Water Potability Prediction Using Machine Learning

Author: Sohaib Farooq
Email: sohaib.farooq@bigacademy.com
Date: January 2026

Abstract

This report presents a comprehensive machine learning approach to predict water potability based on physicochemical properties. We implement and compare four classification algorithms: Logistic Regression, Random Forest, XGBoost, and LightGBM. The study includes detailed exploratory data analysis, feature engineering, model training, evaluation, and deployment of a web application for real-time predictions.

Table of Contents

1. Introduction

Access to clean drinking water is essential for human health. The World Health Organization (WHO) estimates that contaminated water causes approximately 485,000 diarrheal deaths annually. This project aims to develop a machine learning model capable of predicting whether water is safe for human consumption based on measurable physicochemical parameters.

The objective is to build a classification system that can accurately determine water potability, enabling water quality monitoring systems to make rapid assessments without extensive laboratory testing.

2. Dataset Description

The dataset contains water quality metrics for 3,276 water samples with the following features:

Feature	Description	Unit
pH	Measure of acidity/alkalinity	0-14 scale
Hardness	Capacity of water to precipitate soap	mg/L
Solids	Total dissolved solids (TDS)	ppm
Chloramines	Amount of chloramines	ppm
Sulfate	Amount of sulfate dissolved	mg/L
Conductivity	Electrical conductivity	µS/cm
Organic Carbon	Amount of organic carbon	ppm
Trihalomethanes	Amount of trihalomethanes	µg/L
Turbidity	Measure of light-emitting properties	NTU
Potability	Target variable (0: Not Potable, 1: Potable)	Binary

Missing Values Analysis

The dataset contains missing values that require handling before model training:

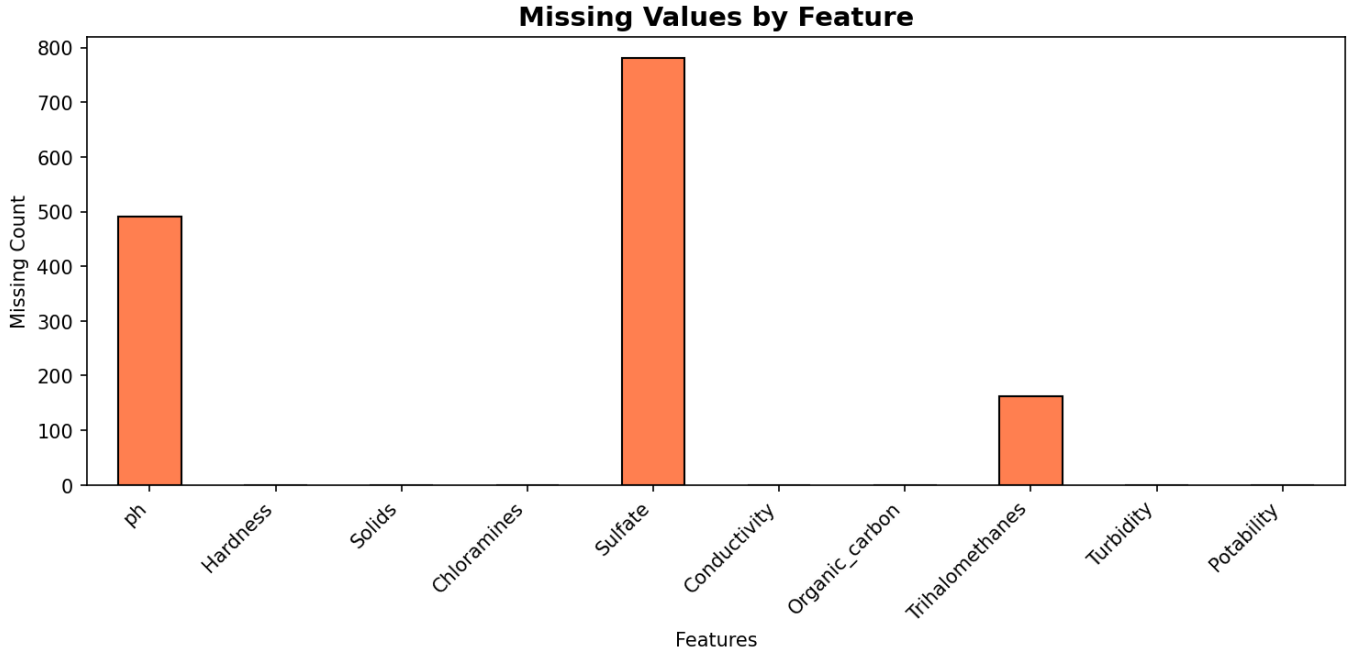


Figure 1: Distribution of missing values across features

---

3. Exploratory Data Analysis

3.1 Target Variable Distribution

The dataset exhibits class imbalance, with approximately 61% non-potable samples and 39% potable samples:

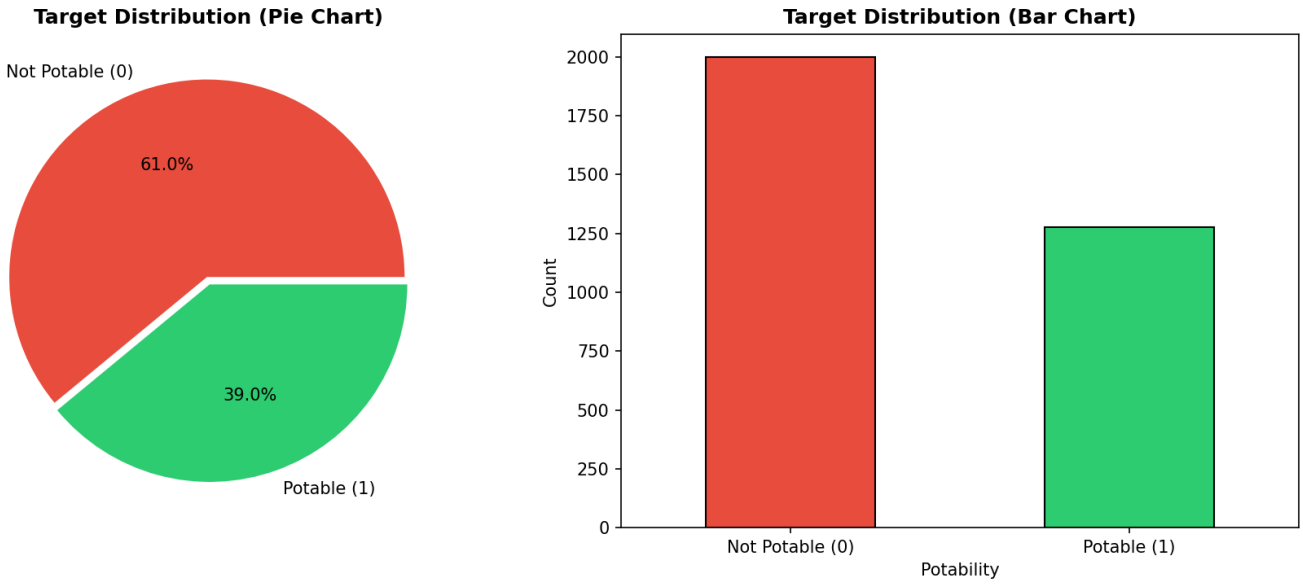


Figure 2: Distribution of water potability classes

3.2 Feature Correlation Analysis

Understanding feature relationships is crucial for feature engineering and model interpretation:

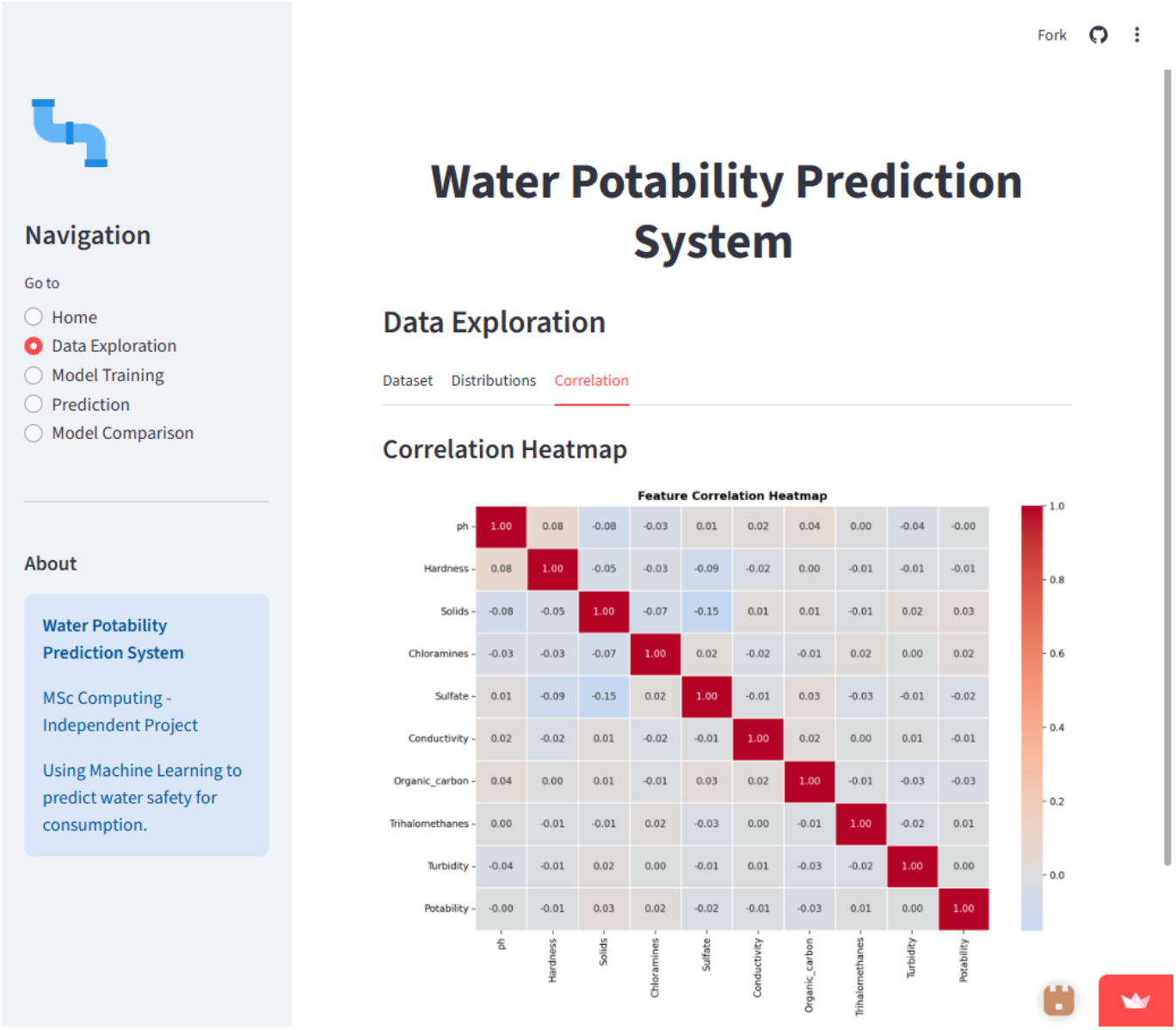


Figure 3: Correlation matrix heatmap showing relationships between features

Key observations:

- Most features show weak correlation with each other
- No strong multicollinearity issues detected
- The target variable has weak correlations with all features, indicating classification complexity

3.3 Feature Distributions

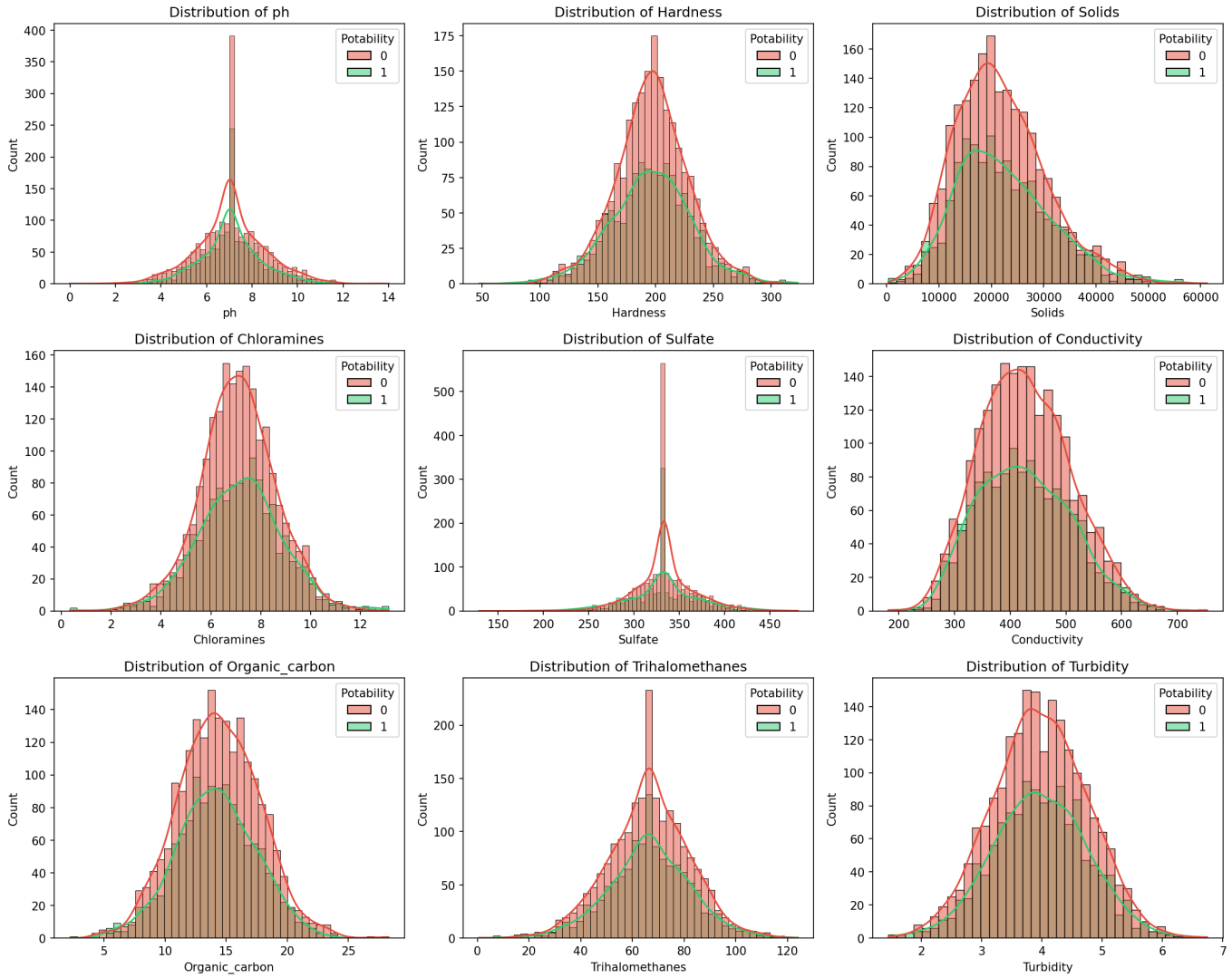


Figure 4: Histograms showing feature distributions by potability class

3.4 Box Plot Analysis

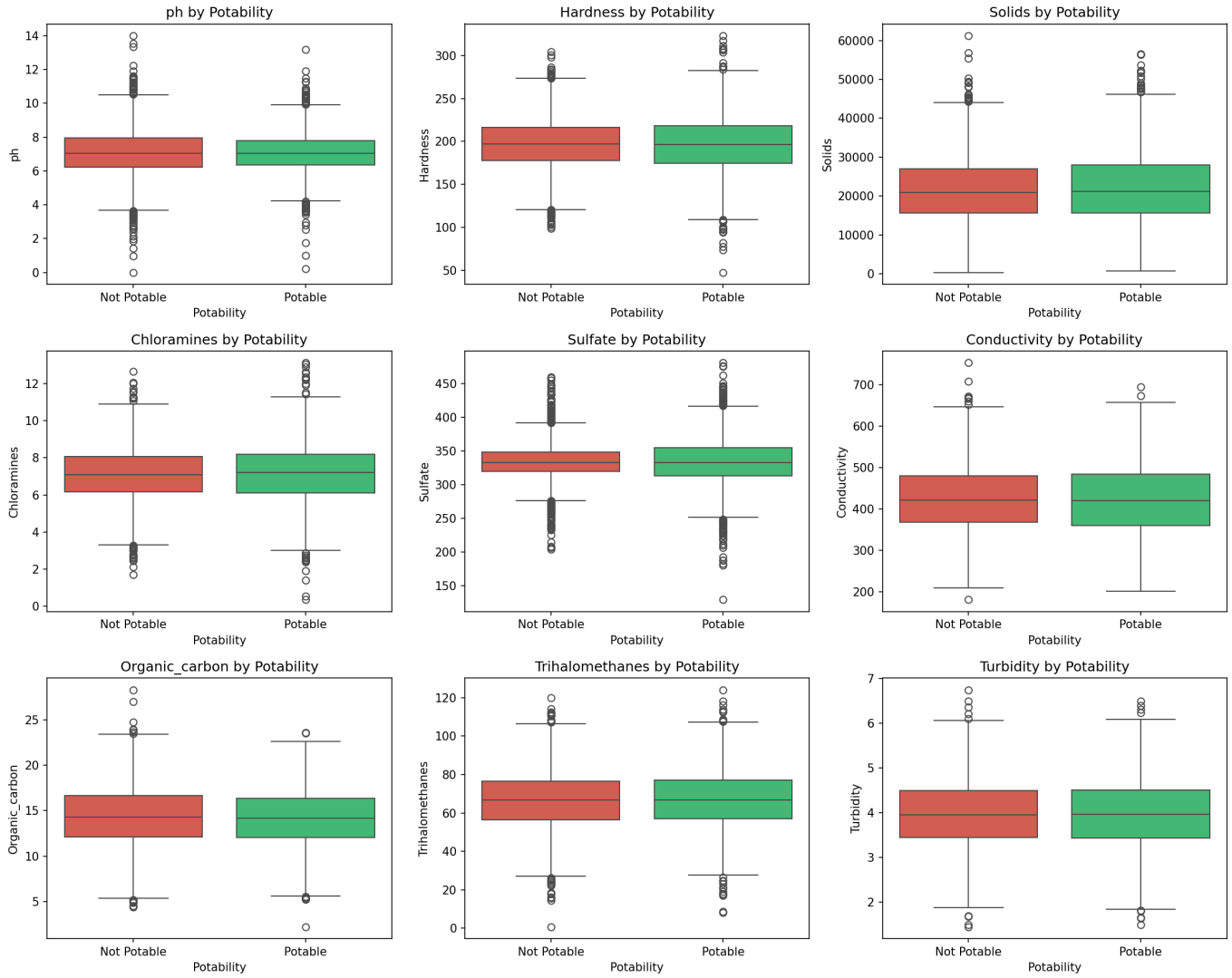


Figure 5: Box plots comparing feature distributions between potable and non-potable water

---

4. Data Preprocessing

4.1 Missing Value Imputation

Missing values were imputed using the median strategy, which is robust to outliers:

$$\tilde{x}*j* = *text{median}(x*{1j}, x_{2j}, \ldots, x_{nj})$$

where $\tilde{x}_j$ is the median value for feature j.

4.2 Feature Scaling

Standard scaling was applied to normalize features:

$$z = \frac{x - \mu}{\sigma}$$

where:

- x is the original feature value
- μ is the mean of the feature
- σ is the standard deviation

- z is the scaled value

4.3 Train-Test Split

The dataset was split using stratified sampling:

- **Training set:** 80% (2,620 samples)
- **Test set:** 20% (656 samples)

---

5. Machine Learning Algorithms

5.1 Logistic Regression

Logistic Regression is a linear classification algorithm that models the probability of binary outcomes using the logistic (sigmoid) function.

Mathematical Formulation

The hypothesis function:

$$h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

where:

- σ is the sigmoid function
- θ represents the model parameters
- x is the input feature vector

Cost Function (Cross-Entropy Loss)

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) \right]$$

where:

- m is the number of training examples
- $y^{(i)}$ is the actual label for sample i
- $h_\theta(x^{(i)})$ is the predicted probability

Gradient Descent Update

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

where α is the learning rate.

Class Weight Balancing

To address class imbalance, we apply balanced class weights:

$$w_c = \frac{n}{k \cdot n_c}$$

where:

- n is the total number of samples
- k is the number of classes
- n_c is the number of samples in class c

5.2 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions.

Algorithm

For $b = 1$ to B (number of trees):

1. Draw a bootstrap sample Z^* of size n from training data
2. Grow a decision tree T_b using recursive partitioning:
    - At each node, select $m \approx \sqrt{p}$ features randomly
    - Find the best split among selected features using Gini impurity
    - Split the node into two child nodes

Gini Impurity

$$G = \sum_{c=1}^{C} p_c(1-p_c) = 1 - \sum_{c=1}^{C} p_c^2$$

where p_c is the proportion of samples belonging to class c at the node.

Information Gain

The best split maximizes information gain:

$$IG(D_p, f) = G(D_p) - \sum_{j \in {left, right}} \frac{n_j}{n_p} G(D_j)$$

where:

- D_p is the parent dataset
- n_p is the number of samples at parent node
- D_j is the child dataset after split

Final Prediction (Majority Voting)

$$\hat{y} = \text{mode}{T_1(x), T_2(x), \ldots, T_B(x)}$$

Feature Importance

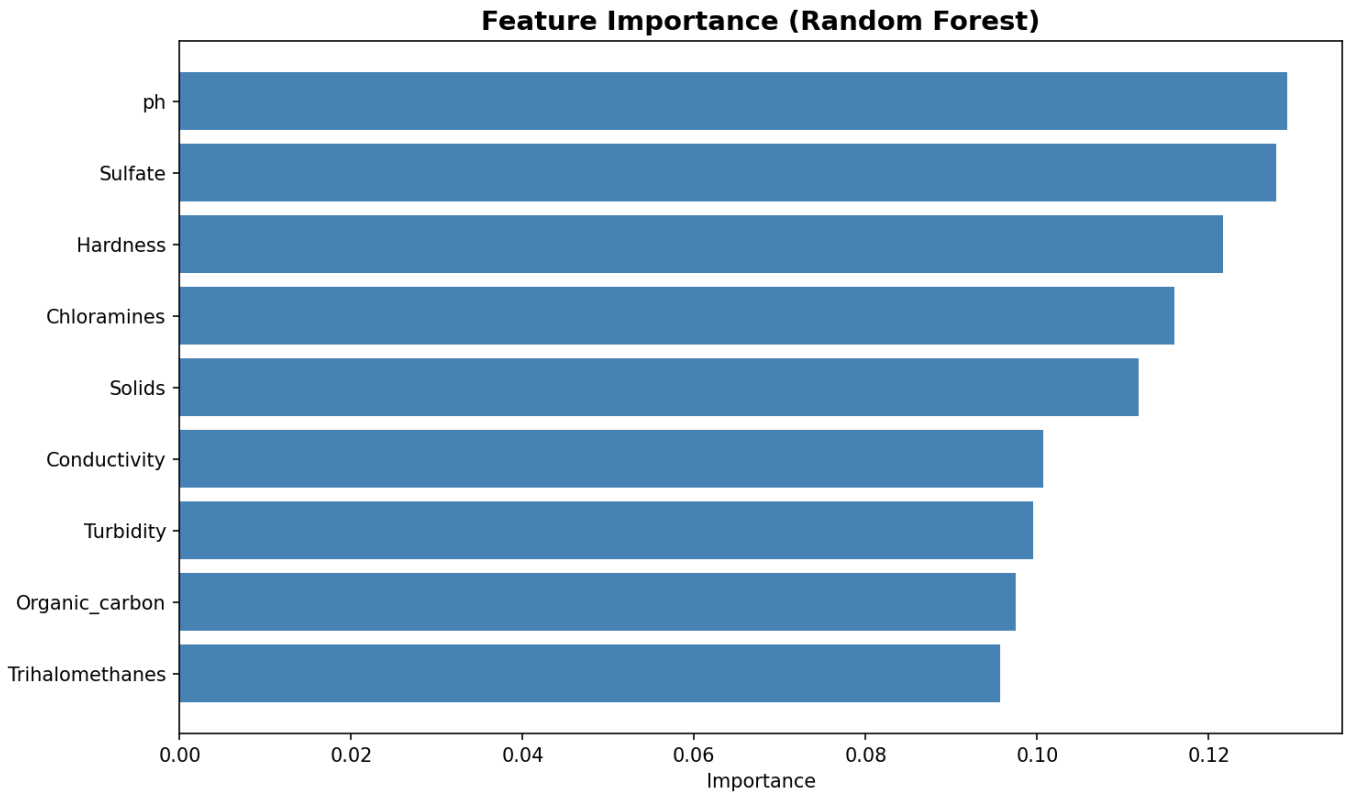


Figure 6: Feature importance scores from Random Forest model

5.3 XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting algorithm with regularization.

Objective Function

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}*i) + *sum*{k=1}^{K} \Omega(f_k)$$

where:

- l is the loss function (logistic loss for classification)
- Ω is the regularization term
- K is the number of trees

Regularization Term

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where:

- T is the number of leaves in the tree
- w_j is the weight of leaf j
- γ and λ are regularization parameters

Second-Order Taylor Expansion

For each iteration t:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

where:

- $g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}$ (gradient)
- $h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial (\hat{y}^{(t-1)})^2}$ (Hessian)

Optimal Leaf Weight

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Split Gain

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

5.4 LightGBM

LightGBM is a gradient boosting framework that uses histogram-based algorithms for efficient training.

Gradient-based One-Side Sampling (GOSS)

GOSS keeps samples with large gradients and randomly samples from small gradient instances:

1. Sort training instances by absolute gradient $|g_i|$
2. Select top $a \times 100\%$ instances with largest gradients
3. Randomly sample $b \times 100\%$ from remaining instances
4. Amplify sampled small gradient data by factor $\frac{1-a}{b}$

Exclusive Feature Bundling (EFB)

EFB bundles mutually exclusive features to reduce dimensionality:

$$\text{Conflict}(f_i, f_j) = \sum_{x \in D} \mathbb{1}[f_i(x) \neq 0 \land f_j(x) \neq 0]$$

Features are bundled if their conflict count is below a threshold.

Leaf-wise Tree Growth

Unlike level-wise growth, LightGBM grows trees leaf-wise:

- Choose the leaf with maximum delta loss
- Can lead to deeper trees and better accuracy
- Uses `max_depth` to prevent overfitting

---

6. Evaluation Metrics

6.1 Confusion Matrix

The confusion matrix provides a complete picture of classification performance:

$$\text{Confusion Matrix} = \begin{bmatrix} TN & FP \ FN & TP \end{bmatrix}$$

where:

- TN = True Negatives (correctly predicted non-potable)
- FP = False Positives (non-potable predicted as potable)
- FN = False Negatives (potable predicted as non-potable)
- TP = True Positives (correctly predicted potable)

6.2 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

6.3 Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision measures the proportion of positive predictions that are actually correct.

6.4 Recall (Sensitivity)

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall measures the proportion of actual positives that are correctly identified.

6.5 F1-Score

The harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

6.6 ROC-AUC

The Area Under the Receiver Operating Characteristic Curve:

$$\text{AUC} = \int_0^1 \text{TPR}(t) , d(\text{FPR}(t))$$

where:

- $\text{TPR} = \frac{TP}{TP + FN}$ (True Positive Rate)
- $\text{FPR} = \frac{FP}{FP + TN}$ (False Positive Rate)

6.7 Average Precision

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are precision and recall at threshold n.

---

7. Results and Analysis

7.1 Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.5259	0.4159	0.5312	0.4666	0.5475
Random Forest	0.6585	0.6311	0.3008	0.4074	0.6407
XGBoost	0.6418	0.5574	0.3984	0.4647	0.6256
LightGBM	0.6540	0.6000	0.3398	0.4339	0.6512

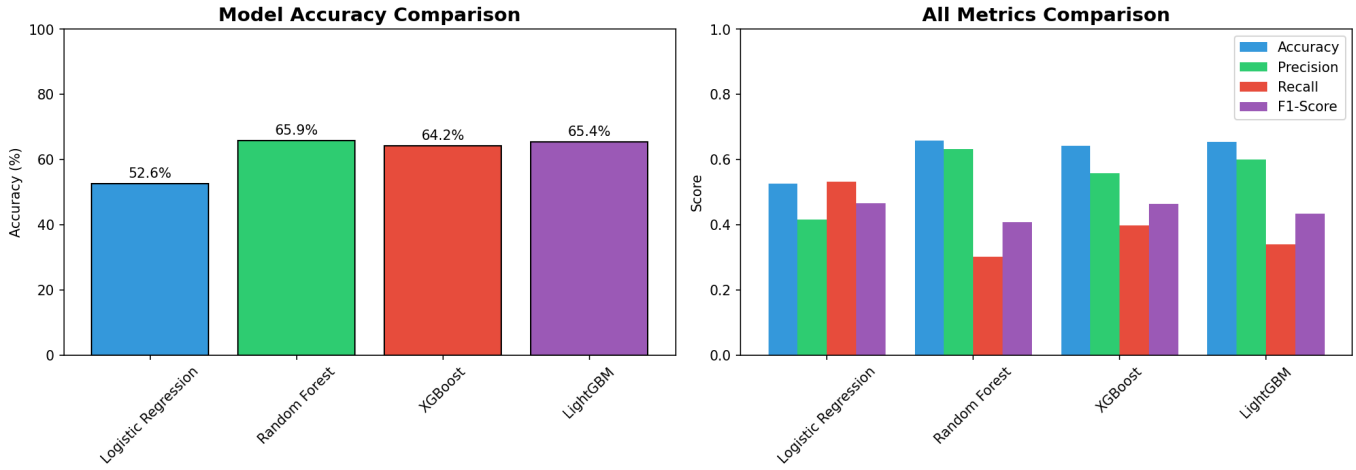


Figure 7: Visual comparison of model performance metrics

7.2 Confusion Matrices

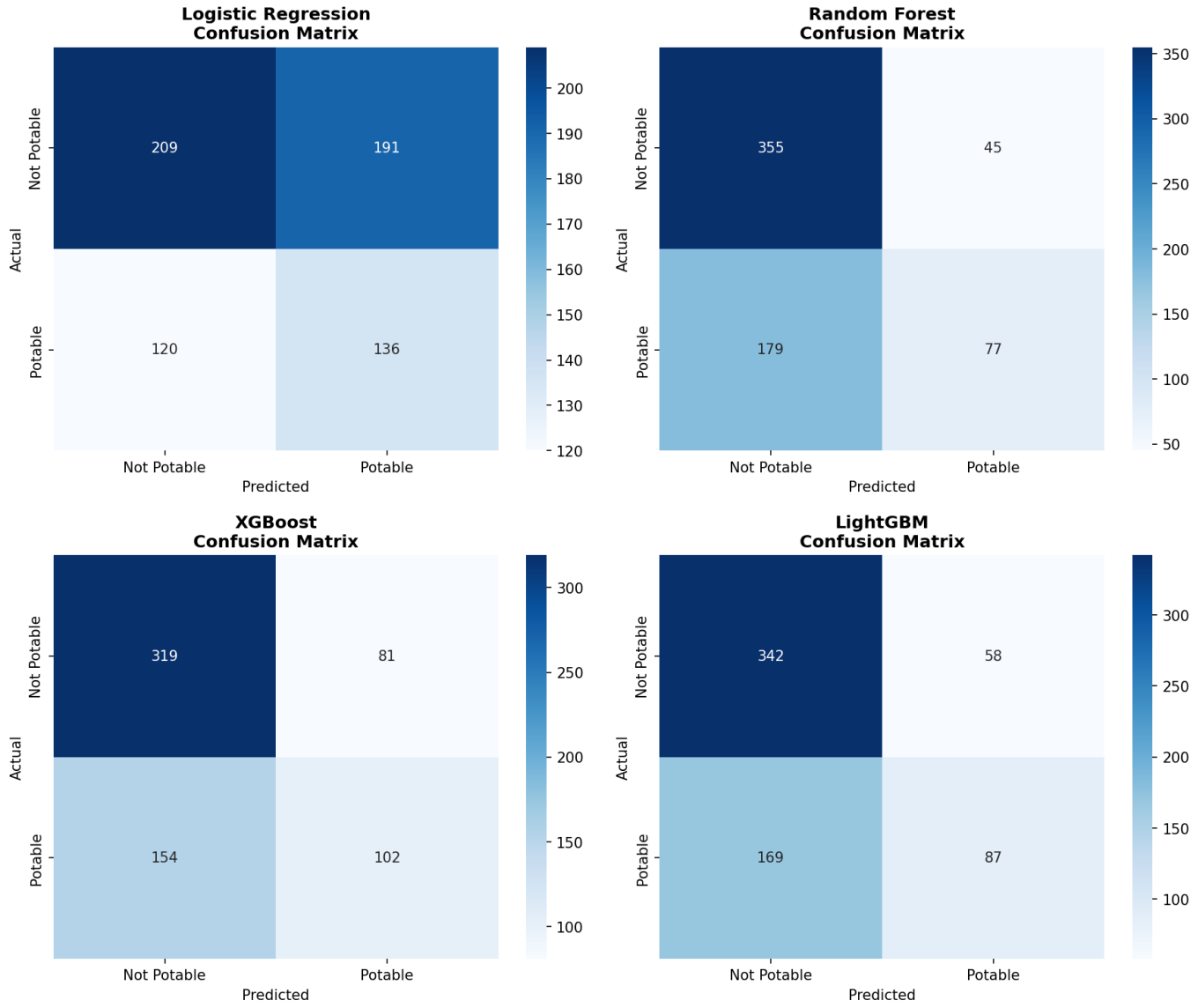


Figure 8: Confusion matrices for all four models

7.3 ROC Curves

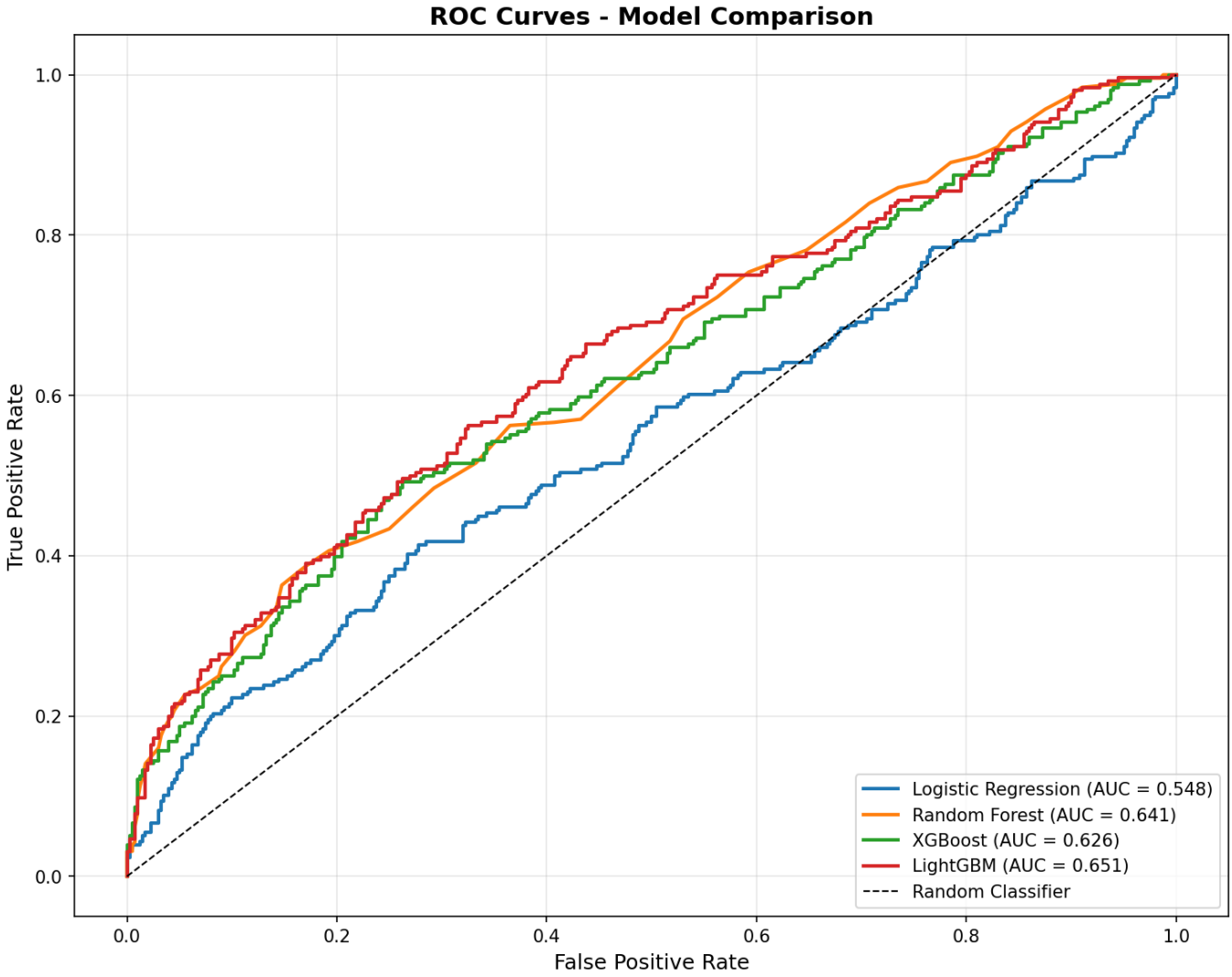


Figure 9: ROC curves comparing model discrimination ability

7.4 Precision-Recall Curves

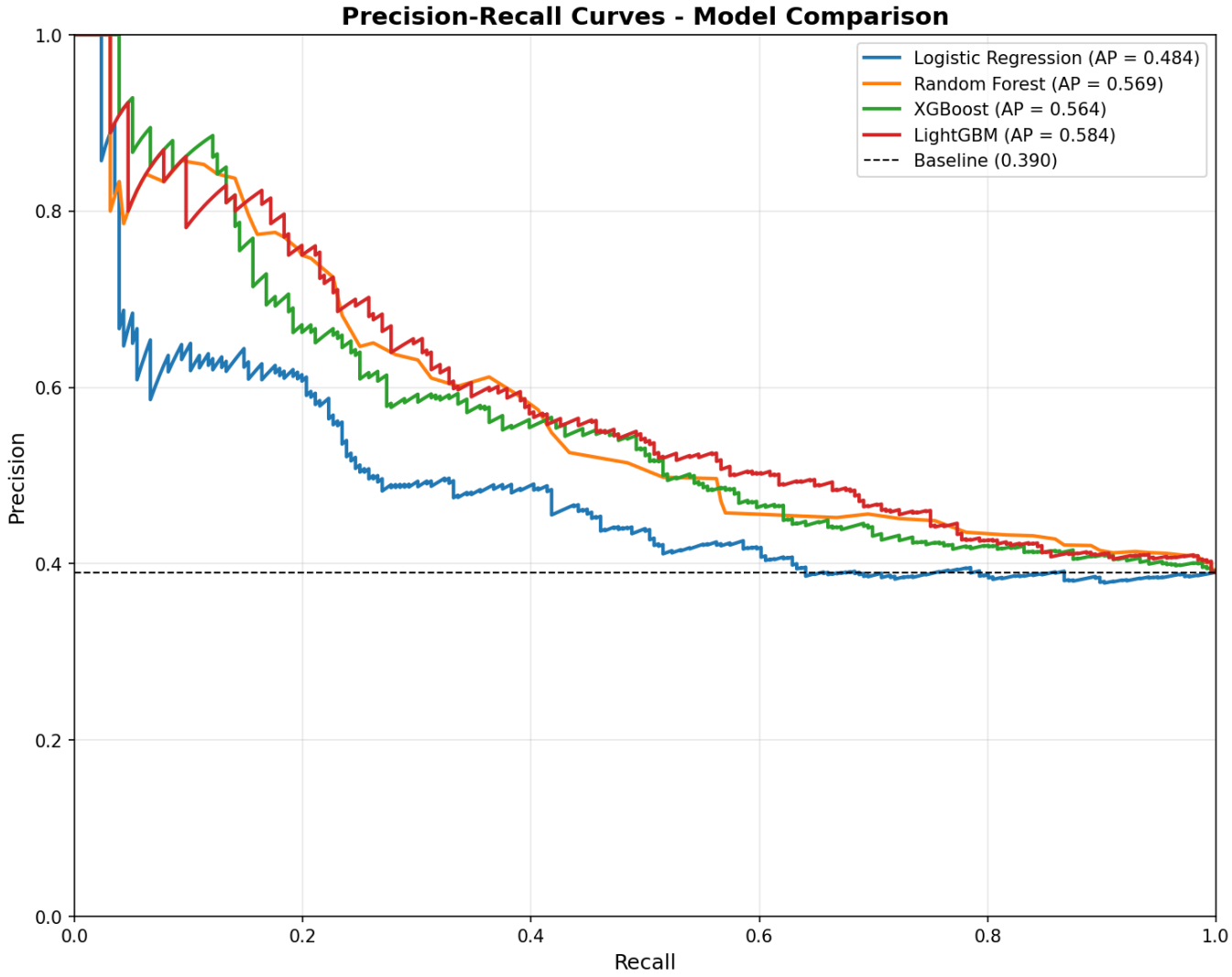


Figure 10: Precision-Recall curves for model comparison

7.5 Key Findings

1. **Best Overall Accuracy:** Random Forest (65.85%)
2. **Best AUC Score:** LightGBM (0.6512)
3. **Best Recall:** Logistic Regression (53.12%) - important for minimizing false negatives
4. **Best Precision:** Random Forest (63.11%)

The class imbalance in the dataset (61% non-potable vs 39% potable) affects model performance. Using balanced class weights in Logistic Regression improved its recall significantly.

---

8. Web Application Deployment

A Streamlit web application was developed and deployed for real-time water potability predictions.

Live Application: https://water-potability-prediction-ip-assignment.streamlit.app/

8.1 Application Home Page

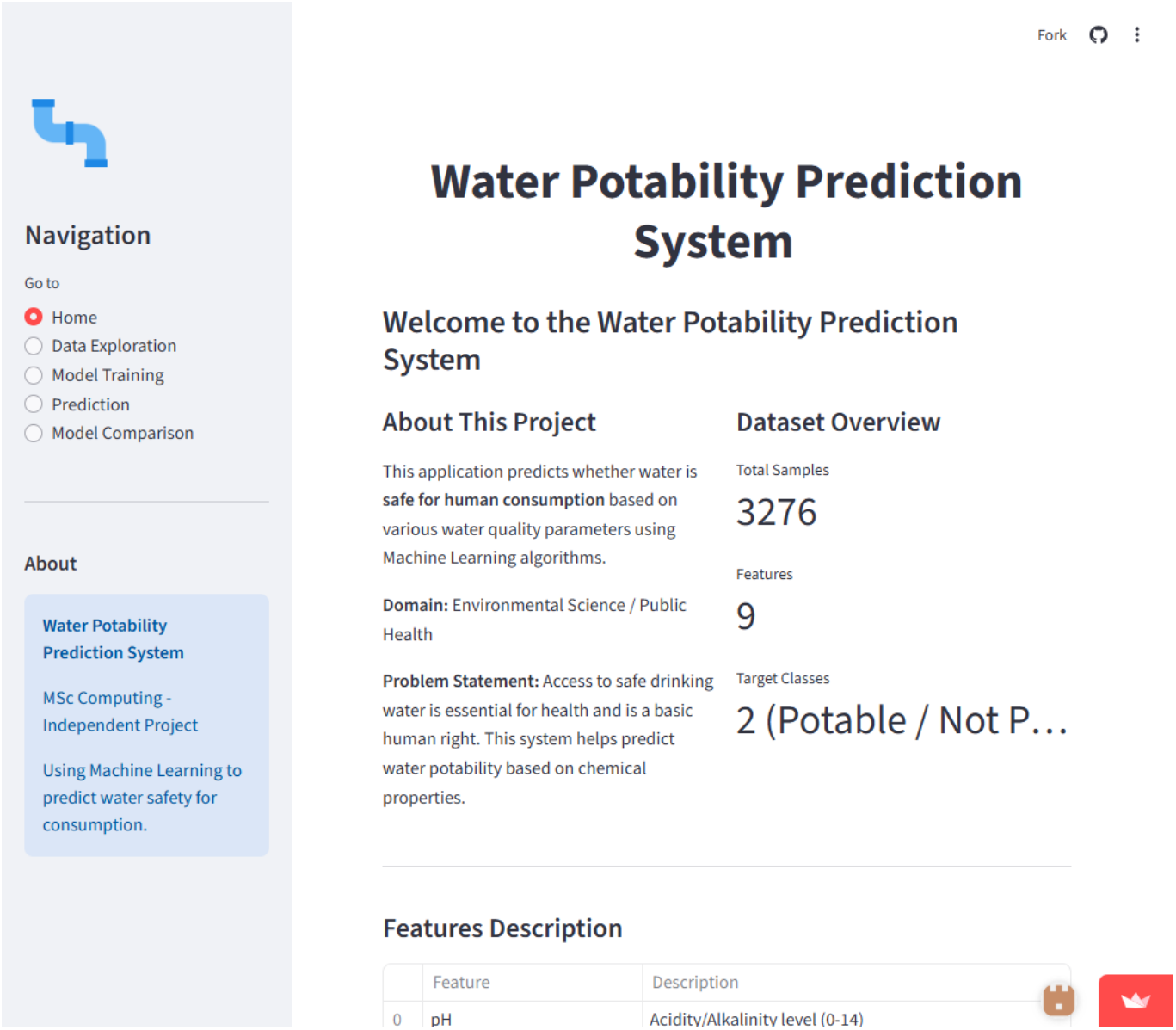


Figure 11: Home page of the Water Potability Prediction application

8.2 Data Exploration Features

The application provides interactive data exploration capabilities:

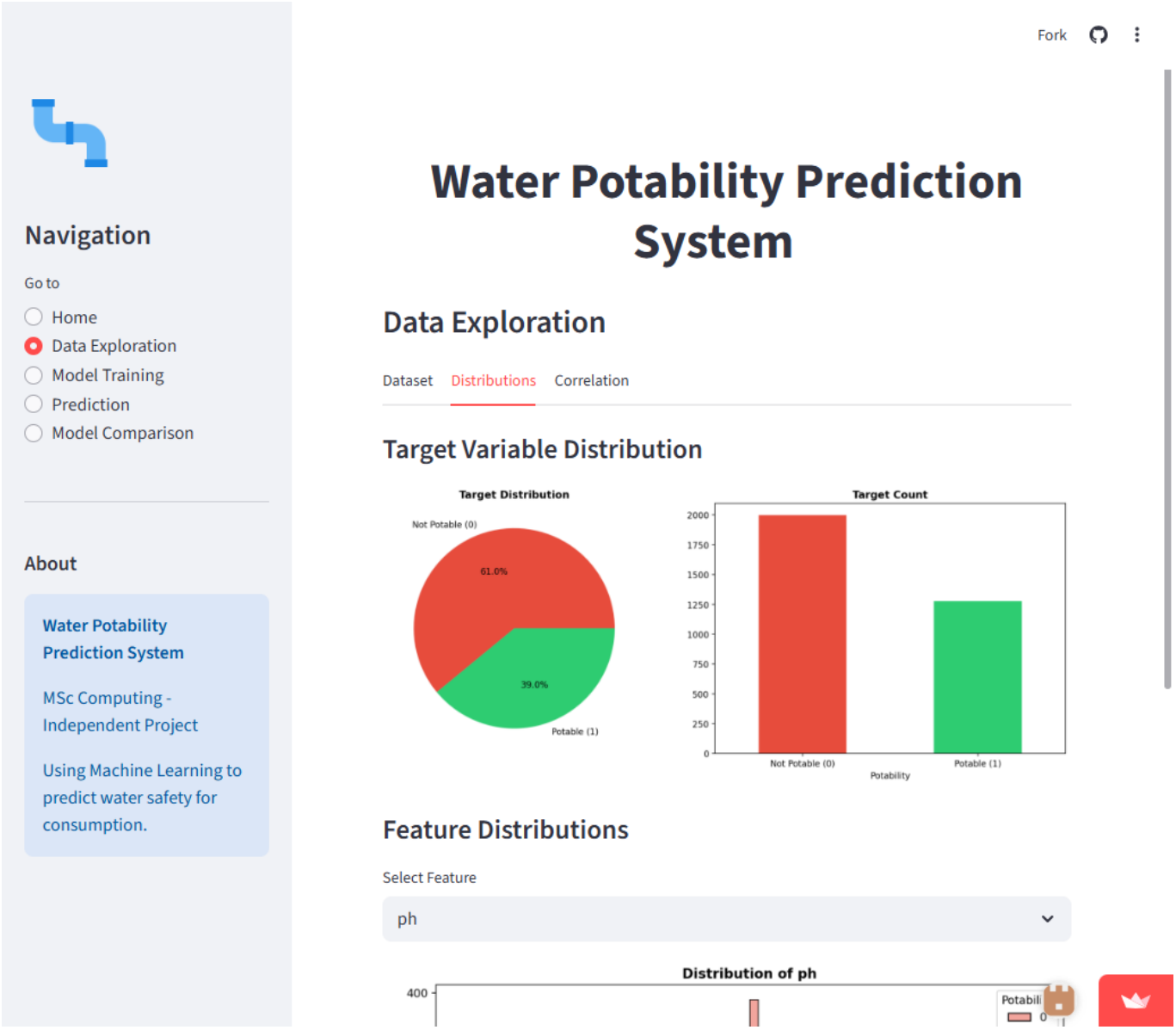


Figure 12: Interactive data distribution visualizations

8.3 Model Training Interface

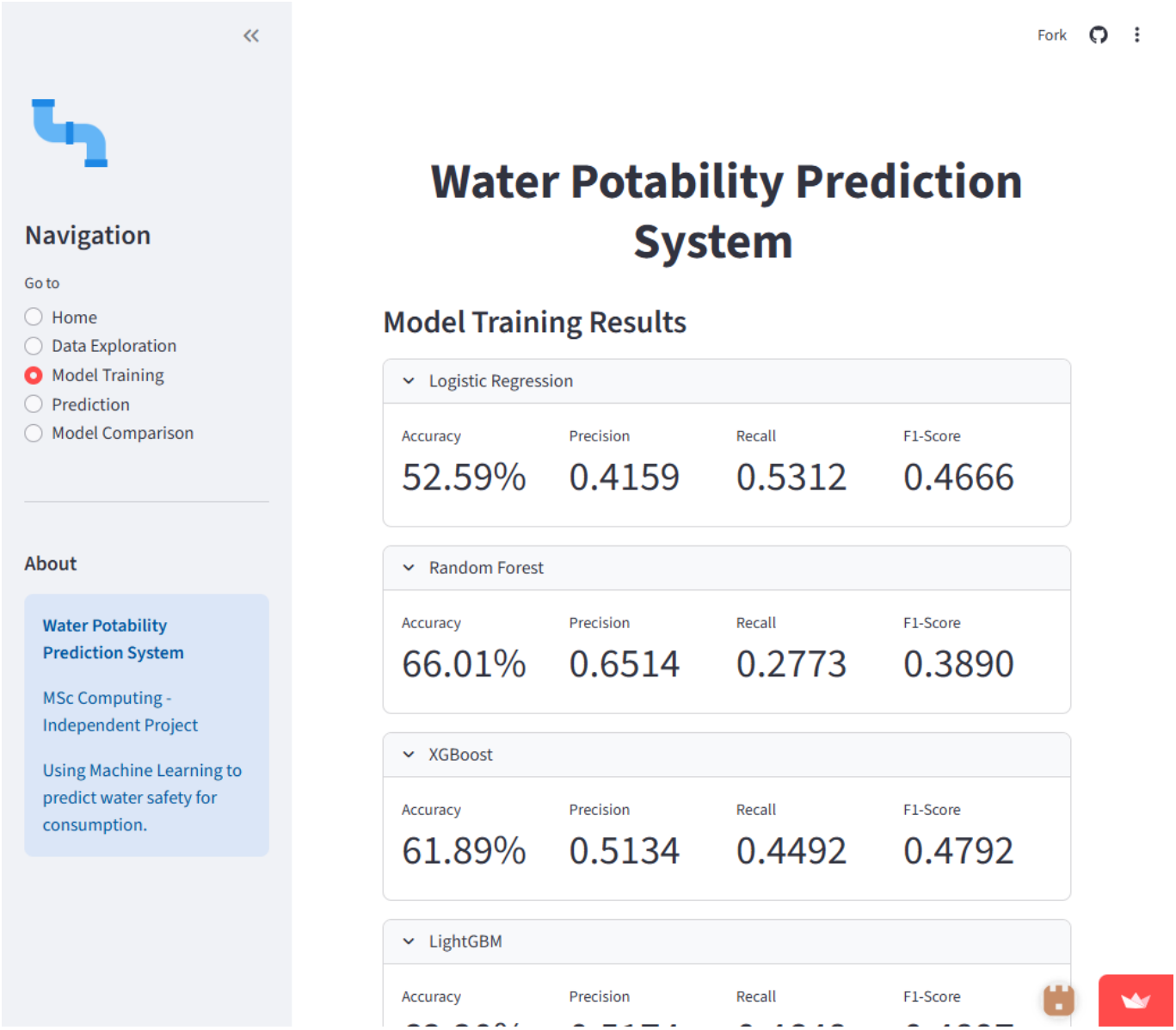
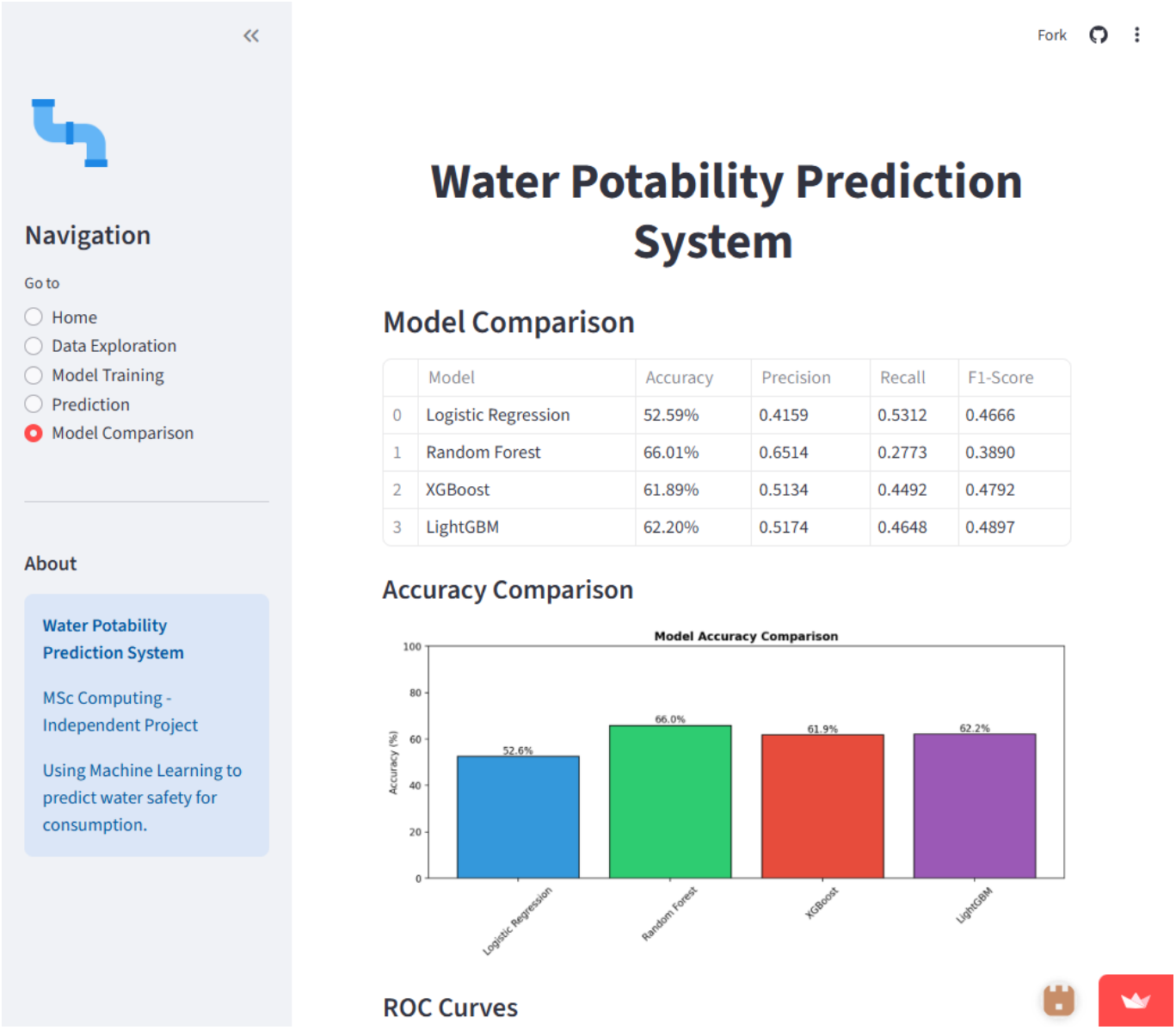


Figure 13: Model training and evaluation interface

8.4 Model Comparison Dashboard



	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	52.59%	0.4159	0.5312	0.4666
1	Random Forest	66.01%	0.6514	0.2773	0.3890
2	XGBoost	61.89%	0.5134	0.4492	0.4792
3	LightGBM	62.20%	0.5174	0.4648	0.4897

Figure 14: Dashboard comparing all trained models

8.5 Prediction Interface

Users can input water quality parameters to get predictions:

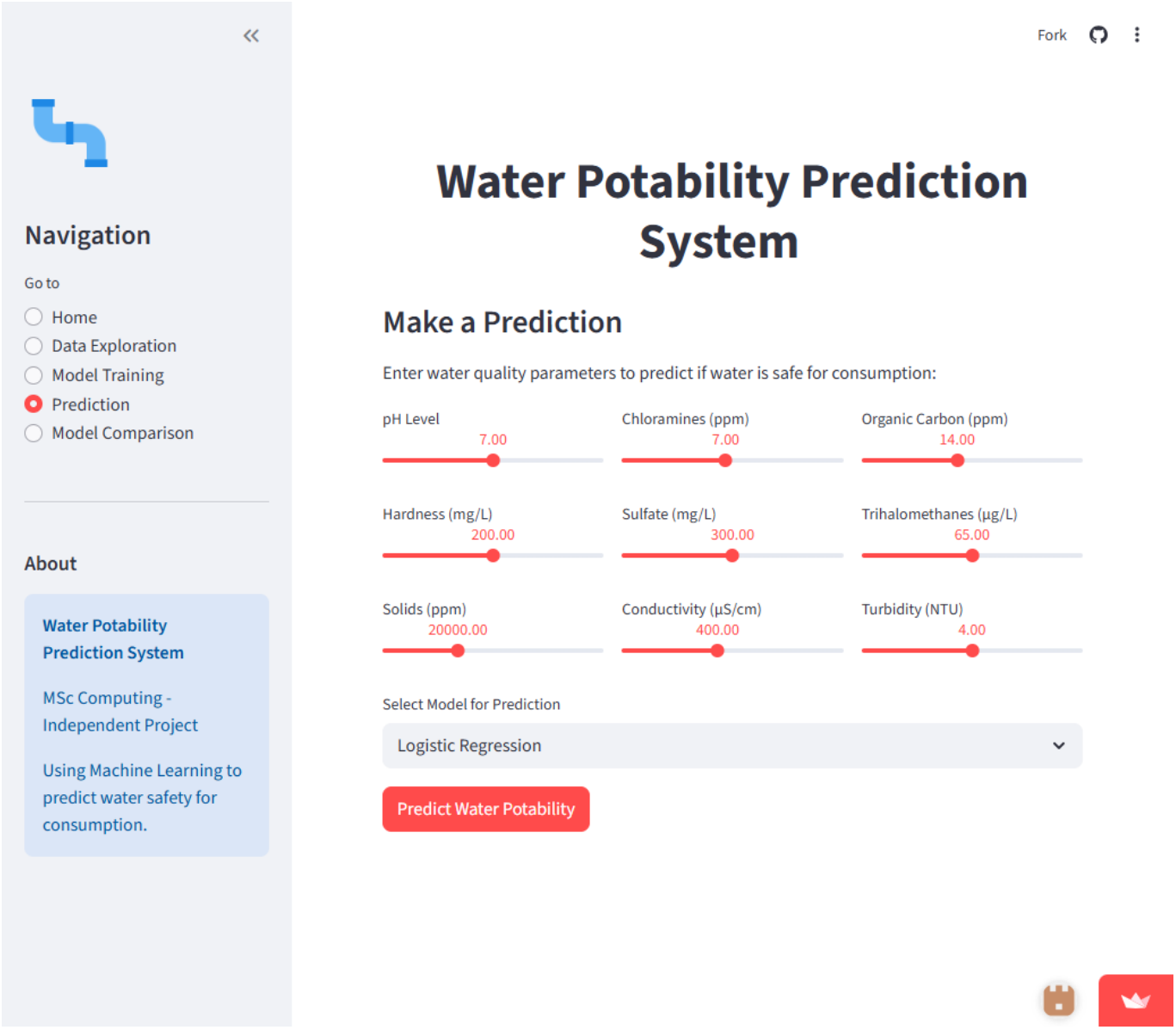


Figure 15: Prediction input interface

8.6 Prediction Results

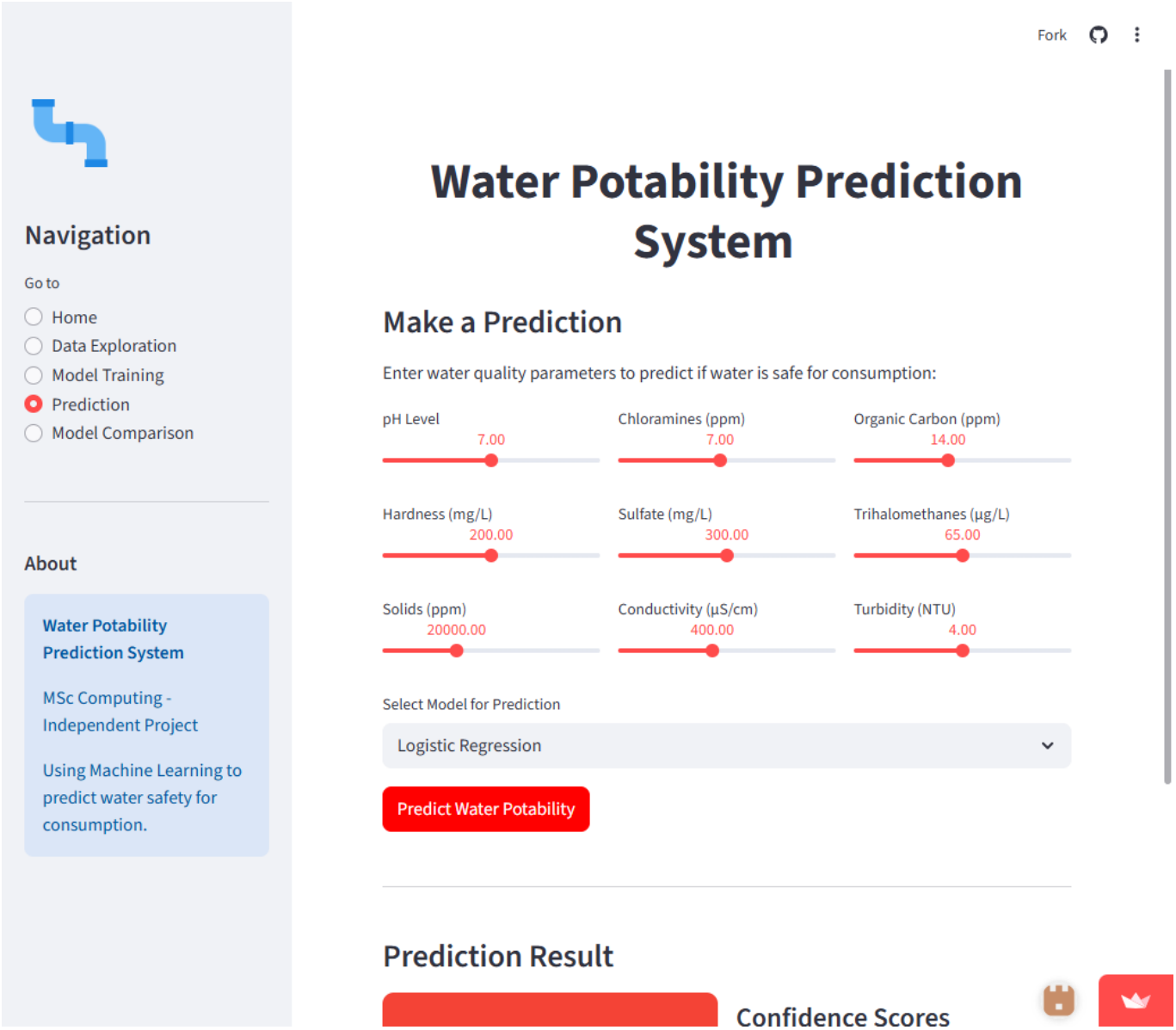


Figure 16: Sample prediction result showing water potability assessment

---

9. Conclusions

9.1 Summary

This project successfully developed and deployed a machine learning system for water potability prediction. Key achievements include:

1. **Comprehensive EDA:** Identified data patterns, missing values, and class imbalance issues
2. **Multiple Model Comparison:** Evaluated four different classification algorithms
3. **Mathematical Foundation:** Provided detailed mathematical formulations for each algorithm
4. **Web Deployment:** Created an accessible web application for real-time predictions

9.2 Recommendations

1. **Collect More Data:** Additional samples, especially potable water samples, would help address class imbalance
2. **Feature Engineering:** Domain-specific features combining multiple water quality parameters

3. **Ensemble Methods:** Combining multiple models could improve prediction accuracy
4. **Threshold Optimization:** Adjusting classification thresholds based on cost-sensitivity analysis

9.3 Future Work

- Implement deep learning approaches (Neural Networks)
- Add time-series analysis for water quality monitoring
- Integrate IoT sensors for real-time data collection
- Develop a mobile application for field testing

---

10. References

1. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.
2. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. KDD '16.
3. Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. NIPS.
4. World Health Organization. (2017). Guidelines for Drinking-water Quality.
5. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression.

GitHub Repository: https://github.com/sohaib3335/water-potability-prediction

---

Report generated as part of MSc Computing Independent Project
Author: Sohaib Farooq | Email: sohaib.farooq@bigacademy.com