Final Assignment

Machine learning and AI

Prof.Amabili, Prof. Petruccelli

Greening Energy Market and Finance 2024/2025

Written By:

Behshad(Beth) Ghaseminezhadabdolmaleki

Email: b.ghaseminezhadabdol@studio.unibo.it

April 2025

Table of Contents

Table of Figure

Introduction

Overview of the water potability prediction project

The goal of this project is to develop a predictive model for determining the potability of water using a dataset containing various chemical properties of water samples. Understanding whether a sample is potable is crucial for water resource management and public health.

Question of assignment:
In this project your task is to build a machine learning model able to predict the potability of water given a set of features. A water is described by the following set of features:
ph: pH of 1. water (0 to 14).
Hardness: Capacity of water to precipitate soap in mg/L.
Solids: Total dissolved solids in ppm.
Chloramines: Amount of Chloramines in ppm.
Sulfate: Amount of Sulfates dissolved in mg/L.
Conductivity: Electrical conductivity of water in μS/cm.
Organic_carbon: Amount of organic carbon in ppm.
Trihalomethanes: Amount of Trihalomethanes in μg/L.
Turbidity: Measure of light emiting property of water in NTU.
Color: is the color of the water
ranking: is a ranking given by some institution
hard_solid: Total dissolved hard solids
Potability: Indicates if water is safe for human consumption.
Potable 1 and Not potable 0

Description of the dataset

- **Number of Samples**: Indicate how many water samples or rows are present in the dataset.
- **Features**: Describe each feature, including:
    - **PH**: Measure of acidity or basicity of water.
    - **Hardness**: Represents the concentration of calcium and magnesium ions in water.
    - **Solids**: Total dissolved solids in water, which includes minerals, salts, and impurities.
    - **Chloramines**: Concentration of chloramines, used for disinfection in water.
    - **Sulfate**: Concentration of sulphate ions.
    - **Conductivity**: Water's electrical conductivity, related to the concentration of ions.
    - **Organic carbon**: Amount of organic molecules in water, indicator of water quality.
    - **Trihalomethanes**: Byproducts of chlorination, potentially harmful substances.
    - **Turbidity**: Measure of water clarity, indicating the presence of suspended particles.
    - **hard solid**: Another indicator of total solid content affecting hardness.
    - **colour**: Categorical colour indicator (e.g., blue, green).
    - **ranking**: Categorical ranking indicator (e.g., a,b,c,etc).

Target Variable:

- **Potability**: The target variable, indicates whether the water is potable (1) or not potable (0).

Key Characteristics

- **Data Type**:
    - **Numerical Features**: Include continuous values (e.g., ph, Hardness, etc.).
    - **Categorical Features**: Include discrete categories (e.g., colour, ranking).
- **Missing Values**:
    - Mention whether the dataset contains missing values and for which attributes. Our preprocessing steps will include handling these values.
- **Imbalances**:
    - Provide information about the distribution of the target variable. For example, if there is a class imbalance where more samples are labelled as 'not potable', note this as it affects choosing suitable modelling and evaluation strategies.

Summary

The dataset is a rich source for predicting water potability using a variety of chemical and physical measurements. The analysis will leverage this data to build and evaluate machine learning models, aiding in efficient water quality assessment. This dataset enables practical insights and decision-making to ensure safe water supply, leveraging both numerical and categorical data to encapsulate holistic water quality metrics.

Data Loading and Initial Exploration

Loading the Dataset

Description:

- **Source:** The dataset was sourced from a file path specified in the script, presumably from a local data repository (CSV file located at my device).
- **Tool Used:** The dataset was loaded using the pandas library, which provides powerful data manipulation capabilities, making it easier to work with tabular data in Python.

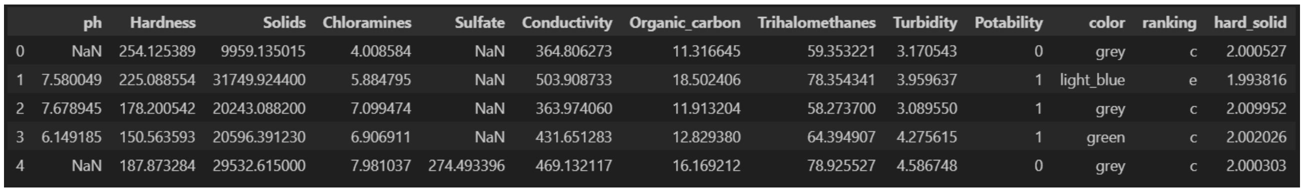
Initial Data Exploration

Once the dataset was loaded into a pandas DataFrame, initial exploration began to understand both its structure and content. The primary steps included:

1. **Checking the First Few Rows:**
    - **Purpose:** To get a quick look at the data, verify correct loading, understand the data structure.
    - **Method:** Use of df.head() to display the first 5 rows of the dataset.

Output:

- Displays columns such as ph, Hardness, Solids, etc., providing a snapshot of the data values.



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	color	ranking	hard_solid
0	NaN	254.125389	9959.135015	4.008584	NaN	364.806273	11.316645	59.353221	3.170543	0	grey	c	2.000527
1	7.580049	225.088554	31749.924400	5.884795	NaN	503.908733	18.502406	78.354341	3.959637	1	light_blue	e	1.993816
2	7.678945	178.200542	20243.088200	7.099474	NaN	363.974060	11.913204	58.273700	3.089550	1	grey	c	2.009952
3	6.149185	150.563593	20596.391230	6.906911	NaN	431.651283	12.829380	64.394907	4.275615	1	green	c	2.002026
4	NaN	187.873284	29532.615000	7.981037	274.493396	469.132117	16.169212	78.925527	4.586748	0	grey	c	2.000303

Figure 1 - DataSet

2. **Overview of Dataframe Structure:**
    - **Purpose:** To comprehend data dimensions, the type of each column, and identify missing data. It provides crucial insights into the dataset's layout and informs subsequent data cleaning and preprocessing steps.
    - **Method:** Use of df.info() to display a summary of the DataFrame.

Output:

- Lists all the columns along with their data types and non-null counts, providing information about potential missing values or incorrectly inferred types.

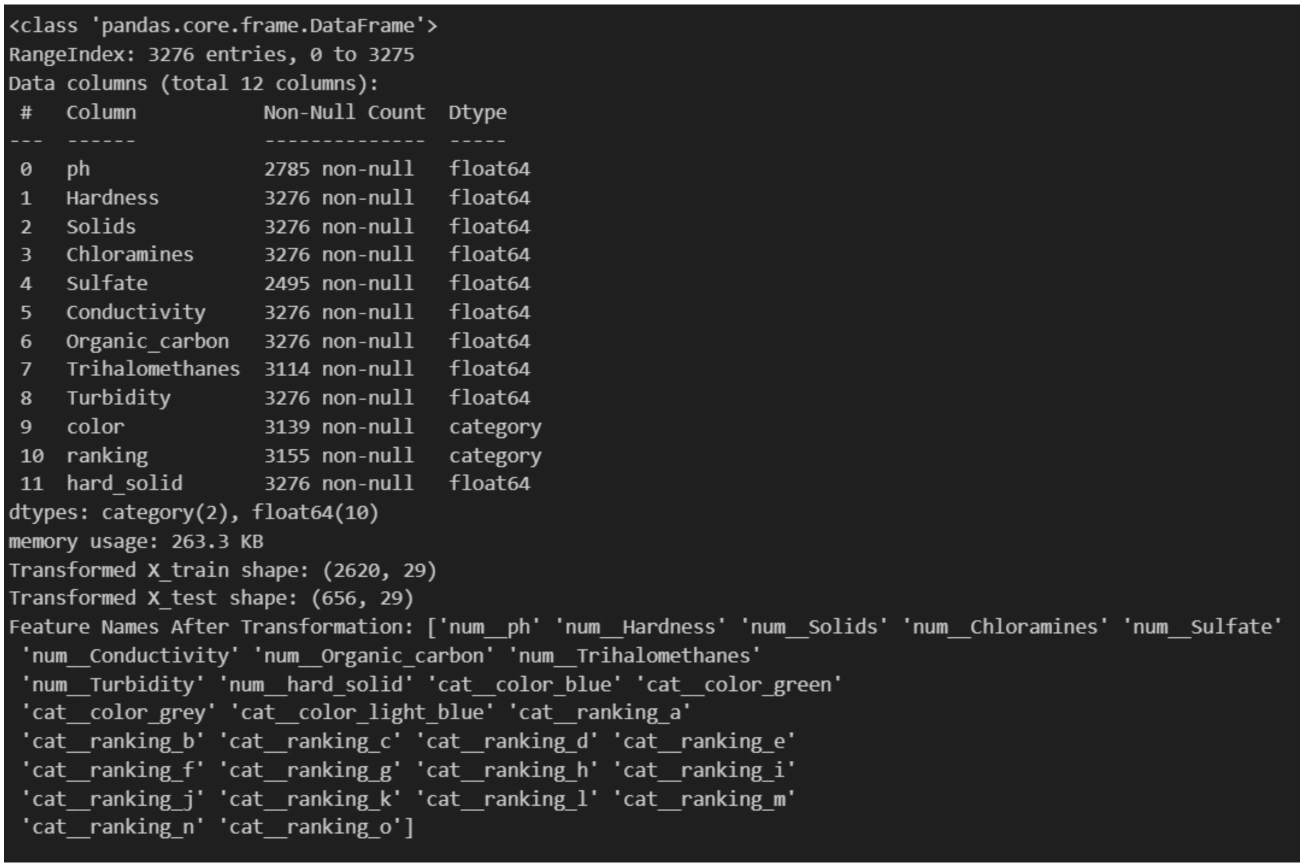
Additional Exploration Details

- **Type Checking and Missing Values:**
    - During exploration, it's noted which columns are categorical or numerical.

This influences how we handle data preprocessing steps like encoding and imputation.

- Missing data are identified using the non-null count, forming the basis for subsequent cleaning operations.

- **Statistical Summary:**
  - Although beyond just df.head() and df.info(), using df.describe() provides a statistical summary (mean, min, max) for numerical fields.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   color            3139 non-null   category
 10  ranking          3155 non-null   category
 11  hard_solid       3276 non-null   float64
dtypes: category(2), float64(10)
memory usage: 263.3 KB
Transformed X_train shape: (2620, 29)
Transformed X_test shape: (656, 29)
Feature Names After Transformation: ['num__ph' 'num__Hardness' 'num__Solids' 'num__Chloramines' 'num__Sulfate'
 'num__Conductivity' 'num__Organic_carbon' 'num__Trihalomethanes'
 'num__Turbidity' 'num__hard_solid' 'cat__color_blue' 'cat__color_green'
 'cat__color_grey' 'cat__color_light_blue' 'cat__ranking_a'
 'cat__ranking_b' 'cat__ranking_c' 'cat__ranking_d' 'cat__ranking_e'
 'cat__ranking_f' 'cat__ranking_g' 'cat__ranking_h' 'cat__ranking_i'
 'cat__ranking_j' 'cat__ranking_k' 'cat__ranking_l' 'cat__ranking_m'
 'cat__ranking_n' 'cat__ranking_o']
```

Figure 2 - Data Summary

Data Preprocessing

Data preprocessing is a critical preparatory step in machine learning pipeline, offering cleaner data through imputation, scaling, encoding, and dimensionality reduction. By addressing these issues, we enhance the models' ability to perceive patterns from features, resulting in better predictive performance and reliability. This careful process ensures subsequent machine learning efforts effectively leverage the dataset's full potential, leading to accurate and actionable insight into water potability.

Handling Missing Values

Objective:

- Missing data can lead to biased estimates and reduce model accuracy, so handling them appropriately is crucial for data integrity and performance.

Strategy:

- **Numerical Features:** Used SimpleImputer with the median strategy for numerical imputation. This is effective for mitigating the impact of outliers, as the median is robust to extreme values.
- **Categorical Features:** Used SimpleImputer with the most_frequent strategy to fill missing values, which replaces missing entries with the most common category in each feature.

Feature Scaling and Encoding Categorical Features

Objective:

- Ensuring features are on a similar scale increases model interpretability and speeds up convergence for gradient-based algorithms. Additionally, properly encoding categorical variables is essential for integrating them into models that require numerical input.

Feature Scaling:

- Applied StandardScaler after imputation for numerical features to standardize them, zero-centering and normalizing to unit variance.

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

Figure 3 - Scaling formula

Categorical Encoding:

- Used OneHotEncoder to convert categorical variables into one-hot vectors. This technique avoids assigning ordinal relationships among categorical values incorrectly.

Applying PCA

Objective:

- Principal Component Analysis (PCA) reduces dimensionality by transforming the data to a new basis (principal components) while preserving variance, enhancing model efficiency and reducing overfitting risks.

Approach:

- Retain 95% of variance while reducing dimensions, ensuring that the essential information is preserved in fewer components.

Mathematics:

- PCA projects the data into its principal components:

$$Z = XW$$

Figure 4 - PCA formula

where Z represents principal components and W is the matrix of eigenvectors. The cumulative explained variance ratio aids in selecting the number of principal components needed to capture the desired variance percentage.

Model Training

This comprehensive approach to model training combines effective preprocessing, robust model selection, and extensive hyperparameter tuning to maximize predictive accuracy. By balancing simplicity with complexity in model choices and implementing grid search for fine-tuning, this strategy effectively captures essential patterns of the data necessary for predicting water potability.

Overview of Models Trained

For this analysis, a range of machine learning models were selected to cover different types of learners and leverage their unique strengths:

1. **Logistic Regression**:
    - A model used for binary classification, capturing the relationship between

the dependent binary variable and one or more independent variables by estimating probabilities using a logistic function.

2. **Decision Tree Classifier**:
    - A non-linear model that splits data into branches to form predictions, particularly useful for interpretability and understanding feature impacts.
3. **Random Forest Classifier**:
    - An ensemble of decision trees that improves performance through bootstrap aggregation, reducing overfitting and improving accuracy.
4. **Gradient Boosting Classifier**:
    - Sequentially builds trees, learning from previous errors, suitable for its ability to improve prediction with sequential corrections.
5. **XGBoost Classifier**:
    - An optimized and efficient implementation of gradient boosting designed to handle sparse data and interact well with large datasets.

Hyperparameter Tuning using GridSearchCV

To optimize model performance and ensure models are tuned for the best accuracy, GridSearchCV was employed:

- **Process:**
    - A range of hyperparameters were tested for each model using a grid search on the training set, leveraging cross-validation to evaluate the combination of hyperparameters.
- **Outcome:**
    - For each model, the best combination of hyperparameters was selected based on the highest mean accuracy yielded by cross-validation, and the model was refitted using these parameters.

Reason of not using other valuable subjects (during the course)

1. **Linear Regression**: This algorithm is primarily used for predicting continuous values, whereas our problem is a binary classification task (0 or 1). Hence, applying linear regression is not appropriate in this context.

2. **Lasso, Ridge, Elastic Net**: These regularization techniques are designed for regression models. While they can be applied to logistic regression to improve generalization, they are not particularly effective for advanced classification models such as Random Forest and XGBoost, which inherently handle feature importance and regularization differently.
3. **Polynomial Regression**: This method is typically employed in regression tasks to capture non-linear relationships. In classification problems, particularly in binary classification, polynomial regression is rarely used. If non-linearity is a concern, kernel methods such as Kernel SVM would be more suitable alternatives.
4. **Kernel Regression**: This technique is mainly applied in continuous regression tasks. In classification problems, it is not commonly used.
5. **Clustering (K-Means, Kernel K-Means, DBSCAN)**: These are unsupervised learning methods used for grouping unlabelled data. Since our dataset includes labelled target values (0 and 1), clustering is unnecessary for model training. However, clustering might be useful for feature engineering or exploratory data analysis.
6. **SoftMax Classification**: This technique is utilized for multi-class classification problems. In our case, the task involves only two classes (binary classification: potable vs. non-potable water).

Model Evaluation

Provide a brief commentary about which models performed best and how different metrics give insight into various aspects of model performance. For example, note if there's a particularly high recall indicating good sensitivity, or if precision was prioritized over others due to the nature of the application (e.g., when false positives are costly).

Evaluation Metrics:

1. **Accuracy:**

Represents the proportion of correctly classified instances among the total instances evaluated.

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

Figure 5 - Accuracy Formula

Where:

- TP (True Positives) are correctly predicted positive instances.
- TN (True Negatives) are correctly predicted negative instances.
- FP (False Positives) are negative instances incorrectly classified as positive.
- FN (False Negatives) are positive instances incorrectly classified as negative.

2. **Precision:**

   Indicates the ratio of true positive observations to the total predicted positive observations.

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

Figure 6 - Precision Formula

3. **Recall (Sensitivity):**

   Measures the model's ability to detect all positive samples.

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

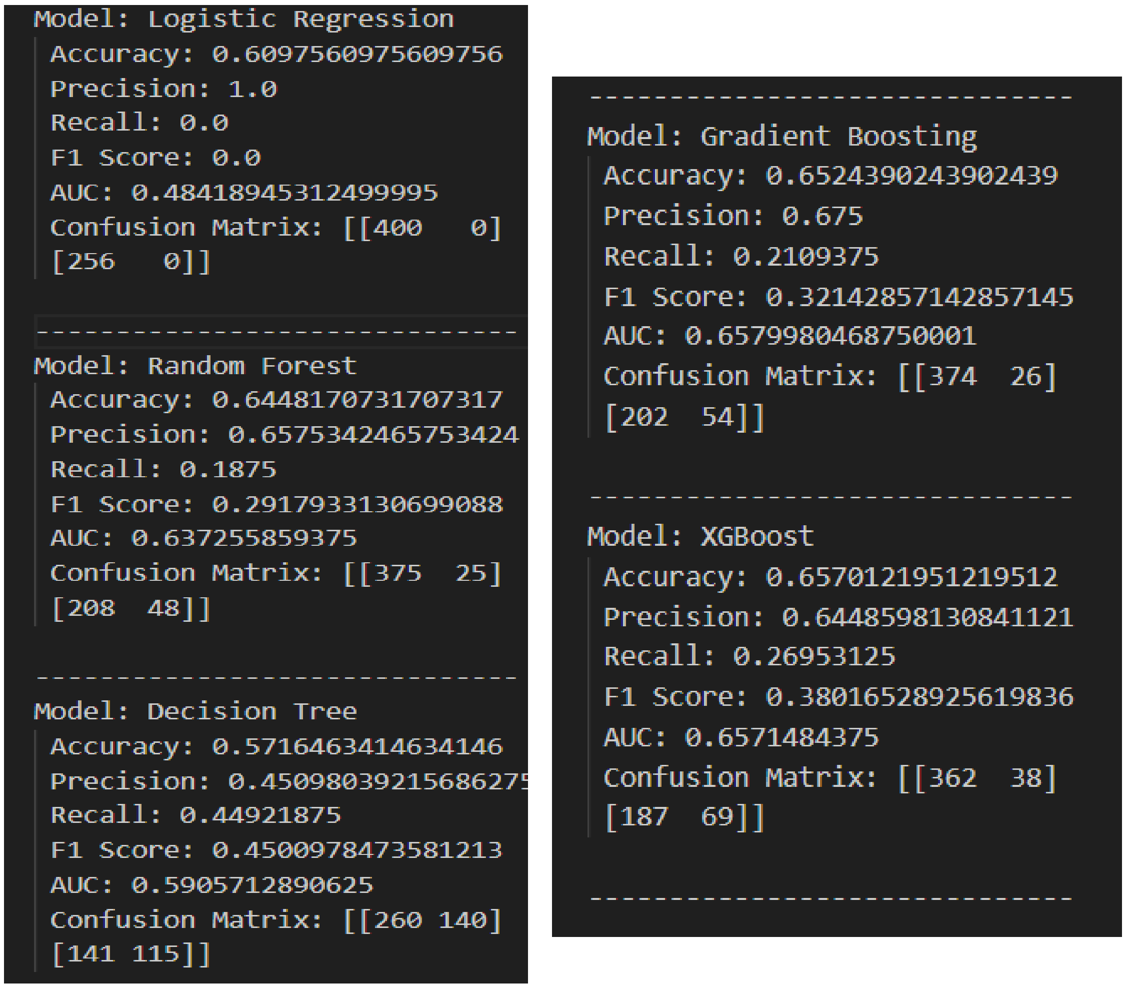
Figure 7 - Recall Formula

4. **F1 Score:**

   The harmonic mean of precision and recall, providing a balance between the two.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 8 - F1 Formula



```
Model: Logistic Regression
  Accuracy: 0.6097560975609756
  Precision: 1.0
  Recall: 0.0
  F1 Score: 0.0
  AUC: 0.48418945312499995
  Confusion Matrix: [[400   0]
  [256   0]]

------------------------------
Model: Random Forest
  Accuracy: 0.6448170731707317
  Precision: 0.6575342465753424
  Recall: 0.1875
  F1 Score: 0.2917933130699088
  AUC: 0.637255859375
  Confusion Matrix: [[375  25]
  [208  48]]

------------------------------
Model: Decision Tree
  Accuracy: 0.5716463414634146
  Precision: 0.4509803921568627
  Recall: 0.44921875
  F1 Score: 0.4500978473581213
  AUC: 0.5905712890625
  Confusion Matrix: [[260 140]
  [141 115]]
```

```
------------------------------
Model: Gradient Boosting
  Accuracy: 0.6524390243902439
  Precision: 0.675
  Recall: 0.2109375
  F1 Score: 0.32142857142857145
  AUC: 0.6579980468750001
  Confusion Matrix: [[374  26]
  [202  54]]

------------------------------
Model: XGBoost
  Accuracy: 0.6570121951219512
  Precision: 0.6448598130841121
  Recall: 0.26953125
  F1 Score: 0.38016528925619836
  AUC: 0.6571484375
  Confusion Matrix: [[362  38]
  [187  69]]

------------------------------
```

Figure 9 - Evaluation Metrics

5. **ROC-AUC (Receiver Operating Characteristic - Area Under Curve):** Evaluates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
    - The ROC curve plots TPR vs. FPR.
    - AUC reflects the overall ability to discriminate between classes.

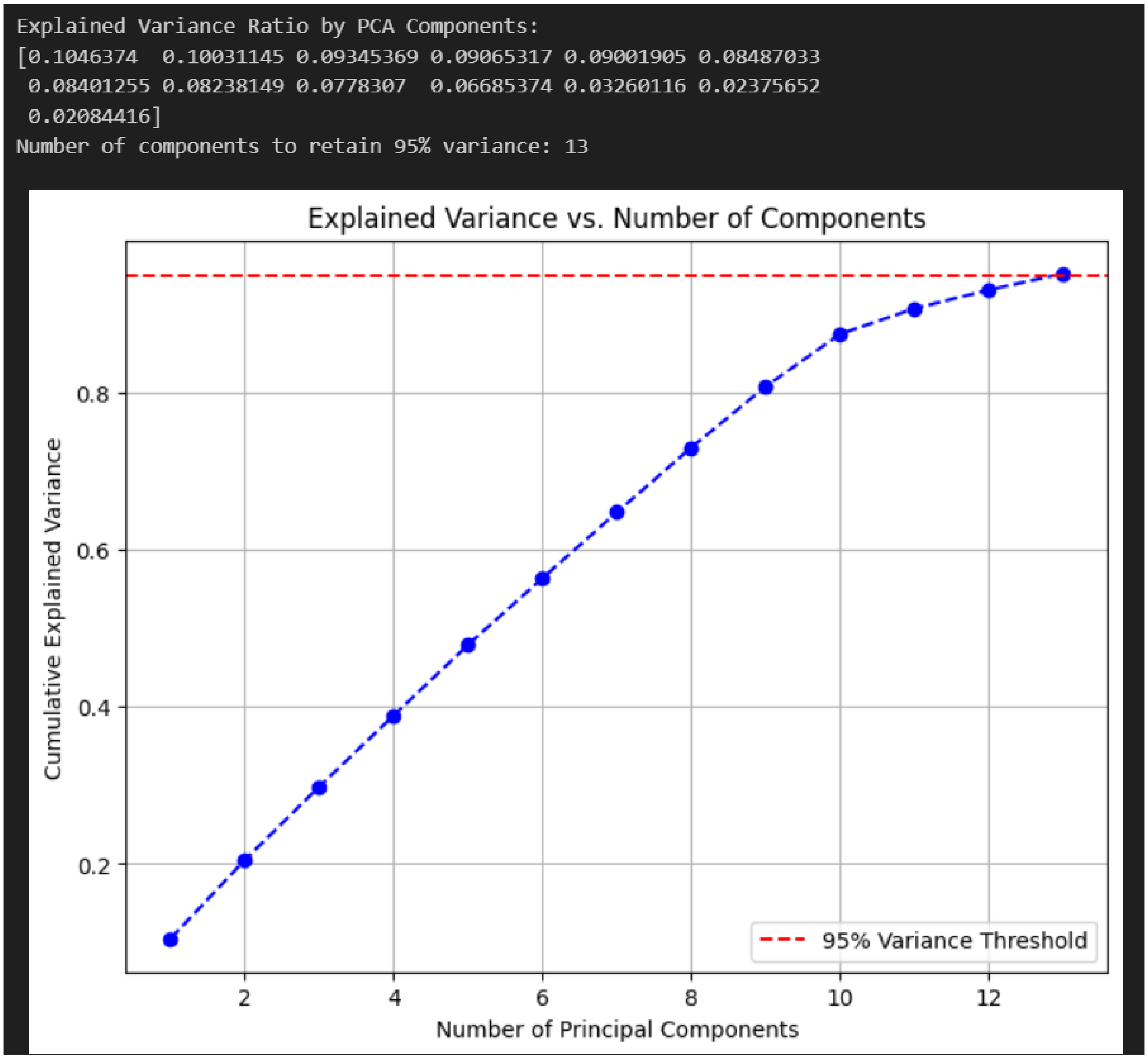
By organizing the model evaluation section in this manner, we offer a clear, methodical, and complete view of how models were assessed, maintaining transparency and rigor in analysis process.

Visualizations

Visualizations significantly enhance the interpretability and explainability of modelling outcomes, guiding stakeholders through complex dataset behaviours and model mechanics. Additionally, consider how each visual output aligns with the problem objectives, adding robust evidence to model performance and interpretability discussions.

Cumulative Explained Variance PCA:

- **Purpose:**
  - This Matrix shows how much of the total variance in the dataset is explained by each principal component when using PCA (Principal Component Analysis). It helps determine the number of components to retain in order to capture a high percentage of the data variation.
- **Description:**
  - X-axis represents the number of principal components.
  - Y-axis shows the cumulative percentage of variance explained.
  - A horizontal line at 95% aids in deciding the number of components required to retain most of the data variance.
- **Insight:**
  - This matrix assists in dimensionality reduction by identifying the minimum number of PCA dimensions needed to capture the majority of the data's variability.



```
Explained Variance Ratio by PCA Components:
[0.1046374  0.10031145 0.09345369 0.09065317 0.09001905 0.08487033
 0.08401255 0.08238149 0.0778307  0.06685374 0.03260116 0.02375652
 0.02084416]
Number of components to retain 95% variance: 13
```

Explained Variance vs. Number of Components
Cumulative Explained Variance
0.2
0.4
0.6
0.8
Number of Principal Components
2
4
6
8
10
12
95% Variance Threshold

Figure 10 - Cumulative Explained Variance PCA

ROC Curves for Model Performance:

- **Purpose:**
    - ROC (Receiver Operating Characteristic) curves help visualize the performance of a classification model at multiple threshold settings.
- **Description:**
    - X-axis represents the False Positive Rate (FPR).

    - Y-axis represents the True Positive Rate (TPR).
    - The Area Under the Curve (AUC) is a singular value summarizing the performance.

- **Insight:**
    - The closer the curve follows the top-left border of the ROC space, the better. A higher AUC indicates a better model capability to distinguish between classes.
- **Our case:**
    - Best models are extreme boosting and gradient boosting.

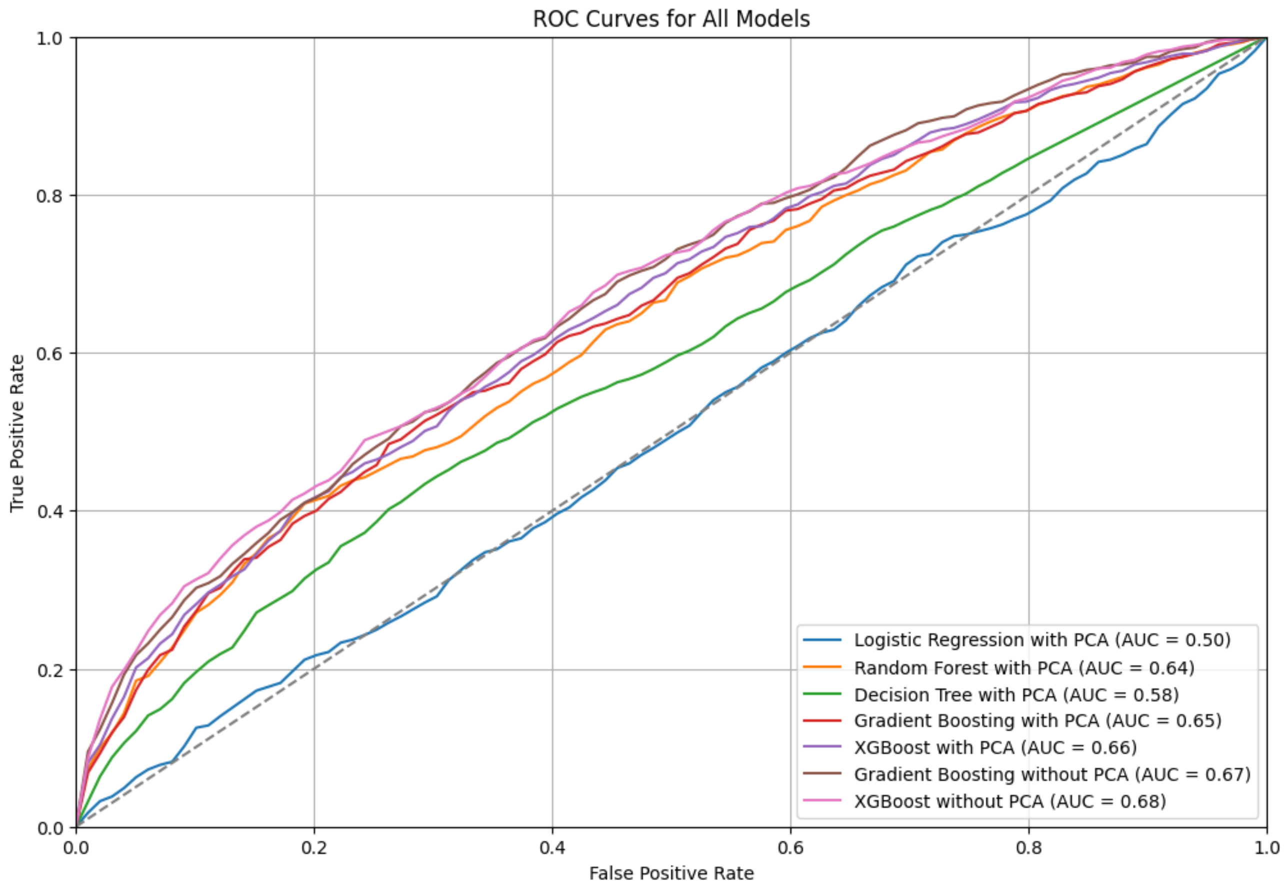


Figure 11 - ROC-AUC Curves for Model Performance

Precision-Recall Curves:

- **Purpose:**
  - Precision-Recall (PR) curves provide a deeper insight when dealing with class imbalance. They highlight the trade-off between precision and recall for different thresholds.
- **Description:**
  - X-axis shows recall.
  - Y-axis shows precision.
  - A perfect model resembles a point at (Recall=1, Precision=1).
- **Insight:**
  - Useful to understand the effectiveness of the classifier on the positive class, especially in imbalanced datasets where ROC curves might be misleading.

For gradient boosting:

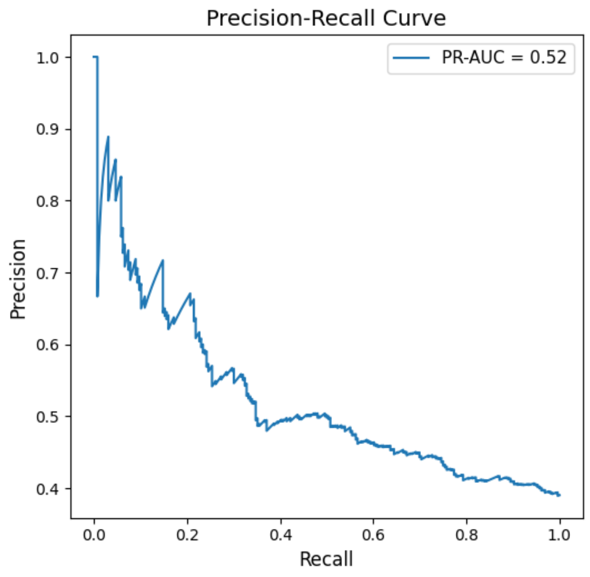


Figure 12 - Precision-Recall Curves GB

For XG Boosting:

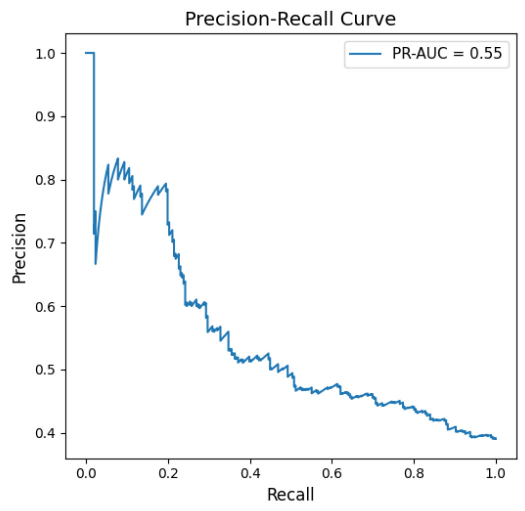


Figure 13 - Precision-Recall Curves XGB

Probability Distribution Histograms:

- **Purpose:**
  - Histograms of predicted probabilities for the positive class, indicating how confident the model is in its predictions.
- **Description:**
  - X-axis indicates the probability of classifying an instance as positive.
  - Y-axis shows the density or count of instances.
  - Separate colours are often used for actual positive and negative samples to illustrate differentiation.
- **Insight:**
  - These visualizations can confirm the separation between classes and help in determining threshold settings.

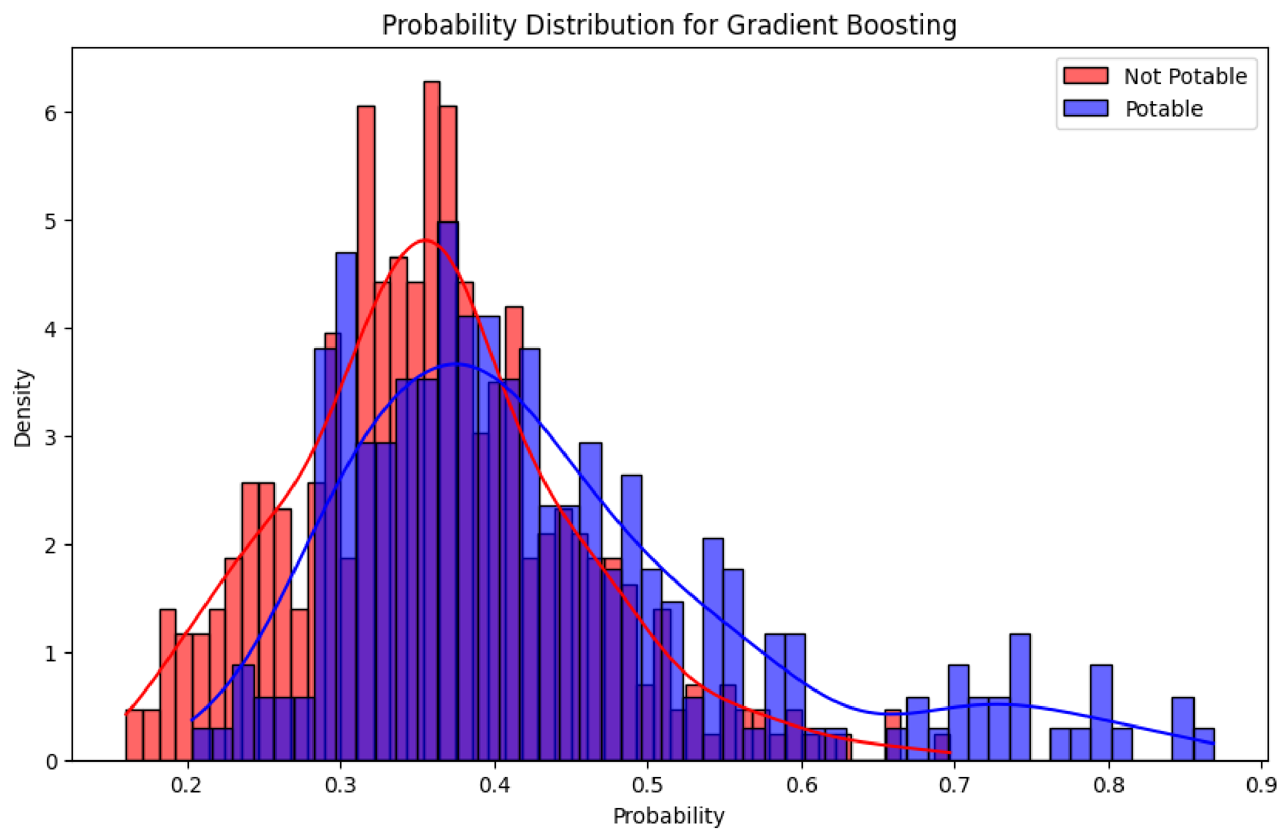


Figure 14 - Probability Distribution GB

Best threshold for Gradient Boosting: **0.28** with F1-score: **0.5910735826296744**

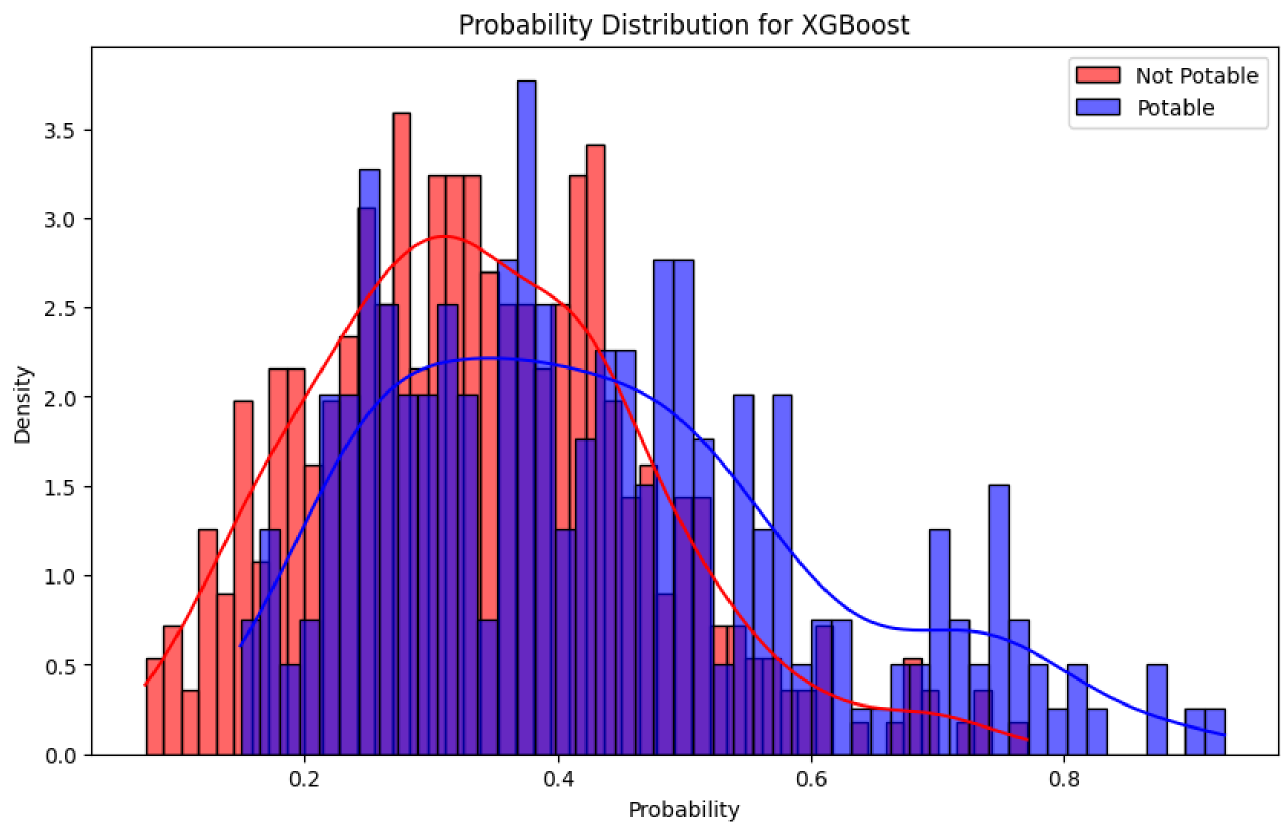


Figure 15 - Probability Distribution XGB

Best threshold for XGBoost: **0.21** with F1-score: **0.5865384615384616**

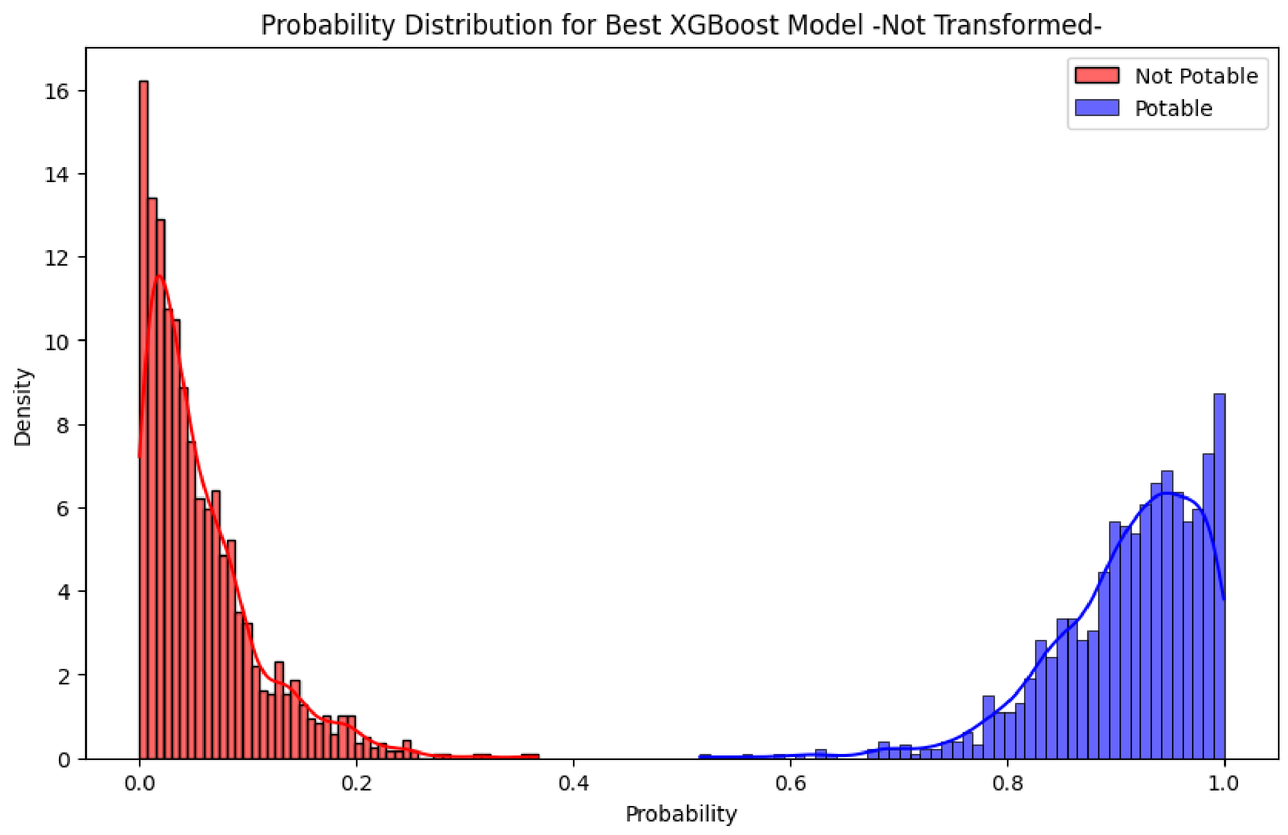


Figure 16 - Potability distribution for XG boosting only on train data

Best threshold for XG boost only on train data : **0.37** (I consider this model at the end, I will explain why in conclusion part)
As you can see this potability distributions are separated from each-other which is what we are looking for!!

Feature Importance and SHAP Analysis:

- **Purpose (Feature Importance):**
  - To identify which features are most influential in making predictions, often used in tree-based models.
- **Description:**
  - Bar plots rank features based on their contribution to model performance.
  - Each bar represents the importance score of a feature.

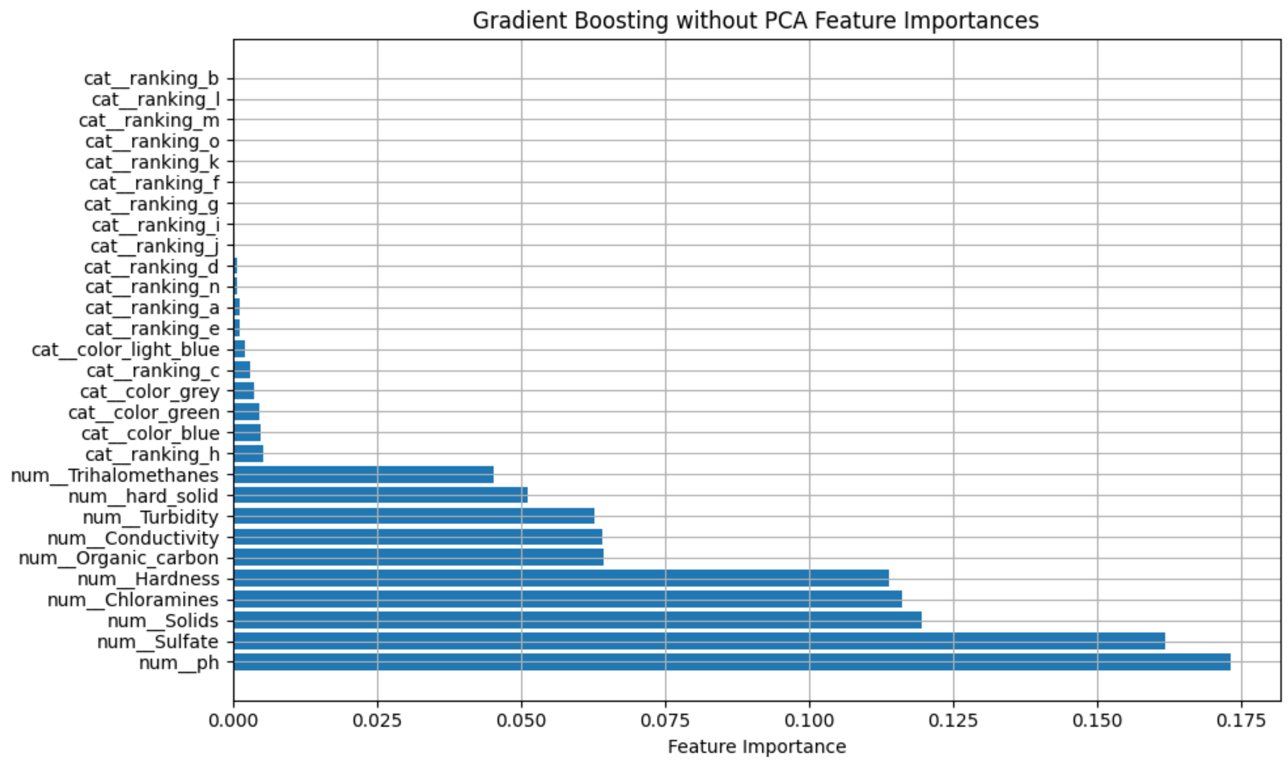


Figure 17 - Feature Importance GB

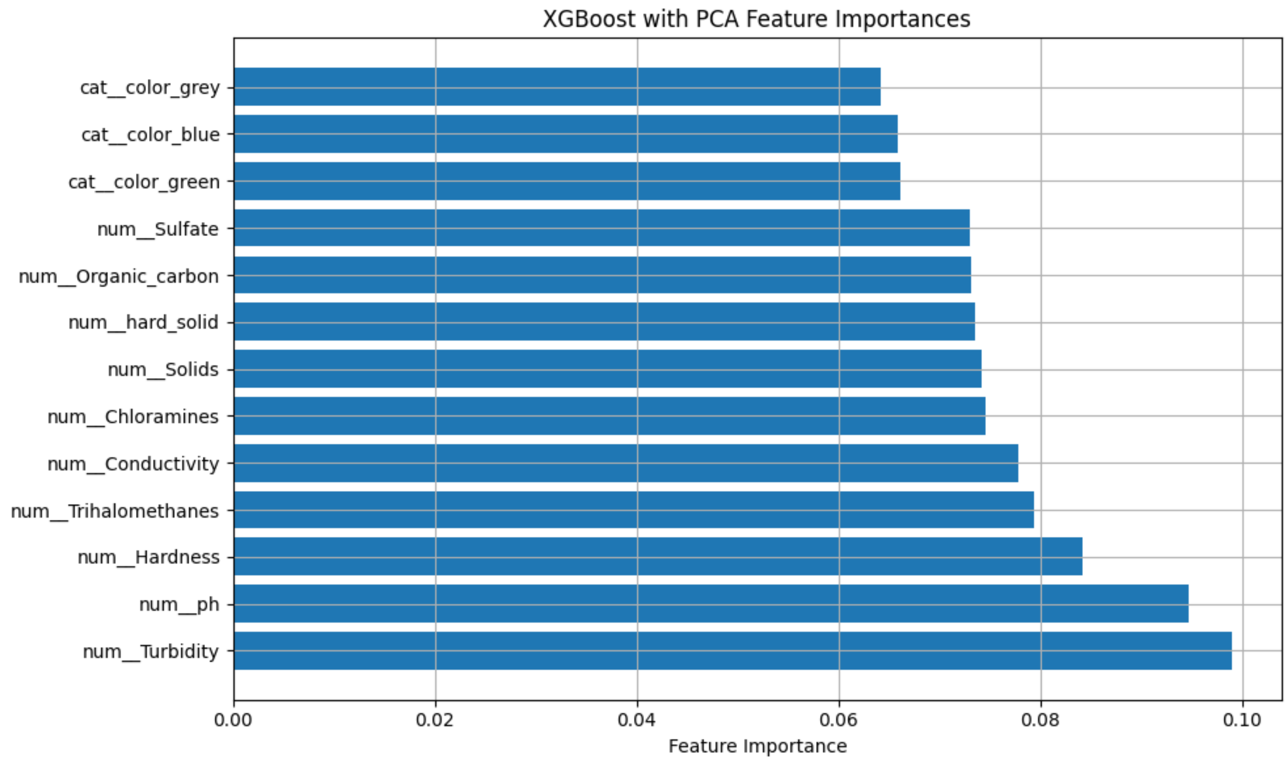


Figure 18 - Feature Importance XGB

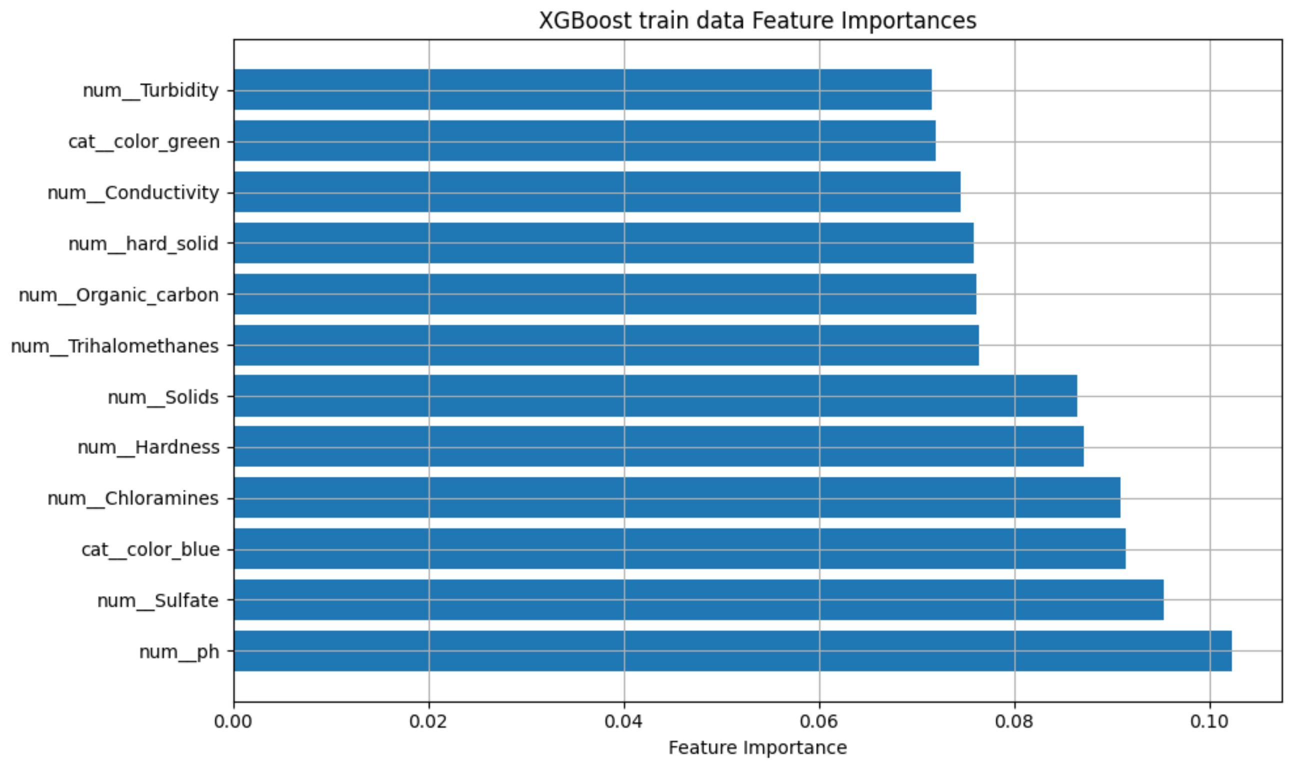


Figure 19 - XG boosting , feature importance only on train data

- **Purpose (SHAP Analysis):**
  - SHAP (SHapley Additive exPlanations) values provide insight into how much each feature contributes to each individual prediction.
- **Description (SHAP):**
  - SHAP summary plots combine feature importance with the effect of the feature on the output.
  - Points on a SHAP summary plot show the feature value's impact on a particular prediction.
- **Insight:**
  - Feature importance plots offer a macro view while SHAP values deliver a micro view, revealing how individual feature variations contribute to the model's decision-making process.

SHAP value for Gradient Boosting:

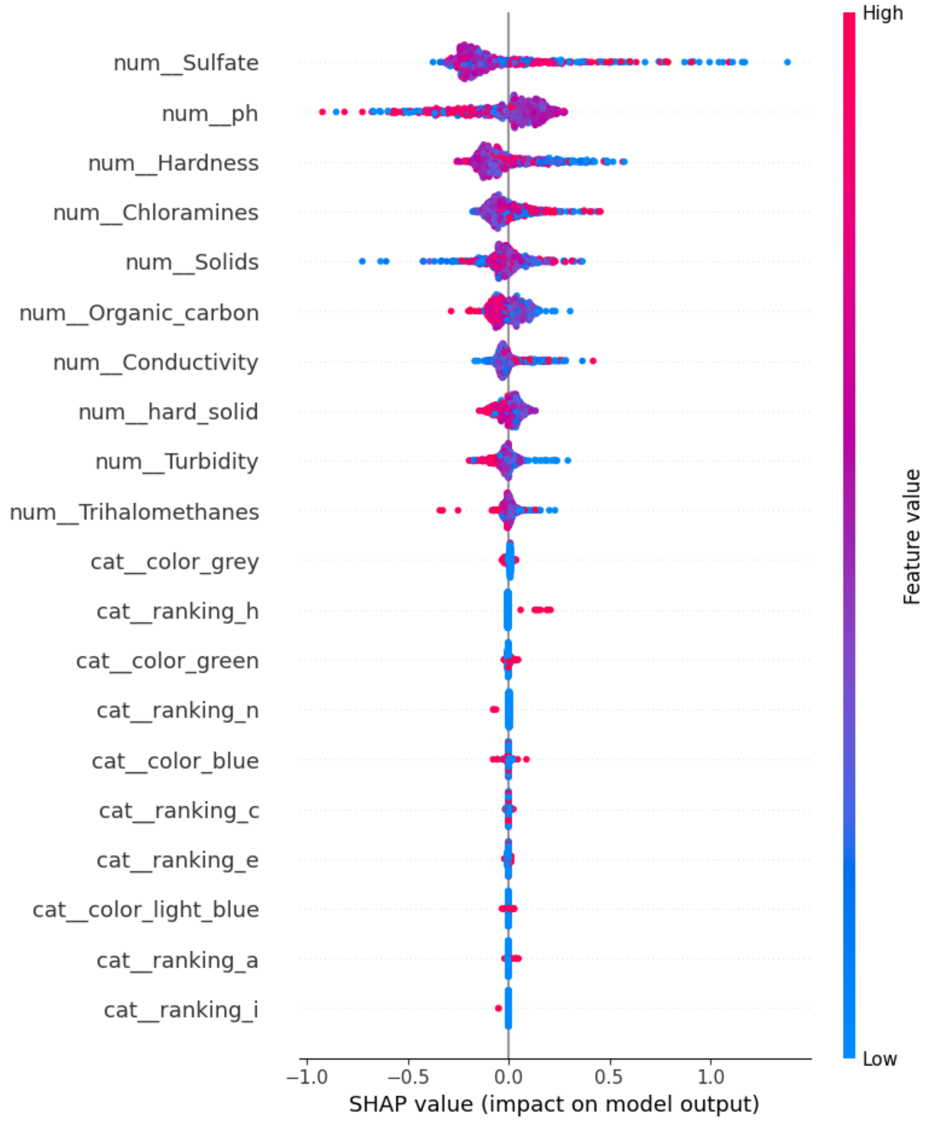


Figure 20 - SHAP value for Gradient Boosting

SHAP value for XG Boosting without PCA:

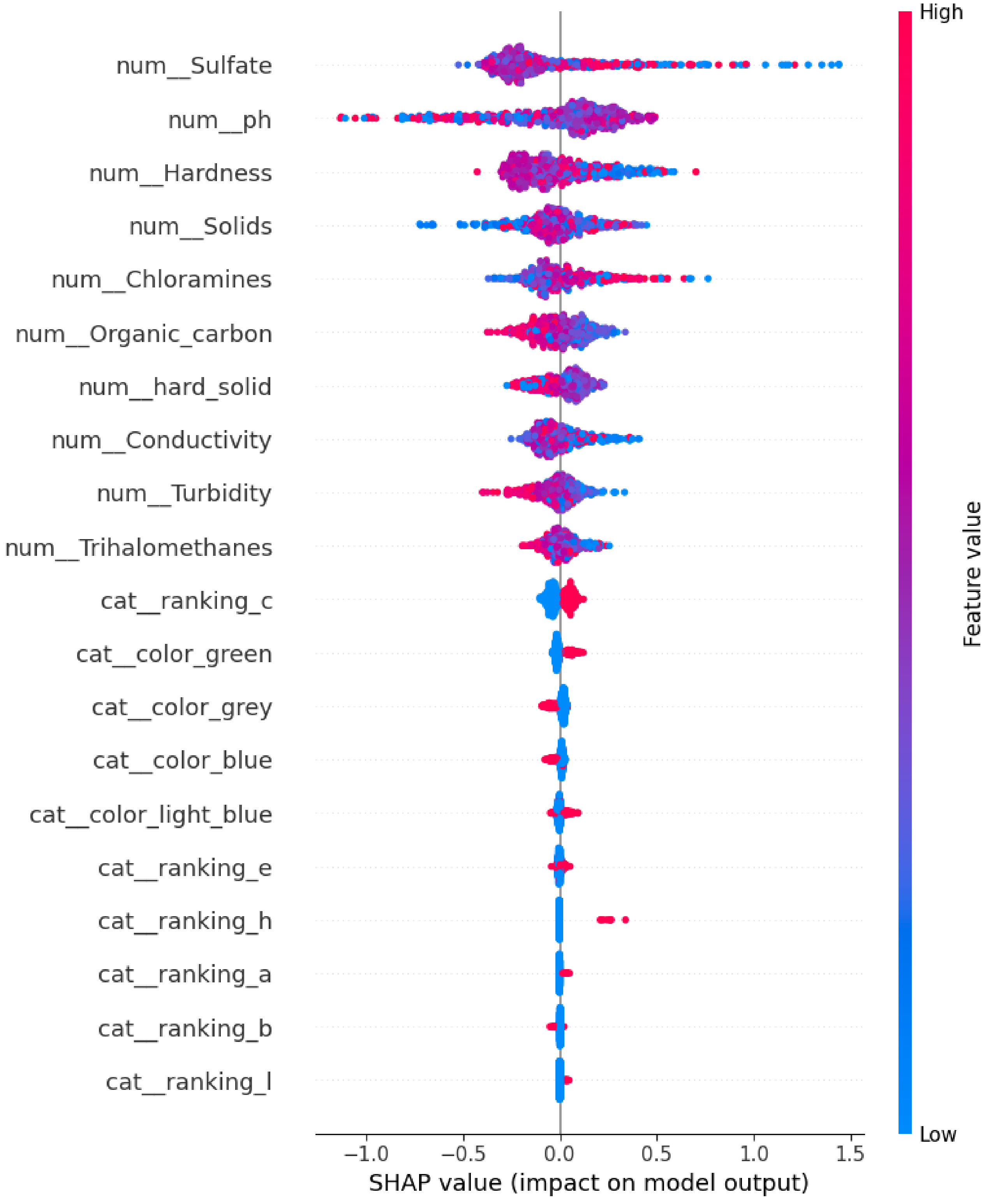


Figure 21 - SHAP value for XG Boosting

Conclusion

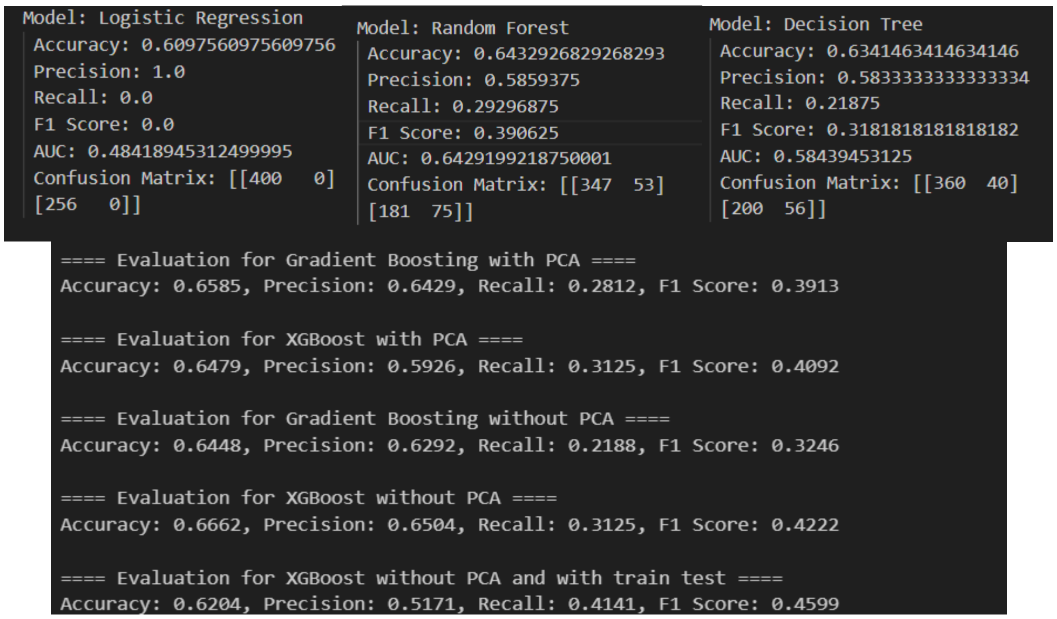
Summary of Findings:

In the analysis of the water quality dataset aimed at predicting potability, several machine learning models were trained and evaluated. Key findings include:

- **Model Performance:**

  For the potable water classification problem, **F1** is the most important metric because False Negatives (FN) and False Positive (TN) are both important with the weights that each have in precession and recall!

  The model demonstrated varying levels of accuracy, with XG Boosting achieving the highest performance, indicated by superior precision, recall, F1 Score, and ROC-AUC scores. This suggests that XG Boosting is well-suited for this classification task, effectively distinguishing between potable and non-potable water. Working on the raw train data in XG boost without filling N.A with mean, scaler , etc we got the better result as we saw in the probability distribution section which potable and not potable histograms were separated. Because XG boost is a powerful model which identifies the Missing as the 'information', not a fake data like mean! So that is why I decided to consider it for my final model.(F1 = 0.46)



```
Model: Logistic Regression
 Accuracy: 0.6097560975609756
 Precision: 1.0
 Recall: 0.0
 F1 Score: 0.0
 AUC: 0.48418945312499995
 Confusion Matrix: [[400   0]
 [256   0]]
```

```
Model: Random Forest
 Accuracy: 0.6432926829268293
 Precision: 0.5859375
 Recall: 0.29296875
 F1 Score: 0.390625
 AUC: 0.6429199218750001
 Confusion Matrix: [[347  53]
 [181  75]]
```

```
Model: Decision Tree
 Accuracy: 0.6341463414634146
 Precision: 0.5833333333333334
 Recall: 0.21875
 F1 Score: 0.3181818181818182
 AUC: 0.58439453125
 Confusion Matrix: [[360  40]
 [200  56]]
```

```
==== Evaluation for Gradient Boosting with PCA ====
Accuracy: 0.6585, Precision: 0.6429, Recall: 0.2812, F1 Score: 0.3913

==== Evaluation for XGBoost with PCA ====
Accuracy: 0.6479, Precision: 0.5926, Recall: 0.3125, F1 Score: 0.4092

==== Evaluation for Gradient Boosting without PCA ====
Accuracy: 0.6448, Precision: 0.6292, Recall: 0.2188, F1 Score: 0.3246

==== Evaluation for XGBoost without PCA ====
Accuracy: 0.6662, Precision: 0.6504, Recall: 0.3125, F1 Score: 0.4222

==== Evaluation for XGBoost without PCA and with train test ====
Accuracy: 0.6204, Precision: 0.5171, Recall: 0.4141, F1 Score: 0.4599
```

Figure 22 - Final Model Performance

- **Feature Importance:**

  Analysis revealed that features such as PH, Sulphate have the strongest influence on the model's predictions. This highlights the significance of these features in water potability, suggesting potential areas of focus for water quality management.

Recommendations for Future Work:

- **Dataset Expansion:**

  Consider acquiring more diverse and extensive data from various geographical locations to increase the generalizability of the model. More data can help in capturing rare patterns and improve the robustness of the predictions.

- **Hyperparameter Optimization:**

  Future efforts can focus on more extensive and automated hyperparameter optimization using techniques to potentially discover better model configurations.

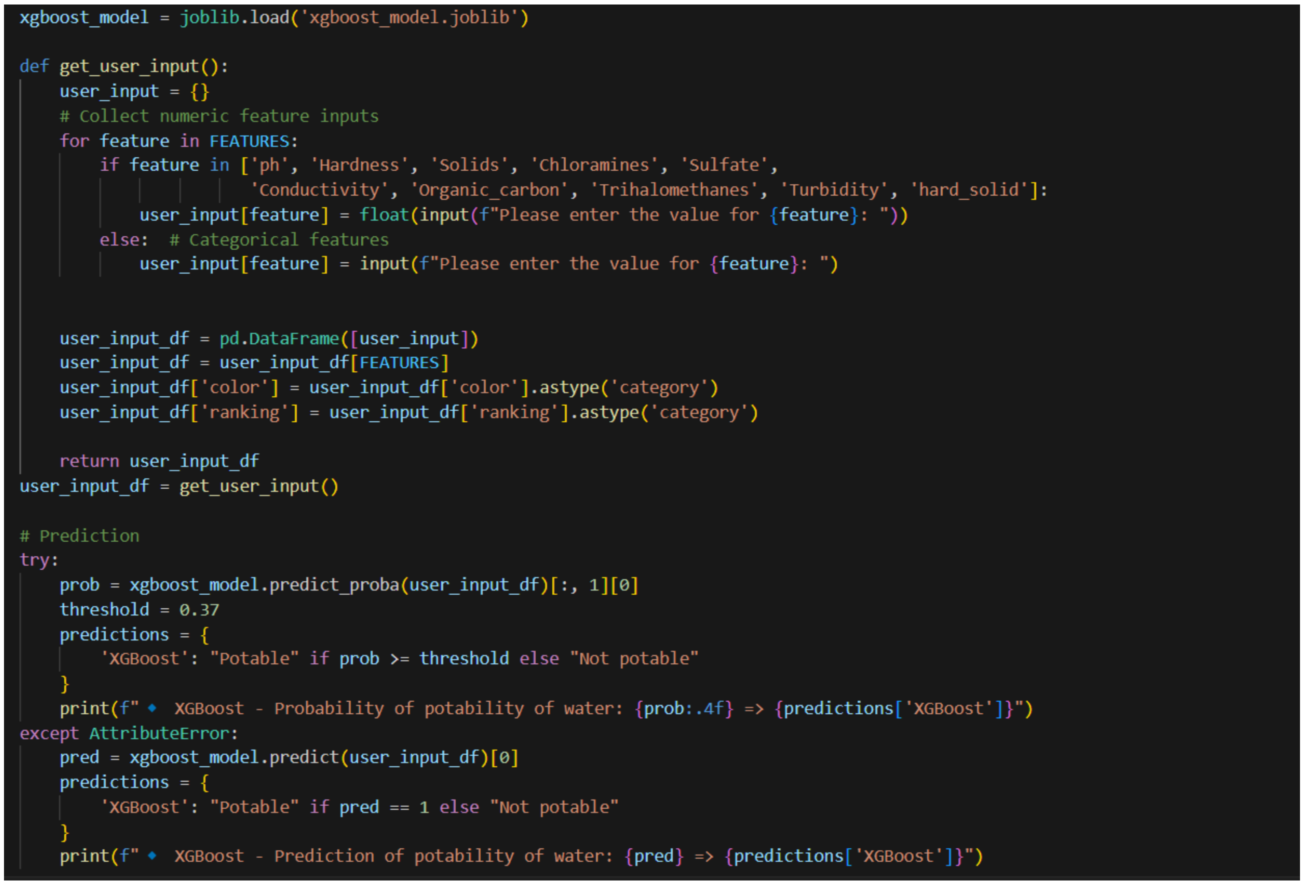
- **Real-time Prediction System:**

  Develop an integrated system that utilizes the trained models for real-time water quality monitoring, providing timely alerts and decision support to stakeholders.

New data for getting probability based on our model:

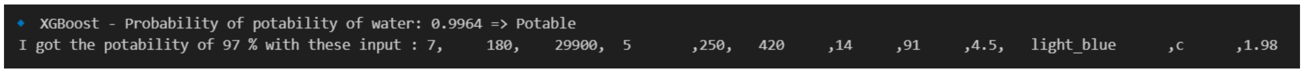
Moreover, I provided the code which by inserting the new data in my XG boosting based on train data model, we can realize the probability of having potable water.

The figure shows one example which was so near to not potable one:



```python
xgboost_model = joblib.load('xgboost_model.joblib')

def get_user_input():
    user_input = {}
    # Collect numeric feature inputs
    for feature in FEATURES:
        if feature in ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
                       'Conductivity', 'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'hard_solid']:
            user_input[feature] = float(input(f"Please enter the value for {feature}: "))
        else:  # Categorical features
            user_input[feature] = input(f"Please enter the value for {feature}: ")


    user_input_df = pd.DataFrame([user_input])
    user_input_df = user_input_df[FEATURES]
    user_input_df['color'] = user_input_df['color'].astype('category')
    user_input_df['ranking'] = user_input_df['ranking'].astype('category')

    return user_input_df
user_input_df = get_user_input()

# Prediction
try:
    prob = xgboost_model.predict_proba(user_input_df)[:, 1][0]
    threshold = 0.37
    predictions = {
        'XGBoost': "Potable" if prob >= threshold else "Not potable"
    }
    print(f"🔹 XGBoost - Probability of potability of water: {prob:.4f} => {predictions['XGBoost']}")
except AttributeError:
    pred = xgboost_model.predict(user_input_df)[0]
    predictions = {
        'XGBoost': "Potable" if pred == 1 else "Not potable"
    }
    print(f"🔹 XGBoost - Prediction of potability of water: {pred} => {predictions['XGBoost']}")
```

Figure 23 - New data for getting probability based on our model



```
🔹 XGBoost - Probability of potability of water: 0.9964 => Potable
I got the potability of 97 % with these input : 7,     180,    29900,   5        ,250,    420      ,14      ,91      ,4.5,    light_blue      ,c       ,1.98
```

Figure 24 - final result

By summarizing the results and proposing avenues for future research and application, this section not only concludes the current work but also sets a roadmap for how these findings can be expanded and utilized effectively.

Thank you for your attention,

Behshad(Beth) Ghaseminezhadabdolmaleki

30/03/2025