Water Quality Prediction

Learning Objectives

- ✓ How to apply multi-output regression techniques with authentic and real-world environmental datasets.
- ✓ Understand how to preprocess water quality data and extract useful features (e.g., year, station ID).
- ✓ Explore the use of RandomForestRegressor within a MultiOutputRegressor wrapper.
- ✓ Evaluate model performance using regression metrics like R^2 Score and Mean Squared Error.
- ✓ Learn how to save, load, and deploy machine learning models using .pkl files in a Streamlit web app.

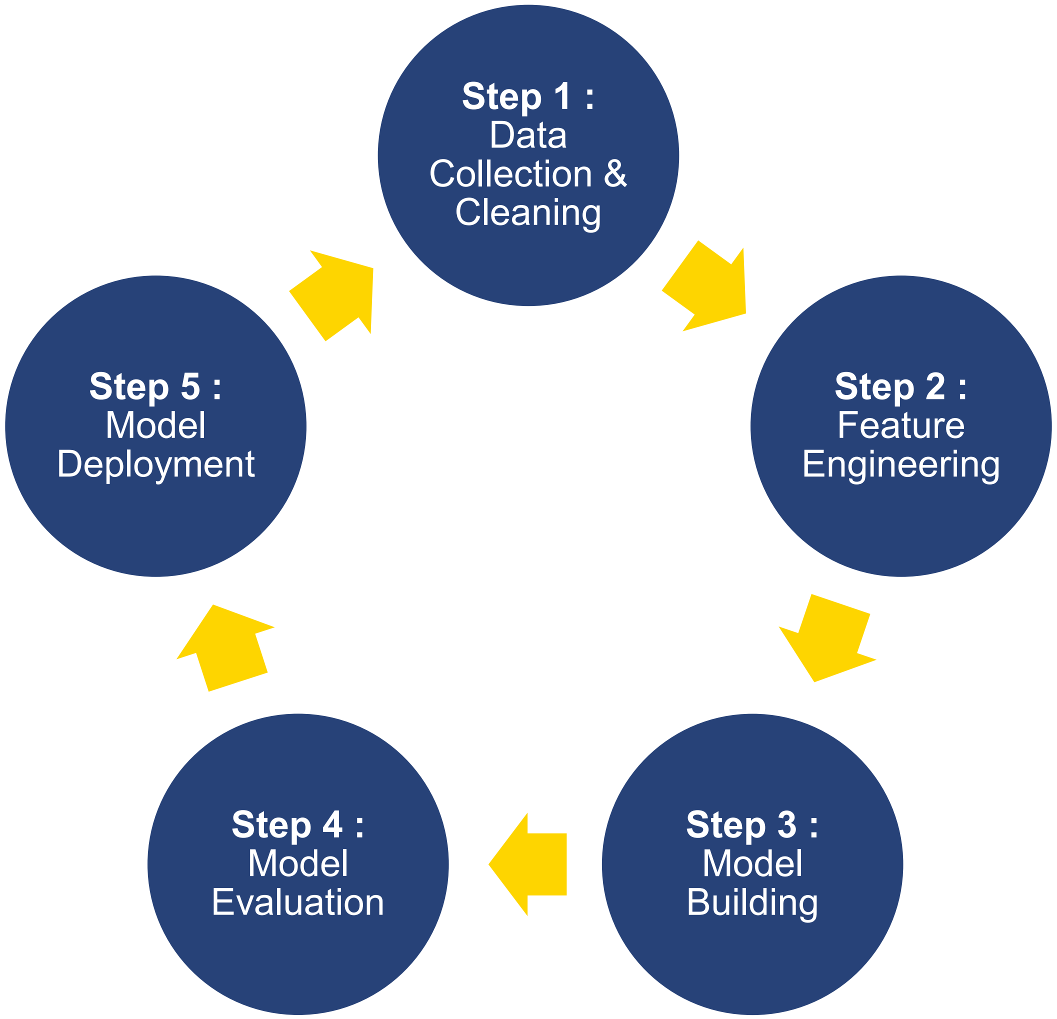
Tools and Technology used

- **Programming Language:**
Python

- **Libraries & Frameworks:**
Data Handling: Pandas, NumPy
Visualization: Matplotlib, Seaborn
ML & Model Saving: Scikit-learn, Joblib
UI & Deployment: Streamlit

- **Platform:**
Development: Google Colab
Deployment: Streamlit Sharing

Methodology



Step 1: Data Collection & Cleaning
Collected water station data (2000–2021), cleaned nulls, extracted year

Step 2: Feature Engineering
Selected year and station ID, applied one-hot encoding for station ID

Step 3: Model Building
Used MultiOutputRegressor with RandomForest to predict 6 pollutants

Step 4: Model Evaluation
Measured performance with R^2 Score and MSE for each pollutant

Step 5: Model Deployment
Saved model using Joblib and deployed using Streamlit web app

Problem Statement:

- Access to clean water is essential, but monitoring water quality manually is slow, expensive, and resource-intensive
- Water pollution varies across time and locations, making consistent testing difficult for authorities
- Traditional systems cannot predict multiple pollutant levels simultaneously
- Delay in identifying pollutants like NO_3, PO_4, or Cl can lead to serious health and environmental consequences
- There is a need for a machine learning-based solution to forecast pollutant levels using historical data — helping with early detection and preventive action

Solution:

- Trained a multi-output regression model on 21 years of water quality data (2000–2021)
- Used minimal but informative features: year and station ID
- Applied One-Hot Encoding for station ID and extracted time-based features from date
- Built a Streamlit-based web interface to predict 6 pollutants simultaneously (O_2, NO_3, NO_2, SO_4 PO_4, Cl)
- Achieved good R^2 scores for pollutants like NO_3, SO_4, and Cl
- Model saved and reused using joblib (.pkl), integrated into UI
- Supports real-time monitoring for pollution control agencies

Screenshot of Output:

Screenshot 1

```python
# Evaluation of the model performance
for i, pollutant in enumerate(pollutants):
    print(f"Pollutant: {pollutant}")
    print(f"R2 Score: {r2_score(y_test.iloc[:, i], y_pred[:, i]):.4f}")
    print(f"Mean Squared Error: {mean_squared_error(y_test.iloc[:, i], y_pred[:, i]):.4f}")
    print("-" * 30)
```

```
Pollutant: O2
R2 Score: -0.0167
Mean Squared Error: 22.2183
------------------------------
Pollutant: NO3
R2 Score: 0.5162
Mean Squared Error: 18.1531
------------------------------
Pollutant: NO2
R2 Score: -78.4207
Mean Squared Error: 10.6074
------------------------------
Pollutant: SO4
R2 Score: 0.4118
Mean Squared Error: 2412.1394
------------------------------
Pollutant: PO4
R2 Score: 0.3221
Mean Squared Error: 0.3850
------------------------------
Pollutant: CL
R2 Score: 0.7358
Mean Squared Error: 34882.8143
------------------------------
```

Model Evaluation: R^2 and MSE for Each Pollutant

- Model evaluated using R^2 Score and Mean Squared Error (MSE) for each pollutant
- CL achieved the best performance (R^2: 0.7358), followed by NO_3 (0.5162) and SO_4 (0.4118)
- NO_2 and O_2 had low/negative R^2 scores, indicating limited prediction accuracy
- PO_4 showed moderate performance (R^2: 0.3221)
- Higher MSE values for SO_4 and CL are due to their larger natural scales

Screenshot of Output:

```
# Saving the trained model
joblib.dump(model, 'water_quality_model.pkl')
print("Model saved as 'water_quality_model.pkl'")

# Saving the feature columns used to train the model
joblib.dump(X_train.columns.tolist(), 'model_columns.pkl')
print("Model columns saved as 'model_columns.pkl'")
```

```
Model saved as 'water_quality_model.pkl'
Model columns saved as 'model_columns.pkl'
```

```
[30] model_loaded = joblib.load('water_quality_model.pkl')

sample = X_test.sample(5, random_state=1)
predictions = model_loaded.predict(sample)

print(pd.DataFrame(predictions, columns=pollutants))
```

	O2	NO3	NO2	SO4	PO4	CL
0	10.115731	4.496948	0.029559	56.472951	0.426765	40.371172
1	5.471103	4.308273	0.728769	53.183141	3.189822	83.221154
2	6.593253	8.597088	0.083556	120.433853	0.414340	64.741407
3	13.860772	5.846808	0.159278	35.990289	0.118727	49.008799
4	9.692809	4.333322	0.619459	40.667637	0.389084	31.940669

Screenshot 2

Model Saving, Loading, and Sample Predictions

- The trained model was saved as a .pkl file for reuse without retraining
- Model columns (features) were saved separately to ensure correct input alignment during prediction
- The saved model was later loaded and used to predict pollutant levels for 5 random test samples
- The output shows predicted concentrations of six pollutants (O_2, NO_3, NO_2, SO_4, PO_4, Cl)
- This confirms successful deployment and end-to-end model usability

Screenshot of Output:

Screenshot 3

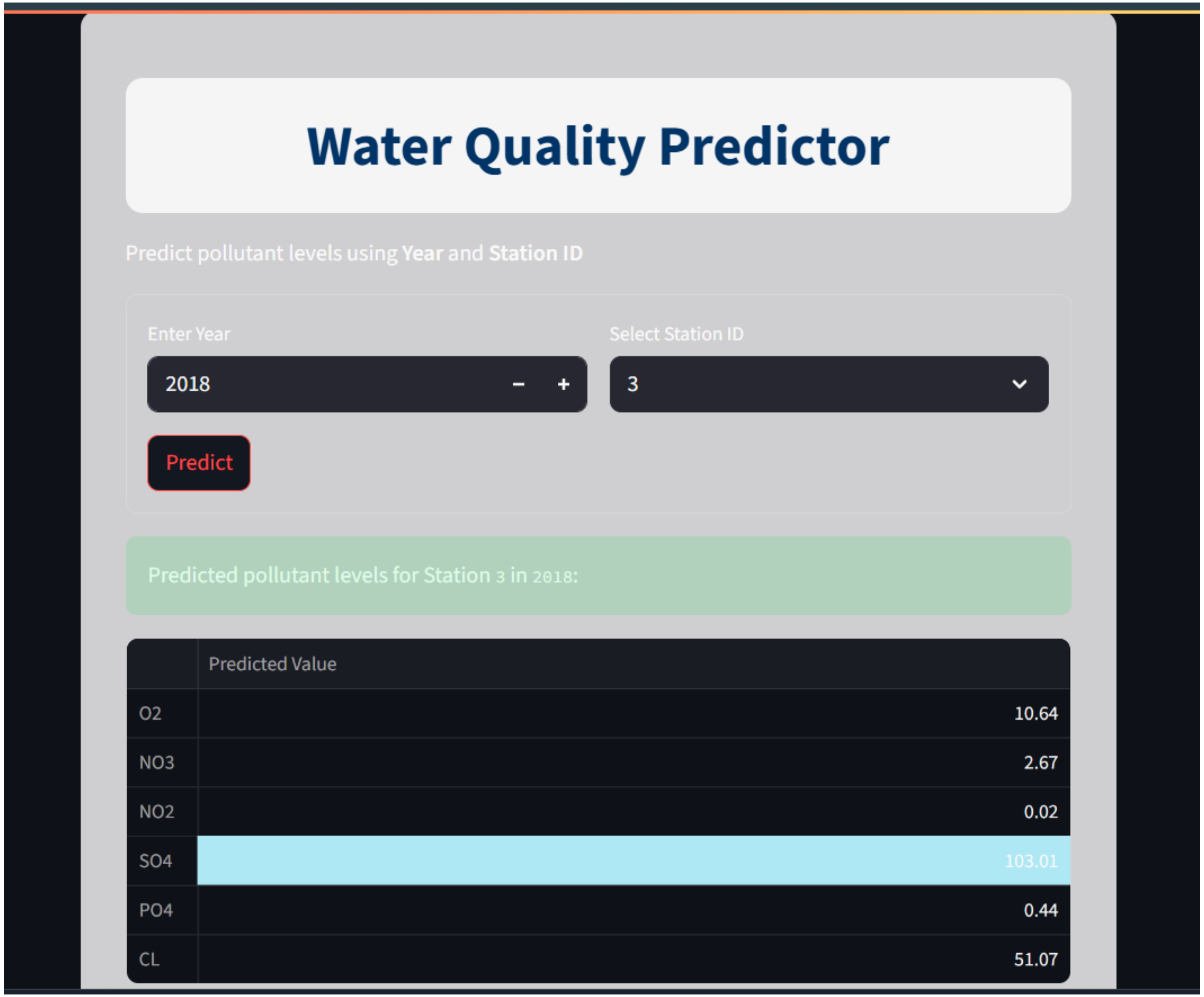
Pollutant Prediction via Streamlit Interface

• Streamlit UI takes Year and Station ID as input

• Loads the trained model and predicts 6 pollutant levels

• Displays results in a clean, user-friendly format

•Output is displayed in a readable format showing predicted values for O_2, NO_3, NO_2, SO_4, PO_4, and Cl



Conclusion:

This project focused on predicting key water pollutants (O_2, NO_3, NO_2, SO_4, PO_4, Cl) using multi-output regression on 21 years of water quality data. We used machine learning techniques like **RandomForestRegressor** within a **MultiOutputRegressor** framework, and built a web interface using **Streamlit** for real-time predictions.

Keywords: Water Quality, Multi-Target Regression, Random Forest, Model Deployment, Streamlit, Joblib

Future Scope:

- Include more features like rainfall, temperature, or industrial activity
- Integrate real-time sensor data
- Add pollution trend visualization and alert system
- Scale predictions to other Indian regions and water bodies for broader impact