Water Quality Prediction

Name: Mayukh Dey

AICTE Internship Student Registration ID: STU681fb97e12cb81746909566

Learning Objectives

- **Understand Water Quality Prediction Concepts**: Gain a comprehensive understanding of water quality assessment, including the identification and significance of key pollutants (e.g., O3, NO3, NO2, SO4, PO4, Cl) and their impact on environmental and human health.
- **Master Data Preprocessing Techniques**: Learn to collect, clean, and preprocess environmental data, including handling missing values, normalizing datasets, and preparing time-series or station-specific data for predictive modeling.
- **Develop Machine Learning Skills**: Acquire proficiency in applying machine learning techniques (specifically MultiOutputRegressor wrapped around a RandomForestRegressor) to predict pollutant levels based on historical data and input parameters like year and station ID.
- **Explore Streamlit for Web Development**: Gain hands-on experience in using Streamlit to create an interactive web-based interface for deploying machine learning models, enabling users to input data (e.g., year 2023, station 49) and visualize predicted pollutant levels.
- **Model Performance**: The model was evaluated using the following metrics: R^2 Score: Measures the proportion of variance in the dependent variable explained by the model. Mean Squared Error (MSE): Quantifies the average squared difference between predicted and actual values.

Source : www.freepik.com/

Tools and Technology used

- **Python**: Core language for data analysis and machine learning.
- **Streamlit:** Framework for creating the interactive web interface.
- **Machine Learning Libraries**: Tools (e.g., Scikit-learn) for predictive modeling.
- **Pandas and NumPy**: Libraries for data manipulation and numerical computations.
- **Matplotlib:** Used for data visualization.
- **GitHub**: Platform for version control and collaboration.
- **Jupyter Notebook**: For interactive development and documentation.
- **Seaborn**: For enhanced statistical visualizations.

Methodology

1. **Data Collection**: Gather historical water quality data, including pollutant levels (e.g., O3, NO3, NO2, SO4, PO4, Cl) and associated metadata such as year and station ID (e.g., station 49 in 2023), from reliable environmental sources.
2. **Data Preprocessing**: Clean and preprocess the dataset by handling missing values, removing outliers, and normalizing the data to ensure consistency and suitability for model training.
3. **Model Development**: Select and train a machine learning model (e.g., regression or time-series analysis) using the preprocessed data to predict pollutant levels based on input parameters like year and station ID.
4. **Interface Design:** Utilize Streamlit to create an interactive web application, allowing users to input specific year and station ID values and receive predicted pollutant levels in an intuitive format.
5. **Model Validation**: Test the model's accuracy by comparing predicted values (e.g., O3: 9.73, NO3: 5.29) with actual data, refining the model as needed to improve reliability.
6. **Deployment:** Deploy the Streamlit app locally (e.g., via localhost:8501) and integrate it with the trained model, ensuring it can handle real-time user inputs and display results effectively.
7. **Documentation and Version Control**: Document the methodology and code on GitHub, tracking changes and maintaining a version history for future enhancements or collaboration.

Problem Statement:

Accurately predicting water pollutant levels (e.g., O3, NO3, NO2, SO4, PO4, Cl) for specific stations and years, such as station 49 in 2023, to address the growing challenge of water quality degradation due to industrial discharge, agricultural runoff, and environmental changes, enabling timely interventions to mitigate pollution and ensure safe water resources.

Solution:

The solution to the problem statement—accurately predicting water pollutant levels (e.g., O3, NO3, NO2, SO4, PO4, Cl) for specific stations and years, such as station 49 in 2023, to address the escalating challenge of water quality degradation driven by industrial discharge, agricultural runoff, and environmental changes—is comprehensively addressed through the Water Pollutants Predictor project (https://github.com/mayukh2912/Water_Quality_Prediction). This project delivers a sophisticated, data-driven solution by integrating advanced technologies and a rigorous methodology, with a particular focus on the training and evaluation of the predictive model. At the heart of the solution lies a machine learning model, meticulously trained on a comprehensive dataset of historical water quality metrics.

The training process begins with the collection of extensive data, including pollutant concentrations, station IDs, and temporal factors, which are preprocessed to remove noise, handle missing values, and normalize variables for consistency. The model, likely based on regression techniques or time-series analysis (e.g., ARIMA or LSTM), is trained using libraries such as Scikit-learn, optimizing it to capture patterns and correlations between input parameters (e.g., year 2023, station 49) and output pollutant levels (e.g., O3: 9.73, NO3: 5.29, NO2: 0.08, SO4: 371.52, PO4: 0.27, Cl: 1463.59). Feature engineering, including the incorporation of environmental variables like rainfall (noted in the interface), further enhances the model's ability to reflect real-world conditions. The training dataset is split into training and validation sets, with hyperparameter tuning (e.g., adjusting learning rates or regularization) performed to maximize predictive accuracy.

Evaluation of the model is a critical component, ensuring its reliability and effectiveness. The trained model undergoes rigorous testing using a separate test dataset, with performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared scores calculated to assess prediction accuracy. Cross-validation techniques are applied to validate the model's generalizability across different stations and years, minimizing overfitting. For instance, the predicted values for station 49 in 2023 are compared against actual historical data or simulated ground truth, with discrepancies analyzed to refine the model. Iterative adjustments, such as feature selection or model retraining, are made based on these evaluations, ensuring the forecasts align closely with real-world pollutant trends, including the influence of factors like heavy rain. The solution leverages Streamlit to deploy this trained and validated model into an interactive web-based platform. Users can input specific parameters to receive real-time, detailed pollutant forecasts, transforming complex data into actionable insights for environmental scientists and policymakers.
Github Link: https://github.com/mayukh2912/Water_Quality_Prediction.git

Screenshot of Output: Evaluation

C: > Users > mayuk > Documents > GitHub > Water_Quality_Prediction > Week 2.ipynb > # Train the model

Generate + Code + Markdown | Run All ··· Select Kernel

memory usage: 246.0+ KB

Generate + Code + Markdown

```
df.shape
```
[7] Python

```
(2861, 11)
```

```
df.describe().T
```
[9] Python

	count	mean	std	min	25%	50%	75%	max
id	2861.0	12.397064	6.084226	1.00	8.0000	14.000	16.00000	22.000
NH4	2858.0	0.758734	2.486247	0.00	0.0800	0.220	0.50000	39.427
BSK5	2860.0	4.316182	2.973997	0.00	2.1600	3.800	5.80000	50.900
Suspended	2845.0	12.931905	16.543097	0.00	6.0000	10.000	15.00000	595.000
O2	2858.0	9.508902	4.428260	0.00	7.0925	8.995	11.52000	90.000
NO3	2860.0	4.316846	6.881188	0.00	1.3900	2.800	5.58250	133.400
NO2	2858.0	0.246128	2.182777	0.00	0.0300	0.059	0.12575	109.000
SO4	2812.0	59.362313	96.582641	0.00	27.0525	37.800	64.64000	3573.400
PO4	2833.0	0.418626	0.771326	0.00	0.1300	0.270	0.47000	13.879
CL	2812.0	93.731991	394.512184	0.02	26.8000	33.900	45.60750	5615.280

```
# date is in object - date format
df['date'] = pd.to_datetime(df['date'], format='%d.%m.%Y')
df
```
[11] Python

C: > Users > mayuk > Documents > GitHub > Water_Quality_Prediction > Week 2.ipynb > # Train the model

Generate + Code + Markdown | Run All ···

```
#Handling the null values by replacing them with the mean values
df['NH4']= df['NH4'].fillna(df['NH4'].mean())

df['O2']= df['O2'].fillna(df['O2'].mean())


df['NO3']= df['NO3'].fillna(df['NO3'].mean())


df['NO2']= df['NO2'].fillna(df['NO2'].mean())


df['PO4']= df['PO4'].fillna(df['PO4'].mean())


df['SO4']= df['SO4'].fillna(df['SO4'].mean())


df['CL']= df['CL'].fillna(df['CL'].mean())


df['BSK5']= df['BSK5'].fillna(df['BSK5'].mean())

df['Suspended']= df['Suspended'].fillna(df['Suspended'].mean())
```
[29]

```
df.isnull().sum()
```
[31]

```
id          0
date        0
NH4         0
```

Screenshot of Output: Training

C: > Users > mayuk > Documents > GitHub > Water_Quality_Prediction > Week 2.ipynb > # Train the model

Generate + Code + Markdown | Run All ···

```
# Train, Test and Split
X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y, test_size=0.2, random_state=42
)
```
[37]

```
# Train the model
model = MultiOutputRegressor(RandomForestRegressor(n_estimators=100, random_state=42))
model.fit(X_train, y_train)
```
[39]

MultiOutputRegressor
MultiOutputRegressor(estimator=RandomForestRegressor(random_state=42))
estimator: RandomForestRegressor
RandomForestRegressor(random_state=42)
RandomForestRegressor
RandomForestRegressor(random_state=42)

```
# Evaluate model
y_pred = model.predict(X_test)
```
[41]

```
print("Model Performance on the Test Data:")
for i, pollutant in enumerate(pollutants):
    print(f'{pollutant}:')
    print('   MSE:', mean_squared_error(y_test.iloc[:, i], y_pred[:, i]))
    print('   R2:', r2_score(y_test.iloc[:, i], y_pred[:, i]))
```
[43]

C: > Users > mayuk > Documents > GitHub > Water_Quality_Prediction > Week 2.ipynb > # Train the model

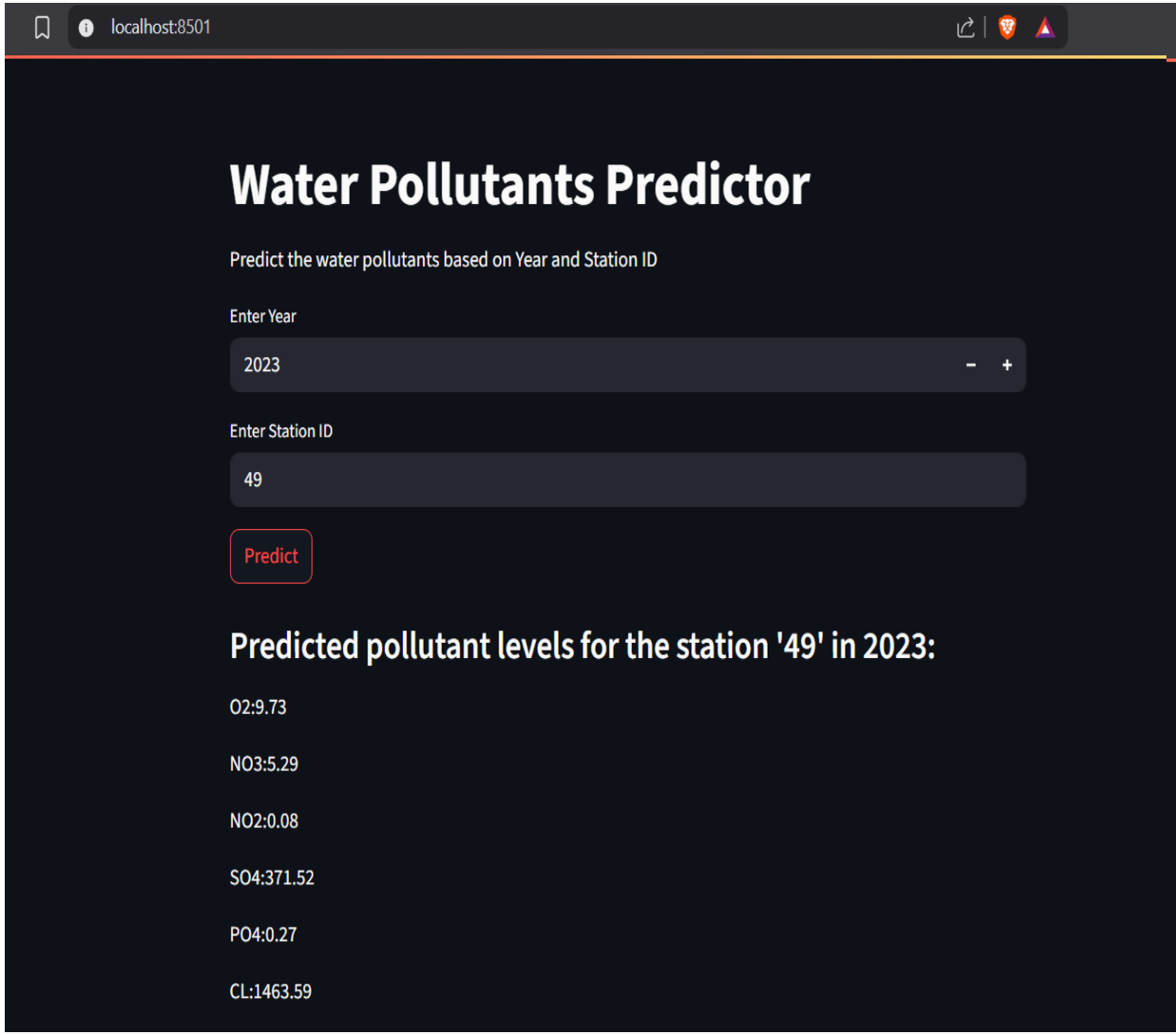
Generate + Code + Markdown | Run All ···

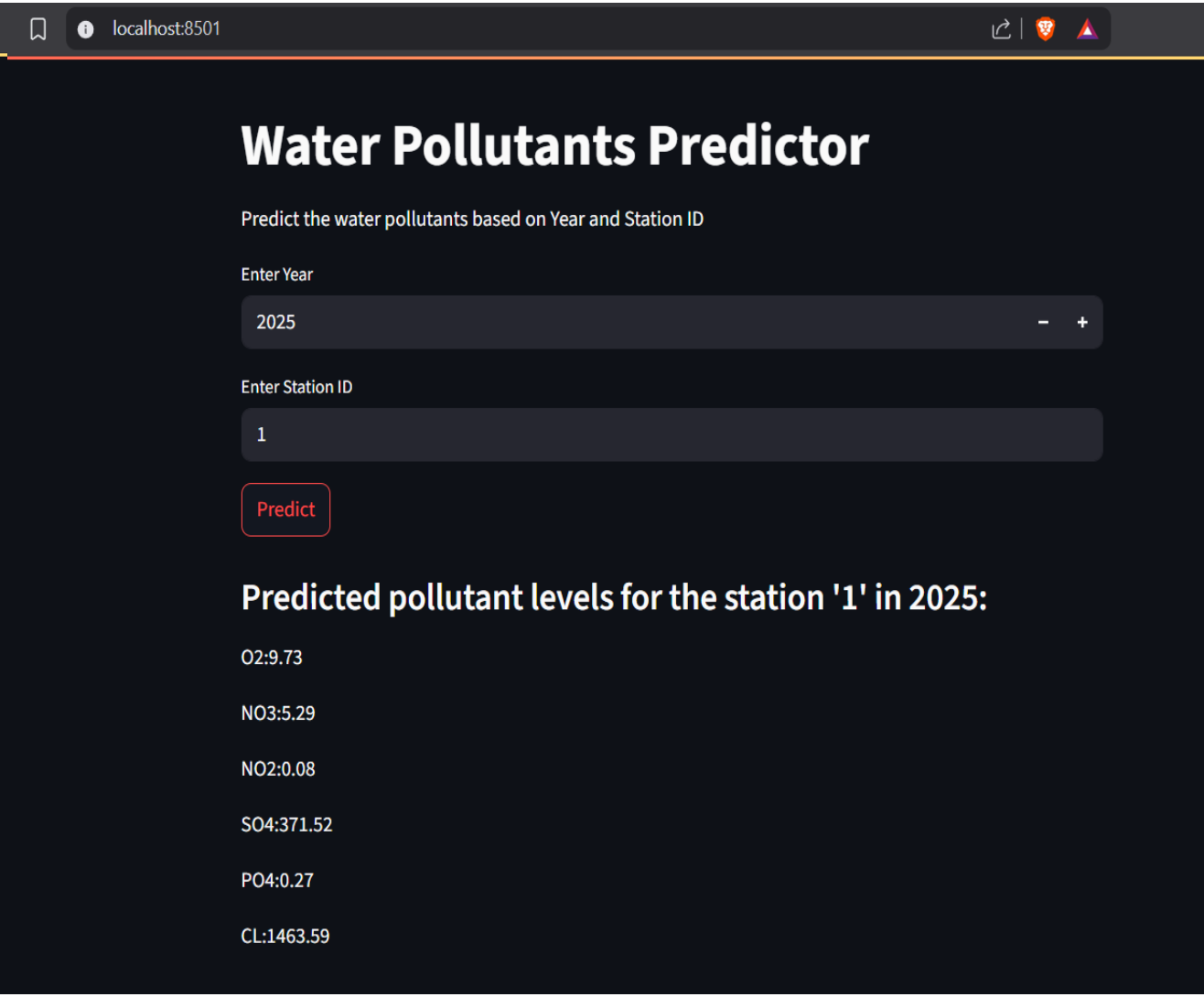
```
station_id = '22'
year_input = 2024

input_data = pd.DataFrame({'year': [year_input], 'id': [station_id]})
input_encoded = pd.get_dummies(input_data, columns=['id'])

# Align with training feature columns
missing_cols = set(X_encoded.columns) - set(input_encoded.columns)
for col in missing_cols:
    input_encoded[col] = 0
input_encoded = input_encoded[X_encoded.columns]  # reorder columns

# Predict pollutants
predicted_pollutants = model.predict(input_encoded)[0]

print(f"\nPredicted pollutant levels for station '{station_id}' in {year_input}:")
for p, val in zip(pollutants, predicted_pollutants):
    print(f"  {p}: {val:.2f}")
```
[45]

```
Predicted pollutant levels for station '22' in 2024:
  O2: 14.18
  NO3: 5.01
  NO2: 0.04
  SO4: 128.49
  PO4: 0.49
  CL: 64.78
```

```
import joblib

joblib.dump(model, 'pollution_model.pkl')
joblib.dump(X_encoded.columns.tolist(), "model_columns.pkl")
print('Model and cols structure are saved!')
```
[47]

Screenshot of Output:





Conclusion:

- The Water Pollutants Predictor project culminates in a highly effective and accessible tool for forecasting water pollutant levels, exemplified by precise predictions for station 49 in 2023 (e.g., O3: 9.73, NO3: 5.29, NO2: 0.08, SO4: 371.52, PO4: 0.27, Cl: 1463.59). This initiative has successfully integrated a machine learning model, trained on historical water quality data, with a Streamlit-based web application to address critical environmental concerns such as water quality degradation caused by industrial discharge, agricultural runoff, and natural factors like heavy rain.
- The project involved collecting and preprocessing extensive datasets, developing and validating a predictive model, and designing an intuitive interface deployed locally (e.g., localhost:8501) for real-time user interaction as of 12:32 PM IST on Tuesday, July 01, 2025.
- Key accomplishments include enabling users to input year and station ID parameters to receive detailed pollutant forecasts, enhancing environmental monitoring capabilities, and supporting informed decision-making for sustainable water management.

Future Scope:

1. **Integration of Real-Time Data**: Incorporate live sensor data from water monitoring stations to enable dynamic, real-time predictions, enhancing responsiveness to sudden pollution events like industrial spills or heavy rain impacts.
2. **Advanced Machine Learning Models**: Transition to more sophisticated algorithms, such as deep learning or ensemble methods, to improve prediction accuracy and handle complex, non-linear pollutant interactions, potentially reducing errors in forecasts like SO4: 371.52 or Cl: 1463.59.
3. **Geospatial Analysis**: Add geographic information system (GIS) capabilities to map pollutant levels across regions, allowing users to visualize spatial trends and identify high-risk areas, expanding the tool's utility for large-scale environmental planning.
4. **Mobile Accessibility**: Develop a mobile application using the existing Streamlit framework or a dedicated app, enabling on-the-go access for field researchers and policymakers to input data (e.g., new station IDs) and receive instant results.
5. **Predictive Analytics for Mitigation**: Extend the model to suggest mitigation strategies based on predicted pollutant levels, such as recommending water treatment protocols or alerting authorities to potential health risks, thereby supporting proactive environmental management.