Predicting Heavy Metal Concentrations in Water Using Machine Learning

Project Aim

To build and evaluate machine learning models for predicting the concentrations of various heavy metals (Lead - Pb, Nickel - Ni, Zinc - Zn, Arsenic - As, Chromium - Cr) in river water using real-world environmental monitoring data, and to assess their utility in water quality analysis and decision-making.

Objectives

1. Preprocess and clean complex environmental water quality data.
2. Generate enriched features using domain-relevant transformations.
3. Train and compare multiple regression models:
    - Linear Regression
    - Random Forest Regressor
    - XGBoost Regressor
4. Visualize predictions and feature importance.
5. Document and publish the project, including reusable code and key insights.

Dataset Summary

Source: Environmental water quality data collected quarterly over 4 years from two sites along the Styr River in northwestern Ukraine—an area impacted by industrial discharge.

Key features:

- Target Metals: Pb, Ni, Zn, As, Cr (each with replicate columns over time)
- Indices: HPI (Pollution Index), HEI (Evaluation Index), DC (Degree of Contamination)
- Time & Location: Year, Month, Si
- Other Trace Elements & Metrics: Additional heavy metals and indicators

Tool/Library	Use Case
pandas, numpy	Data loading, preprocessing, and transformation
matplotlib, seaborn	Visualizations & plots
scikit-learn	Regression modeling, evaluation
xgboost	Advanced gradient boosting model
Jupyter Notebook	Development and documentation
GitHub	Version control and project publishing

Strategy and Implementation

Data Preprocessing

- Skipped metadata row from CSV.
- Removed non-numeric, invalid, or missing rows.
- Selected only relevant columns (excluding textual and constant ones).

Feature Engineering

Created transformed features from numerical columns:

- log_x: Logarithmic transformation
- sqrt_x: Square root transformation
- x_squared: Squared values
- x_cubed: Cubed values

These help improve non-linear pattern recognition for all models.

Modeling Approach

- Train-Test Split: 80/20
- Models Trained on Each Metal:
  - Linear Regression: Simple, interpretable baseline
  - Random Forest: Captures non-linearities, handles high dimensionality
  - XGBoost: High-performing gradient boosting model

Evaluation Metrics

We used:

- MAE (Mean Absolute Error): Average magnitude of error.
- RMSE (Root Mean Squared Error): Penalizes large errors.
- R^2 Score (Coefficient of Determination): Proportion of variance explained.

These together provide a rounded view of performance.

Visualizations

Actual vs Predicted Plots

(Similar plots created for Ni, Zn, As, and Cr)

Feature Importance (XGBoost)

(Reveals which parameters most influence prediction)

Challenges Faced & Solutions

Challenge	Solution
Inconsistent headers	Used skiprows=1 to skip metadata row
Mixed column types	Filtered only numeric features
Model errors (e.g., NaNs)	Dropped invalid rows, handled missing data
Overfitting concerns	Used simple models + test split for validation
Visualization saving	Used plt.savefig() for clean export

Results Summary

- Pb and Cr had the most stable and predictable trends, with XGBoost yielding the best R^2 scores.
- Ni and As had lower R^2 but were still modeled with acceptable MAE/RMSE.
- Zn showed moderate variability, suggesting possible seasonal or site-specific spikes.

Results (Pb):

Model	MAE	RMSE	R^2
Linear Regression	0.0382	0.0690	0.7279
Random Forest	0.0362	0.1011	0.4157
XGBoost	0.0235	0.0660	0.7509

Utility and Value

- Environmental Monitoring: Enables predictive alert systems for water pollution.
- Decision Support: Assists regulators in identifying priority sites and times.
- Scientific Insight: Demonstrates ML’s ability to analyze multi-source environmental data.
- Career Growth: Strong portfolio piece combining sustainability and machine learning.

Future Work

1. Time-Series Modeling: Apply LSTM or Prophet for quarterly trend forecasting.
2. Model Generalization: Add external datasets to improve robustness.
3. Hyperparameter Tuning: Use GridSearchCV or Optuna.
4. Deploy as Dashboard: With Streamlit or Dash for interactive prediction.
5. Cross-validation: Improve evaluation beyond a single train-test split.

GitHub Repository: View Full Project on GitHub

Project by Sunny C. Eke