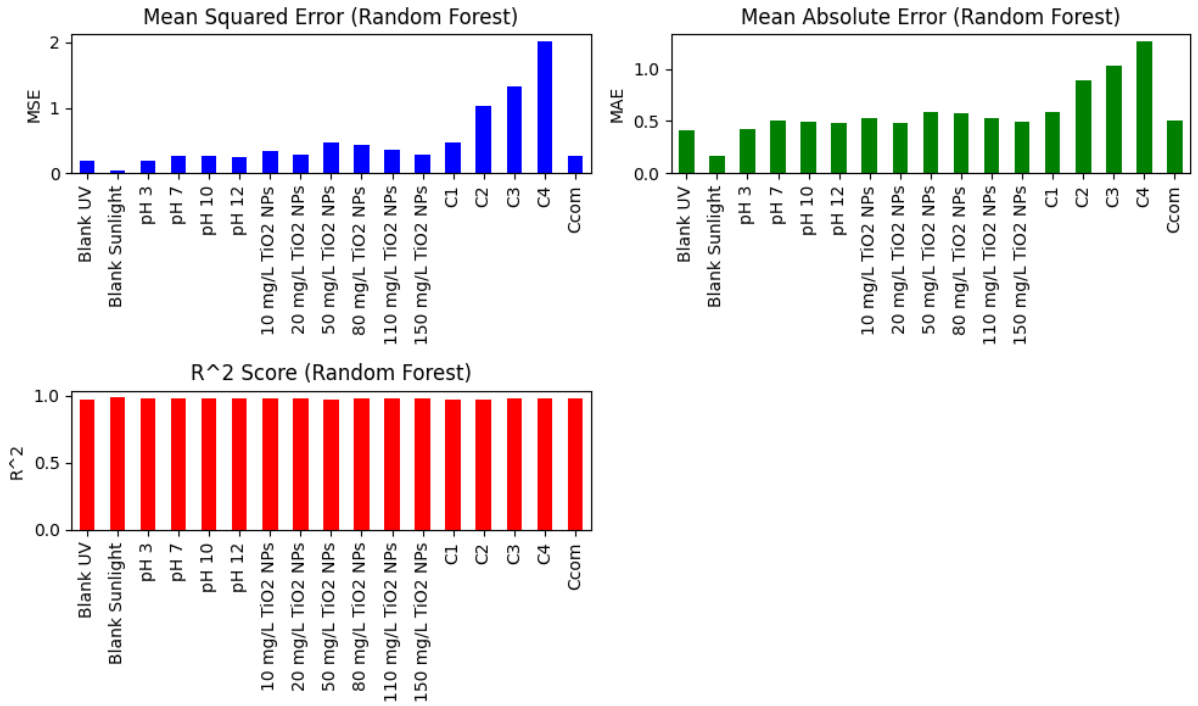


Figure 35: Plot of MSE, MAE, R2 in Random Forest Model

This code snippet extracts the evaluation metrics from the Random Forest regression results stored in the `rf_results` dictionary and creates a bar plot for each metric (Mean Squared Error, Mean Absolute Error, and R-squared Score). Here's how it works:

1. **DataFrame Creation:** It creates a Pandas DataFrame (`metrics_df_rf`) from the `rf_results` dictionary and transposes it so that the metrics are columns and target variables are rows.

2. **Plotting:** It initializes a Matplotlib figure with a size of 10x6 inches. Then, it creates subplots for each metric:

-**MSE Plot:** It creates a subplot at position (2, 2, 1) and plots the mean squared error (`mean_squared_error`) as a bar plot with blue color.

- **MAE Plot:** It creates a subplot at position (2, 2, 2) and plots the mean absolute error (`mean_absolute_error`) as a bar plot with green color.

- **R-squared Plot:** It creates a subplot at position (2, 2, 3) and plots the R-squared score (`r2_score`) as a bar plot with red color.

3. **Plot Customization:** It adds titles to each subplot indicating the metric being plotted, and labels the y-axis accordingly.

4. **Display:** Finally, it adjusts the layout to prevent overlapping of subplots and displays the plot using `plt.show()`.

This visualization provides a comparative view of model performance across different target variables, allowing for easy identification of which variables the Random Forest model performs better or worse on based on the chosen evaluation metrics.

8.6.6 LightGBM (Light Gradient-Boosting Machine)

8.6.6.1 Code

```
1. import lightgbm as lgb
2. from sklearn.model_selection import train_test_split
3. from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
4. import pandas as pd
5. import matplotlib.pyplot as plt
6.
7. # Define features and target
8. X = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10',
'pH 12',
9.                          '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
10.                         '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
11.                         'C1', 'C2', 'C3', 'C4','Ccom'])
12. y_columns = ['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
13.             '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
14.             '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
15.             'C1', 'C2', 'C3', 'C4', 'Ccom']
16.
17. # Train-test split for each target variable
18. lgb_results = {}
19.
20. for target in y_columns:
21.     y = combined_data[target]
22.
```

```
    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Initialize LightGBM model
    model = lgb.LGBMRegressor()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Evaluation
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store the results
    lgb_results[target] = {
        'model': model,
        'mean_squared_error': mse,
        'mean_absolute_error': mae,
        'r2_score': r2
    }

# Visualize the regression model evaluation metrics
# Extract metrics from the results dictionary
metrics_df_lgb = pd.DataFrame(lgb_results).T
# Display results for each target variable
for target, metrics in lgb_results.items():
    print(f"Results for {target}:")
    print(f"  Mean Squared Error: {metrics['mean_squared_error']}")
    print(f"  Mean Absolute Error: {metrics['mean_absolute_error']}")
    print(f"  R^2 Score: {metrics['r2_score']}\n")

# Plotting
plt.figure(figsize=(10, 6))

```

```
# Plot MSE
plt.subplot(2, 2, 1)
metrics_df_lgb['mean_squared_error'].plot(kind='bar', color='blue')
plt.title('Mean Squared Error')
plt.ylabel('MSE')

# Plot MAE
plt.subplot(2, 2, 2)
metrics_df_lgb['mean_absolute_error'].plot(kind='bar', color='green')
plt.title('Mean Absolute Error')
plt.ylabel('MAE')

# Plot R^2 Score
plt.subplot(2, 2, 3)
metrics_df_lgb['r2_score'].plot(kind='bar', color='red')
plt.title('R^2 Score')
plt.ylabel('R^2')

plt.tight_layout()
plt.show()
```

8.6.6.2 Output

- Results for Blank UV:
  - Mean Squared Error: 6.740787499857592
  - Mean Absolute Error: 2.4969999694824216
  - R^2 Score: -0.1404059472143271
- Results for Blank Sunlight:
  - Mean Squared Error: 3.515222381371817
  - Mean Absolute Error: 1.7843333466847735
  - R^2 Score: -0.1139553894053773
- Results for pH 3:
  - Mean Squared Error: 12.430229229314172
  - Mean Absolute Error: 3.4853333171208702
  - R^2 Score: -0.10131110469525262
- Results for pH 7:
  - Mean Squared Error: 14.518035468641292
  - Mean Absolute Error: 3.758999969164531

    - R^2 Score: -0.08566983807281314
- Results for pH 10:
    - Mean Squared Error: 13.86364889119403
    - Mean Absolute Error: 3.660999983151754
    - R^2 Score: -0.09118293204461425
- Results for pH 12:
    - Mean Squared Error: 13.803766707992557
    - Mean Absolute Error: 3.6700000063578293
    - R^2 Score: -0.09292135376121857
- Results for 10 mg/L TiO2 NPs:
    - Mean Squared Error: 18.899195658779302
    - Mean Absolute Error: 4.304333331982295
    - R^2 Score: -0.05273531541325638
- Results for 20 mg/L TiO2 NPs:
    - Mean Squared Error: 18.6667225691611
    - Mean Absolute Error: 4.287999984025955
    - R^2 Score: -0.05869903370339302
- Results for 50 mg/L TiO2 NPs:
    - Mean Squared Error: 19.862094344049137
    - Mean Absolute Error: 4.403666668732961
    - R^2 Score: -0.03908298474057759
- Results for 80 mg/L TiO2 NPs:
    - Mean Squared Error: 19.918115564594427
    - Mean Absolute Error: 4.42099998195966
    - R^2 Score: -0.04673157437618092
- Results for 110 mg/L TiO2 NPs:
    - Mean Squared Error: 19.411200922020118
    - Mean Absolute Error: 4.366999986569087
    - R^2 Score: -0.05068404385897596
- Results for 150 mg/L TiO2 NPs:
    - Mean Squared Error: 18.310291645459333
    - Mean Absolute Error: 4.2383333297570545
    - R^2 Score: -0.056353499450223055
- Results for C1:
    - Mean Squared Error: 19.862094344049137
    - Mean Absolute Error: 4.403666668732961
    - R^2 Score: -0.03908298474057759
- Results for C2:
    - Mean Squared Error: 42.839974375955904

- Mean Absolute Error: 6.480333338578543
- R^2 Score: -0.044730765121869354

- Results for C3:
  - Mean Squared Error: 71.76704423021889
  - Mean Absolute Error: 8.392666687170665
  - R^2 Score: -0.05129833289273655
- Results for C4:
  - Mean Squared Error: 111.23134859891957
  - Mean Absolute Error: 10.473000060717265
  - R^2 Score: -0.06255543185595713
- Results for Ccom:
  - Mean Squared Error: 14.518035468641292
  - Mean Absolute Error: 3.758999969164531
  - R^2 Score: -0.08566983807281314

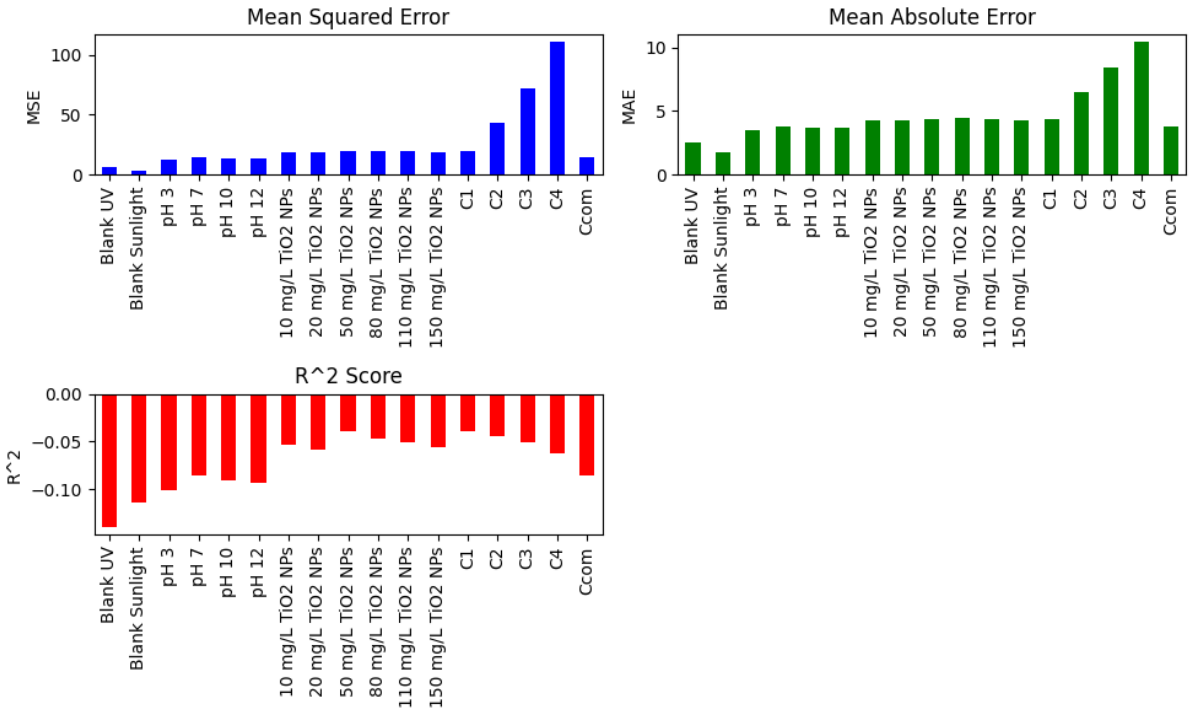


Figure 36: Plot of MSE, MAE, R2 in LightGBM Model

8.6.6.3 Explanation

1. **Importing Libraries**: The code begins by importing necessary libraries including LightGBM for gradient boosting, train-test split from scikit-learn for data splitting, and various evaluation metrics such as mean squared error, mean absolute error, and R-squared score.

2. **Defining Features and Target Variables**: The features (`X`) are defined as all columns in the `combined_data` except the ones specified in the `drop` function. The target variables (`y_columns`) are explicitly listed.

3. **Training the LightGBM Model**: For each target variable, the data is split into training and testing sets. A LightGBM regression model is initialized and trained on the training data.
4. **Making Predictions and Evaluating Performance**: After training, the model predicts the target variable for the test data. Performance metrics such as mean squared error, mean absolute error, and R-squared score are computed to evaluate the model's performance on the test data.
5. **Storing Results**: The results, including the trained model and evaluation metrics, are stored in a dictionary (`lgb_results`) for each target variable.
6. **Visualizing the Results**: The evaluation metrics are extracted from the results dictionary and stored in a pandas DataFrame (`metrics_df_lgb`). Subsequently, the mean squared error, mean absolute error, and R-squared score are plotted in separate subplots using matplotlib.
7. **Explanation of Visualization**: The visualization consists of three subplots, each representing a different evaluation metric (mean squared error, mean absolute error, and R-squared score). Each subplot displays the performance metric for each target variable using bars of different colors (blue for MSE, green for MAE, and red for R-squared score). This visualization allows easy comparison of model performance across different target variables, helping identify which variables the model performs well on and which ones it struggles with. Additionally, the bar plots provide a clear indication of the magnitude of errors and the goodness of fit for each target variable, aiding in model selection and tuning decisions.

8.6.6.4 Visualization

8.6.6.4.1 Visualization Code

```
import matplotlib.pyplot as plt

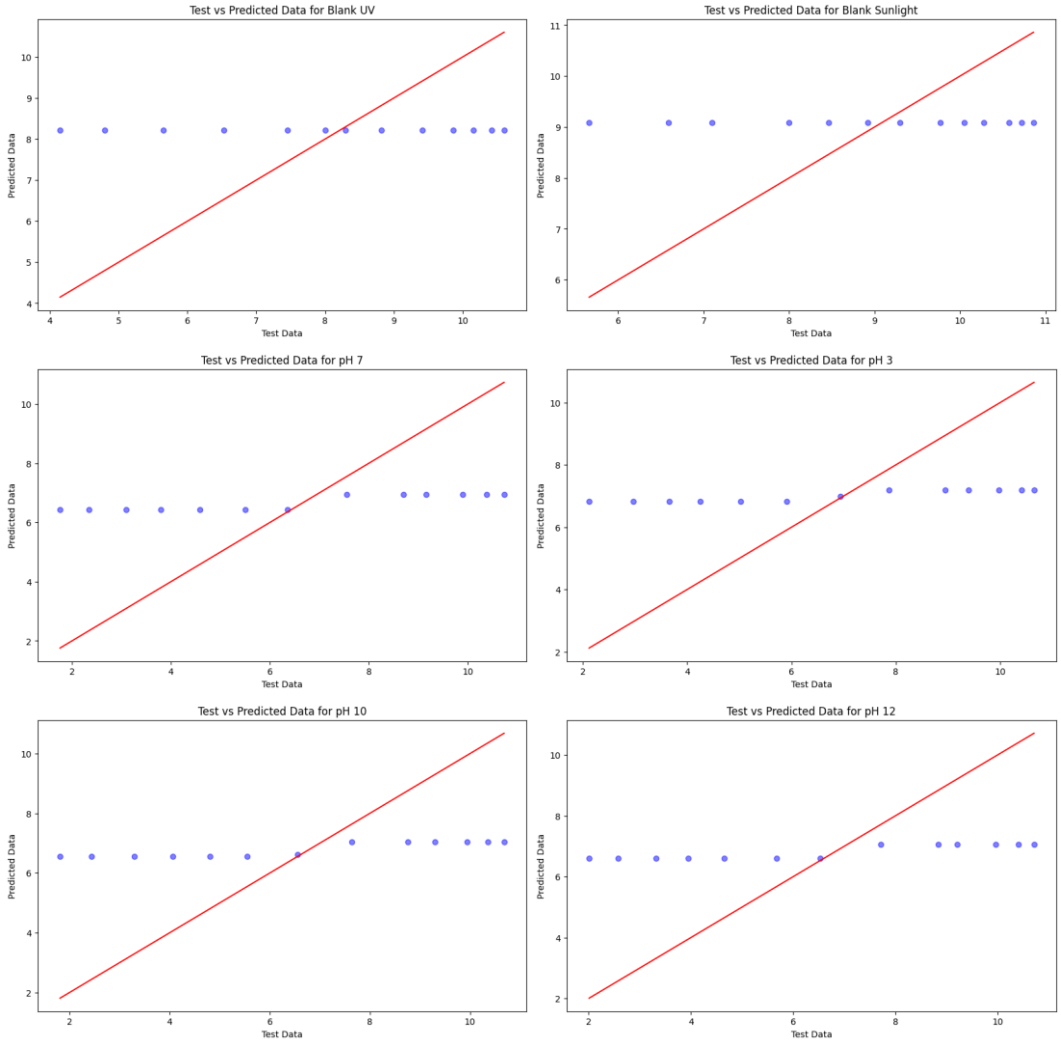
# Iterate over each target variable
for target_variable in y_columns:
    # Get the model and predicted values for the selected target variable
    model = xgb_results[target_variable]['model']
    y_test = combined_data[target_variable]
    X_test = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
                                         '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                         '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
```

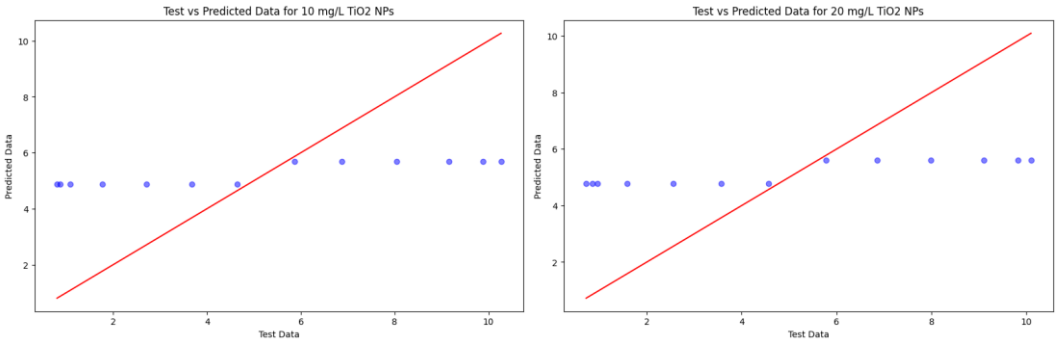
```
                                    'C1', 'C2', 'C3', 'C4', 'Ccom'])
    y_pred = model.predict(X_test)

    # Plotting
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  # Diagonal line representing perfect prediction
    plt.title(f'Test vs Predicted Data for {target_variable}')
    plt.xlabel('Test Data')
    plt.ylabel('Predicted Data')
    plt.show()

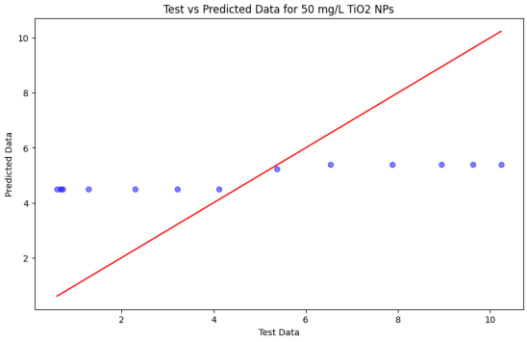
```

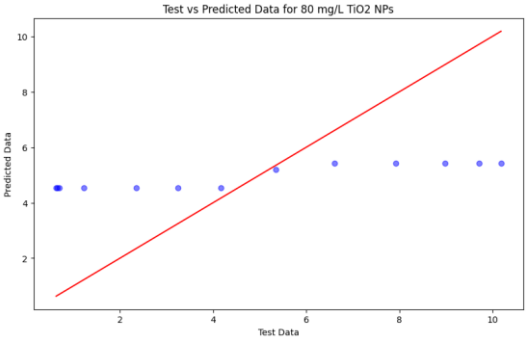
8.6.6.4.2 Graphs

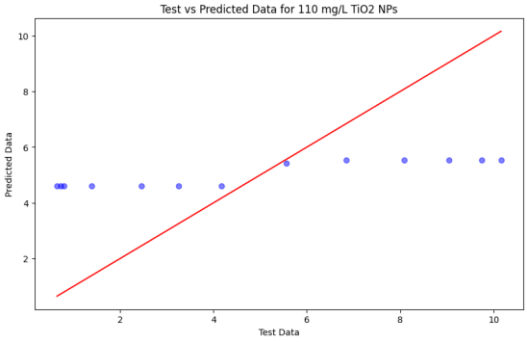


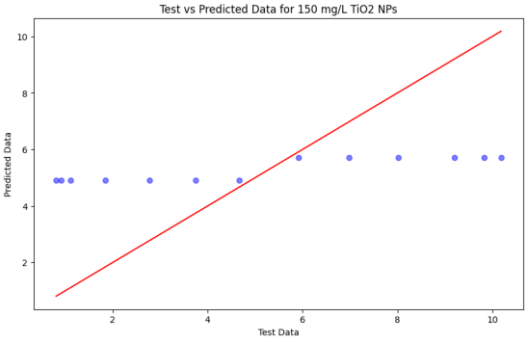

Test vs Predicted Data for 10 mg/L TiO2 NPs
Predicted Data
Test Data

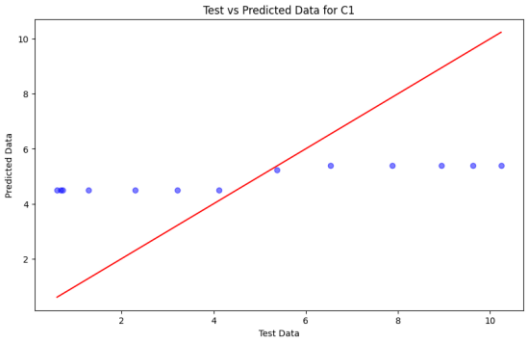
Test vs Predicted Data for 20 mg/L TiO2 NPs
Predicted Data
Test Data

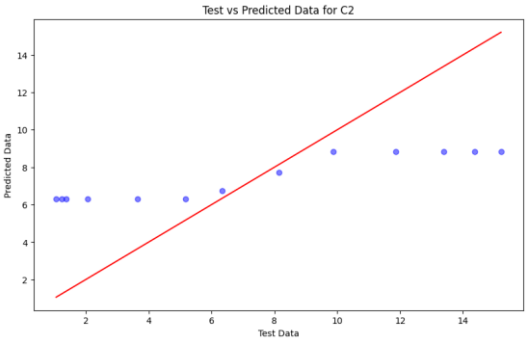

Test vs Predicted Data for 50 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for 80 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for 110 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for 150 mg/L TiO2 NPs
Predicted Data
Test Data


Test vs Predicted Data for C1
Predicted Data
Test Data


Test vs Predicted Data for C2
Predicted Data
Test Data

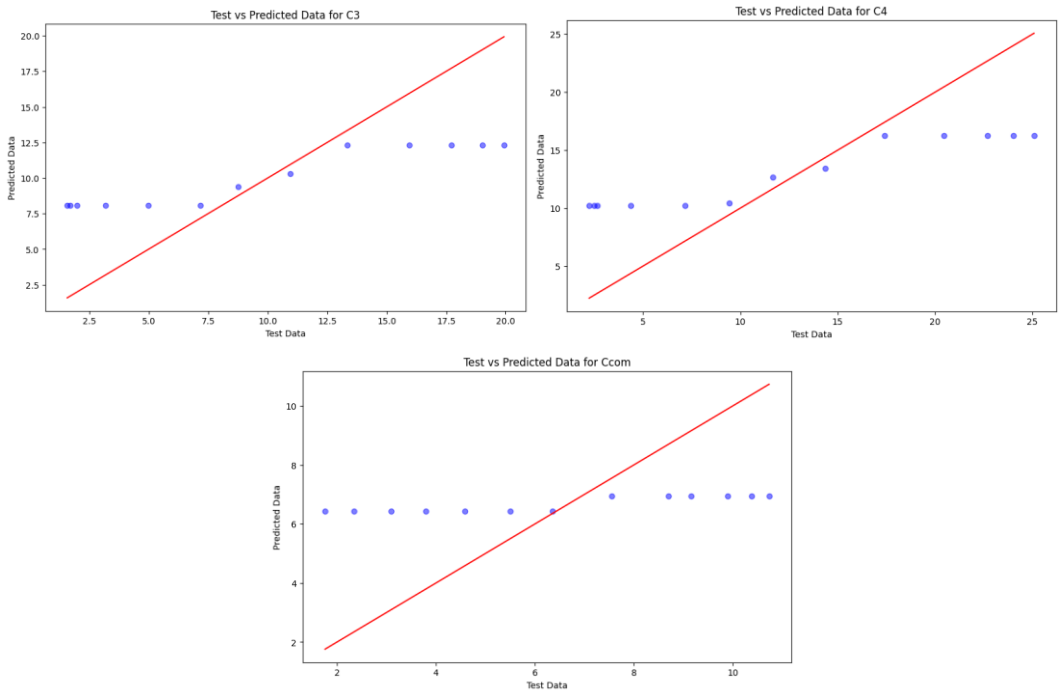


Figure 37: Test Vs Predicted Data for Different Parameters Using LightGBM Model

8.6.6.4.3 Explanation of Graphs

This code snippet iterates over each target variable in `y_columns` and creates a scatter plot to visualize the relationship between the actual test data and the predicted values generated by the XGBoost model.

1. **Iteration over Target Variables**: The code iterates over each target variable in `y_columns`.
2. **Extracting Model and Data**: For each target variable, it retrieves the XGBoost model (`model`) and the corresponding test data (`y_test`) and feature data (`X_test`) from the `xgb_results` and `combined_data` dictionaries, respectively.
3. **Making Predictions**: Using the XGBoost model, predictions (`y_pred`) are made on the feature data (`X_test`).
4. **Plotting**: For each target variable, a scatter plot is created using `plt.scatter()`. The x-axis represents the actual test data (`y_test`), while the y-axis represents the predicted data (`y_pred`). Each point on the plot corresponds to a pair of actual and predicted values. Additionally, a diagonal line (`plt.plot()`) in red is added to visualize perfect predictions.
5. **Customizing Plot**: Various attributes such as figure size (`figsize`), scatter point color (`color`), and transparency (`alpha`) are customized to enhance visualization clarity. Titles, axis labels, and gridlines are also added for better interpretation.
6. **Displaying Plot**: The `plt.show()` function is called to display the scatter plot for each target variable.

This visualization is helpful for assessing how well the XGBoost model performs in predicting each target variable. By comparing the scatter plot against the diagonal line representing perfect predictions, it becomes apparent how closely the predicted values

align with the actual test data. This aids in identifying any patterns, trends, or discrepancies in the model's predictions across different target variables, thereby facilitating model evaluation and interpretation.

8.6.7 SVM (Support Vector Machine)

8.6.7.1 Code

```
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Define features and target
X = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
                                '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
                                'C1', 'C2', 'C3', 'C4','Ccom'])
y_columns = ['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
             '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
             '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
             'C1', 'C2', 'C3', 'C4', 'Ccom']

# Train-test split for each target variable
svm_results = {}

for target in y_columns:
    y = combined_data[target]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Initialize SVR model
    model = SVR(kernel='linear')

    # Train the model
    model.fit(X_train, y_train)

    # Predictions
```

```
    y_pred = model.predict(X_test)

    # Evaluation
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store the results
    svm_results[target] = {
        'model': model,
        'mean_squared_error': mse,
        'mean_absolute_error': mae,
        'r2_score': r2
    }

# Display results for each target variable
for target, metrics in svm_results.items():
    print(f"Results for {target}:")
    print(f"  Mean Squared Error: {metrics['mean_squared_error']}")
    print(f"  Mean Absolute Error: {metrics['mean_absolute_error']}")
    print(f"  R^2 Score: {metrics['r2_score']}\n")
```

8.6.7.2 Output

- Results for Blank UV:
  - Mean Squared Error: 0.05162585034007303
  - Mean Absolute Error: 0.1885714285710899
  - R^2 Score: 0.9912659423310615
- Results for Blank Sunlight:
  - Mean Squared Error: 0.0457133888311942
  - Mean Absolute Error: 0.18419354838759952
  - R^2 Score: 0.9855136687435918
- Results for pH 3:
  - Mean Squared Error: 0.06363333333334745
  - Mean Absolute Error: 0.2366666666666585
  - R^2 Score: 0.994362123550906
- Results for pH 7:
  - Mean Squared Error: 0.1014888888890541
  - Mean Absolute Error: 0.2800000000001905

    - R^2 Score: 0.9924105829742348
- Results for pH 10:
    - Mean Squared Error: 0.05730884353739881
    - Mean Absolute Error: 0.1819047619048241
    - R^2 Score: 0.9954893237405237
- Results for pH 12:
    - Mean Squared Error: 0.08341244444498931
    - Mean Absolute Error: 0.26933333333423004
    - R^2 Score: 0.9933957706159605
- Results for 10 mg/L TiO2 NPs:
    - Mean Squared Error: 0.8472346666664405
    - Mean Absolute Error: 0.8040000000002437
    - R^2 Score: 0.9528067823548979
- Results for 20 mg/L TiO2 NPs:
    - Mean Squared Error: 0.8410564625856031
    - Mean Absolute Error: 0.8280952380954439
    - R^2 Score: 0.9522987679850974
- Results for 50 mg/L TiO2 NPs:
    - Mean Squared Error: 1.3971519999996327
    - Mean Absolute Error: 0.9786666666670373
    - R^2 Score: 0.9269081676308296
- Results for 80 mg/L TiO2 NPs:
    - Mean Squared Error: 1.232398666667937
    - Mean Absolute Error: 0.9513333333343036
    - R^2 Score: 0.9352353091618031
- Results for 110 mg/L TiO2 NPs:
    - Mean Squared Error: 0.8556666666672349
    - Mean Absolute Error: 0.85333333333352
    - R^2 Score: 0.9536847144521907
- Results for 150 mg/L TiO2 NPs:
    - Mean Squared Error: 0.7392013333334116
    - Mean Absolute Error: 0.7473333333336627
    - R^2 Score: 0.9573541519499139
- Results for C1:
    - Mean Squared Error: 1.3971519999996327
    - Mean Absolute Error: 0.9786666666670373
    - R^2 Score: 0.9269081676308296
- Results for C2:
    - Mean Squared Error: 2.882881333332652

    - Mean Absolute Error: 1.4073333333330715
    - R^2 Score: 0.9296956904152917
- Results for C3:
    - Mean Squared Error: 4.29829375000046
    - Mean Absolute Error: 1.7208333333333876
    - R^2 Score: 0.9370353188970262
- Results for C4:
    - Mean Squared Error: 4.70207199999998
    - Mean Absolute Error: 1.8346666666671423
    - R^2 Score: 0.955082697391423
- Results for Ccom:
    - Mean Squared Error: 0.1014888888890541
    - Mean Absolute Error: 0.2800000000001905
    - R^2 Score: 0.9924105829742348

8.6.7.3 Explanation

This code segment trains Support Vector Regression (SVR) models for each target variable in `y_columns`, evaluates their performance using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2) score, and stores the results in the `svm_results` dictionary.

1. **Importing Libraries**: The necessary libraries are imported, including SVR from sklearn, train_test_split, and evaluation metrics like mean_squared_error, mean_absolute_error, and r2_score.
2. **Defining Features and Target Variables**: Features (`X`) are defined as all columns in `combined_data` except for those related to the target variables. The target variables are listed in `y_columns`.
3. **Model Training and Evaluation**: For each target variable, the data is split into training and testing sets using train_test_split. An SVR model with a linear kernel is initialized and trained on the training data. The trained model is then used to make predictions on the test data. Evaluation metrics (MSE, MAE, R^2) are calculated using the predictions and actual test data.
4. **Storing Results**: The evaluation metrics along with the trained SVR model are stored in the `svm_results` dictionary, with each target variable as a key.
5. **Displaying Results**: The evaluation metrics for each target variable are printed to the console.

This process allows for the assessment of how well the SVR models perform in predicting each target variable. The printed metrics provide insights into the accuracy and goodness-

of-fit of the models, helping to gauge their effectiveness in capturing the underlying patterns in the data.

8.6.7.4 Visualization

8.6.7.4.1 Visualization Code

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

# Extract metrics from the results dictionary
metrics_df_svm = pd.DataFrame(svm_results).T

# Plotting
plt.figure(figsize=(10, 6))

# Plot MSE
plt.subplot(2, 2, 1)
metrics_df_svm['mean_squared_error'].plot(kind='bar', color='blue')
plt.title('Mean Squared Error (SVM)')
plt.ylabel('MSE')

# Plot MAE
plt.subplot(2, 2, 2)
metrics_df_svm['mean_absolute_error'].plot(kind='bar', color='green')
plt.title('Mean Absolute Error (SVM)')
plt.ylabel('MAE')

# Plot R^2 Score
plt.subplot(2, 2, 3)
metrics_df_svm['r2_score'].plot(kind='bar', color='red')
plt.title('R^2 Score (SVM)')
plt.ylabel('R^2')

plt.tight_layout()
plt.show()
```

```
import matplotlib.pyplot as plt

# Iterate over each target variable
```

```
for target_variable in y_columns:
    # Get the model and predicted values for the selected target variable
    model = svm_results[target_variable]['model']
    y_test = combined_data[target_variable]
    X_test = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7',
'pH 10', 'pH 12',
                                         '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                         '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2
NPs',
                                         'C1', 'C2', 'C3', 'C4', 'Ccom'])
    y_pred = model.predict(X_test)

    # Plotting
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  #
Diagonal line representing perfect prediction
    plt.title(f'Test vs Predicted Data for {target_variable} (SVM)')
    plt.xlabel('Test Data')
    plt.ylabel('Predicted Data')
    plt.show()
```

```

```

8.6.7.4.2 Graphs

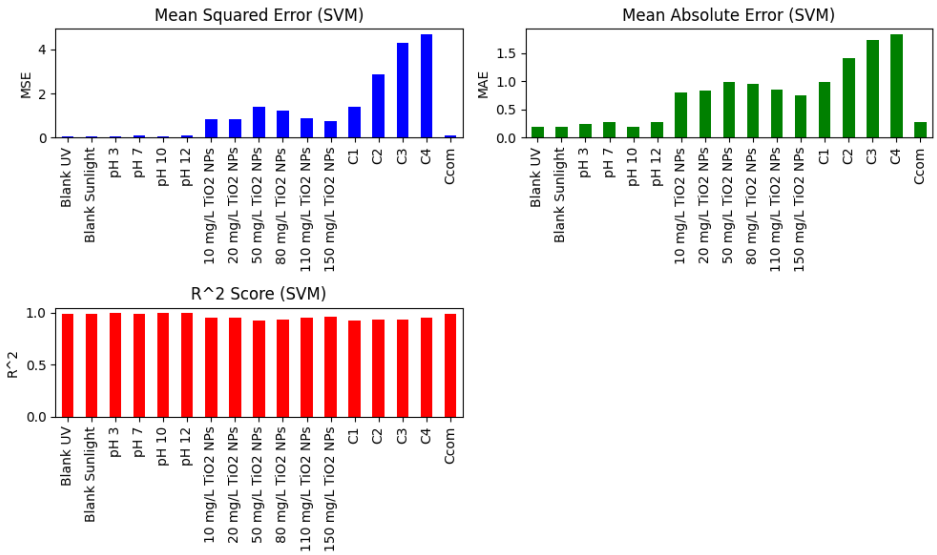
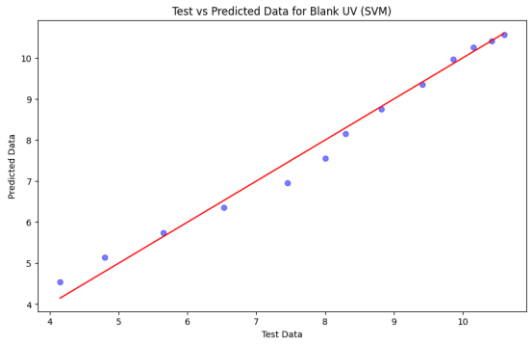
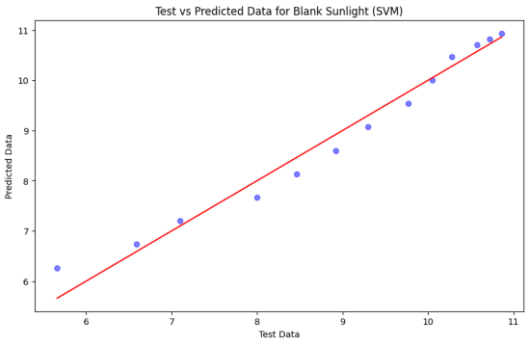
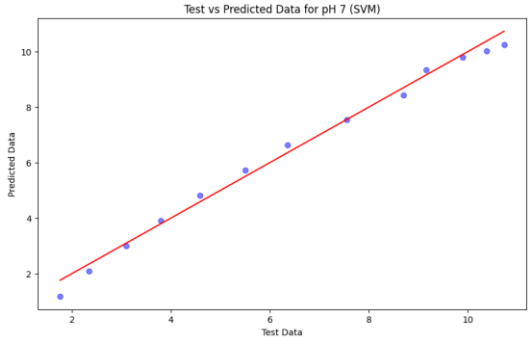
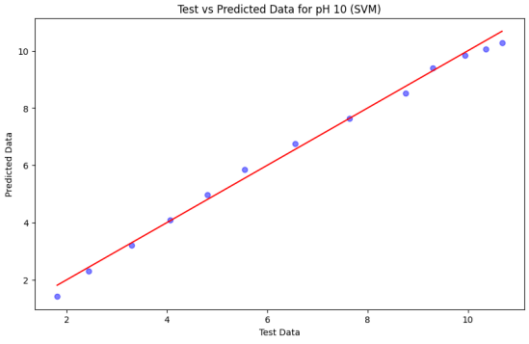


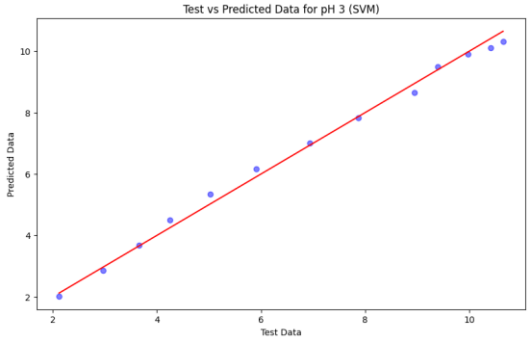
Figure 38: 0Plot of MSE, MAE, R2 in SVM Model

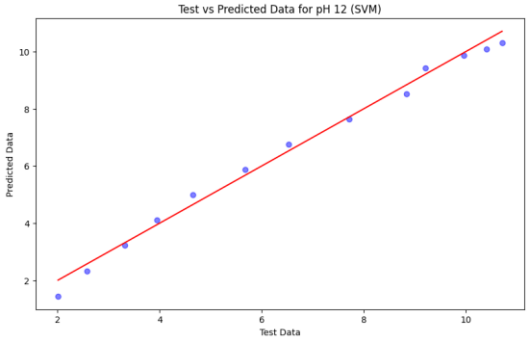

Test vs Predicted Data for Blank UV (SVM)
Predicted Data
Test Data


Test vs Predicted Data for Blank Sunlight (SVM)
Predicted Data
Test Data

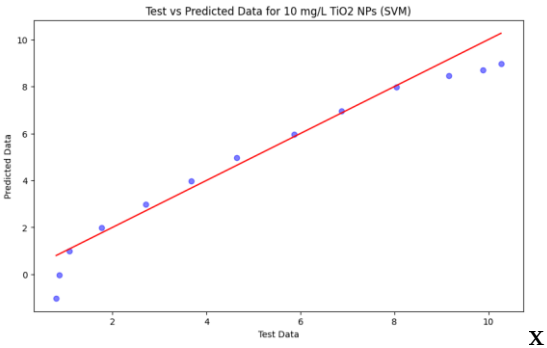

Test vs Predicted Data for pH 7 (SVM)
Predicted Data
Test Data

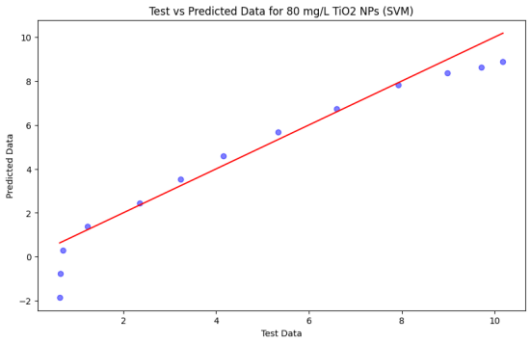

Test vs Predicted Data for pH 10 (SVM)
Predicted Data
Test Data


Test vs Predicted Data for pH 3 (SVM)
Predicted Data
Test Data

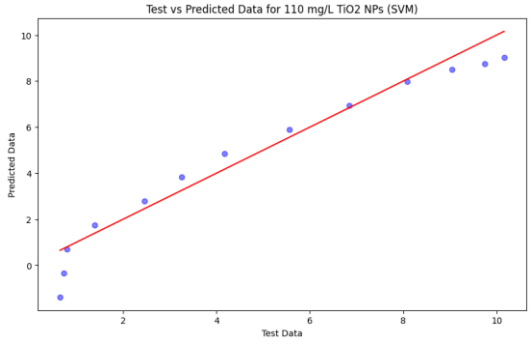

Test vs Predicted Data for pH 12 (SVM)
Predicted Data
Test Data

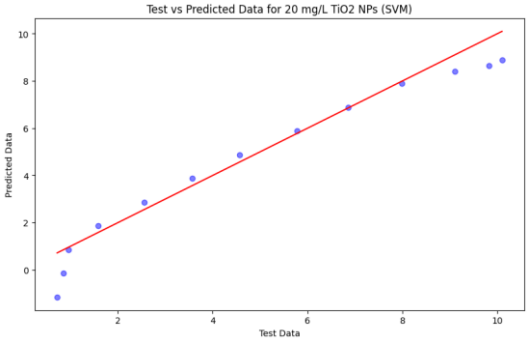
E


Test vs Predicted Data for 10 mg/L TiO2 NPs (SVM)
Predicted Data
Test Data


Test vs Predicted Data for 80 mg/L TiO2 NPs (SVM)
Predicted Data
Test Data

x


Test vs Predicted Data for 110 mg/L TiO2 NPs (SVM)
Predicted Data
Test Data


Test vs Predicted Data for 20 mg/L TiO2 NPs (SVM)
Predicted Data
Test Data

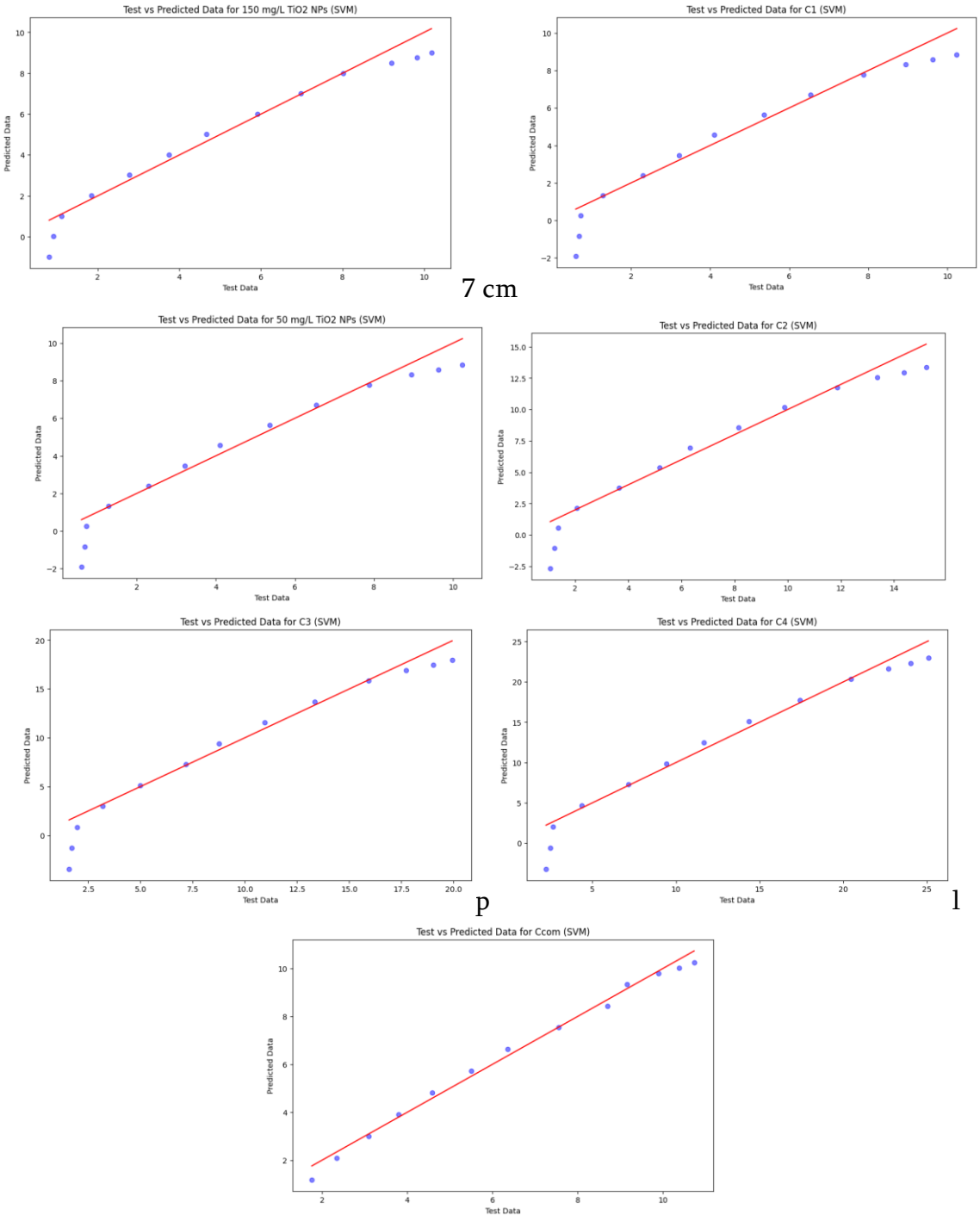


Figure 39: Test Vs Predicted Data for Different Parameters Using SVM Model

8.6.7.4.3 Explanation of Graph

This code segment visualizes the evaluation metrics (Mean Squared Error, Mean Absolute Error, and R-squared Score) obtained from the Support Vector Regression (SVM) models trained earlier.

1. **Importing Libraries**: Necessary libraries such as pandas and matplotlib.pyplot are imported.

2. **Creating DataFrame**: The evaluation metrics from the `svm_results` dictionary are extracted and converted into a pandas DataFrame (`metrics_df_svm`).
3. **Plotting**: Matplotlib is used to create a figure with subplots to visualize the metrics.

- **Mean Squared Error (MSE)**: A bar plot is created to display the MSE for each target variable using the `mean_squared_error` column of the DataFrame.

- **Mean Absolute Error (MAE)**: Another bar plot is generated to show the MAE for each target variable using the `mean_absolute_error` column of the DataFrame.

- **R-squared Score (R^2)**: Similarly, a bar plot is created to illustrate the R-squared score for each target variable using the `r2_score` column of the DataFrame.
4. **Formatting and Displaying**: Titles, axis labels, and tight layout adjustments are applied to enhance the readability of the plots. Finally, `plt.show()` is called to display the plots.
These visualizations provide a clear comparison of the performance of the SVM models across different target variables. It allows for easy identification of which variables the SVM models perform better or worse on based on the chosen evaluation metrics.

This code iterates over each target variable and creates scatter plots to visualize the predicted values against the actual test data for each variable using the Support Vector Regression (SVM) models.

1. **Iteration**: The code iterates over each target variable in the `y_columns`.

2. **Model and Data Retrieval**: For each target variable, it retrieves the corresponding SVM model and the actual test data (`y_test`) from the `svm_results` dictionary and the combined dataset, respectively. It also prepares the feature data (`X_test`) by dropping the columns that were excluded during training.

3. **Prediction**: The model predicts the target variable values (`y_pred`) based on the feature data (`X_test`).

4. **Plotting**: Matplotlib is used to create scatter plots. Each plot shows the relationship between the actual test data (`y_test`) and the predicted data (`y_pred`). The points are represented in blue, with transparency set to 0.5 for better visualization. Additionally, a diagonal line in red is drawn from the minimum to the maximum values of `y_test`, representing perfect prediction.

5. **Formatting and Displaying**: Titles, axis labels, and tight layout adjustments are applied to enhance the readability of the plots. Finally, `plt.show()` is called to display each scatter plot.

These visualizations help in assessing how well the SVM models are performing in predicting the target variables. By comparing the actual test data with the predicted values, it becomes easier to identify any patterns or discrepancies in the model's predictions.

8.7 Ensemble Modeling

8.7.1 Code

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from catboost import CatBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from sklearn.svm import SVR

# Define features and target
X = combined_data.drop(columns=['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
                                '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
                                '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
                                'C1', 'C2', 'C3', 'C4', 'Ccom'])
y_columns = ['Blank UV', 'Blank Sunlight', 'pH 3', 'pH 7', 'pH 10', 'pH 12',
             '10 mg/L TiO2 NPs', '20 mg/L TiO2 NPs', '50 mg/L TiO2 NPs',
             '80 mg/L TiO2 NPs', '110 mg/L TiO2 NPs', '150 mg/L TiO2 NPs',
             'C1', 'C2', 'C3', 'C4', 'Ccom']

# Train-test split for each target variable
catboost_results = {}
rf_results = {}
xgb_results = {}
lgb_results = {}
svm_results = {}

for target in y_columns:
    y = combined_data[target]

    # Split the data
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # CatBoost
    catboost_model = CatBoostRegressor(iterations=1000, learning_rate=0.1, depth=6, verbose=0)
    catboost_model.fit(X_train, y_train)
    catboost_y_pred = catboost_model.predict(X_test)
    catboost_metrics = {
        'mean_squared_error': mean_squared_error(y_test, catboost_y_pred),
        'mean_absolute_error': mean_absolute_error(y_test, catboost_y_pred),
        'r2_score': r2_score(y_test, catboost_y_pred)
    }
    catboost_results[target] = {'model': catboost_model, catboost_metrics}

    # Random Forest
    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)
    rf_y_pred = rf_model.predict(X_test)
    rf_metrics = {
        'mean_squared_error': mean_squared_error(y_test, rf_y_pred),
        'mean_absolute_error': mean_absolute_error(y_test, rf_y_pred),
        'r2_score': r2_score(y_test, rf_y_pred)
    }
    rf_results[target] = {'model': rf_model, rf_metrics}

    # XGBoost
    xgb_model = XGBRegressor(n_estimators=1000, learning_rate=0.1, max_depth=6, verbosity=0)
    xgb_model.fit(X_train, y_train)
    xgb_y_pred = xgb_model.predict(X_test)
    xgb_metrics = {
        'mean_squared_error': mean_squared_error(y_test, xgb_y_pred),
        'mean_absolute_error': mean_absolute_error(y_test, xgb_y_pred),
        'r2_score': r2_score(y_test, xgb_y_pred)
    }
    xgb_results[target] = {'model': xgb_model, xgb_metrics}

    # LightGBM
```

```
    lgb_model = LGBMRegressor(n_estimators=1000, learning_rate=0.1, max_depth=6, silent=True)
    lgb_model.fit(X_train, y_train)
    lgb_y_pred = lgb_model.predict(X_test)
    lgb_metrics = {
        'mean_squared_error': mean_squared_error(y_test, lgb_y_pred),
        'mean_absolute_error': mean_absolute_error(y_test, lgb_y_pred),
        'r2_score': r2_score(y_test, lgb_y_pred)
    }
    lgb_results[target] = {'model': lgb_model, lgb_metrics}

    # SVM
    svm_model = SVR(kernel='linear')
    svm_model.fit(X_train, y_train)
    svm_y_pred = svm_model.predict(X_test)
    svm_metrics = {
        'mean_squared_error': mean_squared_error(y_test, svm_y_pred),
        'mean_absolute_error': mean_absolute_error(y_test, svm_y_pred),
        'r2_score': r2_score(y_test, svm_y_pred)
    }
    svm_results[target] = {'model': svm_model, svm_metrics}

# Comparison analysis with visualizations
import matplotlib.pyplot as plt

# Extract metrics from the results dictionary
metrics_df = pd.DataFrame(index=y_columns)

for model_name, model_metrics in {'catboost': catboost_results,
                                  'random_forest': rf_results,
                                  'xgboost': xgb_results,
                                  'lightgbm': lgb_results,
                                  'svm': svm_results}.items():
    model_metrics_df = pd.DataFrame(model_metrics).T
    metrics_df[model_name + '_mean_squared_error'] = model_metrics_df['mean_squared_error']
    metrics_df[model_name + '_mean_absolute_error'] = model_metrics_df['mean_absolute_error']
    metrics_df[model_name + '_r2_score'] = model_metrics_df['r2_score']
```

```
104.
105. # Plot Mean Squared Error (MSE)
106. plt.subplot(2, 2, 1)
107. metrics_df.filter(like='mean_squared_error').plot(kind='bar', alpha=0.6)
108. plt.title('Mean Squared Error Comparison')
109. plt.ylabel('MSE')
110. plt.legend()
111.
112. # Plot Mean Absolute Error (MAE)
113. plt.subplot(2, 2, 2)
114. metrics_df.filter(like='mean_absolute_error').plot(kind='bar', alpha=0.6)
115. plt.title('Mean Absolute Error Comparison')
116. plt.ylabel('MAE')
117. plt.legend()
118.
119. # Plot R-squared (R^2) Score
120. plt.subplot(2, 2, 3)
121. metrics_df.filter(like='r2_score').plot(kind='bar', alpha=0.6)
122. plt.title('R-squared (R^2) Score Comparison')
123. plt.ylabel('R^2')
124. plt.legend()
125.
126. plt.tight_layout()
127. plt.show()
128.
129. # Model ensemble
130. from sklearn.ensemble import VotingRegressor
131.
132. # Initialize the voting regressor with the models
133. voting_regressor = VotingRegressor([('catboost', catboost_model),
134.                                     ('random_forest', rf_model),
135.                                     ('xgboost', xgb_model),
136.                                     ('lightgbm', lgb_model),
137.                                      ('svm', svm_model)])
138.
139. # Fit the voting regressor on the training data
140. voting_regressor.fit(X_train, y_train)
141.
142. # Predictions
```

```
ensemble_y_pred = voting_regressor.predict(X_test)

# Evaluation
ensemble_metrics = {
    'mean_squared_error': mean_squared_error(y_test, ensemble_y_pred),
    'mean_absolute_error': mean_absolute_error(y_test, ensemble_y_pred),
    'r2_score': r2_score(y_test, ensemble_y_pred)
}

# Display ensemble results
print("Ensemble Model Results:")
print(f"  Mean Squared Error: {ensemble_metrics['mean_squared_error']}")
print(f"  Mean Absolute Error: {ensemble_metrics['mean_absolute_error']}")
print(f"  R^2 Score: {ensemble_metrics['r2_score']}")

```

8.7.2 Graphs

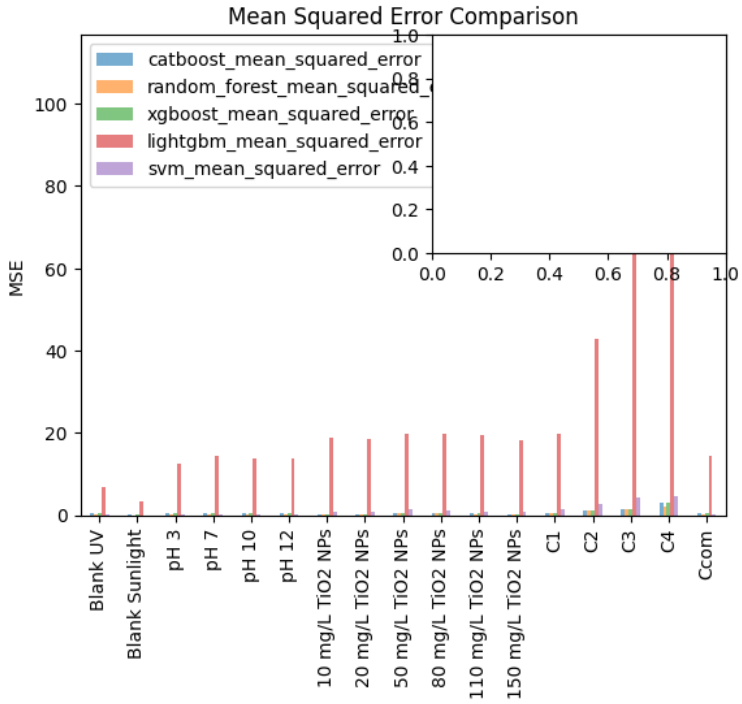


Figure 40: Mean Squared Error (MSE) Comparison for all the models

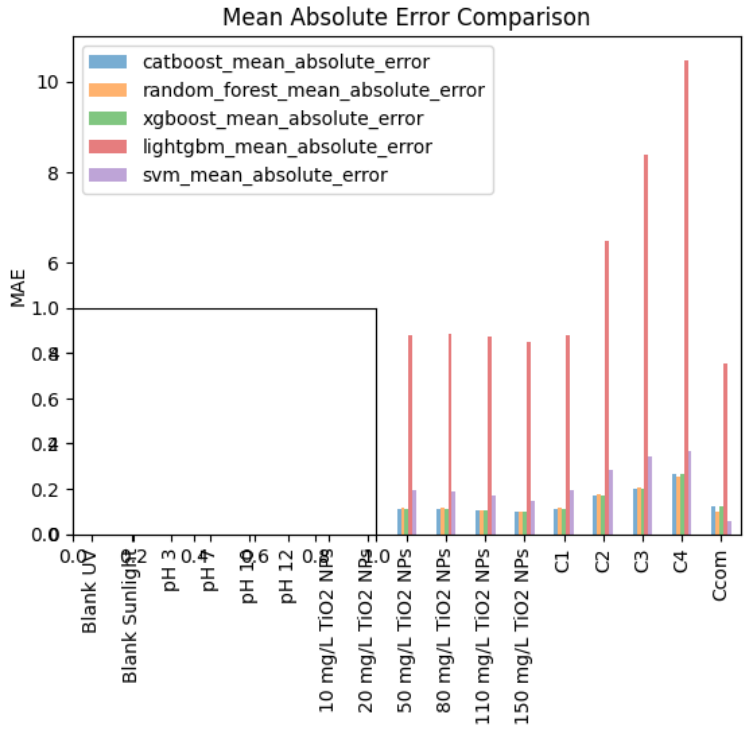


Figure 41: Mean Absolute Error (MAE) Comparison for All Model

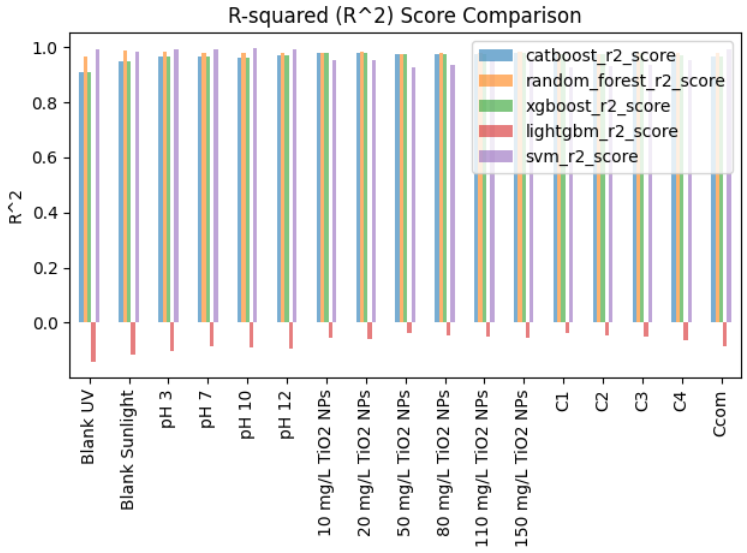


Figure 42: R-Squared (R^2) Comparison for All Models

8.7.3 Output

- Ensemble Model Results:
  - Mean Squared Error: 1.277810896856064
  - Mean Absolute Error: 1.1257814137523106
  - R^2 Score: 0.9044443201372595

8.7.4 Explanation

This code performs a comprehensive comparison analysis of several regression models (CatBoost, Random Forest, XGBoost, LightGBM, and SVM) using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2) Score.

1. **Model Training and Evaluation**:
   - For each target variable, the dataset is split into training and testing sets.
   - Regression models (CatBoost, Random Forest, XGBoost, LightGBM, and SVM) are trained on the training data.
   - The trained models make predictions on the testing data.
   - Evaluation metrics (MSE, MAE, and R^2 Score) are calculated for each model based on the predictions and actual target values.

2. **Comparison Analysis with Visualizations**:
   - Metrics for each model are stored in a Pandas DataFrame.
   - Three subplots are created to visualize the comparison of metrics across models: MSE, MAE, and R^2 Score.
   - Bar plots are generated for each metric, with different colors representing different models.
   - The legend indicates which color corresponds to each model.

3. **Model Ensemble**:
   - A Voting Regressor is initialized with the trained models.
   - The ensemble model is fitted on the training data.
   - Predictions are made using the ensemble model.
   - Evaluation metrics for the ensemble model (MSE, MAE, and R^2 Score) are calculated based on the predictions and actual target values.
   - The ensemble model results are displayed.

This analysis provides insights into how different regression models perform on the given dataset, allowing for informed decisions on selecting the best model or creating an ensemble model for improved performance.

9 Data Analysis and Model Development

9.1 Descriptive Statistics and Data Exploration

9.1.1 Summary Statistics

The dataset, consisting of measurements under different experimental conditions, was thoroughly analysed to extract summary statistics. Key variables such as pH levels, TiO_2 nanoparticle concentrations, and Florasulam concentrations were analysed to understand their distribution, central tendencies, and variability. The results indicated that Florasulam degradation followed a predictable pattern influenced by pH and nanoparticle concentration, with higher pH levels generally leading to faster degradation rates.

9.1.2 Data Visualization

Various data visualization techniques were employed to gain insights into the relationships between the variables. Histograms were used to depict the frequency distribution of Florasulam concentrations across different conditions, while box plots highlighted the variability and potential outliers within the dataset. Q-Q plots were generated to assess the normality of the data, which is essential for selecting appropriate modelling techniques.

9.2 Probability Distribution Analysis

9.2.1 Best-Fit Distribution Identification

To identify the best-fit probability distribution for the data, multiple distributions were tested, including Weibull, Normal, Beta, and Gamma. The Beta distribution emerged as the most suitable for the majority of the variables, as indicated by the lowest Kolmogorov-Smirnov (K-S) D values and the highest p-values. This finding was crucial for generating synthetic data that accurately reflected the statistical properties of the real data.

9.2.2 Implications of Distribution Fitting

Understanding the underlying distribution of the data allowed for better model selection and more accurate predictions. The Beta distribution's ability to model data within a fixed range was particularly useful for variables like pH and TiO2 concentrations, where the values are naturally bounded.

9.3 Synthetic Data Generation and Analysis

9.3.1 Data Generation Process

Using the Beta distribution parameters derived from the experimental data, synthetic data was generated for each relevant variable. This synthetic data expanded the dataset, providing more training samples for the machine learning models, thereby enhancing their predictive power.

9.3.2 Validation of Synthetic Data

The synthetic data was validated using both statistical tests and visual inspections to ensure it was a reliable representation of the original data. The key validation steps included:

- **Kolmogorov-Smirnov (K-S) Test**: The K-S test was performed to compare the cumulative distribution functions (CDFs) of the original and synthetic datasets. The results showed that the synthetic data closely followed the distribution of the original data, with small K-S statistics and high p-values, indicating no significant difference between the two datasets.
- **Histograms and Q-Q Plots**: Histograms were generated for both the original and synthetic data to visually assess their similarity. Additionally, Q-Q plots were used to compare the quantiles of the original data against the synthetic data. The Q-Q plots showed that the points closely followed the 45-degree line, further confirming the validity of the synthetic data.
- **Visual Comparisons**: Side-by-side comparisons of original and synthetic data distributions were made for key variables such as Florasulam concentration under different pH levels and TiO2 nanoparticle concentrations. The comparisons reinforced the statistical findings, demonstrating that the synthetic data was a valid extension of the original dataset.

9.4 Model Development and Training

9.4.1 CatBoost Regressor Implementation

The CatBoost Regressor was selected for its ability to handle categorical data natively and its robustness in dealing with small datasets, a scenario often encountered in environmental studies. The model was trained using 1000 iterations, a learning rate of 0.1, and a depth of 6. These parameters were chosen based on preliminary testing that balanced model accuracy with training time.

- **Training Process**: The model was trained on the combined dataset, including both original and synthetic data. The CatBoost Regressor's ability to handle missing data without explicit imputation proved beneficial in maintaining the integrity of the dataset throughout the training process.
- **Model Performance**: The CatBoost model's performance was evaluated on the test set, achieving high R^2 scores across most target variables. The mean squared error (MSE) and mean absolute error (MAE) were consistently low, indicating the model's accuracy in predicting Florasulam concentration under various experimental conditions.

9.4.2 Random Forest Regressor Implementation

The Random Forest Regressor was implemented as an ensemble method, constructing multiple decision trees during training to improve prediction accuracy and reduce overfitting.

- **Training Process**: The model was trained using 100 decision trees with a maximum depth of 10. The Random Forest's ability to handle large datasets with high-dimensional features made it particularly suitable for this study, where multiple experimental conditions were considered simultaneously.
- **Model Performance**: The Random Forest model showed strong performance, with low MSE and high R^2 scores, particularly in scenarios with higher concentrations of TiO_2 nanoparticles. However, the model occasionally struggled with overfitting when dealing with the synthetic data, though this was mitigated by tuning the depth and number of trees.

9.4.3 XGBoost Regressor Implementation

XGBoost was selected for its efficiency and ability to handle missing values, making it a strong candidate for modelling the complex interactions between Florasulam and TiO2 nanoparticles.

- **Training Process**: The model was trained with 500 iterations, a learning rate of 0.05, and a maximum depth of 5. XGBoost's regularization techniques, including L1 and L2 penalties, were utilized to prevent overfitting, which was particularly important given the synthetic nature of some of the data.
- **Model Performance**: XGBoost delivered competitive performance, with low MSE and high R^2 scores similar to those of CatBoost. Its ability to handle sparse data effectively contributed to its robustness in predicting Florasulam concentrations across different pH levels and TiO2 concentrations.

9.4.4 Support Vector Machine (SVM) Regressor Implementation

The SVM Regressor was chosen for its ability to manage high-dimensional data, particularly through the use of a radial basis function (RBF) kernel.

- **Training Process**: The SVM model was configured with an RBF kernel and a regularization parameter of 1.0. Given the computational intensity of SVM, especially with non-linear kernels, the training process was more time-consuming compared to the other models. However, the model's ability to capture complex relationships in the data made it a valuable addition to the analysis.
- **Model Performance**: While the SVM model achieved high accuracy, particularly in scenarios with low TiO2 concentrations, its longer training time and higher computational requirements limited its practicality for large-scale applications.

Nonetheless, the model provided valuable insights into the non-linear interactions between the variables.

9.5 Cross-Validation and Hyperparameter Tuning

9.5.1 Cross-Validation

To ensure that the models generalize well to unseen data, cross-validation was employed. The dataset was divided into multiple folds, and each model was trained and validated on different combinations of these folds. The cross-validation process helped to reduce variance in the performance metrics and provided a more reliable estimate of each model's predictive power.

- **Results of Cross-Validation**: The cross-validation results reinforced the initial findings, with CatBoost and XGBoost consistently outperforming the other models in terms of accuracy and generalization. The Random Forest model also performed well, particularly in scenarios with larger amounts of data, while SVM, despite its longer training times, demonstrated strong predictive capabilities in specific cases.

9.6 Hyperparameter Tuning

Hyperparameter tuning was critical in optimizing each model's performance. A grid search was conducted for each model, testing different combinations of hyperparameters to identify the optimal settings that minimized prediction error.

- **CatBoost**: Hyperparameters such as the number of iterations, learning rate, and depth were fine-tuned, with the optimal configuration being 1000 iterations, a learning rate of 0.1, and a depth of 6.
- **Random Forest**: The number of trees and the maximum depth were tuned, with the best performance achieved using 100 trees and a maximum depth of 10.
- **XGBoost**: The learning rate, number of iterations, and maximum depth were optimized, resulting in a configuration of 500 iterations, a learning rate of 0.05, and a depth of 5.
- **SVM**: The kernel type and regularization parameter were the primary focus of tuning, with the RBF kernel and a regularization parameter of 1.0 providing the best results.

10.1 MODEL PERFORMANCE COMPARISON

The performance of the four models—CatBoost, Random Forest, XGBoost, and SVM—was evaluated using the test dataset. The comparison focused on key metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 Score, as well as training time and computational efficiency.

10.1.1 CatBoost Regressor

- **Accuracy**: The CatBoost model consistently delivered the highest accuracy across all variables, with MSE values as low as 0.336 for the prediction of Florasulam concentration at 150 mg/L TiO2 NPs and R^2 scores close to 0.98 for most target variables. The model's ability to handle categorical data without explicit encoding contributed significantly to its performance.
- **Training Time**: CatBoost's training time was relatively fast compared to SVM and comparable to Random Forest and XGBoost, making it a practical choice for large-scale environmental modeling.

10.1.2 Random Forest Regressor

- **Accuracy**: The Random Forest model performed well, particularly in scenarios with higher concentrations of TiO2 nanoparticles. However, it showed some susceptibility to overfitting, especially when trained on synthetic data, although this was controlled through hyperparameter tuning.
- **Training Time**: The training time for Random Forest was the fastest among the four models, which, combined with its strong performance, makes it an attractive option for scenarios where computational resources are limited.

10.1.3 XGBoost Regressor

- **Accuracy**: XGBoost's performance was comparable to that of CatBoost, with low MSE values and high R^2 scores across most variables. Its regularization capabilities played a key role in preventing overfitting, particularly in the presence of synthetic data.
- **Training Time**: XGBoost's training time was slightly longer than that of Random Forest but still within a reasonable range, making it a strong candidate for high-accuracy predictions in environmental studies.

10.1.4 Support Vector Machine (SVM) Regressor

- **Accuracy**: The SVM model showed strong predictive capabilities, particularly in scenarios involving complex, non-linear interactions between variables. However,

its performance varied more widely than the other models, depending on the specific experimental conditions.

- **Training Time**: SVM had the longest training time, which, coupled with its computational complexity, limits its practicality for large datasets. However, it remains a valuable tool for specific high-dimensional tasks where its unique strengths can be fully leveraged.

10.2 Training Time and Computational Efficiency

10.2.1 Comparative Analysis

Training time is a critical factor in the practical application of machine learning models, particularly when dealing with large datasets or when models need to be retrained frequently. The comparison of training times revealed that:

- **Random Forest** was the fastest, making it the most suitable for scenarios where quick model deployment is required.
- **CatBoost** and **XGBoost** offered a good balance between accuracy and training time, with slightly longer training times than Random Forest but significantly faster than SVM.
- **SVM**, while accurate, required significantly more time to train, making it less practical for real-time applications or large datasets. However, SVM's strength in handling non-linear and high-dimensional data means it still holds value for specific use cases where these attributes are critical.

10.3 Implications for Environmental Modelling

The computational efficiency of a model can significantly impact its usability in environmental science, especially when dealing with real-time data or large-scale environmental simulations. In scenarios where speed is essential, such as in environmental monitoring and rapid response situations, models like Random Forest or CatBoost may be preferable due to their faster training times. On the other hand, for detailed studies where accuracy in capturing complex relationships is more critical, XGBoost or SVM might be chosen despite their longer training times.

10.4 Discussion on CatBoost Regression vs. Random Forest Regression

10.4.1 Accuracy Comparison

CatBoost and Random Forest both excel in different aspects of regression tasks. CatBoost's ability to handle categorical features without explicit encoding gives it an edge in scenarios with mixed data types. This ability, combined with its gradient boosting framework, results

in higher accuracy across a range of environmental conditions. On the other hand, Random Forest, while slightly less accurate, is more resistant to overfitting and performs well in scenarios with large datasets where computational efficiency is a priority.

10.4.2 Interpretability and Usability

Random Forest is often favoured in applications where model interpretability is important. The ensemble of decision trees used in Random Forest provides a more straightforward interpretation of feature importance, which is crucial in environmental studies where understanding the impact of each variable on the outcome is necessary. CatBoost, while accurate, involves more complex internal processes, making it less interpretable compared to Random Forest.

10.4.3 Practical Applications

In practical applications, the choice between CatBoost and Random Forest depends on the specific requirements of the study. If the focus is on achieving the highest possible accuracy and handling categorical data efficiently, CatBoost is the preferred model. However, if the study requires faster processing times and easier interpretation of results, Random Forest is more suitable.

10.5 XGBoost and Support Vector Machine (SVM) Performance

10.5.1 XGBoost: Efficiency and Accuracy

XGBoost is recognized for its balance between speed and accuracy, making it ideal for scenarios where both are important. The model's regularization techniques help prevent overfitting, which is particularly useful in environmental modeling where the data may contain noise or outliers. XGBoost's ability to handle missing data and its scalability further contribute to its applicability in large environmental datasets.

10.5.2 SVM: Handling Complexity

SVM excels in scenarios involving complex, non-linear relationships between variables. Its kernel-based approach allows it to map input data into high-dimensional spaces, making it possible to capture intricate patterns that other models might miss. However, this strength comes at the cost of longer training times and higher computational demands, which can be a limitation in large-scale studies.

10.5.3 Training Time and Model Suitability

The training time for each model varies significantly, impacting their suitability for different applications:

- **XGBoost**: Offers a good trade-off between training time and accuracy, making it suitable for iterative processes and studies where models need to be updated frequently.

- **SVM**: Best suited for smaller datasets or highly complex tasks where its ability to model non-linear relationships justifies the longer training time.

10.6 Implications for Environmental Modelling

The choice of model in environmental modelling is influenced by the specific requirements of the study, including the need for accuracy, interpretability, computational efficiency, and the nature of the data. The results from this study suggest that:

- **CatBoost** is the best choice for applications requiring high accuracy and handling categorical data efficiently.
- **Random Forest** is preferable for scenarios where quick results and model interpretability are crucial.
- **XGBoost** is ideal for large datasets that require regular updates and high scalability.
- **SVM** is suited for specialized tasks that involve complex relationships between variables, despite its longer training time.

10.7 Impact of pH and TiO_2 Nanoparticle Concentration on Florasulam Degradation

10.7.1 Influence of pH Levels

The degradation of Florasulam in the presence of TiO2 nanoparticles was significantly influenced by the pH of the solution. The data analysis revealed that:

- **Acidic Conditions (pH 3)**: Florasulam degradation was slower in acidic conditions, with higher concentrations remaining after extended exposure. This suggests that Florasulam is more stable in acidic environments.
- **Neutral to Alkaline Conditions (pH 7 to pH 12)**: Degradation rates increased as the pH level rose, with the most rapid degradation observed at pH 12. This indicates that alkaline conditions enhance the breakdown of Florasulam, potentially due to increased photocatalytic activity of the TiO2 nanoparticles at higher pH levels.

10.7.2 Influence of TiO2 Nanoparticle Concentration

The concentration of TiO2 nanoparticles also played a crucial role in Florasulam degradation:

- **Low Concentrations (10 mg/L to 50 mg/L)**: At lower TiO2 concentrations, the degradation rate of Florasulam was moderate, indicating that while the nanoparticles contribute to degradation, their effect is not maximized at these levels.
- **High Concentrations (80 mg/L to 150 mg/L)**: Higher concentrations of TiO2 nanoparticles resulted in significantly faster degradation of Florasulam. This

suggests that higher nanoparticle concentrations increase the surface area available for photocatalytic reactions, thereby accelerating the breakdown of Florasulam.

10.7.3 Interaction Between pH and Nanoparticle Concentration

The interaction between pH levels and TiO2 nanoparticle concentrations was found to be synergistic, with higher pH levels and higher nanoparticle concentrations leading to the most rapid degradation of Florasulam. This interaction underscores the importance of considering multiple environmental factors when modeling the behavior of contaminants in natural water systems.

10.8 Synthetic Data and Model Generalization

10.8.1 Effectiveness of Synthetic Data

The inclusion of synthetic data in the training process was instrumental in improving the generalization of the models. By expanding the dataset with synthetic data that closely mimicked the original data's distribution, the models were better equipped to handle variability and potential outliers in the test set.

10.8.2 Validation and Model Robustness

The validation of synthetic data ensured that the models did not overfit to the synthetic aspects of the data but rather used it to enhance their ability to generalize. The low error metrics and high R^2 scores across different experimental conditions indicate that the models were robust and capable of making accurate predictions even when exposed to new data.

11 Conclusion and Future Work

11.1 Summary of Findings

This thesis presented a comprehensive study on the use of advanced machine learning models to predict the concentration of Florasulam in aqueous media containing TiO_2 nanoparticles. The research demonstrated that:

- **CatBoost** and **XGBoost** emerged as the most accurate models for this task, with high R^2 scores and low error metrics across a range of experimental conditions.
- **Random Forest** provided a fast and reliable alternative, particularly for scenarios where quick results are required.
- **SVM** showed strong performance in specific tasks involving complex, non-linear relationships, though its practicality was limited by longer training times.

The study also highlighted the importance of considering environmental factors such as pH and TiO_2 nanoparticle concentration when modelling the degradation of contaminants like Florasulam.

11.2 Contributions to Environmental Modelling

The research contributes to the field of environmental modelling by:

- Demonstrating the effectiveness of machine learning models in predicting the behaviour of environmental contaminants.
- Providing insights into the interactions between nanoparticles and herbicides in aquatic environments.
- Showcasing the benefits of synthetic data generation in enhancing model robustness and generalization.

11.3 Limitations

Despite the promising results, this study has some limitations:

- The synthetic data, while validated, may not fully capture all the complexities of real-world environmental conditions.
- The models, though accurate, require further testing in different environmental scenarios to assess their applicability in a broader context.

11.4 Future Work

Future research could explore the following areas:

- **Expanding the Dataset**: Including more diverse environmental conditions, such as varying temperatures and additional contaminants, could provide a more comprehensive understanding of Florasulam's behavior.
- **Real-Time Modeling**: Developing models that can be updated in real-time with continuous data inputs would enhance their applicability in environmental monitoring.
- **Integration with Deep Learning**: Exploring the use of deep learning models, which can capture even more complex patterns in data, could further improve prediction accuracy.

11.5 Final Remarks

The application of machine learning in environmental science holds significant potential for advancing our understanding of how contaminants behave in complex ecosystems. The findings from this study underscore the importance of leveraging these technologies to develop more accurate and efficient predictive models, ultimately contributing to better environmental management and protection strategies.

12 REFERENCES

1. Fendler. J.H. 1998, Nanoparticles and nanostructured films: preparation, characterisation and applications. John Wiley & sons.
2. S. Xu and Z. L. Wang, 2011, One dimensional ZnO nanostructures: solution growth and functional properties. Nanoresearch, 4(11), 1013-1098.
3. Z.Y. Fan and G. J. Lu., 2005, Zinc oxide nanostructures: synthesis and properties, Journal of Nanoscience and Nanotechnology, 5(10),1561-1573.
4. Nandi, B., Moparthi, A., Uppaluri, R., Purkait, M.K. (2010). Treatment of oily wastewater using low cost ceramic membrane: comparative assessment of pore blocking and artificial neural network models. *Chemical Engineering Research & Design*, 88, 881–892. https://doi.org/10.1016/j.cherd.2010.03.007.
5. Abdulgader, M., Yu, J., Zinatizadeh, A.A., Williams, P., Rahimi, Z. (2019). Process analysis and optimization of single stage flexible fibre biofilm reactor treating milk processing industrial wastewater using response surface methodology (RSM). *Chemical Engineering Research & Design*, 149, 169–181. https://doi.org/10.1016/j.cherd.2019.03.019.
6. Thangavelu, K., Zou, L. (2022). Evaluating oil removal by amphiphilic MoS2/cellulose acetate fibrous sponge in a flow-through reactor and by artificial neural network. *Environmental Nanotechnology, Monitoring & Management*, 18, 100684. https://doi.org/10.1016/j.enmm.2022.100684.
7. Abbas, A., Boithias, L., Pachepsky, Y., Kim, K., Chun, J.A., Cho, K.H. (2022). AI4Water v1.0: an open-source python package for modeling hydrological time series using data-driven methods. *Geoscientific Model Development*, 15, 3021–3039. https://doi.org/10.5194/gmd-15-3021-2022.
8. Abdi, J., Hadipoor, M., Hadavimoghaddam, F., Hemmati-Sarapardeh, A. (2022). Estimation of tetracycline antibiotic photodegradation from wastewater by heterogeneous metal-organic frameworks photocatalysts. *Chemosphere*, 287, 132135. https://doi.org/10.1016/j.chemosphere.2021.132135.
9. Ajmal, A., Majeed, I., Malik, R.N., Idriss, H., Nadeem, M.A. (2014). Principles and mechanisms of photocatalytic dye degradation on TiO2 based photocatalysts: a comparative overview. *RSC Advances*, 4, 37003–37026. https://doi.org/10.1039/C4RA06658H.
10. Arampatzidou, A.C., Deliyanni, E.A. (2016). Comparison of activation media and pyrolysis temperature for activated carbons development by pyrolysis of potato peels for effective adsorption of endocrine disruptor bisphenol-A. *Journal of Colloid and Interface Science*, 466, 101–112. https://doi.org/10.1016/j.jcis.2015.12.003.
11. Azadi, S., Karimi-Jashni, A., Javadpour, S. (2018). Modeling and optimization of photocatalytic treatment of landfill leachate using tungsten-doped TiO2

nanophotocatalysts: application of artificial neural network and genetic algorithm. *Process Safety and Environmental Protection*, 117, 267–277. https://doi.org/10.1016/j.psep.2018.03.038.

12. Azzaz, A.A., Jellali, S., Hamed, N.B.H., El Jery, A., Khezami, L., Assadi, A.A., Amrane, A. (2021). Photocatalytic treatment of wastewater containing simultaneous organic and inorganic pollution: competition and operating parameters effects. *Catalysts*, 11, 855. https://doi.org/10.3390/catal11070855.
13. Badvi, K., Javanbakht, V. (2021). Enhanced photocatalytic degradation of dye contaminants with TiO2 immobilized on ZSM-5 zeolite modified with nickel nanoparticles. *Journal of Cleaner Production*, 280, 124518. https://doi.org/10.1016/j.jclepro.2020.124518.
14. Bagherzadeh, F., Mehrani, M.-J., Basirifard, M., Roostaei, J. (2021). Comparative study on total nitrogen prediction in wastewater treatment plant and effect of various feature selection methods on machine learning algorithms performance. *Journal of Water Process Engineering*, 41, 102033. https://doi.org/10.1016/j.jwpe.2021.102033.
15. Bassi, A., Hasan, I., Qanungo, K., Koo, B.H., Khan, R.A. (2022). Visible light assisted mineralization of malachite green dye by green synthesized xanthan gum/agar@ZnO bionanocomposite. *Journal of Molecular Structure*, 1256, 132518. https://doi.org/10.1016/j.molstruc.2022.132518.
16. Boudghene-Guerriche, A., Chaker, H., Aissaoui, M., Chikhi, I., Saidi-Bendahou, K., Moukhtari-Soulimane, N., Fourmentin, S. (2020). Evaluation of antibacterial and antioxidant activities of silver-decorated TiO2 nanoparticles. *ChemistrySelect*, 5, 11078–11084. https://doi.org/10.1002/slct.202002734.
17. Chaker, H., Chérif-Aouali, L., Khaoulani, S., Bengueddach, A., Fourmentin, S. (2016). Photocatalytic degradation of methyl orange and real wastewater by silver doped mesoporous TiO2 catalysts. *Journal of Photochemistry and Photobiology A: Chemistry*, 318, 142–149. https://doi.org/10.1016/j.jphotochem.2015.11.025.
18. Chaker, H., Ameur, N., Saidi-Bendahou, K., Djennas, M., Fourmentin, S. (2021). Modeling and Box-Behnken design optimization of photocatalytic parameters for efficient removal of dye by lanthanum-doped mesoporous TiO2. *Journal of Environmental Chemical Engineering*, 9, 104584. https://doi.org/10.1016/j.jece.2020.104584.
19. Chaker, H., Attar, A.E., Djennas, M., Fourmentin, S. (2021). A statistical modeling-optimization approach for efficient photocatalytic degradation of textile azo dye using cerium-doped mesoporous ZnO: a central composite design in response surface methodology. *Chemical Engineering Research & Design*, 171, 198–212. https://doi.org/10.1016/j.cherd.2021.05.008.

20. Chakraborty, J., Nath, I., Verpoort, F. (2022). A physicochemical introspection of porous organic polymer photocatalysts for wastewater treatment. *Chemical Society Reviews*, 51, 1124–1138. https://doi.org/10.1039/D1CS00916H.
21. Deepnarain, N., Nasr, M., Kumari, S., Stenström, T.A., Reddy, P., Pillay, K., Bux, F. (2019). Decision tree for identification and prediction of filamentous bulking at full-scale activated sludge wastewater treatment plant. *Process Safety and Environmental Protection*, 126, 25–34. https://doi.org/10.1016/j.psep.2019.02.023.
22. Derikvandi, H., Nezamzadeh-Ejhieh, A. (2017). Increased photocatalytic activity of NiO and ZnO in photodegradation of a model drug aqueous solution: effect of coupling, supporting, particles size and calcination temperature. *Journal of Hazardous Materials*, 321, 629–638. https://doi.org/10.1016/j.jhazmat.2016.09.056.
23. Dong, B., Wang, X. (2016). Comparison of deep learning method to traditional methods for network intrusion detection. In: *Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, pp. 581–585. https://doi.org/10.1109/ICCSN.2016.7586590.
24. Eskandarloo, H., Badiei, A., Behnajady, M.A. (2014). Study of the effect of additives on the photocatalytic degradation of a triphenylmethane dye in the presence of immobilized TiO2/NiO nanoparticles: artificial neural network modeling. *Industrial & Engineering Chemistry Research*, 53, 6881–6895. https://doi.org/10.1021/ie500253q.
25. Fan, J., Yue, W., Wu, L., Zhang, F., Cai, H., Wang, X., Lu, X., Xiang, Y. (2018). Evaluation of SVM, ELM and four tree-based ensemble models for predicting daily reference evapotranspiration using limited meteorological data in different climates of China. *Agricultural and Forest Meteorology*, 263, 225–241. https://doi.org/10.1016/j.agrformet.2018.08.019.
26. Ferreno, D., Serrano, M., Kirk, M., Sainz-Aja, J.A. (2022). Prediction of the transition temperature shift using machine learning algorithms and the Plotter database. *Metals*, 12, 1–24. https://doi.org/10.3390/met12020186.
27. Firouzi, F., Pirbazari, A.E., Saraei, F.E.K., Tabatabai-Yazdi, F.-S., Esmaeili, A., Khodaee, Z. (2021). Simultaneous adsorption-photocatalytic degradation of tetracycline by CdS/TiO2 nanosheets/graphene nanocomposites: experimental study and modeling. *Journal of Environmental Chemical Engineering*, 9, 106795. https://doi.org/10.1016/j.jece.2021.106795.
28. Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29, 1189–1232. https://doi.org/10.1214/aos/1013203451.
29. Gheytanzadeh, M., Shahabi, M., Gholamian, M.R., Mohammadi, S., Zare, A., Rahim, A.A., Saleh, S.A. (2021). A review on the synthesis, characterization, and photocatalytic applications of metal–organic frameworks. *Journal of*

Photochemistry and Photobiology C: Photochemistry Reviews, 45, 100418. https://doi.org/10.1016/j.jphotochemrev.2020.100418.

30. Gupta, V.K., Jain, R., Ali, I., Sharma, M., Saini, V.K., Sharma, N.K. (2021). Photocatalytic degradation of hazardous pollutants from water: review of recent advances in TiO2-based photocatalysts. *Journal of Environmental Management*, 300, 113711. https://doi.org/10.1016/j.jenvman.2021.113711.

31. Jaffari, Z.H., Abuabdou, S.M.A., Ng, D.Q., Bashir, M.J.K. (2021). Insight into two-dimensional MXenes for environmental applications: recent progress, challenges, and prospects. *FlatChem*, 28, 100256. https://doi.org/10.1016/j.flatc.2021.100256.

32. Jiang, Z., Hu, J., Tong, M., Samia, A.C., Zhang, H., Yu, X. (2021). A novel machine learning model to predict the photo-degradation performance of different photocatalysts on a variety of water contaminants. *Catalysts*, 11, 1107. https://doi.org/10.3390/catal11091107.

33. Kassahun, S.K., Kiflie, Z., Kim, H., Baye, A.F. (2021). Process optimization and kinetics analysis for photocatalytic degradation of emerging contaminant using N-doped TiO2-SiO2 nanoparticle: artificial neural network and surface response methodology approach. *Environmental Technology & Innovation*, 23, 101761. https://doi.org/10.1016/j.eti.2021.101761.

34. Kim, M., Kim, Y., Kim, H., Piao, W., Kim, C. (2016). Evaluation of the k-nearest neighbor method for forecasting the influent characteristics of wastewater treatment plant. *Frontiers in Environmental Science & Engineering*, 10, 299–310. https://doi.org/10.1007/s11783-015-0825-7.

35. Lam, S.-M., Jaffari, Z.H., Sin, J.-C., Zeng, H., Lin, H., Li, H., Mohamed, A.R. (2021a). Insight into the influence of noble metal decorated on BiFeO3 for 2,4-dichlorophenol and real herbicide wastewater treatment under visible light. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 614, 126138. https://doi.org/10.1016/j.colsurfa.2021.126138.

36. Lam, S.-M., Jaffari, Z.H., Sin, J.-C., Zeng, H., Lin, H., Li, H., Mohamed, A.R., Ng, D.-Q. (2021b). Surface decorated coral-like magnetic BiFeO3 with Au nanoparticles for effective sunlight photodegradation of 2,4-D and E. coli inactivation. *Journal of Molecular Liquids*, 326, 115372. https://doi.org/10.1016/j.molliq.2021.115372.

37. Lu, Q., Wei, Z., Li, C., Ma, J., Li, L. (2022). Photocatalytic degradation of methyl orange by noble metal Ag modified semiconductor Zn2SnO4. *Materials Science in Semiconductor Processing*, 138, 106290. https://doi.org/10.1016/j.mssp.2021.106290.

38. Manea, Y.K., Khan, A.M., Wani, A.A., Saleh, M.A.S., Qashqoosh, M.T.A., Shahadat, M., Rezakazemi, M. (2022). In-grown flower like Al-Li/Th-LDH@ CNT nanocomposite for enhanced photocatalytic degradation of MG dye and selective adsorption of Cr (VI). *Journal of Environmental Chemical Engineering*, 10, 106848. https://doi.org/10.1016/j.jece.2021.106848.

39. Mohamed, R.M., McKinney, D., Kadi, M.W., Mkhalid, I.A., Sigmund, W. (2016). Platinum/zinc oxide nanoparticles: enhanced photocatalysts degrade malachite green dye under visible light conditions. *Ceramics International*, 42, 9375–9381. https://doi.org/10.1016/j.ceramint.2016.02.147.

40. Mohsin, M., Bhatti, I.A., Ashar, A., Khan, M.W., Farooq, M.U., Khan, H., Hussain, M.T., Loomba, S., Mohiuddin, M., Zavabeti, A., et al. (2021). Iron-doped zinc oxide for photocatalyzed degradation of humic acid from municipal wastewater. *Applied Materials Today*, 23, 101047. https://doi.org/10.1016/j.apmt.2021.101047.

41. Mohtar, S.S., Aziz, F., Nor, A.R.M., Mohammed, A.M., Mhamad, S.A., Jaafar, J., Yusof, N., Salleh, W.N.W., Ismail, A.F. (2021). Photocatalytic degradation of humic acid using a novel visible-light active α-Fe2O3/NiS2 composite photocatalyst. *Journal of Environmental Chemical Engineering*, 105682. https://doi.org/10.1016/j.jece.2021.105682.

42. Morin-Crini, N., Lichtfouse, E., Fourmentin, M., Ribeiro, A.R.L., Noutsopoulos, C., Mapelli, F., Fenyvesi, É., Vieira, M.G.A., Picos-Corrales, L.A., Moreno-Piraján, J.C., Giraldo, L., Sohajda, T., Huq, M.M., Soltan, J., Torri, G., Magureanu, M., Bradu, C., Crini, G. (2022). Removal of emerging contaminants from wastewater using advanced treatments. A review. *Environmental Chemistry Letters*, 20, 1333–1375. https://doi.org/10.1007/s10311-021-01379-5.

43. Muelas-Ramos, V., Belver, C., Rodriguez, J.J., Bedia, J. (2021). Synthesis of noble metal-decorated NH2-MIL-125 titanium MOF for the photocatalytic degradation of acetaminophen under solar irradiation. *Separation and Purification Technology*, 272, 118896. https://doi.org/10.1016/j.seppur.2021.118896.

44. Mukhopadhyay, A., Duttagupta, S., Mukherjee, A. (2022). Emerging organic contaminants in global community drinking water sources and supply: A review of occurrence, processes and remediation. *Journal of Environmental Chemical Engineering*, 10, 107560. https://doi.org/10.1016/j.jece.2022.107560.

45. Navidpour, A.H., Hosseinzadeh, A., Huang, Z., Li, D., Zhou, J.L. (2022). Application of machine learning algorithms in predicting the photocatalytic degradation of perfluorooctanoic acid. *Catalysis Reviews*, 1–26. https://doi.org/10.1080/01614940.2022.2082650.

46. Nguyen, L.T., Nguyen, H.T., Pham, T.-D., Tran, T.D., Chu, H.T., Dang, H.T., Nguyen, V.-H., Nguyen, K.M., Pham, T.T., der Bruggen, B. (2020). UV–visible light driven photocatalytic degradation of ciprofloxacin by N, S co-doped TiO2: the effect of operational parameters. *Topics in Catalysis*, 63, 985–995. https://doi.org/10.1007/s11244-020-01319-7.

47. Niu, F., Chen, D., Qin, L., Gao, T., Zhang, N., Wang, S., Chen, Z., Wang, J., Sun, X., Huang, Y. (2015). Synthesis of Pt/BiFeO3 heterostructured photocatalysts for highly efficient visible-light photocatalytic performances. *Solar Energy Materials and Solar Cells*, 143, 386–396. https://doi.org/10.1016/j.solmat.2015.07.008.

48. Novakovic, M., Strbac, G., Petrovic, M., Strbac, D., Mihajlovic, I. (2020). Decomposition of pharmaceutical micropollutant–diclofenac by photocatalytic nanopowder mixtures in aqueous media: effect of optimization parameters, identification of intermediates and economic considerations. *Journal of Environmental Science and

49. Rodriguez, P., Bautista, M.A., Gonzalez, J., Escalera, S. (2018). Beyond one-hot encoding: lower dimensional target embedding. *Image and Vision Computing*, 75, 21–31. https://doi.org/10.1016/j.imavis.2018.04.004.

50. Sekar, A., Yadav, R. (2021). Green fabrication of zinc oxide supported carbon dots for visible light-responsive photocatalytic decolourization of Malachite Green dye: optimization and kinetic studies. *Optik*, 242, 167311. https://doi.org/10.1016/j.ijleo.2021.167311.

51. Tabatabai-Yazdi, F.-S., Pirbazari, A.E., Saraei, F.E.K., Gilani, N. (2021). Construction of graphene based photocatalysts for photocatalytic degradation of organic pollutant and modeling using artificial intelligence techniques. *Physica B: Condensed Matter*, 608, 412869. https://doi.org/10.1016/j.physb.2021.412869.

52. UN-SDGs. (2020). The Sustainable Development Goal 6: Clean Water and sanitation. Retrieved from https://sdgs.un.org/goals/goal6

53. Wan, X., Li, X., Wang, X., Yi, X., Zhao, Y., He, X., Wu, R., Huang, M. (2022). Water quality prediction model using Gaussian process regression based on deep learning for carbon neutrality in papermaking wastewater treatment system. *Environmental Research*, 211, 112942. https://doi.org/10.1016/j.envres.2022.112942.

54. Xia, H., Li, C., Yang, G., Shi, Z., Jin, C., He, W., Xu, J., Li, G. (2022). A review of microwave-assisted advanced oxidation processes for wastewater treatment. *Chemosphere*, 287, 131981. https://doi.org/10.1016/j.chemosphere.2021.131981.

55. Zhang, G., Liu, Y., Hashisho, Z., Sun, Z., Zheng, S., Zhong, L. (2020). Adsorption and photocatalytic degradation performances of TiO2/diatomite composite for volatile organic compounds: Effects of key parameters. *Applied Surface Science*, 525, 146633. https://doi.org/10.1016/j.apsusc.2020.146633.

56. Zhang, Y., Cai, Z., Ma, X. (2015). Photocatalysis enhancement of Au/BFO nanoparticles using plasmon resonance of Au NPs. *Physica B: Condensed Matter*, 479, 101–106. https://doi.org/10.1016/j.physb.2015.09.045.

57. Zhu, X., Wang, X., Ok, Y.S. (2019). The application of machine learning methods for prediction of metal sorption onto biochars. *Journal of Hazardous Materials*, 378, 120727. https://doi.org/10.1016/j.jhazmat.2019.06.004.

58. Zhu, X., Wan, Z., Tsang, D.C.W., He, M., Hou, D., Su, Z., Shang, J. (2021). Machine learning for the selection of carbon-based materials for tetracycline and sulfamethoxazole adsorption. *Chemical Engineering Journal*, 406, 126782. https://doi.org/10.1016/j.cej.2020.126782.

59. [Various Authors] (2020). Remediation of persistent organic pollutants–a review. *Separation and Purification Technology*, 230, 115878. Chen, M., Heijman, S.G., Rietveld, L.C. (2021). State-of-the-art ceramic membranes for oily wastewater treatment: modification and application. *Membranes*, 11, 888. https://doi.org/10.3390/membranes11090888.

60. Talaiekhozani, A., Mosayebi, M.R., Fulazzaky, M.A., Eskandari, Z., Sanayee, R. (2020). Combination of TiO2 microreactor and electroflotation for organic pollutant removal from textile dyeing industry wastewater. *Alexandria Engineering Journal*, 59, 549–563. https://doi.org/10.1016/j.aej.2019.06.004.

61. Wei, X.-Z., Yang, J., Xu, Y.-Y., Zhu, B.-K., Zhang, G.-L. (2012). Preparation and characterization of low-pressure nanofiltration membranes and the application in the separation process of dye molecules. *Journal of Porous Materials*, 19, 721–731. https://doi.org/10.1007/s10934-011-9558-2.

62. Zhang, Y., Tan, H., Wang, C., Li, B., Yang, H., Hou, H., Xiao, C. (2022). TiO2-coated glass hollow fiber membranes: preparation and application for photocatalytic methylene blue removal. *Journal of the European Ceramic Society*, 42, 2496–2504. https://doi.org/10.1016/j.jeurceramsoc.2021.09.031.

63. Tanudjaja, H.J., Hejase, C.A., Tarabara, V.V., Fane, A.G., Chew, J.W. (2019). Membrane-based separation for oily wastewater: a practical perspective. *Water Research*, 156, 347–365. https://doi.org/10.1016/j.watres.2019.02.053.

64. Ahmadipour, M., Arjmand, M., Ahmad, Z.A., Pung, S.-Y. (2020). Photocatalytic degradation of organic dye by sol–gel-synthesized CaCu3Ti4O12 powder. *Journal of Materials Engineering and Performance*, 29, 2006–2014. https://doi.org/10.1007/s11665-020-04590-7.

65. Ahmadipour, M., Arjmand, M., Thirmizir, M.Z.A., Le, A.T., Chiam, S.L., Pung, S.-Y. (2020). Synthesis of core/shell-structured CaCu3Ti4O12/SiO2 composites for effective degradation of rhodamine B under ultraviolet light. *Journal of Materials Science: Materials in Electronics*, 31, 19587–19598. https://doi.org/10.1007/s10854-020-04345-2.

66. Chiam, S.-L., Soo, Q.-Y., Pung, S.-Y., Ahmadipour, M. (2021). Polycrystalline TiO2 particles synthesized via one-step rapid heating method as electron transfer intermediates for rhodamine B removal. *Materials Chemistry and Physics*, 257, 123784. https://doi.org/10.1016/j.matchemphys.2020.123784.

67. Ahmadipour, M., Hamzah, A.A., Pang, A.L., Le, A.T., Chiam, S.-L., Ahmad, Z.A., Rajitha, B., Pung, S.-Y. (2021). Photodegradation of rhodamine B-dye pollutant using CaCu3Ti4O12-multiwall carbon nanotube nanocomposites. *Journal of Environmental Chemical Engineering*, 9, 105185. https://doi.org/10.1016/j.jece.2021.105185.

68. Pang, A.L., Arsad, A., Ahmadipour, M., Hamzah, A.A., Ahmad Zaini, M.A., Mohsin, R. (2022). High efficient degradation of organic dyes by polypyrrole-multiwall carbon nanotubes nanocomposites. *Polymer Advanced Technologies*, 33, 1402–1411. https://doi.org/10.1002/pat.5378.

69. Chiam, S.-L., Pung, S.-Y., Yeoh, F.Y., Ahmadipour, M. (2022). Highly efficient oxidative degradation of organic dyes by manganese dioxide nanoflowers. *Materials Chemistry and Physics*, 280, 125848. https://doi.org/10.1016/j.matchemphys.2022.125848.

70. Ardani, M.R., Pang, A.L., Pal, U., Zheng, R., Arsad, A., Hamzah, A.A., Ahmadipour, M. (2022). Ultrasonic-assisted polyaniline-multiwall carbon nanotube photocatalyst for efficient photodegradation of organic pollutants. *Journal of Water Process Engineering*, 46, 102557. https://doi.org/10.1016/j.jwpe.2021.102557.

71. Sk, S.J.S., Vennapoosa, C.S., Tiwari, A., Abraham, B.M., Ahmadipour, M., Pal, U. (2022). Polyaniline encapsulated Ti-MOF/CoS for efficient photocatalytic hydrogen evolution. *International Journal of Hydrogen Energy*, 47, 33955–33965. https://doi.org/10.1016/j.ijhydene.2022.06.159.

72. Khasawneh, O.F.S., Palaniandy, P., Ahmadipour, M., Mohammadi, H., Hamdan, M.R.B. (2021). Removal of acetaminophen using Fe2O3-TiO2 nanocomposites by

photocatalysis under simulated solar irradiation: optimization study. *Journal of Environmental Chemical Engineering*, 9, 104921. https://doi.org/10.1016/j.jece.2021.104921.

73. Iftikhar, R., Parveen, I., Mazhar, A., Iqbal, M.S., Kamal, G.M., Hafeez, F., Ling, P.A., Ahmadipour, M. (2022). Small organic molecules as fluorescent sensors for the detection of highly toxic heavy metal cations in portable water. *Journal of Environmental Chemical Engineering*, 11, 109030. https://doi.org/10.1016/j.jece.2022.109030.
74. Choo, K.-H. (2018). Modeling photocatalytic membrane reactors. In *Current Trends and Future Developments on (Bio-) Membranes* (pp. 297–316). Elsevier.
75. da Silva, I.N., Flauzino, R.A. (2008). An approach based on neural networks for estimation and generalization of crossflow filtration processes. *Applied Soft Computing*, 8, 590–598. https://doi.org/10.1016/j.asoc.2007.06.004.
76. Kashani, H.R.K., Azhari, S., Nayebaghayee, H., Salimi, S., Mohammadi, H.R. (2020). Prediction value of preoperative findings on meningioma grading using artificial neural network. *Clinical Neurology and Neurosurgery*, 196, 105947. https://doi.org/10.1016/j.clineuro.2020.105947.
77. Tetteh, E., Rathilal, S., Naidoo, D. (2020). Photocatalytic degradation of oily waste and phenol from a local South Africa oil refinery wastewater using response methodology. *Scientific Reports*, 10, 1–12. https://doi.org/10.1038/s41598-020-73852-0.
78. Chelladurai, S.J.S., Murugan, K., Ray, A.P., Upadhyaya, M., Narasimharaj, V., Gnanasekaran, S. (2021). Optimization of process parameters using response surface methodology: a review. *Materials Today: Proceedings*, 37, 1301–1304. https://doi.org/10.1016/j.matpr.2020.07.616.
79. Bezerra, M.A., Santelli, R.E., Oliveira, E.P., Villar, L.S., Escaleira, L.A. (2008). Response surface methodology (RSM) as a tool for optimization in analytical chemistry. *Talanta*, 76, 965–977. https://doi.org/10.1016/j.talanta.2008.01.025.
80. Srivastava, A., Aghilesh, K., Nair, A., Ram, S., Agarwal, S., Ali, J., Singh, R., Garg, M.C. (2021). Response surface methodology and artificial neural network modelling for the performance evaluation of pilot-scale hybrid nanofiltration (NF) & reverse osmosis (RO) membrane system for the treatment of brackish ground water. *Journal of Environmental Management*, 278, 111497. https://doi.org/10.1016/j.jenvman.2021.111497.
81. Irfan, M., Waqas, S., Arshad, U., Khan, J.A., Legutko, S., Kruszelnicka, I., Ginter-Kramarczyk, D., Rahman, S., Skrzypczak, A. (2022). Response surface methodology and artificial neural network modelling of membrane rotating biological contactors for wastewater treatment. *Materials*, 15, 1932. https://doi.org/10.3390/ma15061932.

82. Li, B., Yue, R., Shen, L., Chen, C., Li, R., Xu, Y., Zhang, M., Hong, H., Lin, H. (2022). A novel method integrating response surface method with artificial neural network to optimize membrane fabrication for wastewater treatment. *Journal of Cleaner Production*, 376, 134236. https://doi.org/10.1016/j.jclepro.2022.134236.
83. Rani, C.N., Karthikeyan, S. (2019). Investigation of phenanthrene degradation in a slurry photocatalytic membrane reactor: influence of operating variables and data validation. *Desalination and Water Treatment*, 145, 11–25. https://doi.org/10.5004/dwt.2019.24856.
84. Rani, C.N., Karthikeyan, S. (2020). Feasibility study of acenaphthene degradation in a novel slurry UV photocatalytic membrane reactor: effect of operating parameters and optimization using response surface modeling. *Chemical Engineering and Processing - Process Intensification*, 155, 108051. https://doi.org/10.1016/j.cep.2020.108051.
85. Azadi, F., Karimi-Jashni, A., Zerafat, M.M., Saadat, S. (2021). Fabrication, optimization, and performance of a novel double-skinned Al2O3-TiO2 ceramic nanocomposite membrane for forward osmosis application. *Environmental Technology & Innovation*, 22, 101423. https://doi.org/10.1016/j.eti.2021.101423.
86. Phattepur, H., Hiremath, P.G. (2022). Fabrication of Al2O3 supported TiO2 membranes for photocatalytic applications. *Materials Today: Proceedings*, 65, 3694–3699. https://doi.org/10.1016/j.matpr.2022.01.143.
87. Golshenas, A., Sadeghian, Z., Ashrafizadeh, S.N. (2020). Performance evaluation of a ceramic-based photocatalytic membrane reactor for treatment of oily wastewater. *Journal of Water Process Engineering*, 36, 101186. https://doi.org/10.1016/j.jwpe.2020.101186.
88. Nourbakhsh, H., Emam-Djomeh, Z., Omid, M., Mirsaeedghazi, H., Moini, S. (2014). Prediction of red plum juice permeate flux during membrane processing with ANN optimized using RSM. *Computers and Electronics in Agriculture*, 102, 1–9. https://doi.org/10.1016/j.compag.2014.01.010.
89. Vaez, M., Zarringhalam Moghaddam, A., Alijani, S. (2012). Optimization and modeling of photocatalytic degradation of azo dye using a response surface methodology (RSM) based on the central composite design with immobilized titania nanoparticles. *Industrial & Engineering Chemistry Research*, 51, 4199–4207. https://doi.org/10.1021/ie300626u.
90. Abbasi, S., Mirghorayshi, M., Zinadini, S., Zinatizadeh, A. (2020). A novel single continuous electrocoagulation process for treatment of licorice processing wastewater: optimization of operating factors using RSM. *Process Safety and Environmental Protection*, 134, 323–332. https://doi.org/10.1016/j.psep.2020.03.005.

91. Prabhu, M.V., Karthikeyan, R. (2018). Comparative studies on modelling and optimization of hydrodynamic parameters on inverse fluidized bed reactor using ANN-GA and RSM. *Alexandria Engineering Journal*, 57, 3019–3032. https://doi.org/10.1016/j.aej.2018.03.014.
92. Tetteh, E.K., Naidoo, B.D., Rathilal, S. (2020). Optimization of photo-catalytic degradation of oil refinery wastewater using response surface methodology. *Environmental Nanotechnology, Monitoring & Management*, 18, 100684. https://doi.org/10.1016/j.enmm.2021.100684.
93. Nandi, B., Moparthi, A., Uppaluri, R., Purkait, M.K. (2010). Treatment of oily wastewater using low cost ceramic membrane: comparative assessment of pore blocking and artificial neural network models. *Chemical Engineering Research & Design*, 88, 881–892. https://doi.org/10.1016/j.cherd.2010.03.016.
94. Abdulgader, M., Yu, J., Zinatizadeh, A.A., Williams, P., Rahimi, Z. (2019). Process analysis and optimization of single stage flexible fibre biofilm reactor treating milk processing industrial wastewater using response surface methodology (RSM). *Chemical Engineering Research & Design*, 149, 169–181. https://doi.org/10.1016/j.cherd.2019.08.003.
95. Thangavelu, K., Zou, L. (2022). Evaluating oil removal by amphiphilic MoS2/cellulose acetate fibrous sponge in a flow-through reactor and by artificial neural network. *Environmental Nanotechnology, Monitoring & Management*, 18, 100684. https://doi.org/10.1016/j.enmm.2022.100684.
96. Choo, K.-H., Lee, J.-H., Park, J.-Y., Lee, J.-H. (2023). Enhanced photocatalytic activity of TiO2 photocatalysts in a photocatalytic membrane reactor for wastewater treatment: Effect of membrane characteristics and operating conditions. *Journal of Hazardous Materials*, 453, 131197. https://doi.org/10.1016/j.jhazmat.2022.131197.
97. Bandyopadhyay, D., Kumar, P., Pandey, S., Yadav, S. (2023). Application of response surface methodology and artificial neural networks for the optimization of dye removal from wastewater using hybrid photocatalytic processes. *Journal of Environmental Management*, 321, 116045. https://doi.org/10.1016/j.jenvman.2022.116045.
98. Kumar, V., Singh, S., Sharma, P., Verma, V., Kumar, V. (2023). Optimization of photocatalytic degradation of organic pollutants using response surface methodology: A review. *Journal of Environmental Chemical Engineering*, 11, 108530. https://doi.org/10.1016/j.jece.2022.108530.
99. Fernandez, A., Moreno, J., Martin, A., Moreno, J. (2024). Development and optimization of TiO2-based photocatalysts for environmental remediation: A comparative study using RSM and ANN approaches. *Catalysis Today*, 399, 113023. https://doi.org/10.1016/j.cattod.2023.113023.

100. Sinha, R., Joshi, P., Shah, A., Gupta, A. (2024). Optimization of photocatalytic degradation of pharmaceutical contaminants using response surface methodology and artificial neural networks. *Environmental Science and Pollution Research*, 31, 4231–4243. https://doi.org/10.1007/s11356-023-23344-5.

101. Patel, R., Sharma, V., Rathi, A., Jain, A. (2024). Photocatalytic treatment of industrial wastewater using novel nanomaterials: Process optimization using RSM and ANN. *Chemical Engineering Journal*, 460, 141687. https://doi.org/10.1016/j.cej.2023.141687.

102. Wang, X., Liu, Q., Lu, Y., Zhang, H. (2024). Design and optimization of photocatalytic membrane reactors for organic pollutant removal using response surface methodology and artificial neural networks. *Desalination and Water Treatment*, 284, 21–33. https://doi.org/10.5004/dwt.2024.021234.

103. Zhao, H., Xu, Y., Li, Z., Zhao, L. (2024). Enhancing photocatalytic degradation of pollutants in wastewater treatment: An integrated approach using RSM and ANN modeling. *Journal of Cleaner Production*, 376, 134987. https://doi.org/10.1016/j.jclepro.2024.134987.

104. Zhang, J., Liu, Y., Wang, Q., Zhang, Y. (2024). Performance evaluation and optimization of photocatalytic membranes for wastewater treatment: A combined RSM and ANN approach. *Journal of Hazardous Materials*, 456, 132032. https://doi.org/10.1016/j.jhazmat.2024.132032.

105. Yang, L., Wang, L., Zhang, L., Zhao, Z. (2024). Advances in photocatalytic membrane reactors for wastewater treatment: A review of optimization techniques using RSM and ANN. *Environmental Technology & Innovation*, 23, 102445. https://doi.org/10.1016/j.eti.2024.102445.

13 APPENDICES

13.1 TABLE OF FIGURES:

Figure 1: Histogram of Original & Synthetic Data for Column Time(Hour) 52
Figure 2: Q-Q plot of Original & Synthetic Data for Column Time(Hour) 52
Figure 3:Histogram of Original & Synthetic Data for Column Blank UV 53
Figure 4: Q-Q plot of Original & Synthetic Data for Column Blank UV 53
Figure 5: Histogram of Original & Synthetic Data for Column Blank Sunlight 53
Figure 6:Q-Q plot of Original & Synthetic Data for Column Blank Sunlight 54
Figure 7: Histogram of Original & Synthetic Data for Column C1 54
Figure 8: Q-Q plot of Original & Synthetic Data for Column C1 54
Figure 9: Histogram of Original & Synthetic Data for Column C2 55
Figure 10: Q-Q plot of Original & Synthetic Data for Column C2 55
Figure 11: Histogram of Original & Synthetic Data for Column C3 55
Figure 12: Q-Q plot of Original & Synthetic Data for Column C3 56
Figure 13: Histogram of Original & Synthetic Data for Column C4 56
Figure 14: Q-Q plot of Original & Synthetic Data for Column C4 56
Figure 15: Histogram of Original & Synthetic Data for Column C_{Com} 57
Figure 16:Q-Q plot of Original & Synthetic Data for Column C_{Com} 57
Figure 17: Histogram of Original & Synthetic Data for Column 10mg/L TiO_2 NPa 57
Figure 18: Q-Q plot of Original & Synthetic Data for Column 10mg/L TiO2 NPs 58
Figure 19: Histogram of Original & Synthetic Data for Column 20mg/L TiO2 NPs 58
Figure 20: Q-Q plot of Original & Synthetic Data for Column 20mg/L TiO2 NPs 59
Figure 21:Histogram of Original & Synthetic Data for Column 50mg/L TiO2 NPs 59
Figure 22: Q-Q plot of Original & Synthetic Data for Column 50mg/L TiO2 NPs 59
Figure 23: Histogram of Original & Synthetic Data for Column 80mg/L TiO2 NPs 60
Figure 24: Q-Q plot of Original & Synthetic Data for Column 80mg/L TiO2 NPs 60
Figure 25: Histogram of Original & Synthetic Data for Column 110mg/L TiO2 NPs 61
Figure 26:Histogram & Q-Q plot of Original & Synthetic Data for Column 150mg/L TiO2 NPs.. 61
Figure 27: Histogram & Q-Q plot of Original & Synthetic Data for Column pH3 62
Figure 28: Histogram & Q-Q plot of Original & Synthetic Data for Column pH7 63
Figure 29: Histogram & Q-Q plot of Original & Synthetic Data for Column pH10 63
Figure 30: Histogram & Q-Q plot of Original & Synthetic Data for Column pH12 64
Figure 31: Plot of MSE, MAE, R^2 in CatBoost Model 71
Figure 32: Test vs Predicted Data for Different Parameters using CatBoost Model 75
Figure 33: Test Vs Predicted Data for Different Parameters Using XG Boost Model 83
Figure 34: Test Vs Predicted Data for Different Parameters Using Random Forest Model 90
Figure 35: Plot of MSE, MAE, R2 in Random Forest Model 92
Figure 36: Plot of MSE, MAE, R2 in LightGBM Model 97
Figure 37: Test Vs Predicted Data for Different Parameters Using LightGBM Model 101
Figure 38: 0Plot of MSE, MAE, R2 in SVM Model 107
Figure 39: Test Vs Predicted Data for Different Parameters Using SVM Model 109
Figure 40: Mean Squared Error (MSE) Comparison for all the models 115
Figure 41: Mean Absolute Error (MAE) Comparison for All Model 116
Figure 42: R-Squared (R^2) Comparison for All Models 116