Introduction

Heavy metal contamination in drinking water poses a significant threat to public health and environmental sustainability. Among these contaminants, **copper (Cu^{2+})** is a double-edged element—essential as a trace micronutrient but toxic at elevated concentrations. Prolonged exposure to excess copper has been linked to gastrointestinal disorders, liver damage, oxidative stress, and neurotoxicity [1], [2]. In areas adjacent to industrial zones, mining operations, and aging infrastructure, copper levels in water often exceed safe thresholds, posing serious risks to affected populations [3].

Regions such as **Balochistan, Pakistan**, have reported copper contamination in groundwater, particularly near mining sites, where elevated levels were detected alongside other heavy metals. Such cases underscore the urgent need for decentralized, low-cost water quality monitoring solutions to safeguard vulnerable communities.

To mitigate such risks, organizations including the **World Health Organization (WHO)** and the **U.S. Environmental Protection Agency (EPA)** have established maximum allowable copper concentrations of **2 mg/L (≈31.5 µM)** and **1.3 mg/L (≈20.5 µM)**, respectively [4], [5]. While conventional detection methods—such as **Flame Atomic Absorption Spectrometry (FAAS)** and **Inductively Coupled Plasma Atomic Emission Spectrometry (ICP-AES)**—offer high accuracy, they require laboratory-grade instrumentation, trained personnel, and are not feasible for real-time, field-level deployment [6].

In response to these limitations, optical biosensing technologies, including colorimetric and fluorometric detection, have gained attention for their portability and low-cost implementation. Colorimetric sensors yield visible changes in color intensity, while fluorometric sensors detect variations in fluorescence emission under UV excitation [7]. Despite their advantages, these approaches still rely on manual observation, making them sensitive to environmental conditions and prone to subjective interpretation.

A recent advancement was presented by **Pizzoferrato et al.**, who developed an agar-agar-based dual-mode biosensor using *o*-phenylenediamine-derived carbon dots capable of responding to Cu^{2+} through both colorimetric and fluorescent changes [8]. Dr. Ramanand Bisauriya, co-author of this study, contributed significantly to the biosensor development. While dual-mode detection improves flexibility, interpretation remains dependent on human evaluation.

To address these challenges, we propose a **machine learning (ML)-based framework** that automates the classification of copper ion concentrations from biosensor images. Our system leverages both colorimetric and fluorometric imaging modes to provide rapid, scalable, and interpretable water quality assessments.

A dataset of **5010 images per modality** was generated through interpolation of six base copper concentrations (0, 20, 50, 100, 200, and 500 µM), followed by augmentation using geometric and photometric transformations. This resulted in **501 discrete micromolar concentration levels (0–500 µM)**. Various color spaces were evaluated for feature extraction, with **RGB** selected for its superior consistency and classification performance.

Multiple lightweight supervised classifiers were trained and evaluated, including **Logistic Regression**, **Support Vector Machine (SVM)**, **Random Forest**, and **XGBoost**. Unlike regression models, our system performs discrete classification into five practical water quality categories based on copper concentration, as summarized in **Table 1**.

Table 1: Water Quality Risk Levels Based on Copper Ion Concentration

Water Quality Level	**Concentration Range (µM)**	**Interpretation**
Safe	0–19 µM	Suitable for drinking water
Slightly Contaminated	20–39 µM	Minor contamination detected
Moderately Contaminated	40–59 µM	Moderate risk; caution advised
Unsafe	60–99 µM	Unsafe for prolonged consumption
Heavily Contaminated	100–500 µM	Dangerous; immediate action needed

Materials and Methods

Biosensor Fabrication and Imaging

The biosensors used in this study were developed and characterized in prior work by Pizzoferrato et al. [8], co-authored by Dr. Ramanand Bisauriya. These dual-mode biosensors consist of agar-agar hydrogel matrices embedded with *o*-phenylenediamine-derived carbon dots, which exhibit distinct colorimetric and fluorometric responses upon exposure to copper ions (Cu^{2+}). The sensors show a visible color change under white light illumination and a fluorescence intensity change under UV excitation, enabling two complementary optical detection modes.

In this study, we did not fabricate new biosensors or conduct additional physical experiments. Instead, we utilized pre-acquired and published biosensor images provided by Pizzoferrato et al. [8]. These images were captured under standardized lighting conditions to ensure consistency across different copper concentration levels.

The dual-mode optical behavior of the biosensors is illustrated in Figure 1 and Figure 2. Under white light, the biosensors exhibit a visible gradation in color intensity corresponding to the copper concentration. Under UV excitation, a progressive decrease in fluorescence brightness is observed with increasing copper ion levels. These optical changes form the foundation for our machine learning-based classification approach. The images are referenced from the work of Pizzoferrato et al. [8].

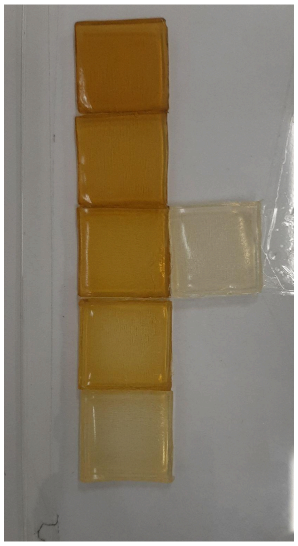


Figure 1: Colorimetric response of the biosensor under white light illumination for different copper ion concentrations (Adapted from [8]).

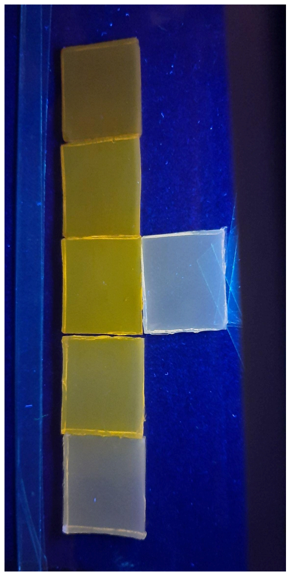


Figure 2: Fluorometric response of the biosensor under UV excitation for different copper ion concentrations (Adapted from [8]).

The image dataset served as the basis for the machine learning pipeline. From the original six base concentration images, additional interpolated samples were digitally generated to cover 501 unique micromolar levels between 0 and 500 μM. Each interpolated image was then subjected to nine augmentation transformations—including rotations, brightness adjustments, flips, and scaling—to simulate real-world variability. This resulted in a total of 5010 images per sensing modality, providing a large and balanced dataset for model training and evaluation.

MACHINE LEARNING

- ## Initialization and Data Preparation

  - In this machine learning project aimed at detecting heavy metals, we begin by setting up known concentrations of heavy metals in micromolar (µM) units.
  - These concentrations, specifically 0, 10, 50, 100, 200 and 500 µM, are initialized to create a structured dataset.
  - Each concentration level reflects different degrees of heavy metal presence, which will eventually help the model learn how to differentiate between varying concentrations.
  - At this stage, setting up these known values is essential to provide a baseline for analyzing changes in image characteristics.

- ## Visual Data Preparation and Display

  - As this project relies on colorimetric and fluorometric image analysis, we start by preparing and displaying a set of images representing each concentration level.
  - By displaying these images, we can visually verify that the initial setup is correct and ensure that each concentration is accurately represented in the images.
  - This step is essential for confirming that the data we will feed into the model is correctly labeled and consistent with the intended concentration levels.
  - Displaying the images at this point serves two main purposes:
    - **Verifying** that the images align with the specified concentrations.
    - **Ensuring** that the visual data captures the differences in color or brightness associated with each heavy metal level.
  - This setup forms the foundation for further analysis, allowing us to proceed with confidence in the accuracy of our data before moving on to more complex processing steps.

Color Models

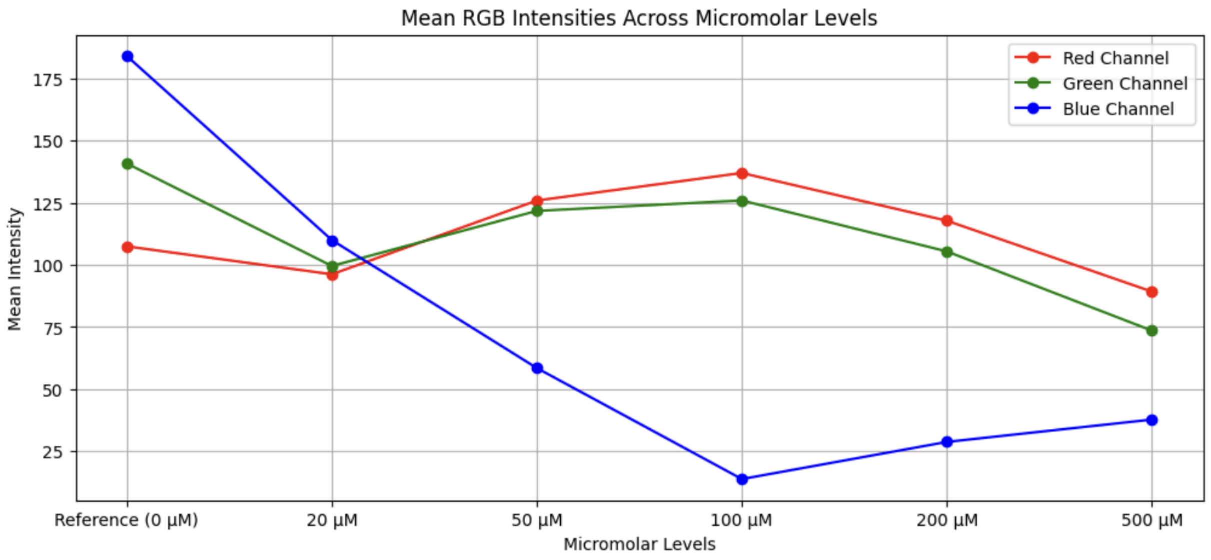
1. RGB (Red, Green, Blue)

About the Color Model

RGB is an **additive color model** that represents colors by combining varying intensities of Red, Green, and Blue light. This model is primarily used for digital displays such as computer monitors, television screens, and cameras. When combined at full intensity, the three colors produce white; when none of the colors are present, the result is black.

How Does RGB Work?

- **Additive Process**: In RGB, the color is created by adding the three primary colors of light (Red, Green, and Blue).
- **Intensity Control**: Each of the RGB components has an intensity value, usually ranging from 0 to 255 (for 8-bit color depth). Higher values correspond to brighter colors.
- **Color Creation**: By adjusting the intensity of each color, you can create any color. For example:
    - Red = (255, 0, 0)
    - Green = (0, 255, 0)
    - Blue = (0, 0, 255)
- **Combination**: When all three components are at their maximum (255, 255, 255), the result is **white**. When all are at zero, the result is **black**.



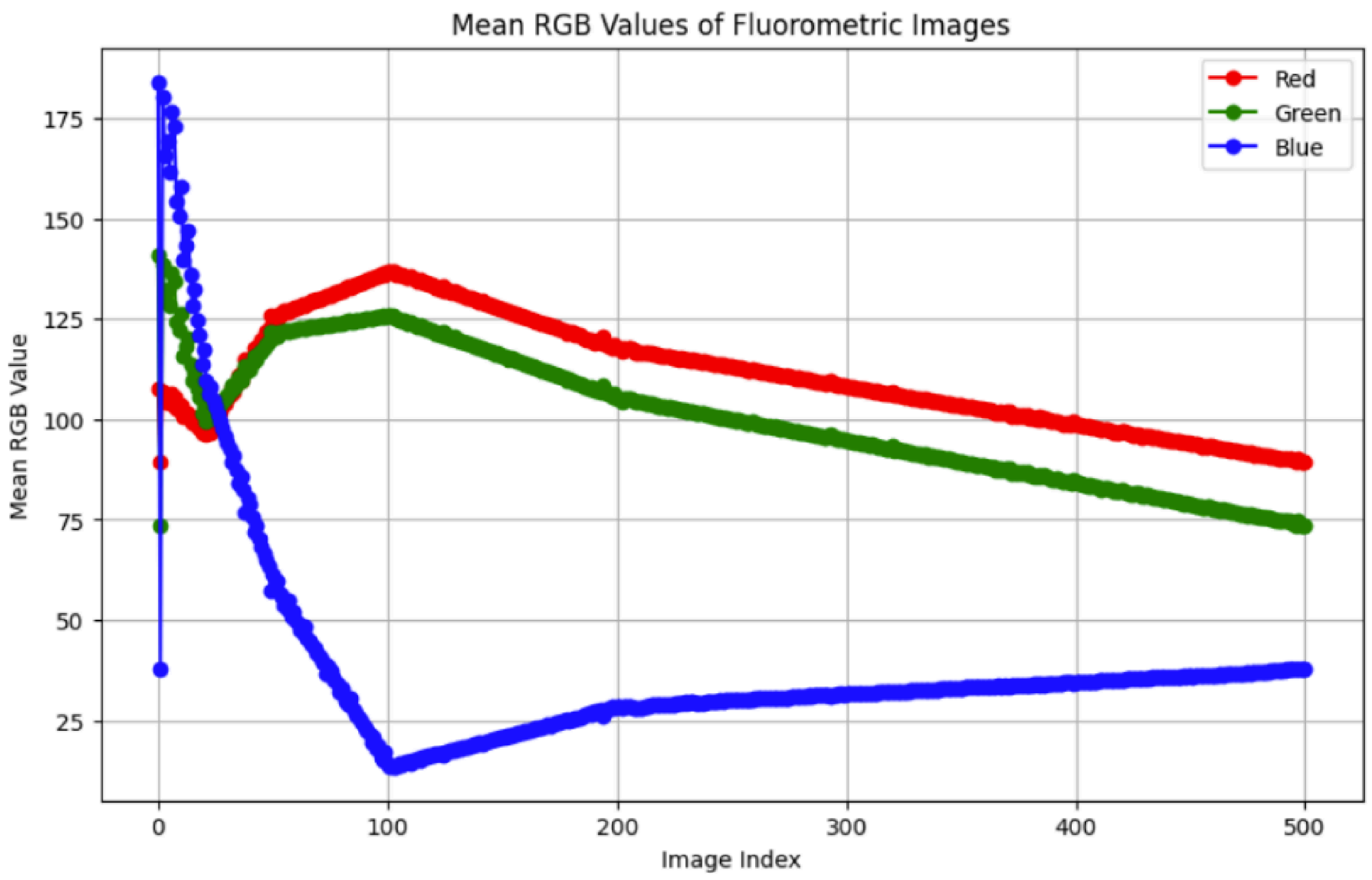


Figure 4 : Mean RGB channel intensities across copper ion concentration levels.

2. CMYK (Cyan, Magenta, Yellow, Key/Black)

About the Color Model

CMYK is a **subtractive color model** used in color printing. It works by subtracting varying percentages of Cyan, Magenta, and Yellow inks from a white background. The "Key" (K) refers to the black ink used to enhance color depth and detail in the printed image. This model is standard for printing, particularly in the production of images on paper.

How Does CMYK Work?

- **Subtractive Process**: CMYK uses the subtractive method, where the colors are created by **removing** specific wavelengths of light using inks.

- **Ink Mixing**: By mixing Cyan, Magenta, and Yellow inks, different colors are created. The more ink you use, the more light is absorbed (subtracted) from the white background.

- **Black Ink**: While mixing Cyan, Magenta, and Yellow together ideally produces a dark brown, it's inefficient, so **Black (K)** is added for true depth and a richer, darker color.

- **Color Creation**: The levels of Cyan, Magenta, Yellow, and Black determine the final color in print.

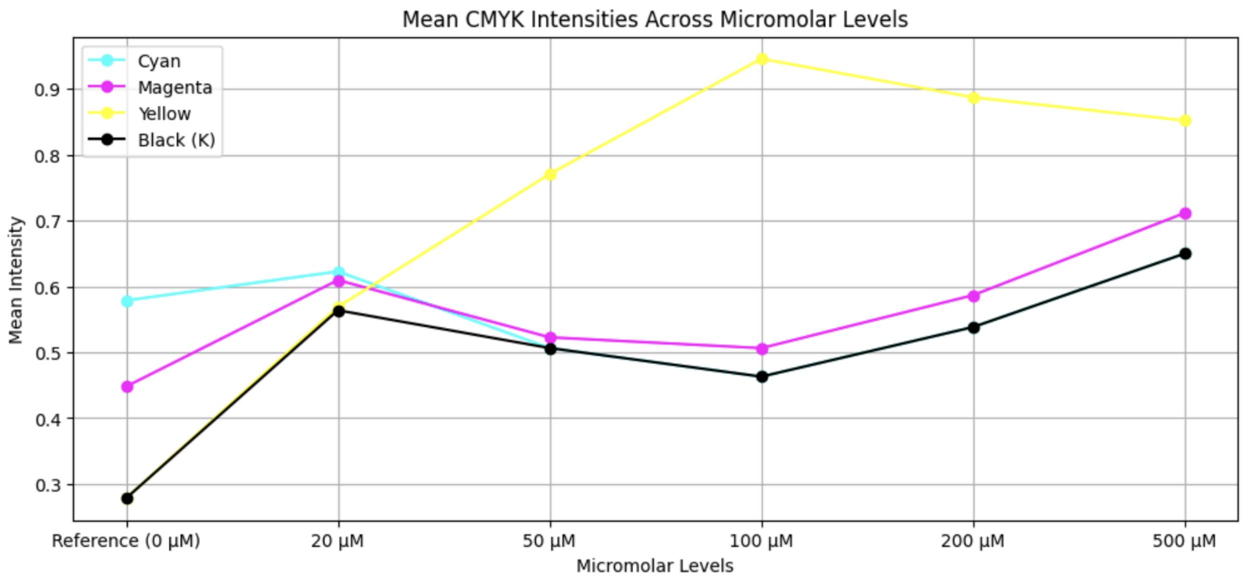


Figure 5 : Mean CMYK channel intensities across copper ion concentration levels.

3. HSV (Hue, Saturation, Value)

About the Color Model

HSV is a **cylindrical representation** of the RGB color model and is more intuitive for humans to understand. It’s widely used in **graphic design** and **image editing** software. HSV focuses on how we perceive colors in terms of **Hue** (type of color), **Saturation** (intensity), and **Value** (brightness).

How Does HSV Work?

- **Hue**: Represents the **color type**, measured in degrees (0 to 360). For example, 0° corresponds to Red, 120° to Green, and 240° to Blue.

- **Saturation**: Indicates the **intensity or vividness** of the color. A high saturation means the color is pure and intense, while low saturation results in a washed-out or grayish color.

- **Value**: Refers to the **brightness** of the color. A value of 0 means the color is completely black, and a value of 100 means the color is at its brightest form.

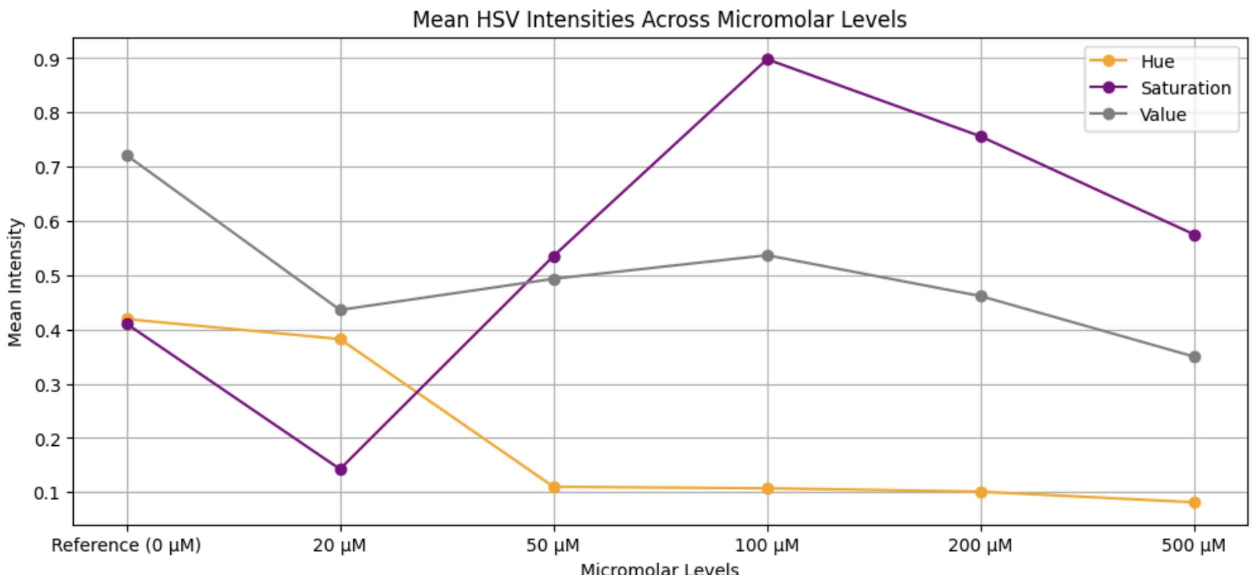


Figure 6: Mean HSV channel intensities across copper ion concentration levels.

4. YCbCr (Luma, Chroma Blue, Chroma Red)

About the Color Model

YCbCr is a color model commonly used in **video compression** and **broadcast television**. It separates the brightness (luminance) information from the color (chrominance) information, which allows for more efficient compression. YCbCr is often used in video encoding formats like JPEG, MPEG, and others.

How Does YCbCr Work?

- **Luma (Y)**: Represents the **brightness** of the image and is derived from the RGB values. It captures the grayscale information.
- **Chroma Blue (Cb)**: Represents the **blue color component**.
- **Chroma Red (Cr)**: Represents the **red color component**.
- **Chroma Subsampling**: YCbCr often uses **subsampling**, where the chrominance (Cb and Cr) components are stored at lower resolution than the luma component. This is because human eyes are more sensitive to changes in brightness than to changes in color.

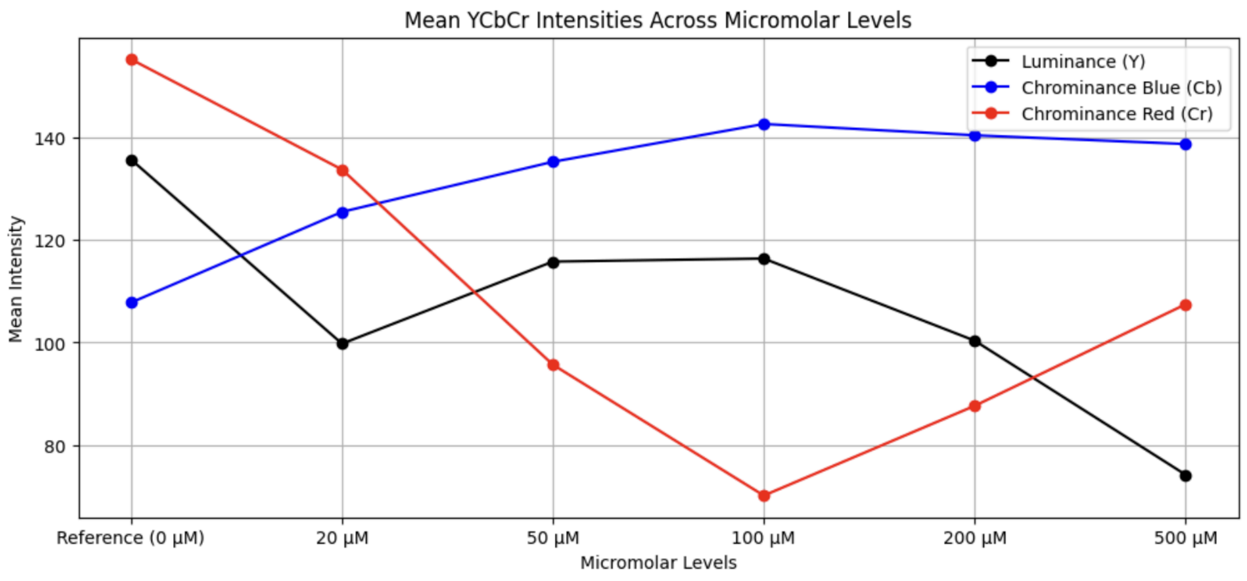


Figure 7: Mean YCbCr channel intensities across copper ion concentration levels.

To simplify the evaluation process, color space comparisons were conducted only on the fluorometric images. Since both the colorimetric and fluorometric biosensor images primarily captured changes in color intensity without substantial structural variation, it was considered unnecessary to evaluate color spaces separately for both modalities. We arbitrarily selected the fluorometric dataset for comparison. Once the color model was finalized based on fluorometric images, the same selection was applied to the colorimetric dataset as well, where it yielded consistent and satisfactory results.

After analyzing the mean channel intensity variations across different color spaces (Figures 5–8) and evaluating preliminary classification performances (Figure 9), **RGB** and **HSV** were identified as the most promising candidates for further exploration. Although RGB showed greater consistency and superior accuracy, HSV offered theoretical advantages such as partial illumination invariance. To ensure comprehensive evaluation, two expanded datasets were generated: one using RGB and the other using HSV representations, each comprising **5010 images per modality** after interpolation and augmentation.

A performance comparison across different color spaces is summarized in **Figure 9**.
It was observed that **RGB images** preserved the relevant visual features most consistently across varying copper concentrations. In contrast, **HSV-transformed images** introduced unwanted **green-tint artifacts**, particularly at lower concentration or highly fluorescent regions, which adversely impacted model performance, as illustrated in **Figure 10**.

Additionally, models trained on HSV, YCbCr, and CMYK images demonstrated slightly lower classification accuracy compared to those trained on RGB images. Although alternative color spaces such as HSV offer theoretical benefits, in this dataset they introduced more noise than useful feature separation.

Based on these comparative analyses, **RGB was selected** as the preferred color space for all downstream modeling due to its superior **feature stability**, **higher classification accuracy**, and **better visual interpretability.**

Interpolation in Image Generation

Interpolation is used to estimate values between known data points. In the case of image generation, the known data points are the reference images at specific concentrations (0, 20, 50, 100, 200, and 500). Interpolation helps to generate images for every concentration in between these reference points, allowing you to create a smooth transition between concentrations.

For example, if you have images at concentrations 0, 20, 50, 100, 200, and 500, interpolation allows you to estimate the image at concentrations like 1, 2, 3, ... up to 499. This provides you with a **continuous range of images** that represent all concentrations between 0 and 500.

Interpolation Workflow

1. **Data Points (Reference Images)**: You have 6 reference images corresponding to concentrations 0, 20, 50, 100, 200, and 500.

2. **Determine Interpolation Method**: Choose an interpolation technique based on the nature of the data:

    ○ **Linear Interpolation**: For simple transitions between two concentrations.

    ○ **Spline or Polynomial Interpolation**: For smoother, more complex transitions between concentrations.

3. **Generate Intermediate Concentrations**:

    ○ For each pair of consecutive known concentrations (e.g., 0 and 20, 20 and 50, etc.), use the chosen interpolation method to calculate the pixel values for intermediate concentrations.

    ○ Repeat this process for each pair of adjacent concentrations from 0 to 500.

4. **Calculate Interpolated Values**:

    ○ For each intermediate concentration, use the interpolation method to estimate pixel values. For example, for concentration 10 (between 0 and 20), you calculate the corresponding pixel values using interpolation.

5. **Generate Intermediate Images**:

    ○ Based on the calculated pixel values for each intermediate concentration, generate the images.

    ○ You will now have a set of images for all concentrations from 0 to 500, including those in between the reference points.

6. **Resulting Images**:

    ○ The final set of images will exhibit smooth transitions between the given concentrations (0, 20, 50, 100, 200, and 500) and will cover every concentration in the range from 0 to 500.

Image Augmentation: Overview

Image augmentation is a technique used in **machine learning** and **computer vision** to artificially increase the size of a dataset by applying various transformations to the original images. This helps improve the robustness and generalization ability of models by introducing variety in the training data. In your case, after interpolating images for different concentrations, you augmented these images using several transformations.

Why is Image Augmentation Important?

- **Increases Dataset Diversity**: By applying transformations to the original images, you create new variants that simulate real-world variations, improving model performance.
- **Prevents Overfitting**: Augmentation helps prevent the model from memorizing the dataset by introducing variability and challenging the model to learn more generalized features.
- **Simulates Real-World Variations**: It mimics real-world changes like rotations, flips, and scaling that could occur in actual conditions, thus making the model more adaptable.

How Image Augmentation is Done Using the Given Factors

Here's how augmentation is performed:

1. Rotation (by –5° and –10°, +5° and +10°)

- **Description**: Rotation involves turning the image around a fixed point, usually the center. By rotating the image by different degrees, you create variations in the angle at which the object or substance in the image is viewed.
- **How It Works**:
    - **Negative Rotation**: Rotating the image by -5° or -10° will tilt the image to the left.
    - **Positive Rotation**: Rotating by +5° or +10° tilts the image to the right.
- **Purpose**: Rotation helps make the model invariant to changes in the orientation of objects or substances in the image. It introduces slight rotations to simulate how the object might appear when viewed from different angles.

2. Horizontal Flipping

- **Description**: Horizontal flipping mirrors the image along its vertical axis (left-to-right flip), creating a horizontally flipped version of the image.
- **How It Works**:
    - The image is flipped horizontally, meaning that the left part of the image becomes the right, and vice versa. This transformation doesn't change the content but changes the position of

the elements within the image.

- **Purpose**: Horizontal flipping ensures that the model can recognize objects or substances regardless of their left or right orientation. It simulates real-world scenarios where an object might appear in different directions.

3. Vertical Flipping

- **Description**: Vertical flipping mirrors the image along its horizontal axis (top-to-bottom flip), creating a vertically flipped version of the image.
- **How It Works**:
    - The top and bottom parts of the image are swapped, so the top half becomes the bottom half, and vice versa.
- **Purpose**: This helps in scenarios where the object could appear upside down or in flipped orientations. Vertical flipping also makes the model more invariant to vertical reflections.

4. Brightness Increase

- **Description**: Increasing the brightness of an image involves enhancing the light intensity across all pixels, making the image appear lighter and more illuminated.
- **How It Works**:
    - The pixel values of the image are increased by a constant factor or percentage, making all areas of the image appear brighter.
- **Purpose**: Brightness adjustment helps the model adapt to images taken under different lighting conditions. Sometimes, images in real-world scenarios are brighter due to overexposure, and this technique ensures that the model can still detect patterns under varying lighting.

5. Brightness Decrease

- **Description**: Decreasing the brightness of an image makes the image appear darker, reducing the light intensity across all pixels.
- **How It Works**:
    - The pixel values of the image are reduced by a constant factor or percentage, which darkens the entire image uniformly.
- **Purpose**: Reducing brightness simulates low-light or poorly lit scenarios. This technique ensures the model can handle images with dim lighting or shadows, where certain features may be less visible.

6. Geometric Scaling (Cropping followed by Resizing)

- **Description**: Geometric scaling involves modifying the size of the image. It can be done by first **cropping** a portion of the image and then **resizing** it to a specified dimension.
- **How It Works**:
    - **Cropping**: A random portion of the image is cut out, removing the edges or any specific part of the image. This simulates zooming into a particular region or excluding some context around the object.
    - **Resizing**: After cropping, the image is resized to the original dimensions or to a predefined size. Resizing can be done either by maintaining the aspect ratio or distorting the image to fit the target size.
- **Purpose**: Geometric scaling helps the model learn to recognize objects or substances at different zoom levels. Cropping simulates varying object sizes, and resizing ensures the model adapts to various aspect ratios, enhancing its ability to generalize.

Augmentation Workflow

1. **Start with Interpolated Images**: Begin with the interpolated images for concentrations 0, 1, 2, ..., 500 that you generated through interpolation.
2. **Apply Rotation**: Rotate each image by -5°, -10°, +5°, and +10°. This will give you four new images per concentration with different orientations.
3. **Apply Horizontal and Vertical Flipping**: For each image, apply horizontal and vertical flipping, which generates two new flipped versions of each image per concentration.
4. **Apply Brightness Changes**:
    - Increase the brightness of each image, resulting in a lighter version of the image.
    - Decrease the brightness of each image, generating a darker version.
5. **Apply Geometric Scaling**:
    - Crop a random portion of the image.
    - Resize the cropped portion back to the original dimensions (or to a new size).
6. **Final Set of Augmented Images**: After applying all the transformations, you will have **9 augmented images** for each concentration (1 from the original and 8 from transformations). The total number of augmented images will be **9 times the number of original interpolated images**.

Transformation	**Number of Variants Created**	**Effect**
Rotation by -5° and -10°	2	Creates slight leftward rotation variations.
Rotation by +5° and +10°	2	Creates slight rightward rotation variations.
Horizontal Flipping	1	Mirrors the image along the vertical axis.
Vertical Flipping	1	Mirrors the image along the horizontal axis.
Brightness Increase	1	Makes the image brighter.
Brightness Decrease	1	Makes the image darker.
Geometric Scaling (Cropping & Resizing)	1	Zooms in on the image, then resizes.

This results in **9 augmented images and 1 original image per concentration**, giving you a total set of **10 times the number of interpolated images** for training the model.

Machine Learning Models

1. Support Vector Machine (SVM)

About the ML Model

Support Vector Machine (SVM) is a supervised learning algorithm primarily used for classification tasks, but it can also be adapted for regression problems. The core idea behind SVM is to find the best boundary (hyperplane) that separates data points of different classes with the maximum margin. The margin is the distance between the hyperplane and the nearest data points from either class, which are called support vectors. SVM can also handle complex datasets by using kernel functions that transform data into a higher-dimensional space where separation is easier.

Workflow of SVM

- **Data Input**: The model receives labeled training data with feature inputs and their corresponding classes.
- **Identify Optimal Boundary**: It tries to find the hyperplane that best separates the classes with the maximum possible margin.
- **Handling Non-linear Data**: When data is not linearly separable, SVM uses kernel functions to transform data into a higher-dimensional space.
- **Training**: The model identifies the support vectors and learns the decision boundary based on them.
- **Prediction**: For new data points, the model determines on which side of the boundary they lie and predicts the class accordingly.

Why to Use SVM

- Very effective in high-dimensional spaces and with datasets having a clear margin of separation.
- Works well for both linear and non-linear problems.
- Highly reliable when the number of features is large compared to the number of samples.
- Memory efficient as it uses only a subset of training points (support vectors).

2. Decision Tree

About the ML Model

A Decision Tree is a simple yet powerful model used for classification and regression. It works like a flowchart where data is split based on certain conditions. Each internal node represents a test on an attribute, each branch represents an outcome, and each leaf node represents a final decision or prediction. Decision Trees are easy to understand, visualize, and interpret, making them very popular in business and operational applications.

Workflow of Decision Tree

- **Data Input**: The model takes training data with features and output labels.
- **Splitting Data**: At each node, the data is split based on the feature that provides the best division, usually by measuring purity using metrics like information gain or Gini impurity.
- **Building the Tree**: The tree grows by recursively splitting subsets of data until certain stopping criteria are met (like reaching maximum depth or minimum samples per node).
- **Making Predictions**: New data points are passed through the tree, following the decisions at each node until a leaf node is reached, where the prediction is made.

Why to Use Decision Tree

- Easy to interpret and explain to non-technical stakeholders.
- Requires very little data preparation or feature scaling.
- Can capture nonlinear relationships between features and output.
- Useful for datasets where interpretability is important.

3. XGBoost (Extreme Gradient Boosting)

About the ML Model

XGBoost is a highly efficient and scalable implementation of gradient boosting for supervised learning tasks. It builds an ensemble of decision trees in a sequential manner, where each new tree tries to correct the errors made by the previous trees. XGBoost is designed for speed and performance, often delivering superior results compared to traditional boosting methods.

Workflow of XGBoost

- **Data Input**: It begins with the training data.
- **Initial Prediction**: A simple prediction (like the average of the target variable) is made.
- **Calculate Errors**: The errors or residuals between the prediction and actual output are calculated.
- **Build Trees**: New trees are added one at a time, each one trained to predict the residual errors from the previous iteration.
- **Update Model**: The predictions are updated by combining the old prediction with the new tree's output.
- **Repeat**: This process continues until the maximum number of trees is reached or no further improvements can be made.

Why to Use XGBoost

- Extremely fast due to parallelization and optimization techniques.
- Regularization features reduce overfitting and improve generalization.
- Can handle missing values automatically.
- Consistently provides top performance in many machine learning competitions.

4. Logistic Regression

About the ML Model

Logistic Regression is a simple, interpretable, and widely-used model for binary classification problems. Despite its name, it is a classification algorithm. It estimates the probability that a data point belongs to a particular class and assigns the class with the highest probability. Logistic Regression is based on the concept of fitting a curve that can separate two classes.

Workflow of Logistic Regression

- **Data Input**: Training data with features and binary labels (e.g., 0 or 1).
- **Model Fitting**: The model learns weights for each feature such that the weighted sum maps to a probability value between 0 and 1.
- **Prediction**: The model predicts the probability of the input belonging to a particular class.
- **Thresholding**: If the probability is above a chosen threshold (commonly 0.5), the input is classified into one class; otherwise, it is classified into the other.

Why to Use Logistic Regression

- Easy to implement and interpret.
- Works well when the relationship between features and output is approximately linear.
- Fast to train and predict.
- Performs well on smaller datasets with binary classification.

5. Naive Bayes

About the ML Model

Naive Bayes is a family of simple probabilistic classifiers based on applying Bayes' Theorem with a strong assumption of feature independence. Despite this "naive" assumption, Naive Bayes often performs very well, particularly for text classification tasks such as spam detection and sentiment analysis.

Workflow of Naive Bayes

- **Data Input**: Training data with feature values and class labels.
- **Calculate Probabilities**: For each class, the model calculates the probability of a data point belonging to that class based on its feature values.
- **Apply Bayes' Theorem**: The model combines the probabilities to compute the most likely class for a given input.
- **Prediction**: Assigns the input to the class with the highest probability.

Why to Use Naive Bayes

- Very fast to train and predict.
- Performs surprisingly well even with the strong independence assumption.
- Works particularly well for large datasets with high-dimensional feature spaces (e.g., text data).
- Requires a relatively small amount of training data to estimate parameters.

6. Neural Networks

About the ML Model

Neural Networks are machine learning models inspired by the human brain's structure. They consist of layers of interconnected nodes (neurons), where each connection has an associated weight. Neural Networks can model highly complex, non-linear relationships and are the foundation of deep learning.

Workflow of Neural Networks

- **Data Input**: The model takes in input features through an input layer.
- **Forward Propagation**: The input data passes through hidden layers where neurons apply transformations based on learned weights and activation functions.
- **Loss Calculation**: The output is compared to the actual value to calculate the loss or error.
- **Backpropagation**: The model adjusts the weights by calculating gradients and applying optimization algorithms to minimize the loss.
- **Iteration**: This process is repeated over multiple epochs to improve accuracy.

Why to Use Neural Networks

- Can capture very complex relationships between input and output.
- Highly flexible and can be adapted to a wide range of tasks (vision, speech, text, etc.).
- Capable of handling large amounts of unstructured data.
- Form the base for more advanced techniques like deep learning and reinforcement learning.

7. Random Forest

About the ML Model

Random Forest is an ensemble learning technique that builds multiple decision trees and merges their outputs to produce a more accurate and stable prediction. It combines the concept of "bagging" (Bootstrap Aggregating) and randomness in feature selection, which helps reduce overfitting and increases model generalization. Instead of relying on a single decision tree, it takes the majority vote (for classification) or average prediction (for regression) from multiple trees.

Workflow of Random Forest

- **Data Input**: The model receives training data with features and corresponding labels.
- **Bootstrap Sampling**: Multiple subsets of the original data are created by randomly sampling with replacement.
- **Building Decision Trees**: For each subset, a decision tree is built. While splitting nodes, only a random subset of features is considered, adding more randomness and diversity among trees.
- **Prediction Aggregation**:
  - For classification tasks, each tree votes for a class, and the majority class is chosen.
  - For regression tasks, the average of all tree outputs is taken as the final prediction.
- **Model Output**: The final output is more robust because it comes from an ensemble rather than a single tree.

Why to Use Random Forest

- Provides higher accuracy than individual decision trees by reducing variance and preventing overfitting.
- Handles missing values and maintains accuracy even when a large portion of data is missing.
- Can handle large datasets and high-dimensional feature spaces effectively.
- Provides feature importance, helping in feature selection and understanding model behavior.
- Works well for both classification and regression problems.

8. K-Nearest Neighbors (KNN)

About the ML Model

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm used for both classification and regression tasks. It assumes that similar data points exist close to each other. KNN classifies a new point based on the majority class of its K nearest neighbors in the feature space, making it a lazy learner (it doesn't learn a model in advance but rather stores the training data and computes predictions only when needed).

Workflow of KNN

- **Data Input**: The training data is stored, without actually building an explicit model.
- **Choosing 'K'**: Select the number of neighbors (K) to consider when making predictions.
- **Distance Calculation**: When a new data point needs to be classified, the distance (typically Euclidean) from the new point to all training points is calculated.
- **Identifying Neighbors**: The K closest training examples are identified based on the calculated distances.
- **Voting/Predicting**:
  - For classification: The most common class among the K neighbors is assigned.
  - For regression: The average value of the K neighbors is assigned.
- **Prediction Output**: The class or value determined by the neighbors is outputted as the final prediction.

Why to Use KNN

- Simple to implement and understand.
- No training phase, making it quick for small datasets.
- Naturally handles multi-class problems.
- Highly flexible, can adapt to complex decision boundaries with the right choice of K and distance metric.
- Effective for problems where the decision boundary is irregular.

6-Fold Cross-Validation and Evaluation Metrics

6-Fold Cross-Validation

Cross-validation is a technique used to **evaluate the generalization ability** of a machine learning model. Instead of simply splitting the dataset into one training and one testing set, **cross-validation** systematically splits the data multiple times to get a more reliable estimate of performance.

Specifically, in 6-fold cross-validation:

- The entire dataset is **randomly divided into 6 equally sized parts** (called "folds").
- In each round of validation:
  - **5 folds are used for training**, and
  - **1 fold is used for testing**.
- This process is repeated **6 times**, each time with a different fold serving as the test set.
- **At the end**, the results (accuracy, precision, etc.) from all 6 rounds are **averaged** to give the final reported performance.

Benefits of 6-fold cross-validation:

- **More stable and unbiased** than a single train/test split.
- Helps in **reducing variance** caused by random data division.
- Ensures **every sample** gets a chance to be in the test set exactly once.
- **Balances computational efficiency and evaluation robustness** (especially compared to higher-fold CV like 10-fold or leave-one-out).

Metrics Used in 6-Fold Cross-Validation

In your study, for each fold, the following metrics are calculated based on the **confusion matrix** of predicted vs true labels:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

All metrics below are derived from **TP, TN, FP, FN**.

1. Accuracy (%)

Definition:
The proportion of total predictions (both positive and negative) that are correct.

Accuracy = (TP + TN) / (TP + TN + FP + FN) × 100

In context:
Accuracy tells us **how often the model correctly identifies the copper ion concentration**, whether high, medium, or low.

2. Precision (%)

Definition:
Of all the instances predicted as positive (high copper ion concentration, for example), how many were actually positive.

Precision = TP / (TP + FP) × 100

In context:
High precision means **few false alarms** — when the model says "high concentration," it's **very likely to be correct**.

3. Recall (%) or Sensitivity (%)

Definition:
Of all the actual positive cases, how many were correctly predicted.

Recall (Sensitivity) = TP / (TP + FN) × 100

In context:
High recall ensures **few missed cases** — i.e., if the concentration is high, the model **rarely misses it**.

4. F1-Score (%)

Definition:
The harmonic mean of precision and recall. It **balances** the trade-off between them.

F1-Score = 2 (Precision × Recall) / (Precision + Recall)

In context:
F1-score is important when **both false positives and false negatives are costly**, as might be the case in **critical ion detection**.

5. Specificity (%)

Definition:
Of all the actual negative cases, how many were correctly predicted.

Specificity = TN / (TN+FP) × 100

In context:
Specificity measures the ability to **correctly identify "normal" or "low" concentrations** without falsely classifying them as high.

How Metrics Are Reported After 6-Fold Cross-Validation

For each fold:

- Calculate the confusion matrix.
- From the confusion matrix, calculate Accuracy, Precision, Recall, Specificity, and F1-Score.

After 6 folds:

- **Average** each metric across all folds:
    - 6 accuracies → averaged into a final **6-fold accuracy**.
    - 6 precisions → averaged into final **6-fold precision**, etc.

Thus, the final reported values (e.g., the tables you shared earlier) represent the **mean performance** of the model across all cross-validation folds, making it **more reliable and less biased** than a single evaluation.

Importance of 6-Fold Metrics in Your Copper Ion Study

- **Accuracy** gives a broad view but can be misleading if the classes are imbalanced (e.g., more samples of low concentration).
- **Precision** is crucial if **false positives are dangerous** (e.g., falsely indicating high contamination).
- **Recall (Sensitivity)** is critical if **missing a real high-concentration sample is risky**.
- **F1-score** balances precision and recall — highly useful if both false alarms and missed detections are undesirable.
- **Specificity** ensures that **clean samples are not mistakenly classified as contaminated**, which could prevent unnecessary alarm.

By reporting all these metrics from **6-fold cross-validation**, your study **demonstrates the robustness, reliability, and practical applicability** of your models for copper ion concentration classification.

Model Training Setup

Stage	Percentage of Data Used
Training	80%
Validation	10%
Testing	10%

What Each Stage Means

1. Training Set (80% of the Data)

- The model **learns patterns** from this part.
- It **adjusts its internal parameters** (e.g., weights for a neural network, split criteria for a decision tree).
- **Goal: Minimize error** on this set during training.
- The model sees these examples multiple times during **epochs** (especially in deep learning).

Example:
Imagine you have 1000 images of copper ion concentrations.
You will use **800 images** to **teach the model** how different concentration levels look.

2. Validation Set (10% of the Data)

- Used to **monitor model performance during training**.
- Helps in **hyperparameter tuning**:
    - E.g., deciding the learning rate, number of hidden layers, regularization strength.
- **Not used** to update the model weights.
- It **checks whether the model is generalizing** well to unseen examples, and helps **avoid overfitting** (memorizing training data too much).

- Often, **early stopping** is based on validation loss:
    - If validation error starts increasing, training is stopped to prevent overfitting.

Example:
From the 1000 images, **100 images** are kept aside during training. After each epoch, the model checks how well it is doing on these 100 images — if performance starts to degrade, it adjusts.

3. Test Set (10% of the Data)

- **ONLY used once**, after the model is fully trained and tuned.
- **Completely unseen** by the model.
- Gives the **final unbiased evaluation** of how well the model would perform in real-world deployment.
- Important: **No information from the test set leaks** into training or validation.

Example:
The remaining **100 images** are used **only at the very end**, to report the final performance numbers (accuracy, precision, recall, etc.) in your research paper or report.

Why This 80/10/10 Split?

- **Enough data for training:** 80% ensures the model learns enough patterns.
- **Validation is crucial:** 10% is enough to **detect overfitting** without wasting too much data.
- **Testing is unbiased:** 10% test set is standard to **report honest final results**.

Experimental Results

Model Performance Comparison

In this study, we systematically evaluated multiple supervised learning algorithms for the task of **copper ion concentration classification** based on two imaging modalities: **colorimetric** and **fluorometric** images. Each model's performance was assessed using five critical evaluation metrics: **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **Specificity**. These metrics collectively provide a comprehensive view of both sensitivity to true positives and robustness against false positives. The detailed numerical results are summarized in Tables 1 and 2.

Performance on Colorimetric Images

Colorimetric images capture color changes resulting from the interaction between copper ions and chromogenic agents. These images, while useful, often present **subtle and gradual color variations**, leading to **lower feature separability**.

Among all models:

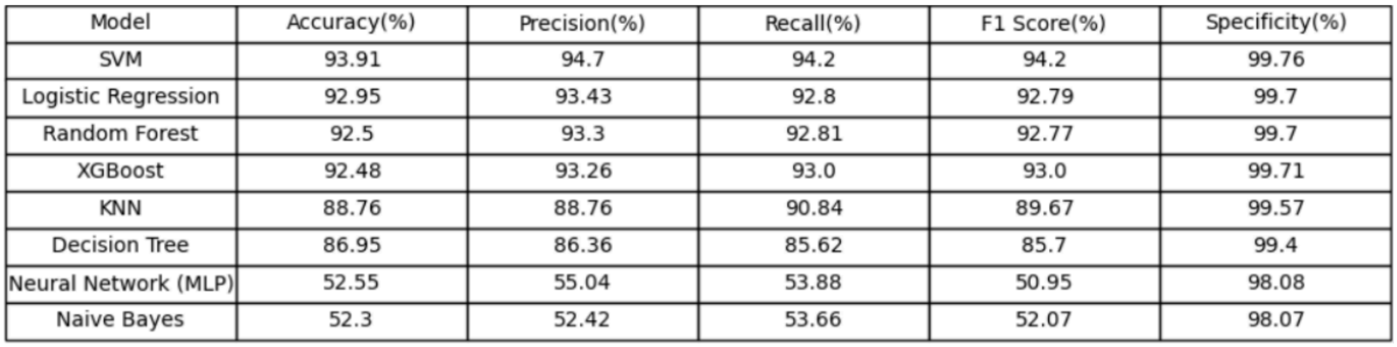
- **Support Vector Machine (SVM)** achieved the **highest overall performance** with an **Accuracy of 93.91%**, **Precision of 94.7%**, and **Recall of 94.2%**.
    - SVM's superior performance can be attributed to its **ability to construct high-dimensional decision boundaries (hyperplanes)**, which are well-suited for **datasets with subtle inter-class variation** like colorimetric images.
    - The high **Specificity (99.76%)** shows that SVM is effective at minimizing false positives, a critical factor when misclassification could lead to wrong concentration estimates.
- **Logistic Regression (LR)** and **Random Forest (RF)** followed closely behind:
    - LR achieved **92.95%** accuracy, while RF achieved **92.5%**.
    - Logistic Regression benefits from its **probabilistic interpretation** and performs well when data is **linearly separable**, even if marginally.
    - Random Forest leverages **ensemble learning**, aggregating multiple decision trees to **reduce overfitting** and **improve generalization**, which is valuable for noisy colorimetric images.
- **XGBoost**, a more sophisticated ensemble method, achieved a **92.48% accuracy** and a **93.00% F1-Score**.
    - XGBoost's **boosting approach** corrects misclassifications iteratively, making it adept at **handling complex, non-linear boundaries**, which are common in real-world colorimetric datasets.

Mid-tier performers:

- **K-Nearest Neighbors (KNN)** and **Decision Tree** models showed moderate performance:
    - KNN achieved **88.76%** accuracy. Its instance-based nature makes it sensitive to **local variations**, but it struggles with **high-dimensional noise** in color data.
    - Decision Tree obtained **86.95%** accuracy. Though simple and interpretable, decision trees are prone to **overfitting** when working with subtle gradient variations typical of colorimetric data.

Poor performers:

- **Multilayer Perceptron (MLP)** and **Naive Bayes** underperformed:
    - MLP achieved **52.55%** accuracy, while Naive Bayes achieved **52.3%**.
    - The MLP's poor performance could be due to its **inadequate training** (possible overfitting, underfitting, or unsuitable architecture for small feature shifts).
    - Naive Bayes assumes **feature independence**, which is unrealistic for color-based features where pixel intensities are often **highly correlated**, leading to major classification errors.



Model	Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	Specificity(%)
SVM	93.91	94.7	94.2	94.2	99.76
Logistic Regression	92.95	93.43	92.8	92.79	99.7
Random Forest	92.5	93.3	92.81	92.77	99.7
XGBoost	92.48	93.26	93.0	93.0	99.71
KNN	88.76	88.76	90.84	89.67	99.57
Decision Tree	86.95	86.36	85.62	85.7	99.4
Neural Network (MLP)	52.55	55.04	53.88	50.95	98.08
Naive Bayes	52.3	52.42	53.66	52.07	98.07

Table 2: Results for Colorimetric Images

Performance on Fluorometric Images

Fluorometric images capture **fluorescence emission** induced by copper ion concentration. Unlike colorimetric images, fluorometric data tends to exhibit **sharper contrast and more distinct feature patterns**, leading to **higher inter-class separability**.

Here, **all models improved** compared to their performance on colorimetric images:

- **Support Vector Machine (SVM)** again achieved **the best results**:
  - **Accuracy of 96.77%**, **Precision of 97.15%**, **Recall of 97.00%**, **F1-Score of 97.02%**, and an **impressive Specificity of 99.88%**.
  - The sharper boundaries in fluorometric images allowed SVM to **find a more optimal separating hyperplane**, leading to improved overall classification.
- **Logistic Regression** also excelled with **96.29% accuracy** and a **96.81% F1-score**.
  - This highlights that, when features are well-separated (as with fluorometric images), even **simple linear models** can perform extremely well.
- **Random Forest and XGBoost** continued to demonstrate **strong robustness**:
  - Random Forest: **95.19% accuracy**, **95.42% F1-Score**.
  - XGBoost: **94.65% accuracy**, **96.01% F1-Score**.
  - These ensemble methods are particularly powerful for fluorometric images due to their ability to **model non-linear interactions** between features and **boost the impact of informative patterns**.

Mid-level models:

- **KNN** performed decently, with **93.33% accuracy**, although still lagging behind SVM and ensemble methods.
  - KNN's reliance on **local feature space distances** works better with fluorometric data but can still be affected by **noise and outliers**.
- **Decision Tree** reached **88.82% accuracy**:
  - While better than on colorimetric images, Decision Trees still **overfit** easily unless heavily pruned, limiting their maximum achievable accuracy.

Low performers:

- **MLP** achieved **only 52.97% accuracy**, and **Naive Bayes** reached **67.17%**.
  - The MLP's poor results suggest that its **learning architecture might not have been**

complex enough or lacked sufficient regularization techniques to deal with high-dimensional image features.

- ○ Naive Bayes, again limited by its **independence assumption**, failed to model the feature correlations inherent even in fluorometric data.

Model	Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	Specificity(%)
SVM	96.77	97.15	97.0	97.02	99.88
Logistic Regression	96.29	97.05	96.8	96.81	99.87
Random Forest	95.19	95.6	95.4	95.42	99.81
XGBoost	94.65	96.18	96.0	96.01	99.83
KNN	93.33	94.28	94.0	94.01	99.75
Decision Tree	88.82	88.46	87.6	87.67	99.48
Naive Bayes	67.17	68.72	67.63	67.62	98.65
Neural Network (MLP)	52.97	64.05	64.84	61.18	98.54

Table 3: Results for Fluorometric Images

Results for Metrics Used in 6-Fold Cross-Validation:

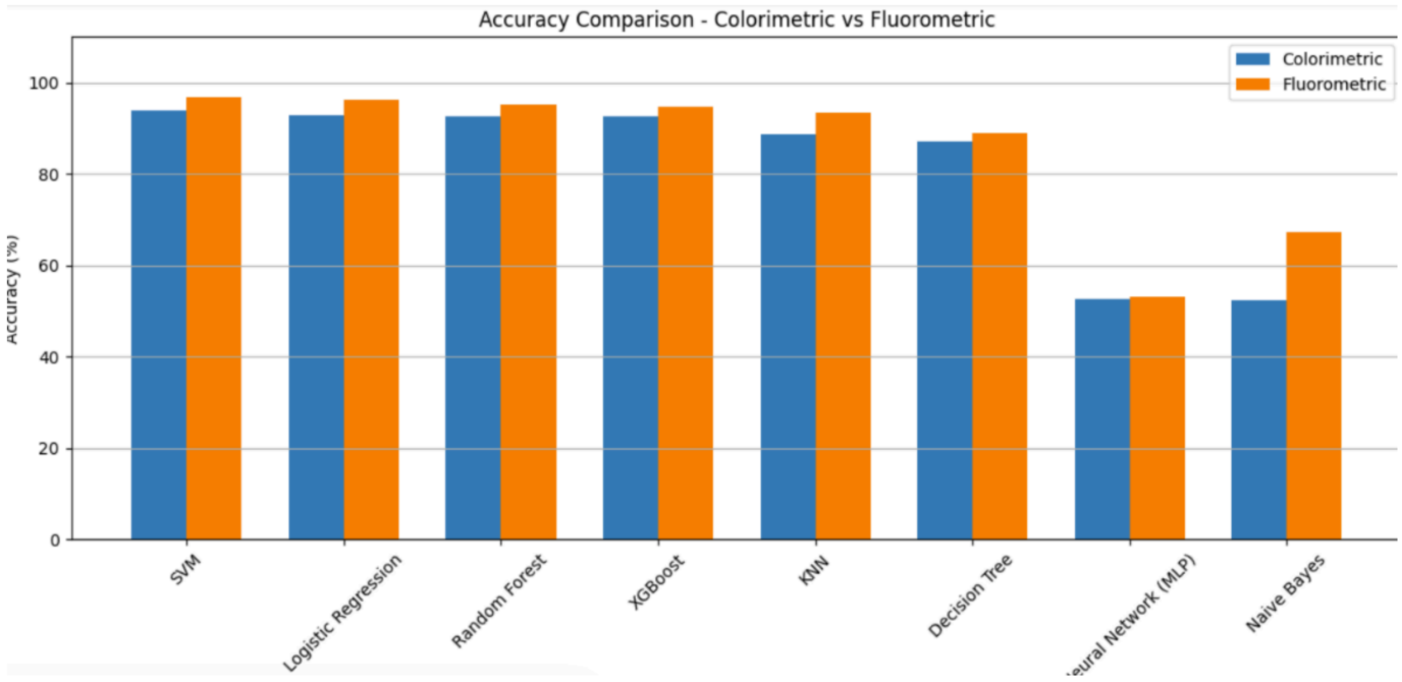


Figure 8: Accuracy Comparison- Colorimetric vs Fluorometric

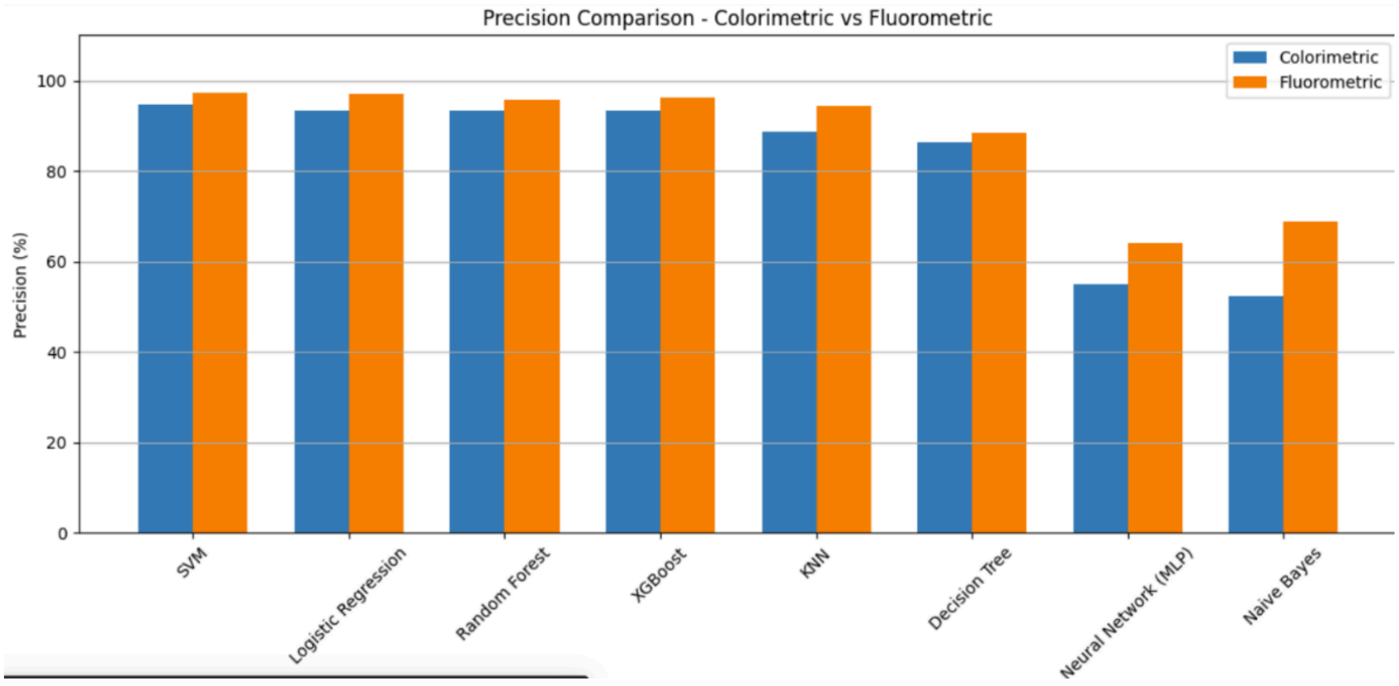


Figure 9 : Precision Comparison- Colorimetric vs Fluorometric

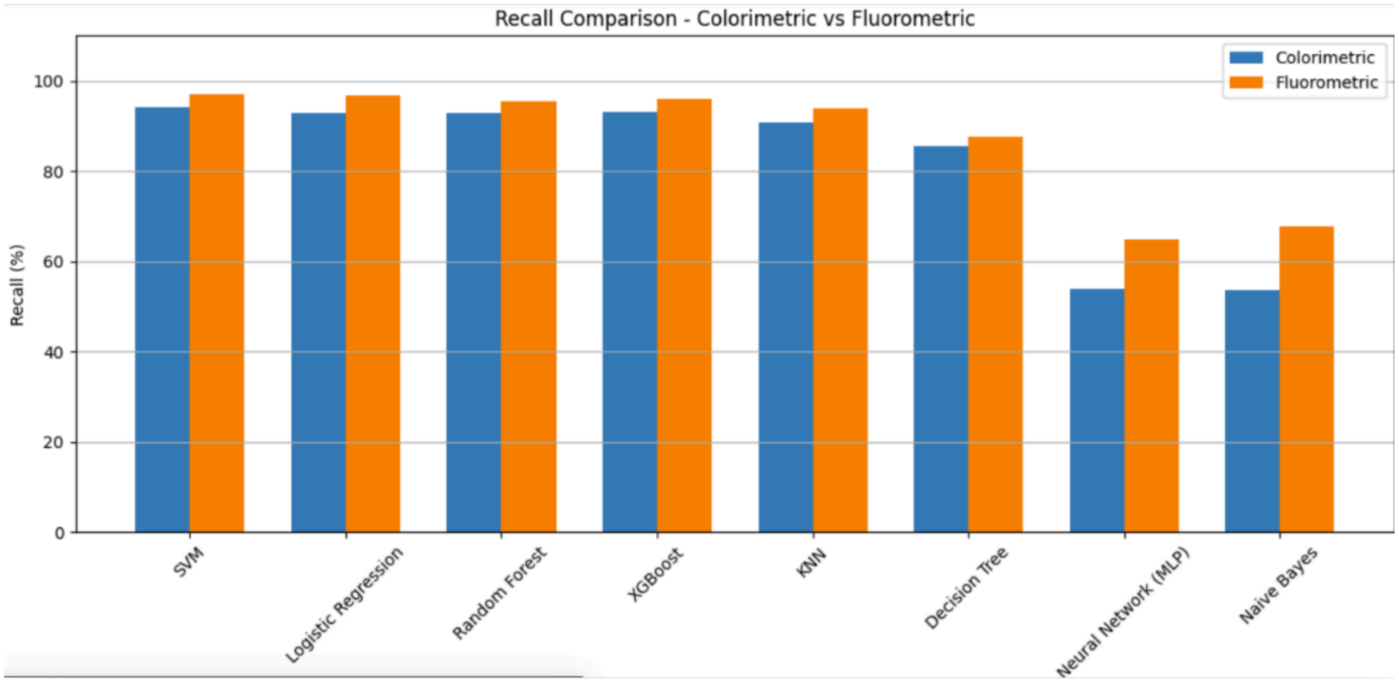


Figure 10 : Recall Comparison- Colorimetric vs Fluorometric

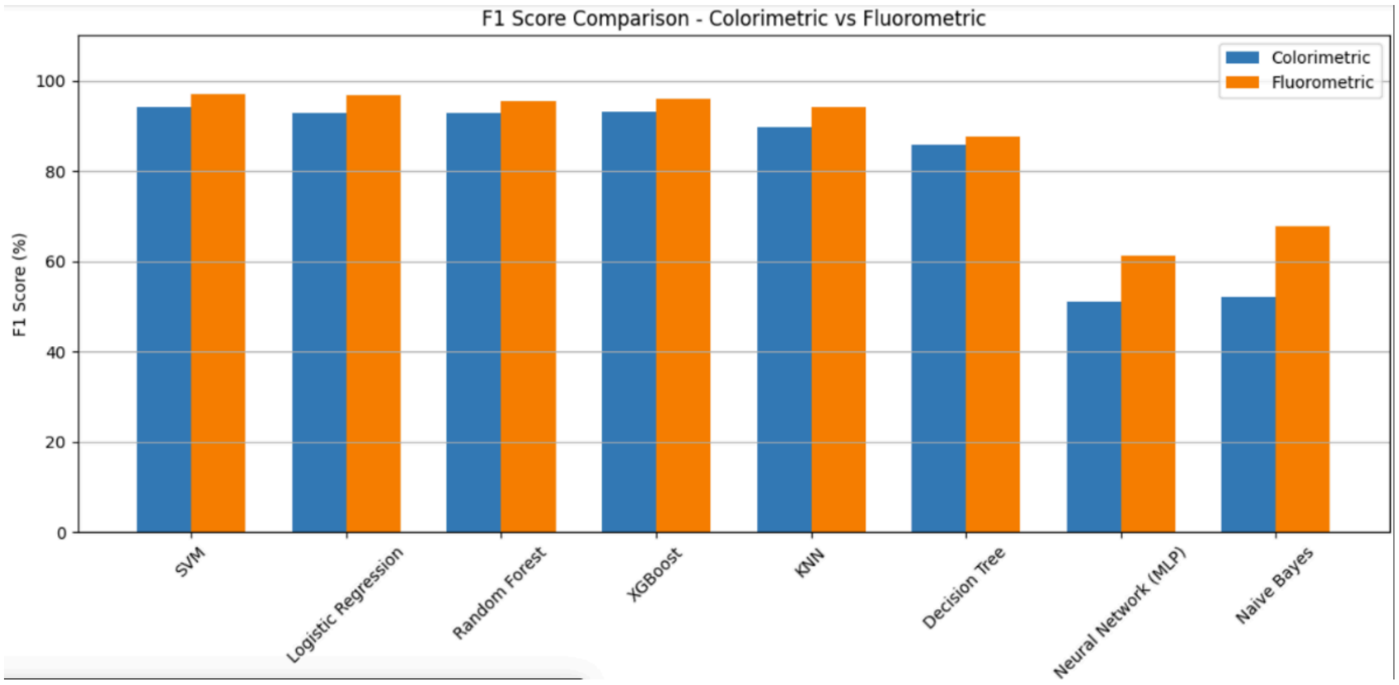


Figure 11 : F1 Score Comparison- Colorimetric vs Fluorometric

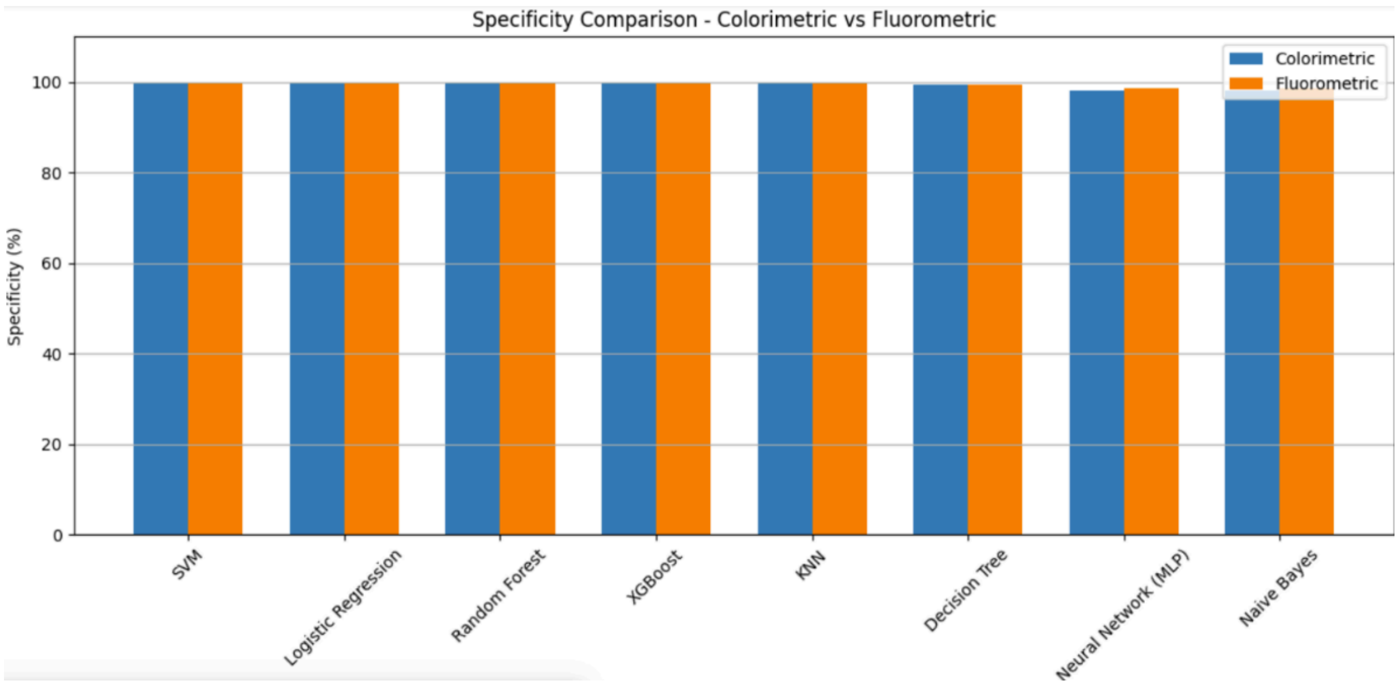


Figure 12 : Sensitivity Comparison- Colorimetric vs Fluorometric

Observations and Key Insights

- **Fluorometric imaging is significantly more informative** than colorimetric imaging for classifying copper ion concentration, given the consistently higher scores across all models.

- **SVM consistently outperformed other models** across both datasets, highlighting its versatility and robustness, especially in moderate to high-dimensional feature spaces.

- **Ensemble models** like Random Forest and XGBoost provided **strong alternatives**, combining multiple weak learners to achieve competitive and often near-optimal performance.

- **Simple models** (Logistic Regression) can be extremely effective **when feature separability is high**, as seen in fluorometric images.

- **Neural networks (MLP)** and **Naive Bayes** are **not ideal** in this context without significant tuning or feature engineering.

Water Quality Analysis Application

Executive Summary

The Water Quality Analysis application is a web-based tool designed to analyze water samples for copper concentration levels. The system utilizes machine learning to process uploaded images of water samples, determine copper concentrations, and provide risk assessments based on established safety thresholds. The application features a complete user authentication system, an intuitive dashboard for viewing historical analyses, and detailed visualization of analysis results.

System Architecture

The application is built using a Flask web framework with a structured MVC architecture. Key components include:

1. User Authentication System: Complete login, signup, and session management functionality
2. Machine Learning Model: An SVM-based model (`svm_colorimetry_model.pkl`) for analyzing water sample images
3. Database Layer: SQLAlchemy ORM with User and Analysis models
4. Web Interface: Responsive Bootstrap-based UI with interactive visualizations

Core Functionality

Water Sample Analysis

The core functionality revolves around analyzing water samples for copper concentration:

1. Users upload images of water samples through the analysis interface
2. The system preprocesses the image for the machine learning model
3. The model predicts copper concentration levels in mg/L
4. Results are classified into risk categories: Safe (<1.0 mg/L), Normal (1.0-1.3 mg/L), Elevated (1.3-2.0 mg/L), Risky (2.0-2.5 mg/L), or Hazardous (>2.5 mg/L)
5. Additional parameters like pH level are measured and displayed

Data Visualization

The application presents analysis results through several visualization techniques:

1. Risk Indicator: Color-coded badges (green, blue, yellow, red) indicating risk level
2. Concentration Display: Prominent display of copper concentration value in mg/L
3. pH Scale Visualization: Interactive gradient scale showing water acidity/alkalinity
4. Risk Gauge: Canvas-based gauge visualization of concentration levels

User Experience Flow

The application provides a seamless user journey:

1. Authentication: Users begin by creating an account or logging in
2. Dashboard: Upon login, users see their analysis history with quick-access controls
3. Analysis: Users can upload new water sample images for immediate analysis
4. Results: Detailed analysis results are presented with interpretations and recommendations
5. History: All analyses are stored and accessible from the dashboard for tracking water quality over time

Technical Implementation

Machine Learning Component

The image analysis system:

1. Takes uploaded water sample images as input
2. Resizes images to 64x64 pixels for standardization
3. Normalizes pixel values by dividing by 255.0
4. Flattens the image array into a feature vector
5. Uses the trained SVM model to predict copper concentration
6. Returns the concentration value and associated risk level

Database Schema

Two primary data models support application functionality:

1. User Model:
   - Basic user information (username, email, password hash)
   - Account creation timestamp
   - Relationship to analyses

2. Analysis Model:
   - Foreign key relationship to user
   - Timestamp of analysis
   - Image path for the uploaded sample
   - Copper concentration value
   - Risk level classification
   - JSON-encoded detailed analysis data

User Interface

The interface employs modern design principles:

1. Card-Based Layout: Clean organization of information in shadow cards
2. Responsive Design: Bootstrap grid system ensures compatibility across devices
3. Interactive Elements: Dynamic loading states, image previews, and animated visualizations
4. Color-Coding: Consistent color scheme for risk levels (green, blue, yellow, red)
5. Clear Navigation: Intuitive links between dashboard, analysis, and results pages

Recommendations for Future Development

Based on the system review, several potential enhancements could be considered:

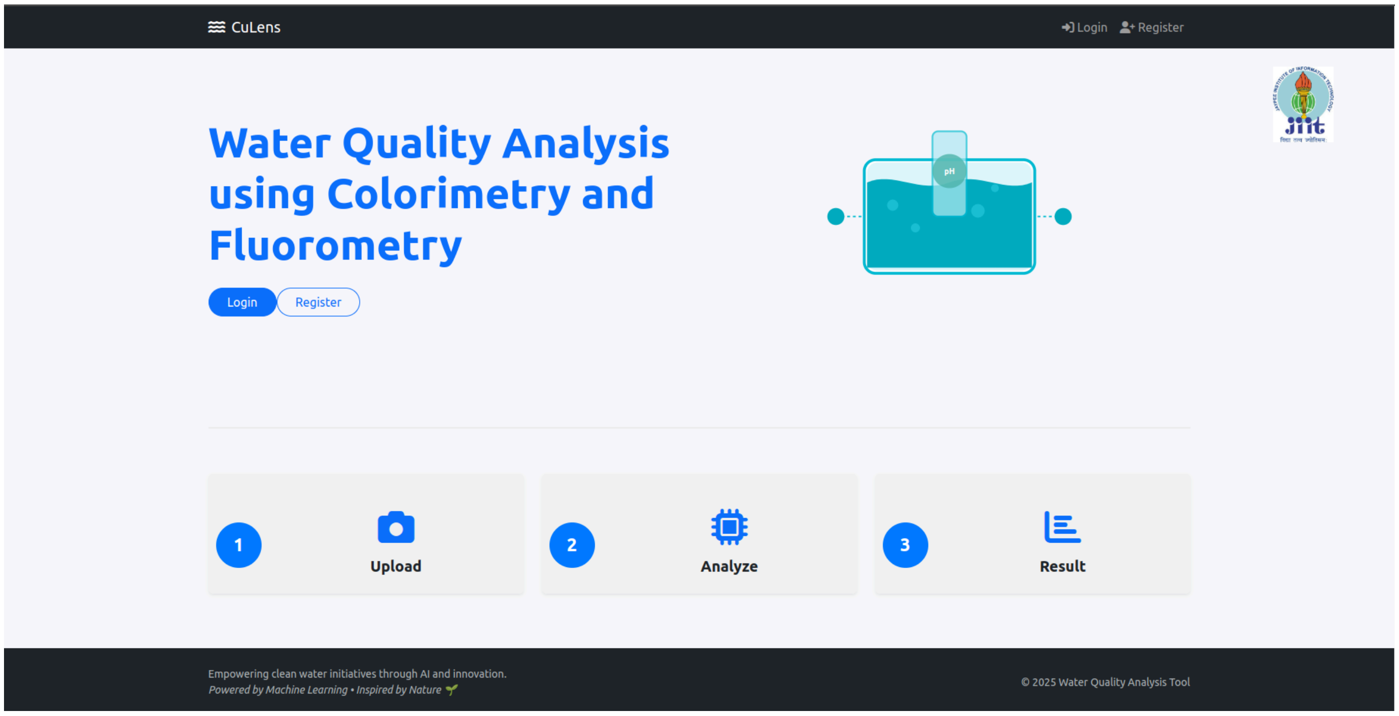
1. Model Refinement: Incorporate additional water quality parameters beyond copper concentration
2. Trend Analysis: Add time-series visualization of water quality changes across multiple analyses
3. Geolocation: Integrate location tracking for analyses to map water quality across regions
4. Mobile App Integration: Develop a companion mobile application for field testing
5. Automated Alerts: Implement notification system for dangerous water quality readings
6. API Endpoints: Create RESTful API endpoints to allow integration with other systems

Conclusion

The Water Quality Analysis application successfully implements a comprehensive system for analyzing copper concentration in water samples. The integration of machine learning with an intuitive user interface provides an accessible tool for water quality monitoring. The application demonstrates strong technical architecture while maintaining focus on user experience and practical utility.

UI INTERFACE:

Home Page

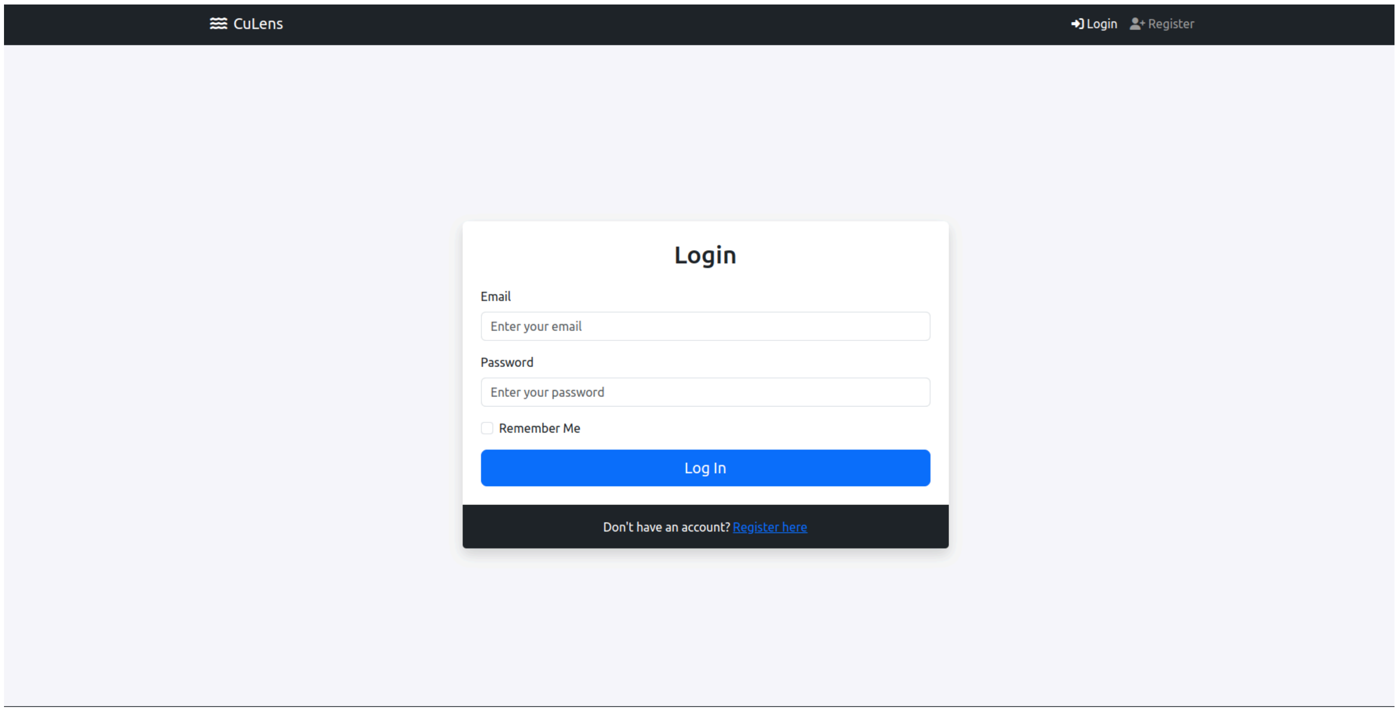


Water Quality Analysis using Colorimetry and Fluorometry

Empowering clean water initiatives through AI and innovation.
Powered by Machine Learning • Inspired by Nature

© 2025 Water Quality Analysis Tool

Login Page:



Login

Email

Enter your email

Password

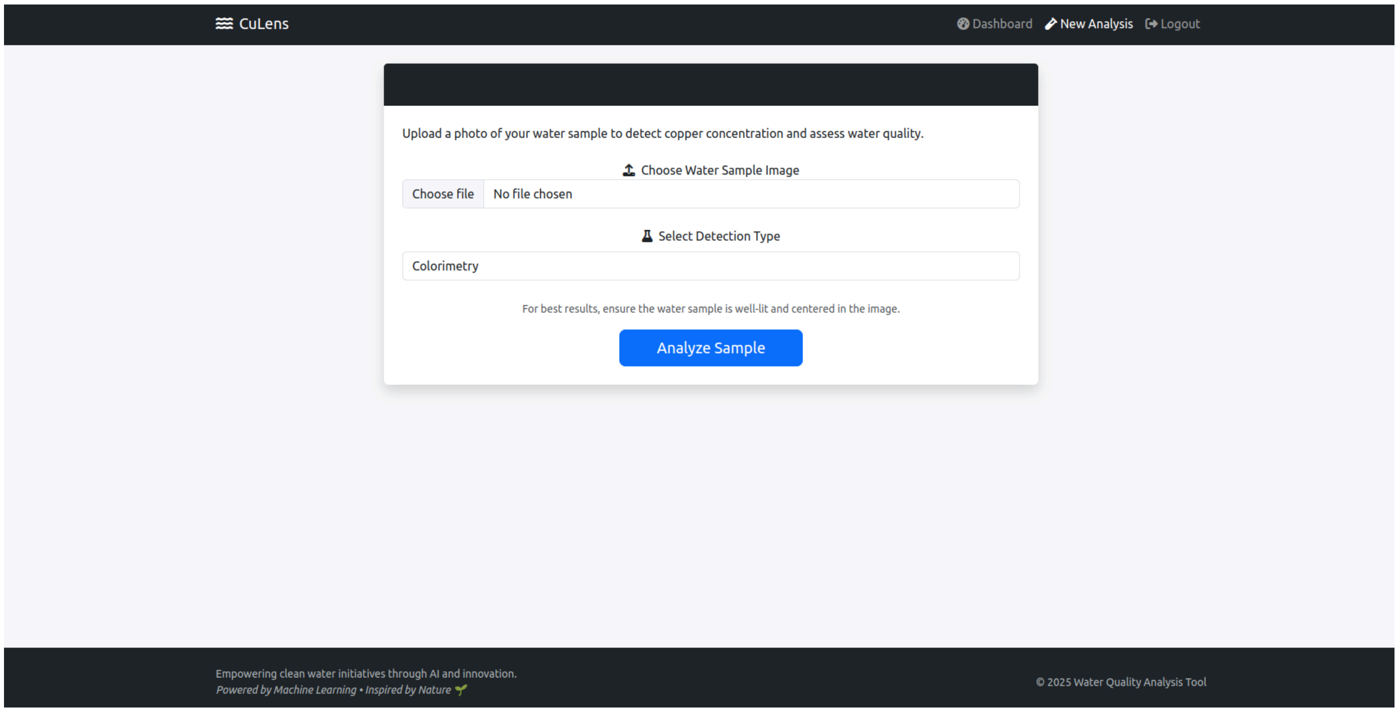
Enter your password

Remember Me

Log In

Don't have an account? Register here

Analyze Page:



Upload a photo of your water sample to detect copper concentration and assess water quality.

Choose Water Sample Image

Choose file No file chosen

Select Detection Type

Colorimetry

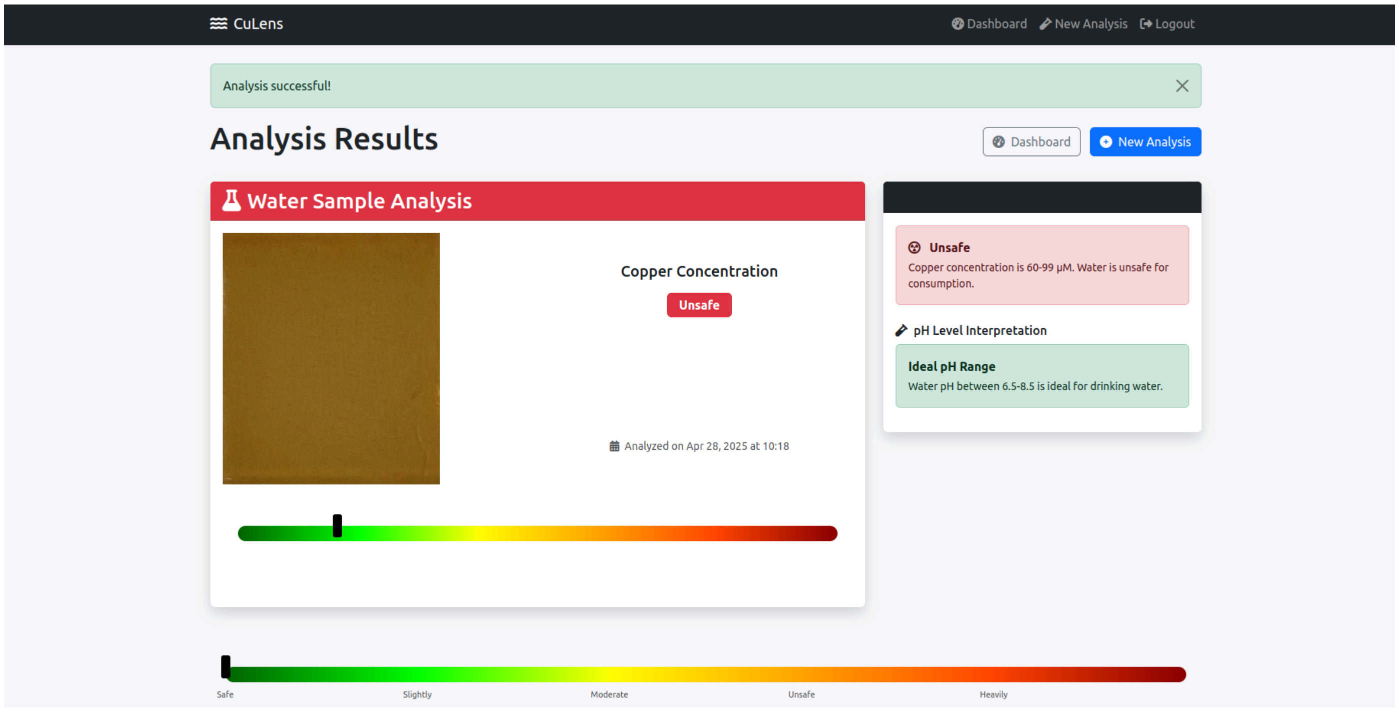
For best results, ensure the water sample is well-lit and centered in the image.

Analyze Sample

Empowering clean water initiatives through AI and innovation.
Powered by Machine Learning • Inspired by Nature

© 2025 Water Quality Analysis Tool

Result Page:



CuLens

Dashboard New Analysis Logout

Analysis successful!

Analysis Results

Dashboard New Analysis

Water Sample Analysis

Copper Concentration

Unsafe

Analyzed on Apr 28, 2025 at 10:18

Unsafe

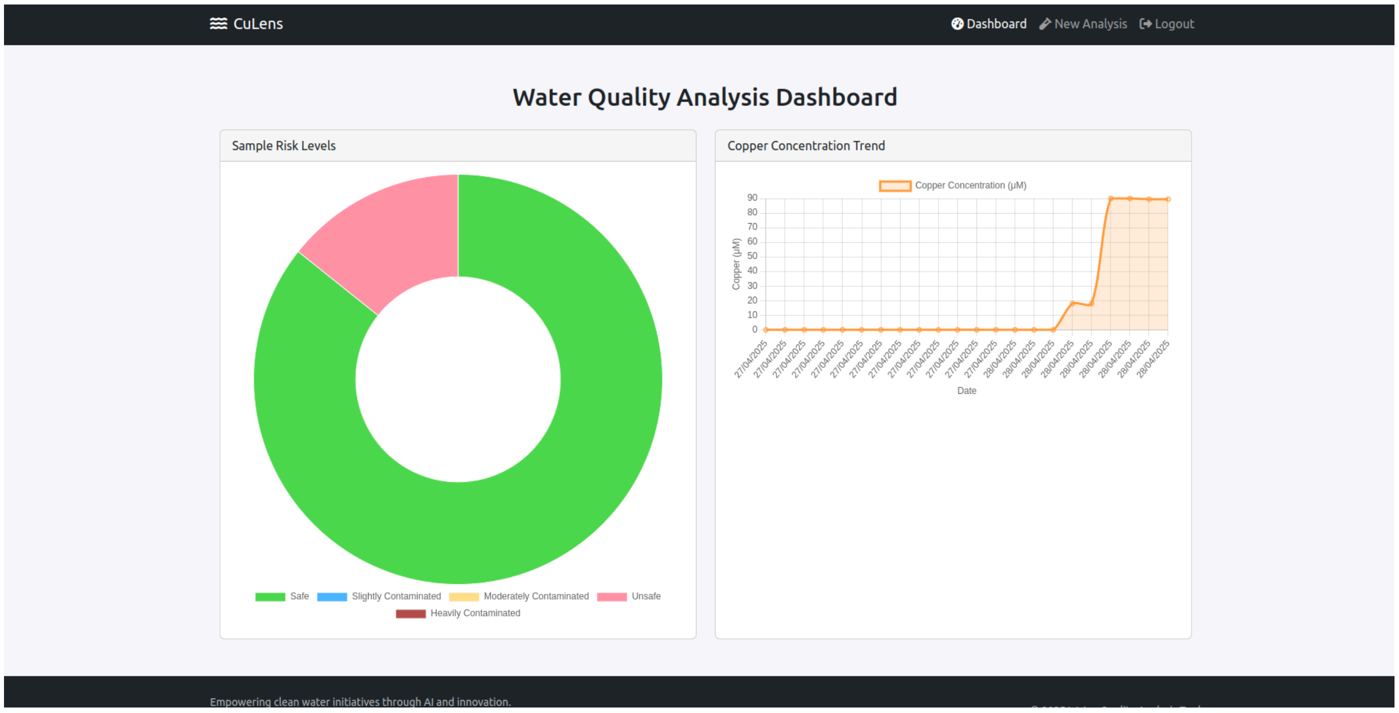
Copper concentration is 60-99 μM. Water is unsafe for consumption.

pH Level Interpretation

Ideal pH Range

Water pH between 6.5-8.5 is ideal for drinking water.

User Dashboard:



CuLens
Dashboard
New Analysis
Logout
Water Quality Analysis Dashboard
Sample Risk Levels
Copper Concentration Trend
Copper Concentration (μM)
Copper (μM)
90
80
70
60
50
40
30
20
10
0
27/04/2025
28/04/2025
Date
Safe
Slightly Contaminated
Moderately Contaminated
Unsafe
Heavily Contaminated
Empowering clean water initiatives through AI and innovation.