Project Report

Machine Learning Approaches for Predicting Water Potability Using Physicochemical Parameters

Course Title: Introduction Machine Learning

Courser Code: EEE 436

Semester: Fall 2025

Section: 01

Submitted By: Group 5

Members Name	**Members ID**
Rahad Ahmed Iram	2022-1-80-038
Mainuddin Rahman Eshan	2022-2-80-022
Niaz Morshed Razon	2022-2-80-008

Submitted to:

Dr. Fakir Mashuque Alamgir

Assistant Professor

Dept. Electrical and Electronic Engineering

East West University

19 December, 2025

Contents

1.1 Introduction & Problem Statement

1.1.1 *Abstract*

Safe drinking water is necessary for human health, and detecting dirty water can prevent many diseases and waste of water. Our easy machine learning system can detect the potability of water by using data of measured quantities of pH, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, and turbidity. Some data values are missing, which are filled by every column's average value. After scaling every feature, Logistic Regression, K-Nearest Neighbors, Random Forest, XG Boost models are trained and tested, and the quality is justified by accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix. The random forest result is better than other models. Its accuracy and F1-score are higher than those of other models. So the possibility of safe drinking water can be predicted well. The feature importance results also show which water measurements affect the decision most, which can help people and organizations improve water testing and management.

1.1.2 *Introduction*

Safe drinking water quality justification is necessary to ensure safe drinking water and prevent waterborne diseases. Examine water in the lab is very time-consuming and costly, and everywhere, every time, it is hard to use. This project was built with a dataset that labeled the potability and non-potability of water [1]. This Dataset contains 3276 samples and 10 features. that is pH, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, and turbidity. Some feature values are missing, which are filled for data stability before training the model. Then, Logistic Regression, K-Nearest Neighbors, Random Forest, and XG Boost models were used. All these models' performance is justified by accuracy, precision, recall, F1-score, and ROC-AUC to analyze the process of quality detection of water. Finally, feature importance is studied to find which measurements affect the decision most, so that people can focus on the key water tests when planning monitoring and management.

1.1.3 *Problem Statement*

Safe drinking water is essential, but regular examination of water in a lab is very time-consuming and costly. To examine the water potential in a smart way is necessary. The available water quality dataset has numeric features and some values that make it hard to examine the water quality and are untrustworthy. To examine the large number of water samples and the quick taking step for risk is tough due to the unavailability of a good predictive model. This project aims to build and compare machine learning models that can accurately predict water potability and highlight the most important water quality parameters.

1.2 Background & Related Work

1.2.1 *Study 1: Water Potability Prediction Using Machine Learning 2023*

Ensuring the availability of safe drinking water remains a significant global challenge due to the increasing impact of urbanization, industrialization, and environmental degradation. To show this problem, Patel et al. [2] (2023) conducted an extensive study on the application of ML methods to predict the potability of drinking water [2]. This study is widely related to the present work. And it uses a drinking water dataset collected from Kaggle and whose structure and physicochemical parameters are consistent with the Water Potability dataset used in this report.

The dataset used in this study includes a total of 3,276 water samples. Here each sample is described by nine important physicochemical properties. These properties involve pH, hardness, total dissolved solids, chloramines, sulfate concentration, trihalomethanes, organic carbon, conductivity, and turbidity. These parameters are widely recognized as significant indicators for determining the safety of drinking water as recommended by various international organizations which include the World Health Organization (WHO). Here, the target variable is 'potability', a binary variable that indicates whether a water sample is fit for human consumption or not.

To creat an effective prediction system, the researchers used multiple supervised machine learning algorithms, including Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, and Extreme Gradient Boosting

(XG Boost). Before training the model, the important preprocessing steps are performed on the dataset, such as missing value management, normalization, feature analysis, and splitting the data into training and testing sets. Numerous widely used performance indicators including accuracy, precision, recall and F1-score were used to evaluate the performance of the models.

1.2.2 *Study 2: Water Quality Prediction using Machine Learning 2024*

With a focus on environmental monitoring and public health, Vidyashree and Vishnujnath (2024) analyzed their other relevant study on the prediction of overall water quality using machine learning algorithms [3]. Although the main focus of this study is not only on the potability of drinking water, but also on water quality, its methodology and results are quite applicable to the problem of classification of drinking water.

The writer highlights the limitations of standard water quality assessment methods, which frequently rely on expensive and time-consuming laboratory tests. These methods are not ideal for massive or live monitoring. To show these problems, the study proposes a machine learning-based framework that is able to effectively predict water quality by combining chemical and physical properties.

Various machine learning models were used in the study, like: Support Vector Machine (SVM), Random Forest, and Gradient Boosting. The research dataset included many water quality indicators, including pH, turbidity, total dissolved solids, conductivity, and various chemical concentrations. This study is mainly important because these indicators are in theory steady with those used in the Water Potability dataset on Kaggle.

From the view , metrics such as accuracy, precision, recall, and F1- score were used to assess the performance of the model which included data preparation, feature selection through correlation analysis, model training, and performance. The dataset was divided into two parts which are training and testing. These sections to ensure a fair and unbiased assessment of the model’s performance.

The results showed that the gradient boosting model surpassed both Random Forest and SVM, with a max prediction accuracy of about 85%. These results further show the effectiveness of creating learning methods with complex environmental datasets. The study also highlights the significance of feature selection and hyperparameter tuning to improve model accuracy.

The analysis of the actual implementation of the study is also an important contribution. In line with modern smart water management technologies, the authors propose that machine learning models can be connected to web-based interfaces and real-time monitoring systems. This method can greatly enhance early water pollution detection and rapid decision-making.

The study marks that feature diversity, dataset size, and data quality have a major impact on prediction accuracy. The authors proposed possible research directions which include the addition of IoT sensors and advanced deep learning methods to further improve performance in the future.

1.2.3 *Relevance to the Present Study*

Both results clearly show that machine learning provides a scalable and efficient approach for predicting water quality and potability using physicochemical indices. In particular, ensemble techniques such as Random Forest and XG Boost consistently outperform conventional classifiers [4]. These results influenced the model selection and experimental design of the present work, where various machine learning algorithms were applied to the Kaggle Water Potability dataset to assess drinking water safety.

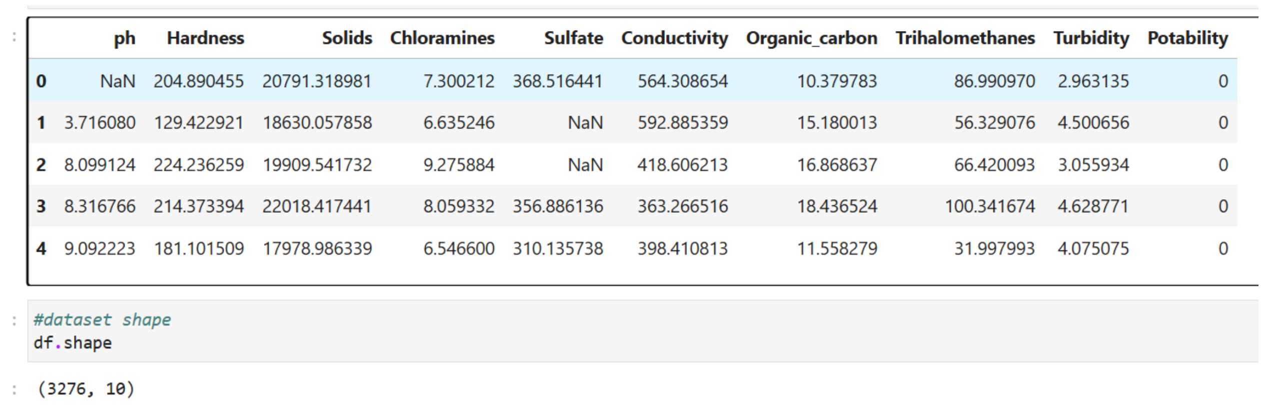
1.3 Dataset Description & Preprocessing

1.3.1 *Dataset Source and Basic Information*

The goal of this project is very much dependable on the quality of the dataset. The reliable prediction of a model can be achieved only when we have a well-prepared dataset. So, we have search across the internet and to came up with a well structure and informative dataset. The information of the dataset is discussed below. We worked with the Water Potability dataset from Kaggl.

Dataset: Water Quality
Source: Kaggle
Author: Aditya Kadiwal
License: CC0

We worked with the Water Potability dataset from Kaggl. Since it’s under a CC0 Public Domain license, we can use and share it freely.



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

```
#dataset shape
df.shape
```

```
(3276, 10)
```

The dataset contains 3276 water samples and 10 columns. It has nine features that describe the chemical properties of water and one target column called **Potability**.

1.3.2 *Meaning of Each Feature (in simple words)*

We explained each feature based on what we learned from WHO guidelines and our basic chemistry knowledge:

1. **ph**: measures how acidic or alkaline the water is (ideal range 6.5–8.5)
2. **Hardness**: amount of calcium and magnesium (mg/L); very high hardness makes water “hard”
3. **Solids**: total dissolved solids or TDS (mg/L); usually below 500 is considered good
4. **Chloramines**: disinfectant added to kill bacteria (mg/L); safe up to about 4 mg/L
5. **Sulfate**: natural salt present in water (mg/L)

6. **Conductivity**: how well electricity passes through water (µS/cm); shows the amount of ions

7. **Organic carbon**: carbon coming from plants or pollution (ppm)

8. **Trihalomethanes**: by-products formed when chlorine reacts with organic matter (µg/L)

9. **Turbidity**: how cloudy the water is (NTU); clear water should be less than 5 NTU

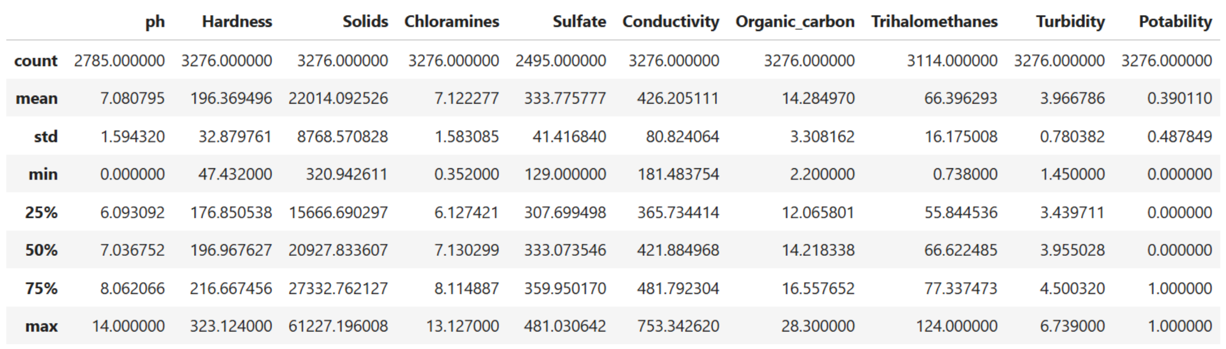
10. **Potability**: our target: 1 = potable (safe), 0 = not potable

1.3.3 *Ethical and Privacy Considerations*

There's no personal info in the data no names, locations, or anything that could identify someone. It looks either man made or very anonymous because privacy isn't a concern. Still, we can see that in real life, a wrong prediction (like saying unsafe water is safe) could hurt people, especially in places where clean water is already hard to get. That's why we have to pay extra attention to false negatives when picking our metrics.

1.3.4 *Exploratory Data Analysis (EDA)*

Descriptive Statistics Data:



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

Figure 1: Descriptive Statistics of Water Quality Parameters

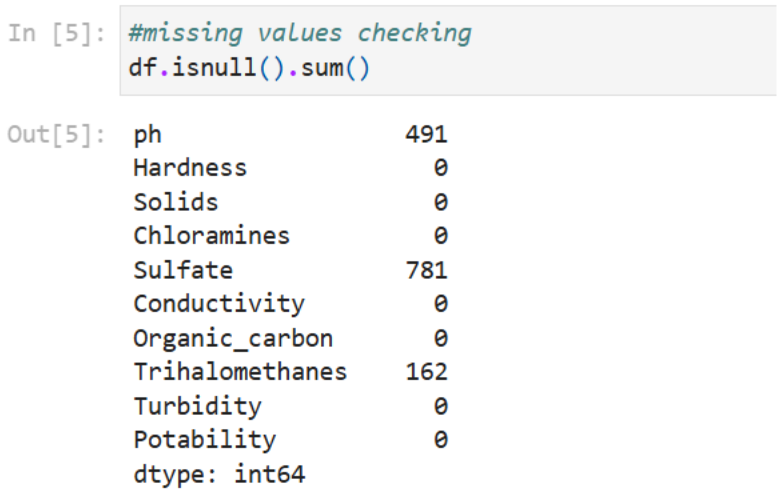
This figure shows a summary of all the features in the Water Potability dataset. For every variable, it lists the count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum.

There are 3,276 samples in the dataset. Many features, such as pH, Sulfate, and Trihalomethanes, have missing values. The average pH is about 7.08. That means the water is generally neutral. Solids (TDS) and Conductivity have a lot of variation, as shown by their wide ranges and high standard deviations.

The Potability column is uneven. About 39% of the samples are labeled as potable, and 61% are non-potable. This inequality was taken into account when training and evaluating the model.

Overall, this summary provides a foundation. This summary gives a basic overview of the dataset and helps to make decisions about preprocessing, like how to handle missing values and whether to scale the data.

Missing Values



```
In [5]: #missing values checking
        df.isnull().sum()

Out[5]: ph                 491
        Hardness             0
        Solids               0
        Chloramines          0
        Sulfate            781
        Conductivity         0
        Organic_carbon       0
        Trihalomethanes    162
        Turbidity            0
        Potability           0
        dtype: int64
```

Figure 2: Missing Values

The check the missing values and found: there are three columns with missing values ph (491), sulfate (781) and trihalomethane (162). So, in total, there are about 1434 missing values. All other columns are complete.

Class Distribution

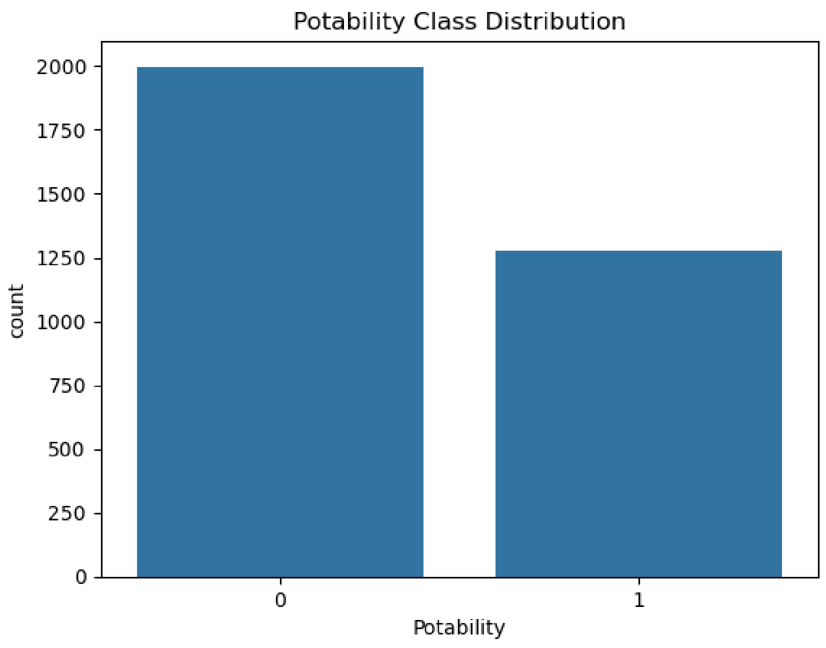


Figure 3: Potability Class Distribution

Figure 1 (Potability Class Distribution) clearly shows imbalance:

Not potable (0): around 2000 samples

Potable (1): only about 1276 samples

About 61% of the samples are class 0, and 39% are class 1. This imbalance worried us because models might just predict 0 most of the time.

Distribution of Features

We plotted histograms for all nine features. (Figure 4: Histogram of Features). Most of them look roughly bell-shaped (normal)

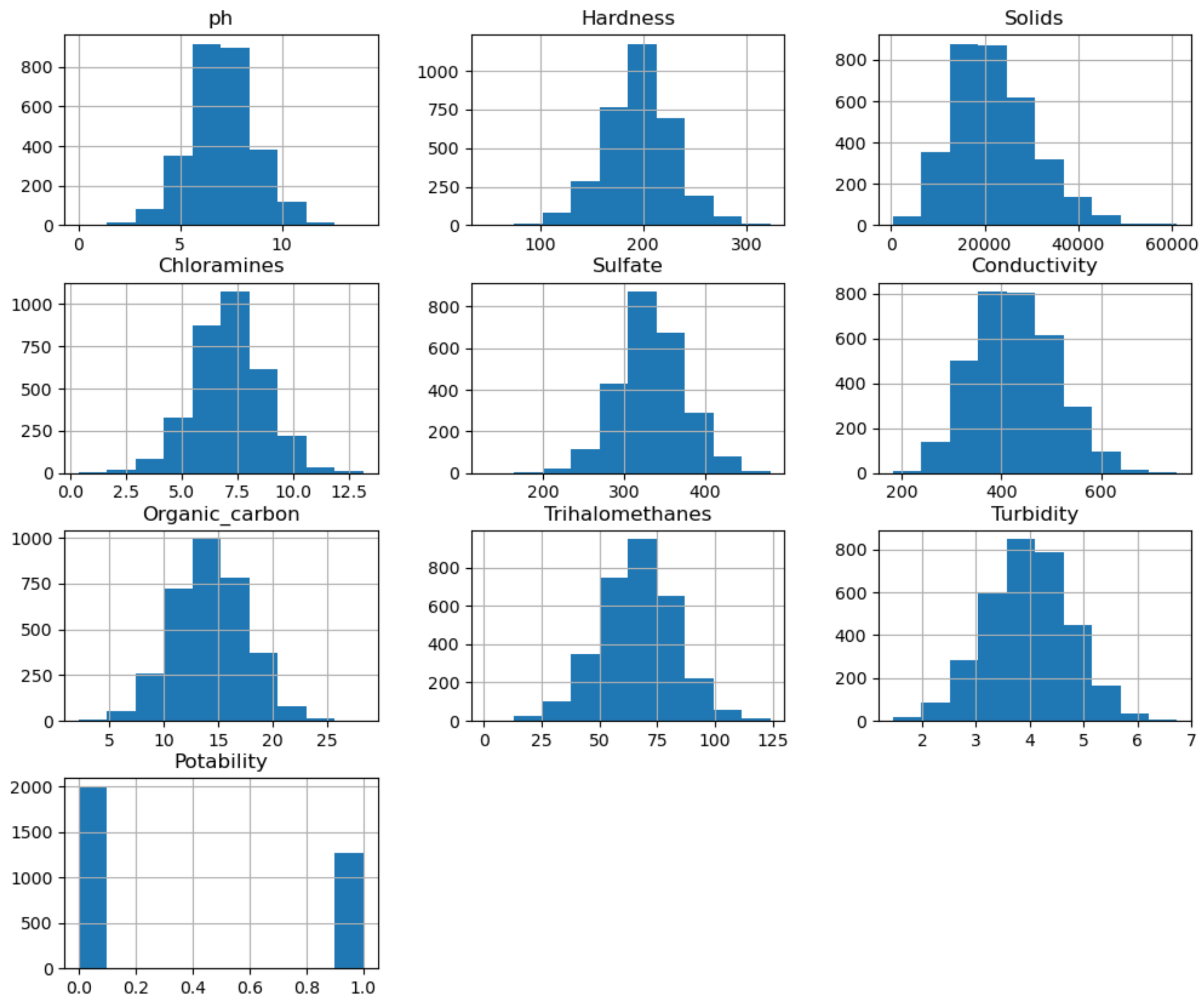
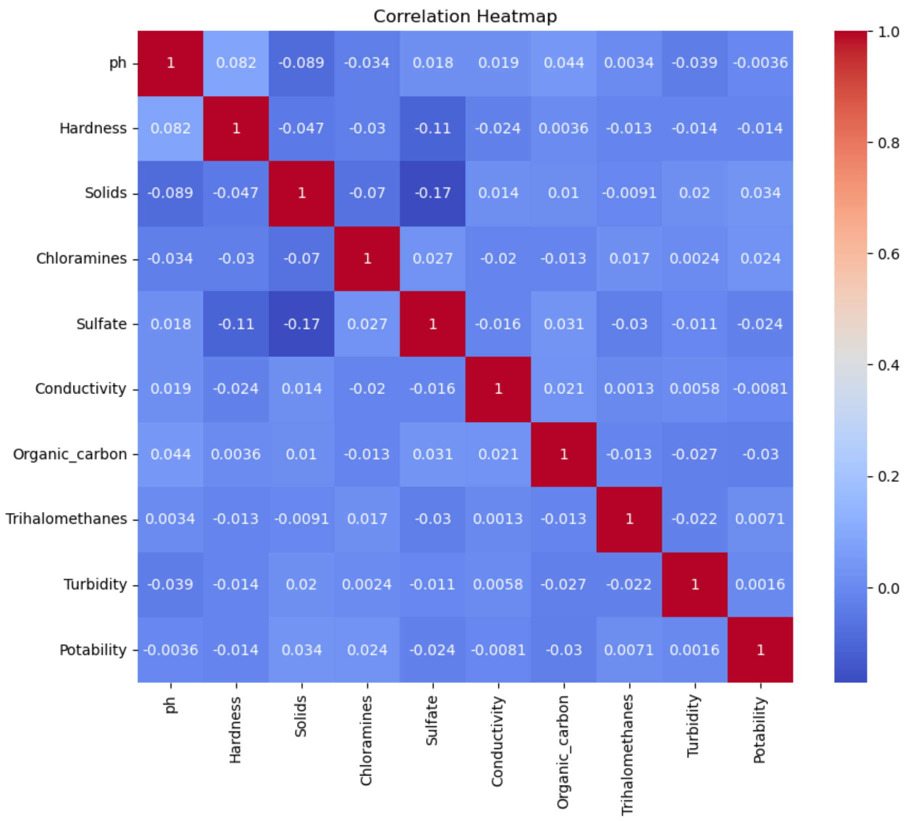


Figure 4: Histogram of Features

- Solids is heavily right-skewed (some samples have very high TDS)
- Turbidity also has a tail on the right side
- pH, Hardness, Chloramines, etc., are quite symmetric

Correlation Heatmap



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1	0.082	-0.089	-0.034	0.018	0.019	0.044	0.0034	-0.039	-0.0036
Hardness	0.082	1	-0.047	-0.03	-0.11	-0.024	0.0036	-0.013	-0.014	-0.014
Solids	-0.089	-0.047	1	-0.07	-0.17	0.014	0.01	-0.0091	0.02	0.034
Chloramines	-0.034	-0.03	-0.07	1	0.027	-0.02	-0.013	0.017	0.0024	0.024
Sulfate	0.018	-0.11	-0.17	0.027	1	-0.016	0.031	-0.03	-0.011	-0.024
Conductivity	0.019	-0.024	0.014	-0.02	-0.016	1	0.021	0.0013	0.0058	-0.0081
Organic_carbon	0.044	0.0036	0.01	-0.013	0.031	0.021	1	-0.013	-0.027	-0.03
Trihalomethanes	0.0034	-0.013	-0.0091	0.017	-0.03	0.0013	-0.013	1	-0.022	0.0071
Turbidity	-0.039	-0.014	0.02	0.0024	-0.011	0.0058	-0.027	-0.022	1	0.0016
Potability	-0.0036	-0.014	0.034	0.024	-0.024	-0.0081	-0.03	0.0071	0.0016	1

Figure 5: Correlation Heatmap

We made a correlation heatmap (Figure 3). Almost everything is only weakly correlated with Potability where all values are below ±0.1. Even between the features, nothing varied over ±0.3 but there's no strong linear relationship, which is probably why simple models don't get high accuracy here.

Key Observations

- The dataset has 3276 rows, which is decent for a student project but a bit small for really complex models.
- Only three columns have missing values. Using the median to fill them in seems safe, especially since there are outliers.
- There's a clear class imbalance (61:39), so we'll need to use class weights or oversampling later.
- Most features are normally distributed, but Solids and Turbidity are skewed.

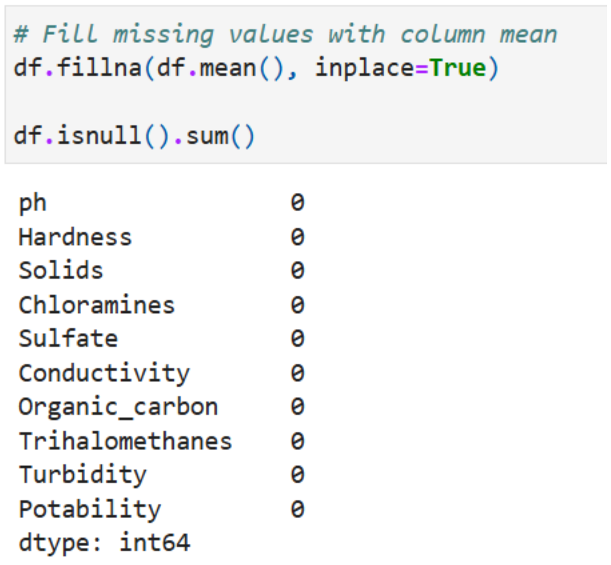
- There’s a very weak correlation with the target, so tree-based models or ensembles will probably work better than linear ones.
- There’s no multicollinearity among the features, so we can keep all nine without dropping any of the features. The data is clean but definitely needs careful handling of missing values and imbalance.

1.4 Methodology

1.4.1 *Data Processing*

1.4.1.1 Filling the Values

After completing the EDA, we proceeded to clean the data. We addressed missing values by applying median imputation, as some features such as Solids contained outliers. Simple Imputer from scikit-learn with strategy set to 'mean' was used on the entire dataset.



```
# Fill missing values with column mean
df.fillna(df.mean(), inplace=True)

df.isnull().sum()
```

```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

Figure 6: After filling the missing values

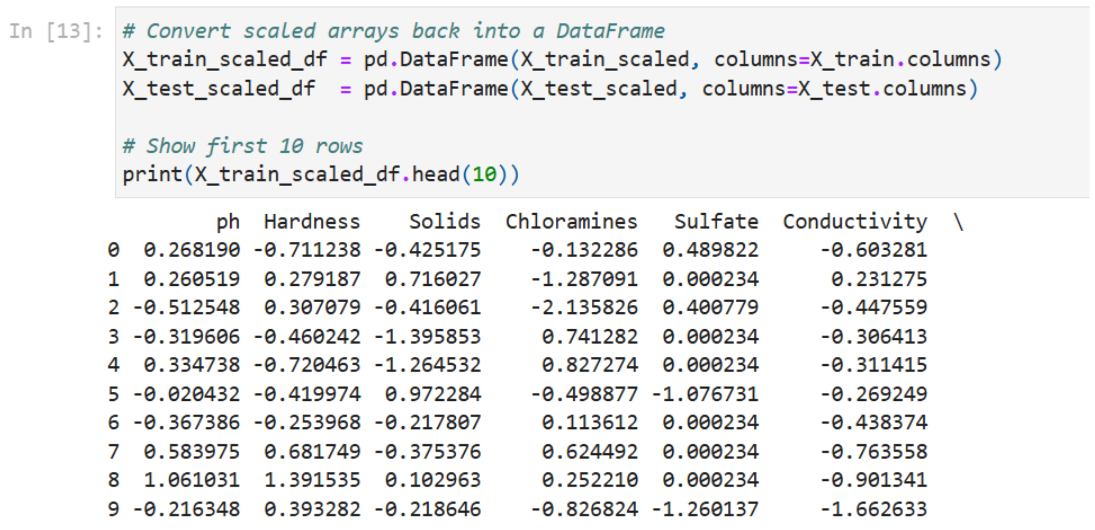
1.4.1.2 Identifying Feature and Target

We have separated the features (X) and target (y = Potability). No new features were added this time, so we kept the original nine columns. All features are numerical, so no encoding needed.

1.4.1.3 *Feature Scaling*

Earlier we have noticed that the features have different range of values. Some features like Solid, Hardness, sulfate and conductivity have very high range of values, while the other features have lower range values. And we know that Logistic Regression and KNN models doesn't perform well with different with different range of data.

Therefore, standard scaler was applied to scale the features, as models such as KNN and Logistic Regression perform better with standardized inputs. The scaler was fitted exclusively on the training data to prevent data leakage.



```
In [13]: # Convert scaled arrays back into a DataFrame
         X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns)
         X_test_scaled_df  = pd.DataFrame(X_test_scaled, columns=X_test.columns)

         # Show first 10 rows
         print(X_train_scaled_df.head(10))
```

```
         ph  Hardness    Solids  Chloramines   Sulfate  Conductivity  \
0  0.268190 -0.711238 -0.425175    -0.132286  0.489822     -0.603281
1  0.260519  0.279187  0.716027    -1.287091  0.000234      0.231275
2 -0.512548  0.307079 -0.416061    -2.135826  0.400779     -0.447559
3 -0.319606 -0.460242 -1.395853     0.741282  0.000234     -0.306413
4  0.334738 -0.720463 -1.264532     0.827274  0.000234     -0.311415
5 -0.020432 -0.419974  0.972284    -0.498877 -1.076731     -0.269249
6 -0.367386 -0.253968 -0.217807     0.113612  0.000234     -0.438374
7  0.583975  0.681749 -0.375376     0.624492  0.000234     -0.763558
8  1.061031  1.391535  0.102963     0.252210  0.000234     -0.901341
9 -0.216348  0.393282 -0.218646    -0.826824 -1.260137     -1.662633
```

Figure 7: After scaling values of data

As we can see here on the figure, after scaling the data all the feature's data range is pretty much closer.

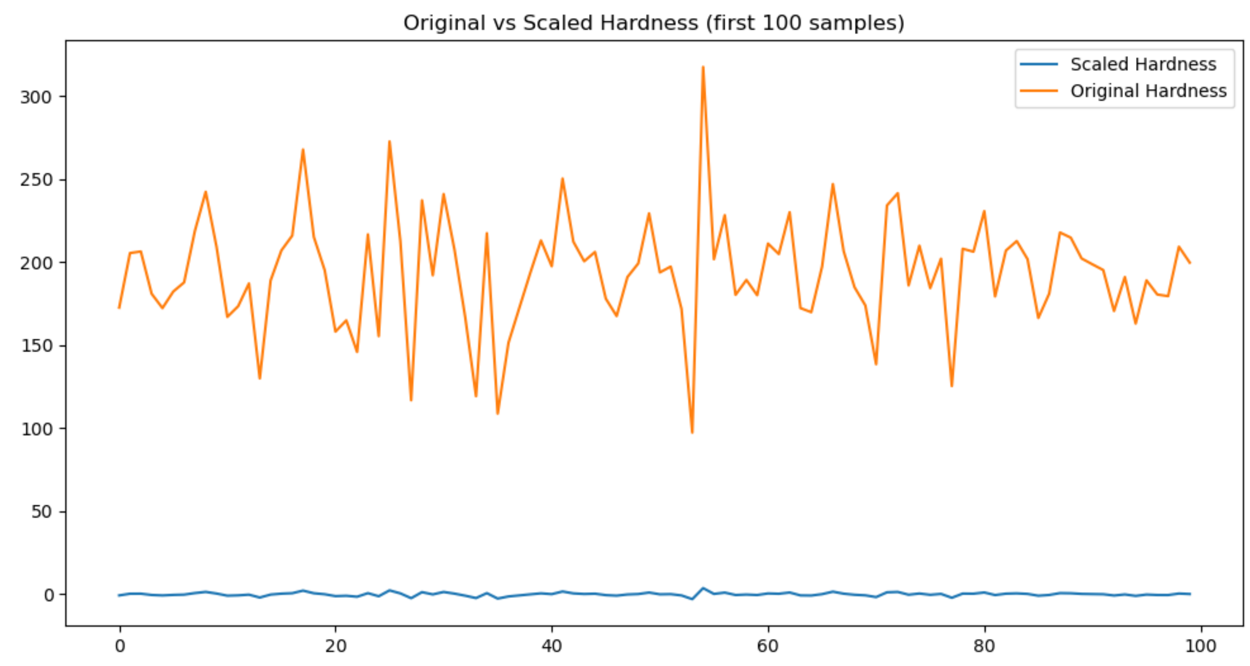


Figure 8: Comparison between original and scaled value of data

This figure shows how the Hardness values change before and after scaling for the first 100 samples. The orange line is the original Hardness, which varies a lot from about 100 up to over 300. The blue line is the scaled version, after using Standard Scaler to set the mean to 0 and the variance to 1.

1.4.1.4 Data Splitting

The dataset was divided into training and testing sets using an 80/20 split, random_state set to 42, and stratify=y to maintain class balance. A separate validation set was not created; models were trained on the training split and evaluated on the test split. Cross-validation was used on the training data for hyperparameter tuning as required.

1.4.2 *Used Models*

We selected four models for this project which are simpler approaches and progressing to more advanced ones. The following models show our rationale for each choice, informed by our EDA and the project guidelines.

Logistic Regression: We used this as our general model. It is a linear method, fast to train, and straightforward to explain. For binary classification work such as determining safety, it gives a useful reference point for model performance. As we see the low correlations observed in the heatmap, we did not expect high accuracy. However, it helped illustrate the challenge of the problem.

K-Nearest Neighbors (KNN): This distance-based method arranges samples based on their similarity to others. We regarded it because it may detect local patterns in the data, which are groups of safe samples with similar pH and Solids values. Even though there are outliers, scaling the data could mitigate their impact. KNN also does not require assumptions about the data distribution, making it suitable for this numerical dataset.

Random Forest: This collecting method combines multiple decision trees. We chose it because our EDA indicated non-linear relationships and weak linear associations. Which decision trees can address by dividing on features in various ways. Random Forest also deals with class inequality when class weights are set and provides feature importance scores. That help identify which parameters, such as Solids or pH, are most relevant for potability. It is also robust to outliers and less prone to overfitting.

XGBoost: This gradient boosting model builds trees orderly to correct previous errors. We selected it because it usually performs well on tabular data, mainly when there is class imbalance . As the scale_pos_weight parameter can be adjusted to address this. XGBoost is broadly used in water quality prediction studies with strong results. That is why we included it for comparison. It is a powerful method but requires careful tuning to prevent overfitting.

1.4.3 *Hyperparameter Tuning*

For each model, we did some tuning to get better results.

- Logistic Regression: Mostly default, but tried different solvers and C values.
- KNN: Tuned the number of neighbors (k from 3 to 5) and tried different distance weights.
- Random Forest: Used “Grid Search CV” to try different n_estimators (100, 200, 300), max_depth, and min_samples_split. Also set class_weight=‘balanced’ to help with the imbalance.

- XGBoost: Tuned learning_rate, max_depth, n_estimators, and subsample. We used scale_pos_weight to handle the class imbalance (set it to ratio of negative to positive samples).

1.4.4 *Training and Evaluation Setup*

All models were trained on the scaled training data (80% of dataset). There was not separate validation set for final selection. Instead the best hyperparameters came from cross-validation. And the final performance was measured using the test_set (20%).

Since the dataset has class imbalance, therefore we just cannot use accuracy to evaluate our models. So other evaluation matrix like precision, recall, and F1-score, with particular emphasis on recall for the class-1 (portable water), as identifying unsafe water was prioritized. Al well as the confusion metrics are shown to judge the misclassifications.

Project Pipeline:

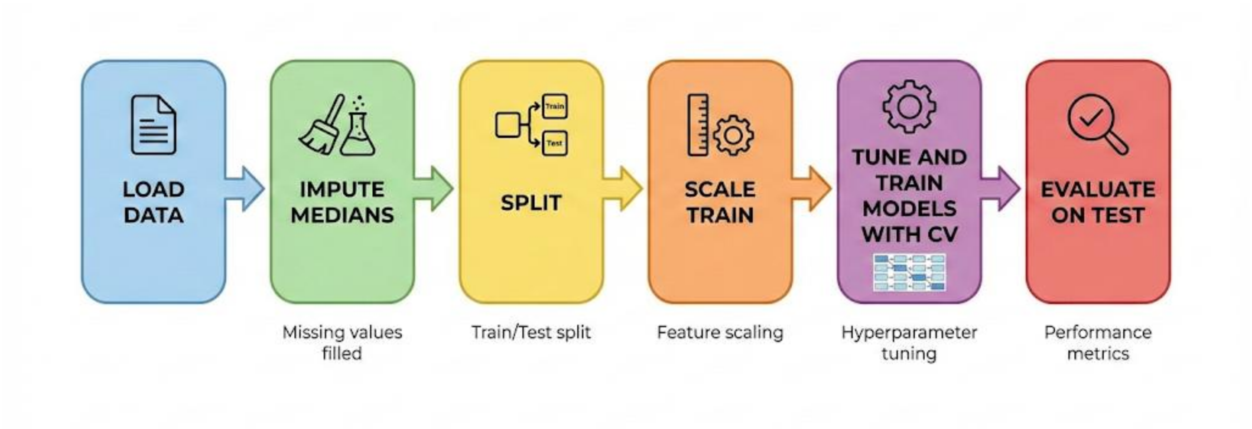
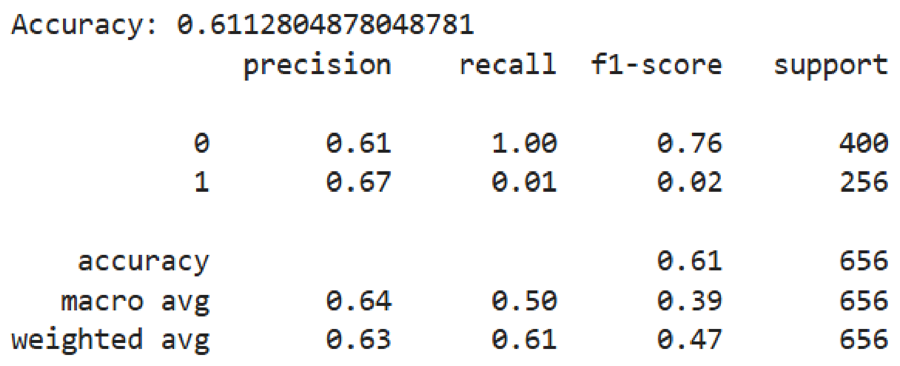


Figure 9: Pipeline

1.5 Experimental Results & Discussion

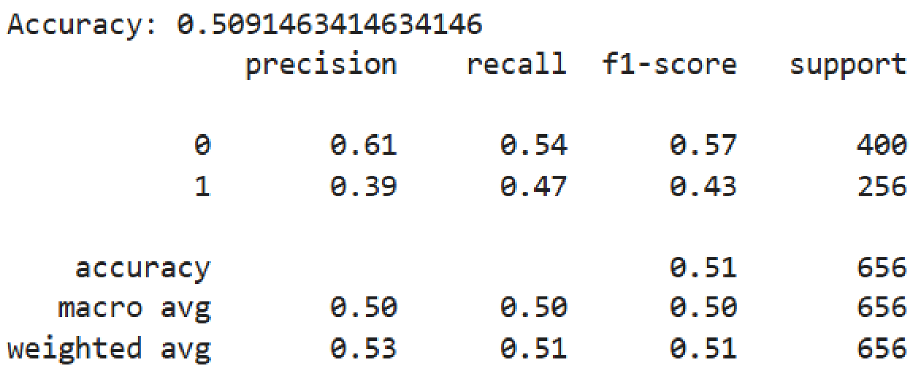
1.5.1 *Logistic Regression Model*



```
Accuracy: 0.6112804878048781
              precision    recall  f1-score   support

           0       0.61      1.00      0.76       400
           1       0.67      0.01      0.02       256

    accuracy                           0.61       656
   macro avg       0.64      0.50      0.39       656
weighted avg       0.63      0.61      0.47       656
```

Figure 10: Result of Logistic regression model

As we can see here, the Logistic Regression model achieved overall 61.13% accuracy. Which is good, but if we dive into the class-wise metric we can spot some imbalance. The table shows the Recall of predicting 0 (non-Portable) is 1, which means the model is extremely good at detecting unsafe water. The model is able to detect 100% of the unsafe water. On the other hand, the recall of predicting 1 (Portable) is 0.01, which shows the model's poor performance at detecting safe water. It only able to classify 1% of the safe water. The model bias toward the non-portable class, which is due to the imbalance in the dataset. With such performance the model is not reliable, therefore we need to bring some changes within the model.

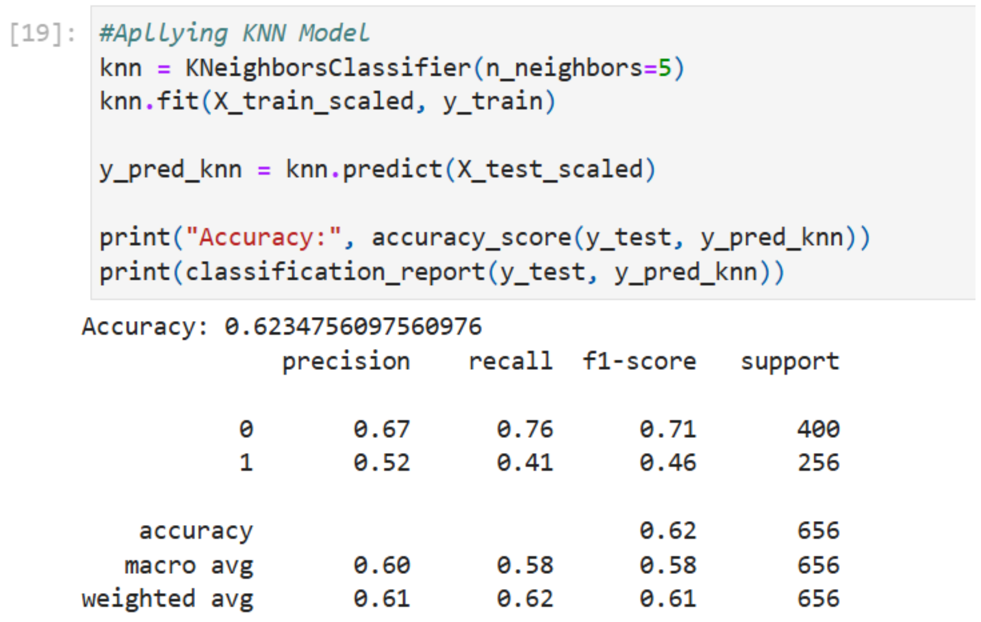


```
Accuracy: 0.5091463414634146
              precision    recall  f1-score   support

           0       0.61      0.54      0.57       400
           1       0.39      0.47      0.43       256

    accuracy                           0.51       656
   macro avg       0.50      0.50      0.50       656
weighted avg       0.53      0.51      0.51       656
```

Figure 11: Logistic Regression model with balanced weight

After balancing the weight of the model, though the accuracy reduced to 50.91% but the class-wise metric shows that now the model can identify safe water better the before. The balanced weight helps the model to reduce the biasness. As a result, now it treated both of the classes equally. Now overall, the model is balanced and more reliable.

1.5.2 *K-Nearest Neighbor (KNN) Model*



```
[19]: #Apllying KNN Model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

y_pred_knn = knn.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

```
Accuracy: 0.6234756097560976
              precision    recall  f1-score   support

           0       0.67      0.76      0.71       400
           1       0.52      0.41      0.46       256

    accuracy                           0.62       656
   macro avg       0.60      0.58      0.58       656
weighted avg       0.61      0.62      0.61       656
```

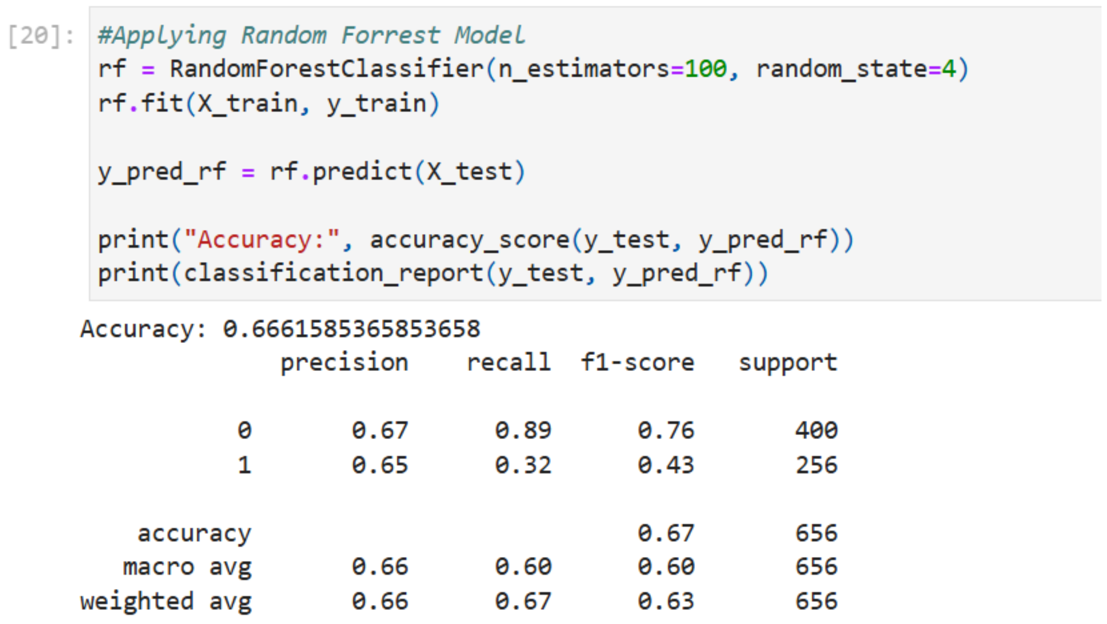
Figure 12: Evaluation Metrics for K-Nearest Neighbors (KNN) Model

Here, the KNN classifier with k=5 performed on the test set. It reached an overall accuracy of 62.35%, which is a bit higher than the Logistic Regression.

Same as the regression model the KNN model performed well for predicting class 0, which is non-potable water. It had a precision of 0.67, recall of 0.76, and an F1-score of 0.71, so it reliably identified unsafe samples. For class 1, potable water, the results were lower: precision was 0.52, recall was 0.41, and the F1-score was 0.46. The figure also shows macro average F1-score is 0.58, which shows the model can handle both of classes reasonably well despite of having imbalance in the data. KNN was improved after scaling features and it was able to find local patterns in the data, but it still struggles with noisy features and high-dimension data.

Overall, the KNN model is a reliable baseline model for predicting water potability and it shows a better balance than Logistic Regression.

1.5.3 *Random Forest Model*



```
[20]: #Applying Random Forrest Model
      rf = RandomForestClassifier(n_estimators=100, random_state=4)
      rf.fit(X_train, y_train)

      y_pred_rf = rf.predict(X_test)

      print("Accuracy:", accuracy_score(y_test, y_pred_rf))
      print(classification_report(y_test, y_pred_rf))
```

```
Accuracy: 0.6661585365853658
              precision    recall  f1-score   support

           0       0.67      0.89      0.76       400
           1       0.65      0.32      0.43       256

    accuracy                           0.67       656
   macro avg       0.66      0.60      0.60       656
weighted avg       0.66      0.67      0.63       656
```

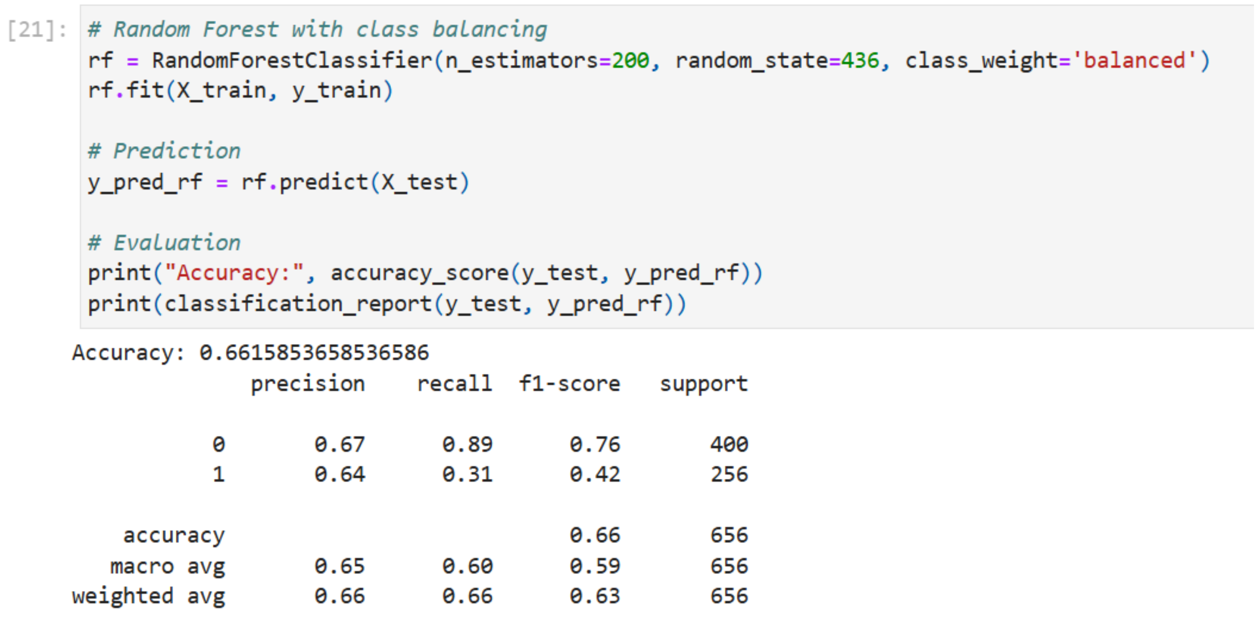
Figure 13: Evaluation Metrics for Random Forest Model

Here, the figure show how the Random Forest Classifier performed when trained with 100 estimators. It reached an overall accuracy of 66.62%, which was better than both Logistic Regression and KNN.

For class 0, which is non-potable water, the model did well: precision was 0.67, recall was 0.89, and the F1-score was 0.76. This means it reliably picked out unsafe water samples.

For class 1 (potable water), the model had a precision of 0.65, recall of 0.32, and an F1-score of 0.43. These numbers are lower than for class 0, but they still show an improvement compared to earlier models.

1.5.3.1 Random Forest with Balanced Class Weights



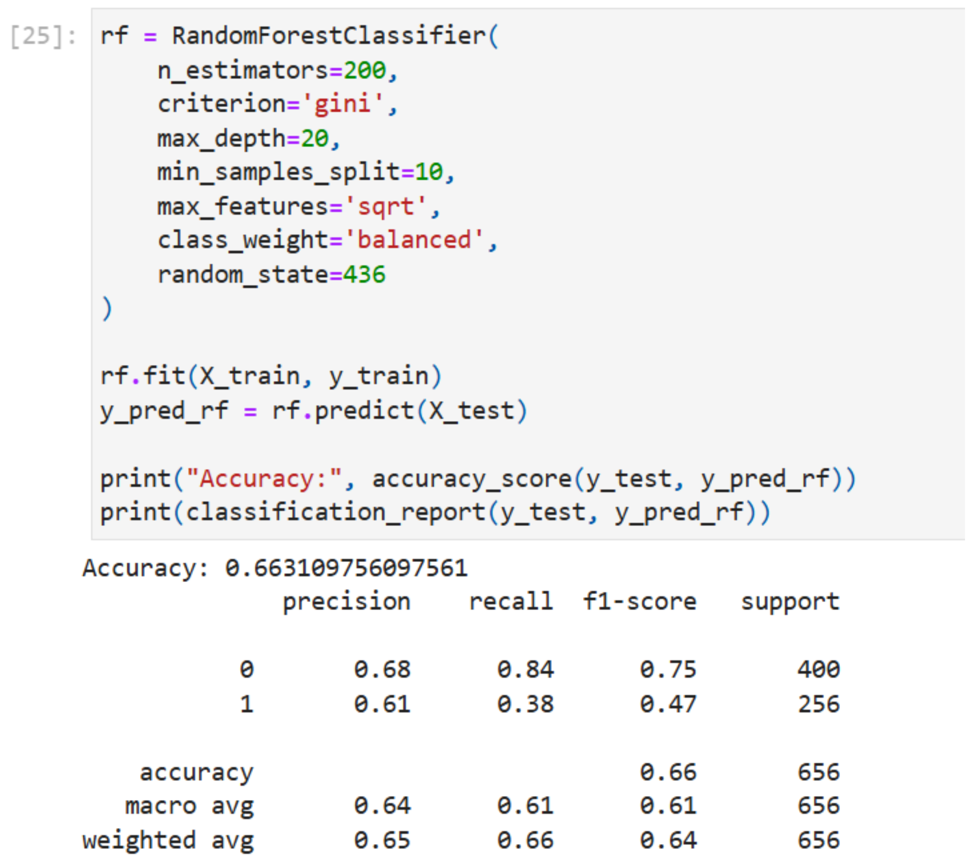
```
[21]: # Random Forest with class balancing
      rf = RandomForestClassifier(n_estimators=200, random_state=436, class_weight='balanced')
      rf.fit(X_train, y_train)

      # Prediction
      y_pred_rf = rf.predict(X_test)

      # Evaluation
      print("Accuracy:", accuracy_score(y_test, y_pred_rf))
      print(classification_report(y_test, y_pred_rf))
```

```
Accuracy: 0.6615853658536586
              precision    recall  f1-score   support

           0       0.67      0.89      0.76       400
           1       0.64      0.31      0.42       256

    accuracy                           0.66       656
   macro avg       0.65      0.60      0.59       656
weighted avg       0.66      0.66      0.63       656
```

Figure 14: Evaluation Metrics for Random Forest with Balanced Class Weights

The figure shows Random Forest Classifier performance when it was trained with 200 estimators. Also with balanced class weights. The model reached an overall accuracy of 66.16%, which is same as the default, but this time fairness between the classes improved a bit. The recall here is a little better than it was in unbalanced version.

1.5.3.2 Random Forest with Hyperparameter Tuned



```
[25]: rf = RandomForestClassifier(
          n_estimators=200,
          criterion='gini',
          max_depth=20,
          min_samples_split=10,
          max_features='sqrt',
          class_weight='balanced',
          random_state=436
      )

      rf.fit(X_train, y_train)
      y_pred_rf = rf.predict(X_test)

      print("Accuracy:", accuracy_score(y_test, y_pred_rf))
      print(classification_report(y_test, y_pred_rf))
```

```
Accuracy: 0.663109756097561
              precision    recall  f1-score   support

           0       0.68      0.84      0.75       400
           1       0.61      0.38      0.47       256

    accuracy                           0.66       656
   macro avg       0.64      0.61      0.61       656
weighted avg       0.65      0.66      0.64       656
```

Figure 15: Evaluation Metrics for Tuned Random Forest Model

After tuning hyperparameters Random Forest model performance with 200 estimators, max depth 20, and balanced class weights reached 66.31% accuracy.Now, class 0 have a strong F1-score of 0.75. Recall for class 1 has been improved to 38%, reducing bias toward the majority class. So, we can say Hyperparameter tuning helped balance the classes and overall performance.

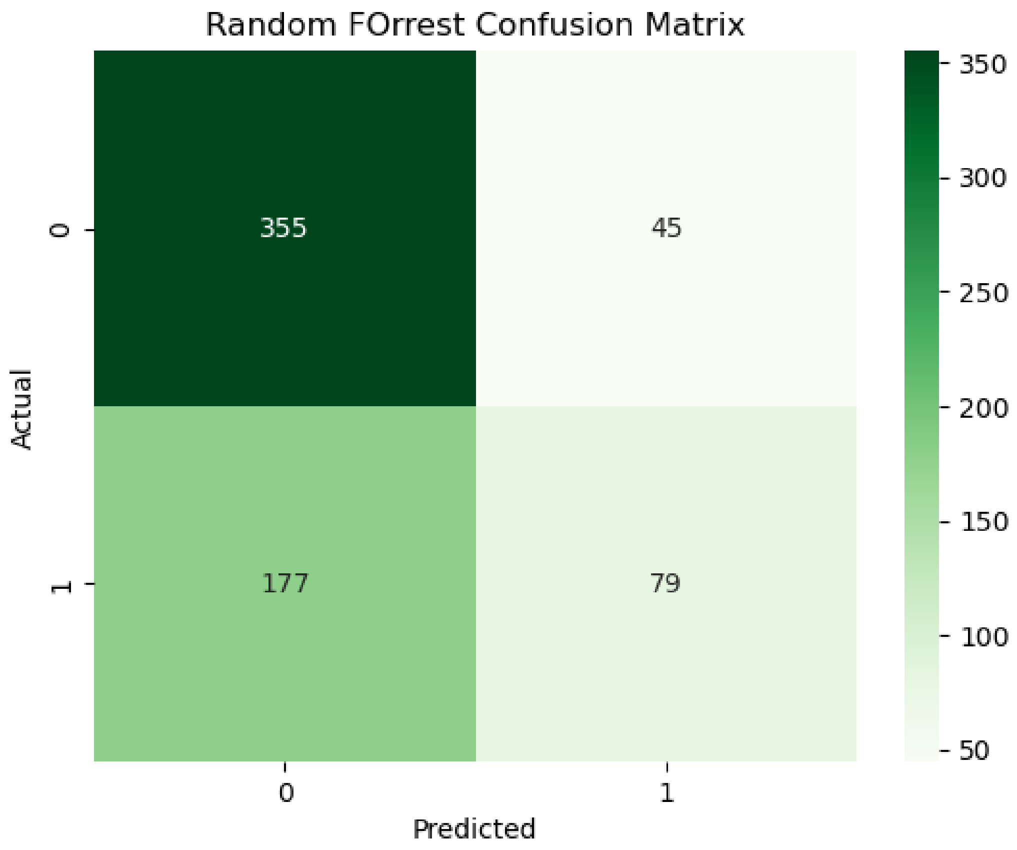


Figure 16: Random Forest Confusion Matrix

We can understand the Random Forest model working process by the confusion matrix. It detects 355 non-potable water samples and 79 potable samples. Also mistakenly classified 45 non-potable drinking water as potable drinking water and 177 potable drinking water as non-potable drinking water. True positive and true negative are the same except for the models. Enhance the detection of potable drinking water for recall. which shows that this model can handle an imbalance well and detect the complex pattern of data.

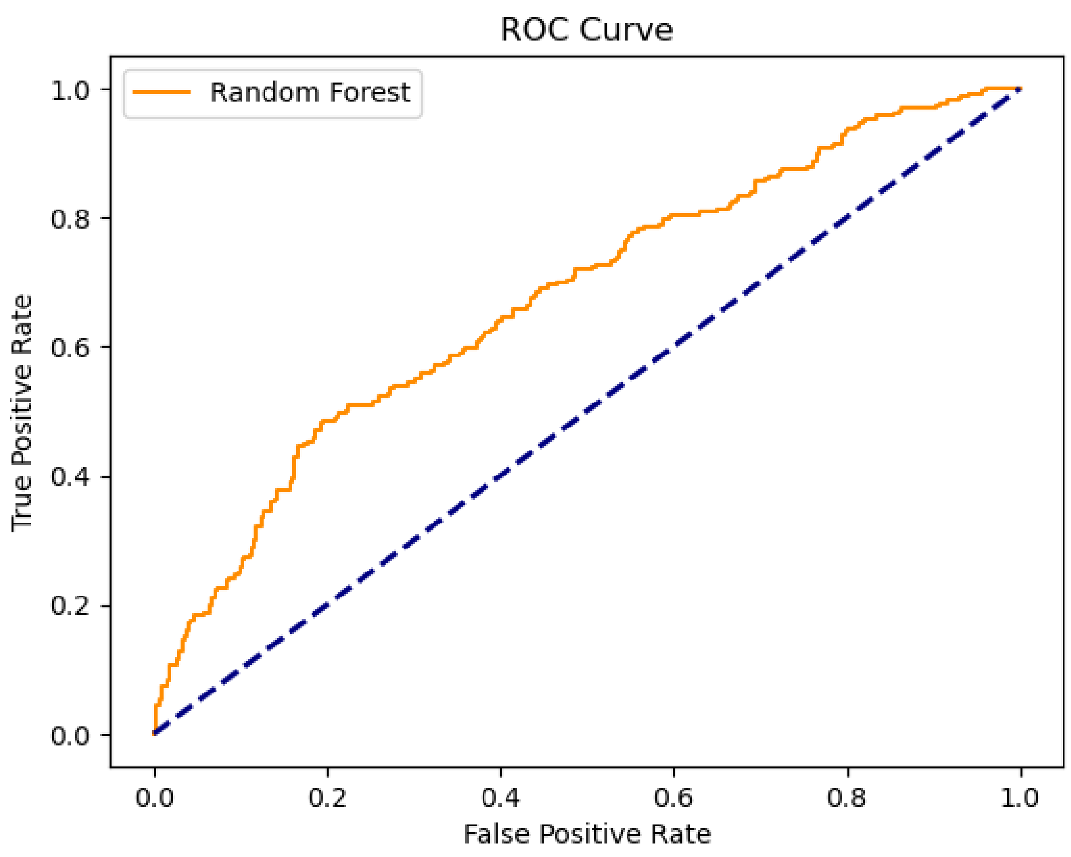


Figure 17: AUC Curve of Random Forest

This ROC curve shows that how the Random Forest model can differentiate between the portable and non-portable class. By looking at then orange line, we can see how much the model is doing better than a simple guessing. Since the curve bends toward the top-left which means the model can make better prediction than chance and it balance between sensitivity and specificity.

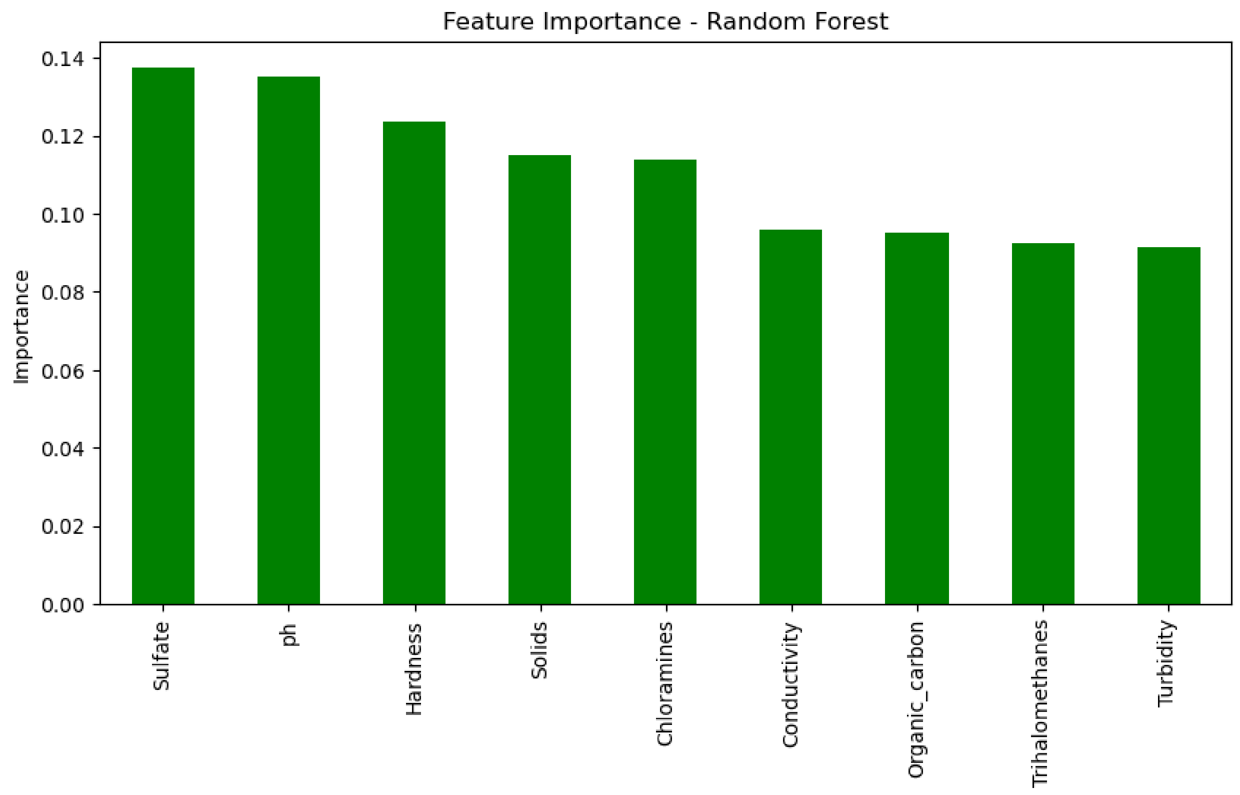
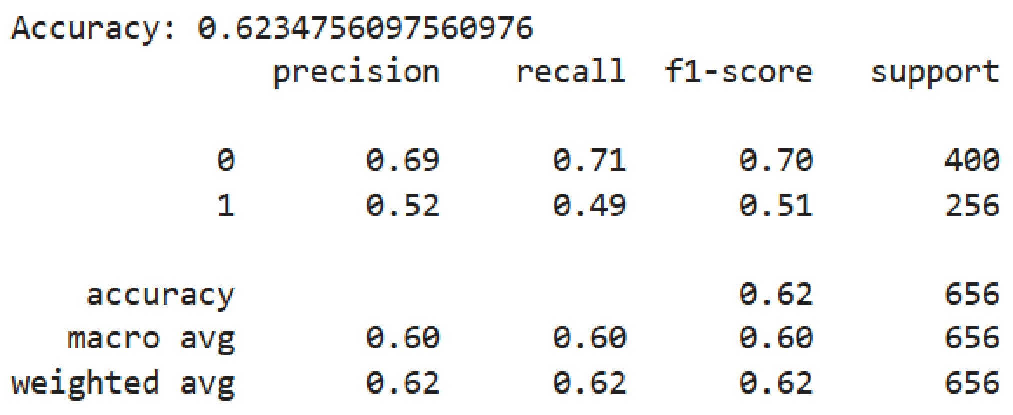


Figure 18: Feature Importance of Random Forest

The bar chart shows which water quality features mattered most to the Random Forest model when predicting potability. Sulfate and pH were the most important, with Hardness, Solids, and Chloramines next. Trihalomethanes and Turbidity had less influence. This matches what we know about the main factors that affect water safety.

1.5.4 *XG Boost Model*



```
Accuracy: 0.6234756097560976
              precision    recall  f1-score   support

           0       0.69      0.71      0.70       400
           1       0.52      0.49      0.51       256

    accuracy                           0.62       656
   macro avg       0.60      0.60      0.60       656
weighted avg       0.62      0.62      0.62       656
```

Figure 19: XG Boost Model Evaluation Matrix

XGBoost classifier trained by 200 estimators, maximum depth 5, and learning rate 0.1. To decrease the imbalance of class, we use the "scale_pos_weight" parameter. This model gets 62.35% accuracy. It works well in both classes. with an F1-score of 0.70 for class 0 and 0.51 for class 1. macro F1-score 0.6 and weighted average F1-score 0.62. Overall, the XGBoost model detects well non-linear patterns and increases recall for water potability (class 1), compared to other models

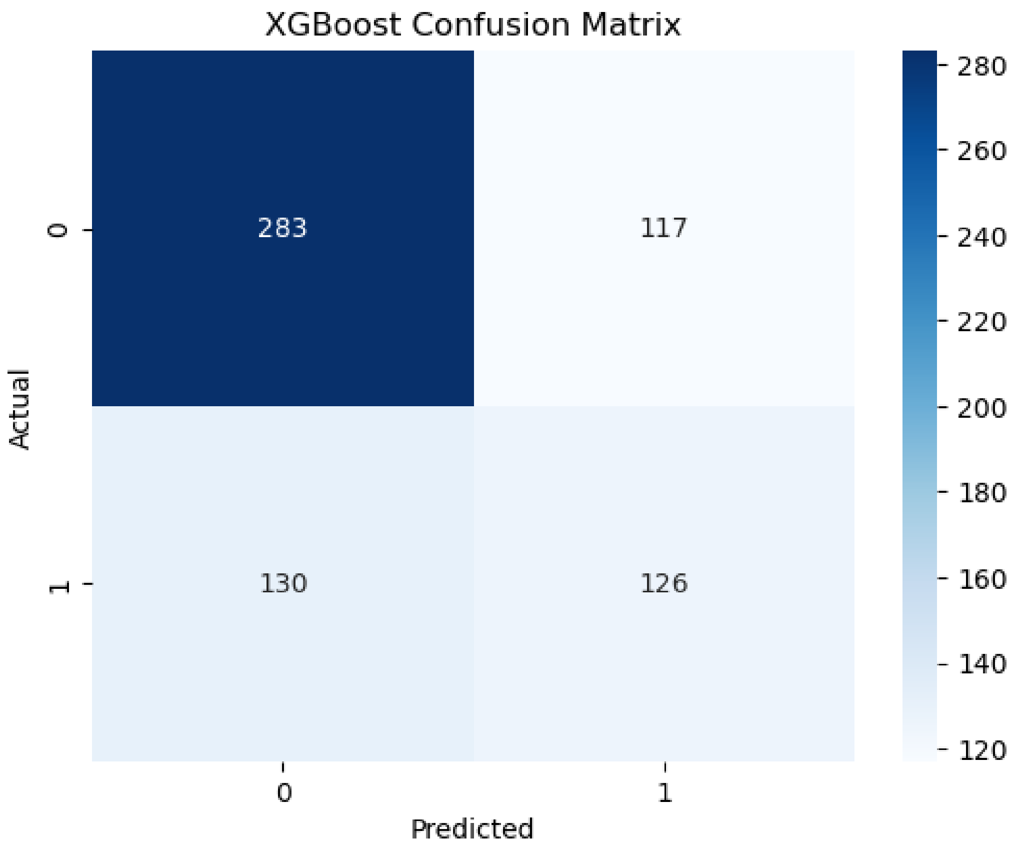


Figure 20: XG Boost Model Confusion Matrix

We can understand the XGBoost model working process by the confusion matrix. It detects 283 non-potable water samples and 126 potable samples. Also mistakenly classified 117 non-potable drinking water as potable drinking water and 130 potable drinking water as non-potable drinking water. True positive and true negative are the same except for the models. Enhance the detection of potable drinking water for recall. which shows that this model can handle an imbalance well and detect the complex pattern of data.

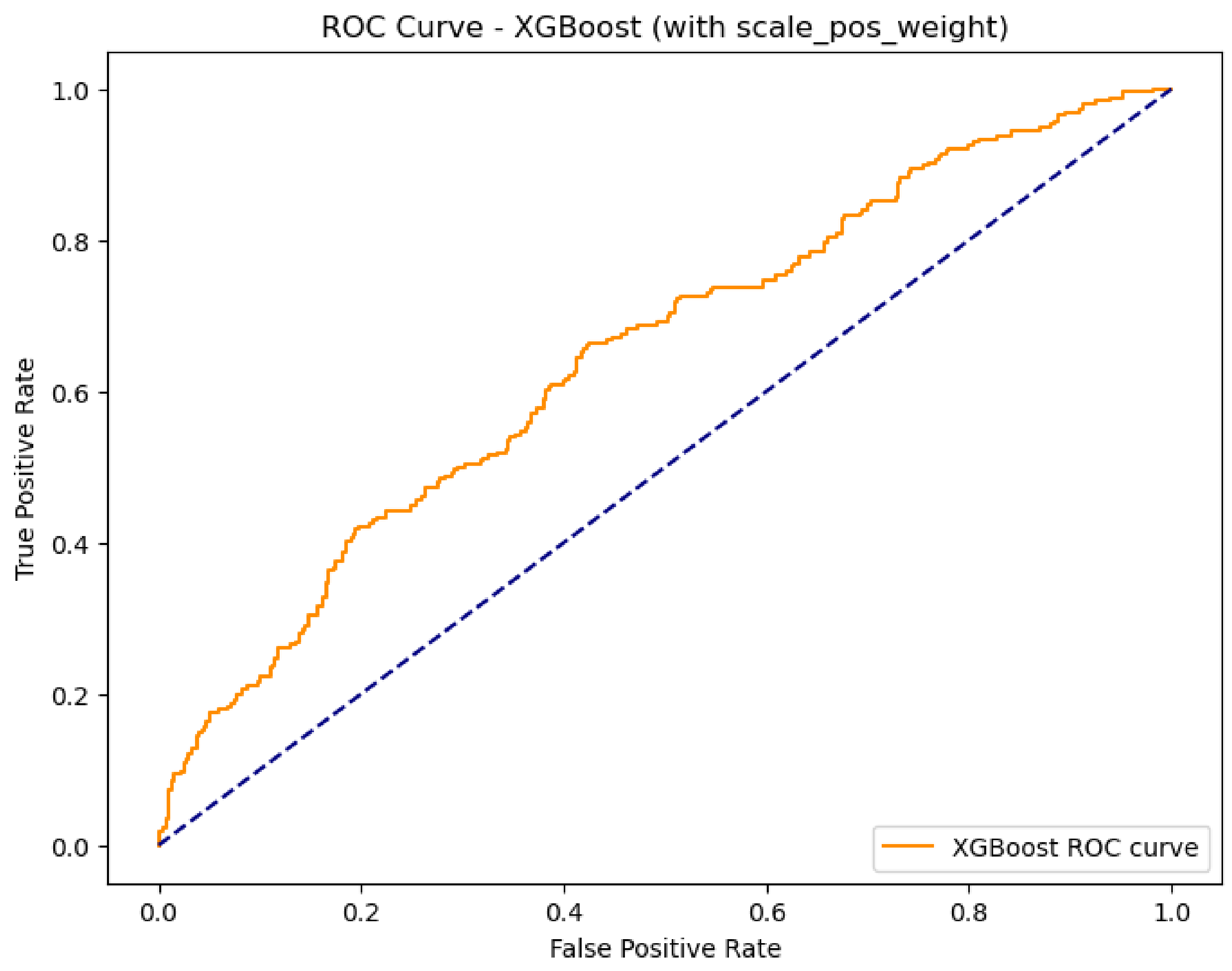


Figure 21: XG Boost Model ROC Curve

This ROC curve shows that how the XG Boost model can differentiate between the portable and non-portable class. By looking at then orange line, we can see how much the model is doing better than a simple guessing. Since the curve bends toward the true positive which means the model can make better prediction than chance and it balance between sensitivity and specificity.

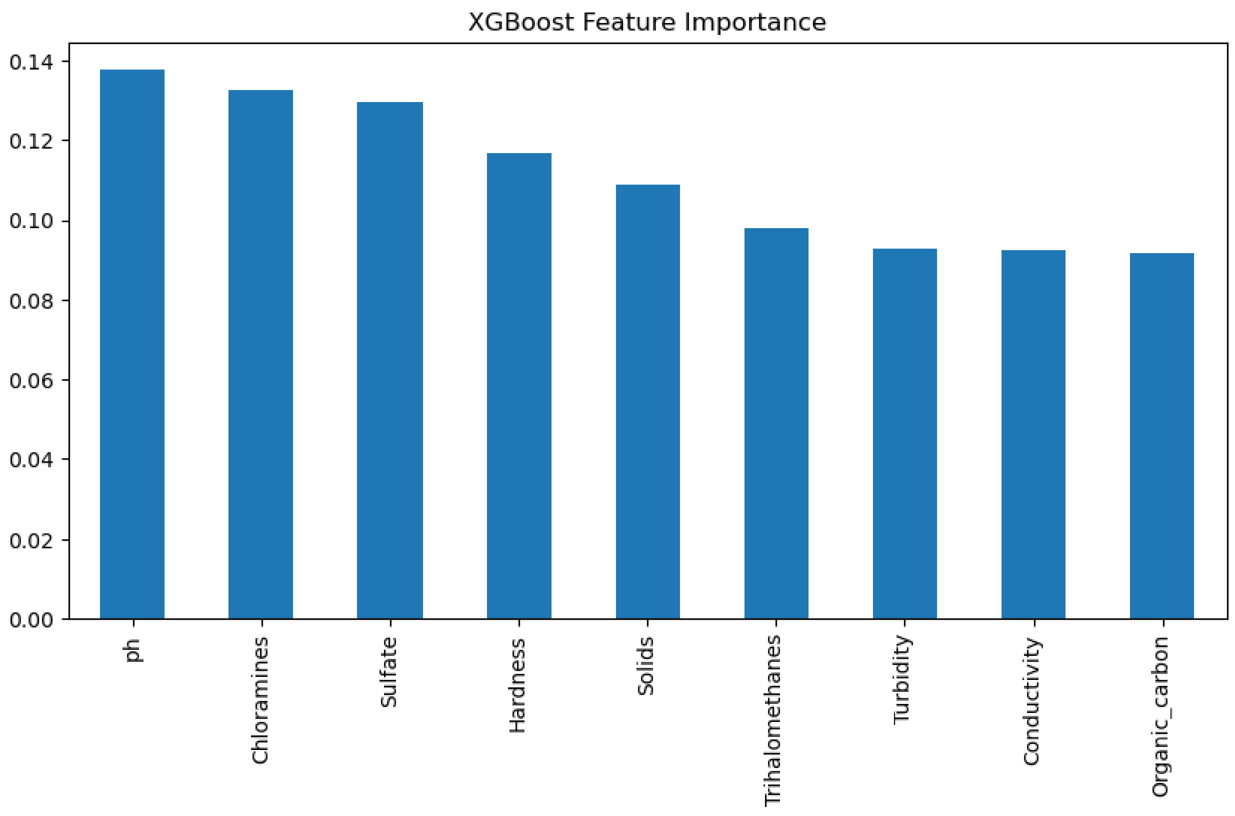
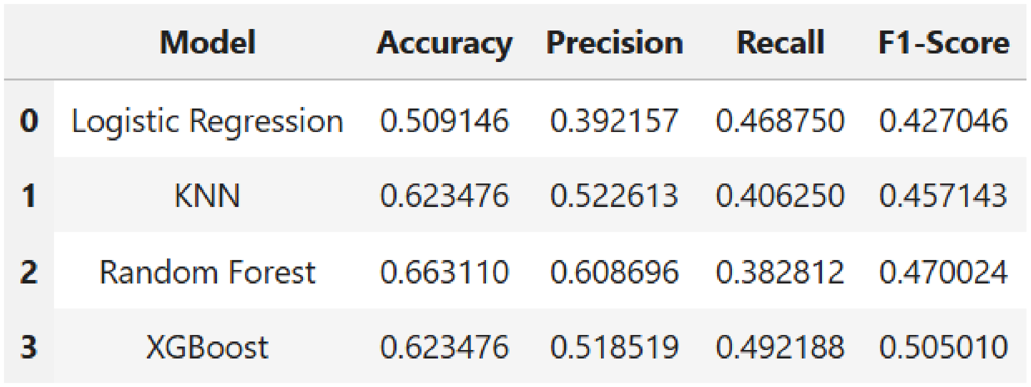


Figure 22: XG Boost Feature Importance

The bar chart shows which water quality features are essential for drinking water potability in the XGBoost model. pH, Chloramines, and sulfate are the most important features, and hardness and solids are next to them. Organic carbon effects are small. It explains which type of chemical feature is most essential, and in the future, it will be clear which type of data or sensors can be helpful for the next step.

1.5.5 *Overall Comparison*



	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.509146	0.392157	0.468750	0.427046
1	KNN	0.623476	0.522613	0.406250	0.457143
2	Random Forest	0.663110	0.608696	0.382812	0.470024
3	XGBoost	0.623476	0.518519	0.492188	0.505010

Figure 23: Performance Comparison Table of Models

The table shows that the random forest model accuracy is maximum, 66.3%. XGBoost model F1-score is better, 0.505, and recall is 0.492. Logistic regression was comparatively weak, 50%. The score was moderate, but this is not unrealistic because EDA shows that the relation of features and the possibility of water is weak. No model had a strong pattern for detection, so the best model also stopped at around 66%. Logistic regression was bad because it expects an easy linear relationship, but those aren't there.

KNN was average; it performed well after scaling the outliers of solids and turbidity. On the other hand, tree-based models worked well. Random Forest performed the best in terms of accuracy and good precision, so the chances of Wrong 'safe' detection are low. XGBoost was comparatively balanced. Its recall and F1-score were large, so this can also predict more accurately not so small.

The feature importance of Random Forest and XGBoost shows that solids, pH, and sulfate were most important and logical because they are significant factors for the classification of real water standards. For an imbalance of classes, the possibility of a drinking water recall becomes low. Most prediction goes for 'not safe' for higher accuracy. In reality, it is safe because unsafe water is a risk.

Overall, due to the small size of the dataset, missing values, and some synthetic features, very high scores were not achieved. Nevertheless, the ensemble models performed clearly better than the simple models, which highlighted how much data quality impacts the results. In the future, these models could be useful for rapid water testing in practical scenarios.

1.5.6 *Discussion*

Four machine learning models- Logistic Regression, KNN, Random Forest, and XGBoost- were used in this project to predict the potability of drinking water and compared. Logistic regression does not work well because it can not detect the nonlinear or complex patterns of data. So we understand that complex environmental issues are not enough for linear models. KNN performs poorly, but it is influenced by the process of scaling, and the availability of large features decreases the performance, so in real life, Logistic Regression and KNN are helpful for just a primary step, but not for the main step. Random forest performed better than other models. because it can detect the complex pattern of data. It can especially detect non-potable drinking water, so this model is suitable for detecting unsafe drinking water. Although some potable samples are missed due to class imbalance, by tuning the model's settings, both classes work well. By extracting the importance of features, we see that pH, sulfate, and hardness have emerged as the most important features in determining security. XGBoost is a strong model, especially since it can handle the imbalance of classes better than random forest. All model faces some challenges in detecting the potential of drinking water. mainly consists of a large amount of non-potable drinking water samples in data, which improves in an ensemble way, the potable classes of recall increase in future resampling, and improve the performance matrix in an ensemble way. Overall, Random Forest and XGBoost performed best and were most accurate, and can balance explainable and class sensitivity.

In summary, Random Forest and XG Boost were the best models overall. They balanced accuracy, interpretability, and sensitivity to different classes. Still, their limits show that choosing a model is not just about performance. It is also important to think about fairness, how much computing power is needed, and what happens if the model makes a mistake.

1.6 Conclusion & Future Work

1.6.1 *Future Work*

The present work shows that physicochemical data can be used to predict the potability of drinking water by a machine learning system, but it also has some areas for improvement. Now, a big limitation is that it depends on a public dataset. In the future, it can be improved by collecting data from various places and times. By using diverse datasets, the model's predictions will be balanced and enhanced.

The class imbalance in the datasets is a potential area of improvement. For potable water samples in particular, improved resampling methods or cost-sensitive learning techniques can be used to reduce the bias towards the dominant class and increase the reliability of the predictions.

To identify complex patterns in water quality data, future research could use more advanced modeling approaches such as deep learning or hybrid models. Explainable AI (explainable artificial intelligence) methods can also be used to determine the most important factors affecting the potability of drinking water, which will increase the transparency and credibility of the model.

Finally, connecting the model to real-time sensor data and implementing it as a web- or mobile-based decision-support tool can improve the practical effectiveness of the system. Such developments will enable continuous water quality monitoring and rapid intervention to ensure safe drinking water.

1.6.2 *Conclusion*

This project shows how to automatically detect water quality by using machine learning. This project successfully distinguishes the potability of water by using a labeled dataset and the right preprocessing. Random forest model and F1-score have performed better than other models. The system is suitable for practical use and especially suitable for testing water where examination in a lab is difficult. By analyzing the feature, this project can identify which physicochemical parameters influence potability, which will be helpful for future research. Overall, the proposed approach can help authorities and organizations monitor water quality faster, cheaper, and more systematically.

1.7 References

[1] A. Kadiwal, "Water Potability," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/adityakadiwal/water-potability.

[2] S. Patel, K. Shah, S. Vaghela, M. Aglodiya, and R. Bhattad, "Water Potability Prediction Using Machine Learning," Research Square, May 2023, doi: 10.21203/rs.3.rs-2965961/v1.

[3] V. R and A. G. Vishvanath, "Water Quality Prediction using Machine Learning," International Advanced Research Journal in Science, Engineering and Technology, vol. 11, no. 7, pp. 156-162, July 2024, doi: 10.17148/IARJSET.2024.11724.

[4] Y. Barzegar, A. Barzegar, F. Bellini, S. Marrone, P. Pisani, and L. Verde,
"Data-Centric Water Safety Monitoring: A Machine Learning Pipeline with Intelligent Feature Selection for Potability Prediction, in Proc. 29th Int. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2025), Rome, Italy, 2025, pp. 6035–6044. doi: 10.1016/j.procs.2025.10.073.